TRABAJO FIN DE GRADO

INGENIER´IA INFORMÁTICA Y ADMINISTRACIÓN
Y DIRECCIÓN DE EMPRESAS

Sistema de Gestión para Supermercados

Diseño e Implementación de una Aplicación Web con un Sistema para gestionar los pedidos y existencias en Supermercados

Autor

Eulalio Martínez Ríos

Director

José Manuel Benítez Sánchez

Escuela Técnica Superior de Ingenierías Informática
y de Telecomunicaciones

—

Granada, Julio de 2024

Sistema de Gestión para Supermercados

Eulalio Martínez Ríos

Palabras clave: *Sistema de Gestión, Supermercado, Aplicación Web, Programación.*

Resumen

Contar con un sistema que ayude en la administración de un negocio de tamaño mediano o pequeño se ha vuelto indispensable en la actualidad. Usar programas anticuados que tienen más de veinte años o llevar las cuentas a mano nos parece algo muy pasado de moda. Este proyecto introduce un sistema completo creado para resolver las deficiencias de estos sistemas, ofreciendo una plataforma web actualizada que simplifica la organización logística de supermercados medianos.

La estructura del proyecto consta de múltiples etapas, iniciando con un minucioso análisis de las necesidades del supermercado y los obstáculos a los que se enfrentan los empleados del establecimiento con el control de fechas de caducidad y demás tareas diarias. Con base en esta evaluación, se diseñó un sistema que no solo automatiza dichas funciones, sino que también mejora la experiencia del usuario que lo utiliza mediante una interfaz fácil de usar y accesible, sin apenas curva de aprendizaje. La integración del sistema en una plataforma web fue parte del proceso de desarrollo e implementación. Durante la etapa final de pruebas, se llevaron a cabo análisis para detectar y solucionar errores potenciales, garantizando que el sistema cumpliera con los requisitos iniciales y las expectativas del cliente. Este trabajo se consideró exitoso gracias a la retroalimentación de los usuarios, la cual demostró una mejora significativa tanto en la eficacia esperada como en la satisfacción personal. Este proyecto no solo cumple con los objetivos planteados inicialmente, sino que también establece una base sólida para futuras expansiones. Se pueden realizar futuras acciones, como agregar funciones extras para manejar las nóminas o emplear inteligencia artificial para mejorar las operaciones del supermercado, entre otras posibilidades. Puedo confirmar que este proyecto muestra que es posible y beneficioso actualizar los sistemas de gestión de un negocio mediante una plataforma web, proporcionando soluciones adaptables que mejoran la eficiencia laboral y la experiencia del usuario.

Supermarket Management System

Eulalio Martínez Ríos

Keywords: *Management System, Supermarket, Web Application, Programming.*

Abstract

Having a system that assists in the administration of a medium or small-sized business has become indispensable today. Using outdated programs that are more than twenty years old or keeping accounts by hand seems very outdated to us. This project introduces a comprehensive system designed to address the deficiencies of these systems, offering an updated web platform that simplifies the logistical organization of medium-sized supermarkets.

The structure of the project consists of multiple stages, starting with a thorough analysis of the supermarket's needs and the obstacles faced by the employees, such as managing expiration dates and other daily tasks. Based on this evaluation, a system was designed that not only automates these functions but also enhances the user experience through an easy-to-use and accessible interface, with virtually no learning curve. The integration of the system into a web platform was part of the development and implementation process. During the final testing phase, analyses were conducted to detect and resolve potential errors, ensuring that the system met the initial requirements and client expectations. This work was considered successful thanks to user feedback, which demonstrated a significant improvement in both expected efficiency and personal satisfaction. This project not only meets the initial objectives but also lays a solid foundation for future expansions. Future actions could include adding extra features to handle payroll or employing artificial intelligence to enhance supermarket operations, among other possibilities. I can confirm that this project shows it is possible and beneficial to update a business's management systems through a web platform, providing adaptable solutions that improve work efficiency and user experience.

D. **José Manuel Benítez Sánchez**, catedrático del departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Sistema de Gestión para Supermercados***, ***Diseño e Implementación de una Aplicación Web con un Sistema para gestionar los pedidos y existencias en Supermercados***, ha sido realizado bajo su supervisión por **Eulalio Martínez Ríos**, y autoriza la defensa de dicho trabajo ante el tribunalquecorresponda.

Y para que conste, expide y firma el presente informe en Granada a 10 de Junio de 2024

El director:

José Manuel Benítez Sánchez

Agradecimientos

Llegar a este momento que culmina mi carrera universitaria no habría sido posible sin el apoyo incondicional de aquellos que me han acompañado en este camino.

A mi familia. Gracias

Índice

Índice de figuras

IMÁGENES

Índice de tablas

TABLAS

Capítulo 1
Introducción

Parece obvio que el primer paso a dar en la redacción de esta memoria sea el de comentar aquello que sienta las bases del presente proyecto así como los aspectos fundamentales que lo contextualizan. Por ello, este capítulo trata sobre cómo surge la idea de realizar este proyecto, destacando cómo desde la experiencia y los retos del día a día a los que se enfrenta el propietario de un supermercado surge una idea y la motivación que impulsa la creación de esta solución personalizada. Se comenzará hablando sobre la motivación tanto personal como profesional que moldea esta iniciativa (véase apartado 1.1).

También se contextualizará la situación actual de ciertos sistemas de gestión actuales utilizados en los supermercados, destacando las limitaciones y deficiencias que enfrentan (véase apartado 1.2). Acto seguido se detallan los objetivos fundamentales que guiarán el desarrollo del proyecto (véase apartado 1.3). Y por último se hace una aclaración de las distintas partes que componen esta memoria (véase apartado 1.4).

1.1. Motivación

Este proyecto nace de la necesidad concreta de abordar un problema actual en la logística de los supermercados de hoy en día. La motivación que lo impulsa surge desde mi propia experiencia, que es muy cercana a esta problemática enfrentada, ya que tengo una relación muy estrecha con la familia propietaria de dos establecimientos de este tipo, en el corazón de la ciudad de Granada. Durante múltiples visitas a dichos supermercados y en

charlas con ellos, he presenciado los obstáculos a los que se enfrentan diariamente en la gestión de sus negocios. Así que la planificación de un sistema personalizado no solo resuelve un problema, sino que también permite aplicar los conocimientos adquiridos en mis estudios de Ingeniería Informática y Administración y Dirección de Empresas. Puedo decir que este proyecto no solo representa un reto técnico, sino también una posibilidad para crear un impacto positivo en entornos empresariales reales. Principalmente en los momentos previos a comenzar, aunque también durante el desarrollo del propio proyecto, he llevado a cabo entrevistas exhaustivas con la gerencia de los establecimientos, en adelante en el proyecto, el cliente; para comprender sus requerimientos y expectativas. Estas conversaciones han sido cruciales para poder saber y definir los objetivos y el alcance del sistema de información; asegurando claro, que la solución propuesta fuera efectiva y pertinente.

1.2. Contexto y definición del problema

En el contexto actual, los sistemas de gestión utilizados en algunos supermercados tradicionales a menudo adolecen de obsolescencia y tienen limitaciones significativas. Estos programas heredados han sido diseñados para satisfacer necesidades específicas en el pasado, pero su capacidad para adaptarse a las demandas cambiantes del mercado es cada vez más limitada. En concreto, el programa utilizado actualmente por este supermercado tiene una interfaz muy rudimentaria, similar al estilo de un terminal MS-DOS, lo que dificulta considerablemente la navegación por los menús y el uso eficiente del sistema.

La interfaz poco amigable contribuye a una curva de aprendizaje elevadísima, lo que hace que la tarea de aprender a utilizar el programa sea ardua y poco intuitiva para los empleados. Esta falta de usabilidad no solo ralentiza las operaciones diarias, sino que también aumenta los costos asociados a la formación del personal y a la ejecución de tareas rutinarias. Además, uno de los principales problemas es el manejo ineficiente de las fechas de caducidad en los productos. La ausencia de un sistema automatizado para el seguimiento y la gestión de las fechas de vencimiento resulta en una carga adicional para el propietario, quien debe llevar un control en la memoria de los productos y sus fechas de caducidad. Este desafío se traduce en un desgaste mental significativo y en pérdidas económicas debido a la incapacidad para aprovechar plenamente los productos antes de su caducidad.

Esta situación subraya la necesidad urgente de desarrollar soluciones innovadoras y personalizadas que mejoren la experiencia del usuario y optimicen los procesos logísticos en el sector minorista de alimentos. El proyecto propuesto busca abordar esta problemática al proporcionar un sistema de información moderno y fácil de usar, diseñado específicamente

para superar las limitaciones de los sistemas heredados y mejorar la eficiencia operativa en los supermercados.

1.3. Objetivos

El objetivo fundamental de desarrollar este proyecto es obtener una aplicación web que integre un sistema de gestión para un supermercado. Acompañando a este objetivo principal, hay una serie de objetivos que abarcan desde la dicha conceptualización hasta la implementación y evaluación del sistema propuesto. Por tanto los principales objetivos que se persiguen son los siguientes:

- **Integración y optimización de un sistema en una plataforma web**: El objetivo principal del proyecto es diseñar, desarrollar e implementar un sistema de información personalizado para la gestión logística de supermercados, integrado en una plataforma web.

- **Automatización de la gestión de inventario y fechas de caducidad**: Se pretende implementar funcionalidades que permitan la automatización del seguimiento y gestión de inventario, incluyendo un sistema de seguimiento para las fechas de caducidad de los productos. Esto facilitará la toma de decisiones y reducirá los errores humanos asociados a la gestión manual de inventario.

- **Mejora de la usabilidad y experiencia del usuario**: Se busca mejorar la usabilidad del sistema, reduciendo la curva de aprendizaje y proporcionando una experiencia de usuario más fluida y agradable. Esto se logrará mediante el diseño de una interfaz intuitiva y amigable, así como la incorporación de funcionalidades que agilicen las tareas cotidianas de los usuarios.

- **Implementación de medidas de seguridad**: Se establecerán medidas de seguridad robustas para proteger la integridad y confidencialidad de los datos del sistema. Esto incluirá desde la implementación de protocolos de autenticación y autorización hasta la encriptación de datos sensibles.

- **Escalabilidad y adaptabilidad**: Se diseñará el sistema con la capacidad de escalar para satisfacer las necesidades futuras del negocio, así como para adaptarse a cambios en el entorno tecnológico. Se emplearán tecnologías y arquitecturas que permitan una fácil escalabilidad y mantenimiento del sistema a largo plazo.

1.4. Estructura de la memoria

Para facilitar la navegación del lector por el presente documento se incluye a continuación su estructura:

- **Resumen** (Véase Resumen): Este apartado proporciona una síntesis concisa de los objetivos, métodos y resultados del proyecto.

- **Introducción** (Véase Capitulo 1): Aquí se contextualiza el proyecto, describiendo el problema a abordar y los objetivos que se pretenden alcanzar. Se explica también la estructura general del presente documento.

- **Estado del Arte** (Véase Capítulo 2): En el que se habla de las distintas opciones que rivalizan con la solución propuesta; así como las diferencias entre ellas y los motivos de la elección.

- **Planificación** (Véase Capítulo 3): Donde se detallan las distintas fases del proyecto, desde la conceptualización inicial hasta la finalización del desarrollo. Contiene también un estudio presupuestario detallado.

- **Análisis** (Véase Capítulo 4): En esta sección se presentan los resultados del análisis realizado, incluyendo la especificación de requisitos y los modelos de caso de uso y comportamiento del sistema. Incluye también un estudio sobre la documentación del contexto que rodea la solución.

- **Diseño** (Véase Capítulo 5): Se describe la arquitectura y diseño del sistema, incluyendo el diseño de clases y de la interfaz de usuario.

- **Implementación** (Véase Capítulo 6): Aquí se detalla el proceso de implementación del sistema, incluyendo las herramientas utilizadas y los pasos del despliegue.

- **Pruebas y Validación** (Véase Capítulo 7): Sección en la que se comentan y muestran las diferentes pruebas que se han llevado a cabo.

- **Conclusiones** (Véase Capítulo 8): Se presentan las conclusiones obtenidas a partir del desarrollo del proyecto, destacando los logros alcanzados, el aprendizaje adquirido y las recomendaciones para futuros trabajos.

- **Bibliografía** (Véase Bibliografía): Se enumeran las fuentes consultadas durante la investigación y el desarrollo del proyecto.

- **Anexos** (Véase Anexos): Se adjuntan los materiales complementarios, como el código fuente, manuales de usuario u otros documentos relevantes.

Capítulo 2
Estado del Arte

En este capítulo se presenta un análisis del estado del arte de los sistemas de gestión de supermercados disponibles en el mercado, es decir las opciones que ya existen y que son competidoras a la solución aquí propuesta. Empezando por asentar la idea y el concepto general de un sistema de gestión (véase apartado 2.1), se continuará con un análisis que incluye una revisión de las características, ventajas y desventajas de los sistemas más relevantes, así como una comparación con el sistema desarrollado en este proyecto (véase apartado 2.2 y apartado 2.3). El objetivo es proporcionar un contexto claro sobre las soluciones actuales y justificar las decisiones de diseño y funcionalidad del sistema aquí propuesto.

2.1. Definición de los sistemas de gestión

Es de recibo comenzar con una descripción general de los sistemas de gestión, hablar de su importancia en el entorno empresarial y de su evolución. Para, posteriormente, aterrizar en los sistemas específicos para supermercados, detallando las características y funciones principales que estos poseen.

Podríamos decir que los sistemas de gestión son herramientas integrales que permiten a las organizaciones (ya sean directivos, gestores u simples responsables) administrar de manera eficiente sus recursos y

operaciones [1]. Estas soluciones tecnológicas facilitan la planificación, ejecución y control de los procesos empresariales, proporcionando datos en tiempo real y mejorando la toma de decisiones. Es cierto que a lo largo de los años, al mismo tiempo que lo hacía la tecnología los sistemas de gestión han evolucionado significativamente, integrando nuevas tecnologías y adaptándose a las necesidades cambiantes de las empresas. Haciendo un breve repaso por la historia, los primeros surgieron en las décadas de 1960 y 1970. Estos sistemas se centraban en la recopilación y procesamiento de datos, proporcionando informes básicos para la toma de decisiones. Más tarde, en las décadas de 1980 y 1990, la aparición de los *Sistemas de Planificación de Recursos Empresariales* marcó un importante hito [2]. Los famosos ERP integraron múltiples funciones empresariales en una única plataforma, abarcando áreas como finanzas, recursos humanos, producción y ventas. Con la entrada en el nuevo siglo, se evolucionó hacia sistemas basados en la nube [3] y a soluciones móviles lo que permitió a las empresas acceder a sus datos y gestionar sus operaciones desde cualquier lugar y en cualquier momento.

Habiendo ubicado tanto su definición como la evolución año tras año que han sufrido los sistemas de gestión en término general, es turno de focalizar sobre aquello que nos concierne de verdad en este proyecto, la gestión de una tienda minorista. Los sistemas de gestión de supermercados son una categoría específica dentro de los sistemas de gestión empresarial, diseñados para abordar las necesidades particulares del sector minorista [4]. Estos sistemas integran diversas funciones que son esenciales para la operación eficiente de un supermercado:

- ✓ **Gestión de Inventario**: Permite el seguimiento en tiempo real de los niveles de inventario y la reposición de productos. Ayuda a minimizar el exceso de stock y a evitar la escasez de productos.

- ✓ **Punto de Venta:** Incluye la gestión de las transacciones en las cajas registradoras, el procesamiento de pagos y la emisión de recibos.

- ✓ **Gestión de Proveedores**: Facilita la creación y seguimiento de órdenes de pedidos, la gestión de relaciones con proveedores y el control de costes.

- ✓ **Gestión de Precios y Promociones**: Permite establecer precios, aplicar descuentos y gestionar promociones de manera eficiente, asegurando una estrategia de precios competitiva y atractiva para los clientes.

- ✓ **Gestión de Clientes**: Incluye funcionalidades para la gestión de programas de fidelización, la recopilación de datos de clientes y la personalización de ofertas y promociones.

2.2. Sistemas actuales

En el mercado actual existen diversas opciones de sistemas de gestión de supermercados, cada una con características, puntos fuertes y desventajas particulares. Estos sistemas varían en funcionalidad, coste, facilidad de uso y capacidad de personalización, adaptándose a las diferentes necesidades y tamaños de los supermercados. A continuación se compararán las características generales de las principales soluciones disponibles, evaluando sus ventajas competitivas y las posibles limitaciones que presentan. Este análisis permitirá entender mejor las opciones existentes y justificar la elección del sistema desarrollado en este proyecto.

SAP Retail [5] es una solución de software empresarial diseñada para gestionar las operaciones de venta al por menor. Ofrece funcionalidades para la planificación de mercancías, la gestión de inventarios, la optimización de precios y promociones, y la integración con proveedores y clientes. Son muchas las ventajas que ofrece, como su amplia funcionalidad ya que cubre todas las áreas de gestión posibles de un supermercado, la integración con otros módulos de SAP, como son finanzas y recursos humanos, además de herramientas muy avanzadas de análisis y reportes. Nada menos que esperar de la multinacional alemana líder en software europeo y top mundial. Factor clave; ya que su mayor virtud también es su gran defecto ya que esta solución conlleva, un coste elevado de implementación y mantenimiento, requiere un alto grado de personalización y configuración inicial y la curva de aprendizaje es pronunciada debido a la complejidad del sistema.

Microsoft Dynamics 365 for Retail [6] es una solución integrada para la gestión de operaciones minoristas, que abarca desde la gestión de tiendas físicas y en línea hasta la administración de inventarios y ventas. Como gran ventaja cuenta con la integración con el ecosistema de Microsoft, facilitando la interoperabilidad con otras herramientas del paquete Office 365, que es una plataforma basada en la nube, lo que permite actualizaciones y escalabilidad bastante sencilla, así como interfaces de usuario intuitivas y modernas. Por contraparte esta requiere suscripción continua, lo que puede resultar costoso a largo plazo, necesita personalización adicional para satisfacer necesidades específicas de cada negocio y también depende de una conexión a internet fiable para el uso pleno de sus funcionalidades.

Oracle Retail [7] ofrece una suite completa de soluciones para la gestión de la cadena de suministro, la planificación de la mercancía, la optimización de precios, y la gestión de las tiendas. Su principal ventaja es que es una solución robusta y escalable apta para grandes cadenas de supermercado, cuenta con funcionalidades avanzadas de análisis de datos e inteligencia de negocio además de que permite la integración con otras soluciones de Oracle, como la base de datos y el ERP. Similar a las

opciones anteriores su principal desventaja es el coste de implementación y mantenimiento, la complejidad en la implementación y personalización del sistema y también la necesidad de formación específica para el personal que lo vaya a utilizar.

2.3. Comparativa

El análisis anterior de las diferentes opciones actuales revela que existen numerosas soluciones muy avanzadas para la gestión de supermercados, pero estas están orientadas a grandes cadenas, tienen costes elevados y la curva de aprendizaje para poder utilizarlos y exprimirlos al máximo es algo elevada. Escogiendo una de estas soluciones estaríamos ejemplificando perfectamente el popular dicho “matar moscas a cañonazos” ya que podríamos dar solución a nuestro problema pero ¿a qué coste? El sistema de gestión para supermercados desarrollado en este proyecto ofrece una alternativa viable para supermercados medianos y pequeños, proporcionando una solución equilibrada entre funcionalidad, coste y facilidad de uso. Este se diferencia de las soluciones mencionadas en varios aspectos clave:

- **Coste:** A diferencia de las soluciones comerciales de SAP, Microsoft y Oracle, este sistema está diseñado para ser una opción más asequible, ideal para supermercados medianos y pequeños que no pueden asumir los altos costes de licencias y mantenimiento de las grandes soluciones empresariales.

- **Simplicidad y Usabilidad:** El sistema propuesto se centra en ofrecer una interfaz de usuario muy intuitiva y fácil de usar, reduciendo la curva de aprendizaje y facilitando la adopción por parte del personal del supermercado.

- **Personalización y Flexibilidad:** Las soluciones comerciales ofrecen una gran cantidad de funcionalidades y estas resultan excesivas y complicadas para negocios de menor tamaño, que no quieren complicarse y solo necesitan dar respuesta a ciertas tareas. El sistema ofrece las funciones necesarias para la gestión de un supermercado, con la posibilidad de añadir módulos adicionales según las necesidades futuras del cliente.

- **Implementación y Mantenimiento:** El sistema desarrollado está diseñado para una implementación más rápida y sencilla, con menores requisitos de personalización inicial y mantenimiento continuo, en comparación con las soluciones más complejas de SAP, Microsoft y Oracle.

Esta comparación hace decantarse al cliente por una solución a

medida, más sencilla que las opciones del mercado y que no complique la tarea de adaptar tecnológicamente su negocio; lo que lo convierte a la solución propuesta en una opción atractiva, tanto para este negocio como para otros que buscan modernizar sus operaciones diarias sin incurrir en grandes gastos y sin necesidad de formación extra para el personal.

Capítulo 3
Planificación

En el presente capítulo, se adentra en la planificación detallada del proyecto, delineando las fases clave y los recursos necesarios para llevar a cabo la implementación de la solución propuesta. Comenzando con la fase inicial, se establecen los cimientos esenciales para el desarrollo futuro del proyecto, definiendo objetivos generales y visualizando el producto final pasando por distintas fases como son la de análisis, la de desarrollo y una última de prueba y correcciones (véase apartado 3.1).

También se detalla la planificación temporal del proyecto a través de un diagrama de Gantt, que proporciona una representación visual de las tareas y su cronograma de ejecución. Este diagrama facilita la gestión del tiempo y el seguimiento del progreso del proyecto (véase apartado 3.2). Como no podía ser de otra forma en este capítulo se presenta un desglose detallado de los costes, junto con un cálculo del presupuesto total necesario para llevar a cabo el proyecto (véase apartado 3.3).

Podríamos decir que la planificación detallada de las fases del proyecto permite una gestión eficiente de los recursos y un seguimiento riguroso del progreso. A través de una combinación de análisis y acción, se busca garantizar que el proyecto avance de manera fluida y efectiva hacia la consecución de los objetivos.

3.1. Fases

En esta sección, se detallan las distintas fases que conforman el

proyecto. Estas fases están pensadas para garantizar un enfoque estructurado y eficiente en el desarrollo del proyecto, permitiendo una gestión efectiva de los recursos y una entrega exitosa del producto final.

Durante estas fases, se abordan aspectos clave como la definición de objetivos, la comprensión de los requisitos del sistema, el diseño arquitectónico, la implementación del software y las pruebas exhaustivas para garantizar la calidad y funcionalidad del sistema.

3.1.1. Fase inicial

En esta primera fase del proyecto se establecen las bases fundamentales para el futuro desarrollo. Durante esta etapa, se definen los objetivos generales del proyecto y se delinean los pasos a seguir de la futura solución a implementar. Este es un momento crucial en el que se plantean las ideas iniciales y se visualiza el producto final.

Uno de los aspectos clave de esta fase es la conceptualización del sistema propuesto. Pensando un esquema detallado del sistema, que sirve como guía visual para el diseño y desarrollo posterior. Este esquema muestra cómo los usuarios interactúan con la interfaz para realizar peticiones a través de una API, la cual se comunica con el servidor a través de los *endpoints* para acceder a la base de datos subyacente.

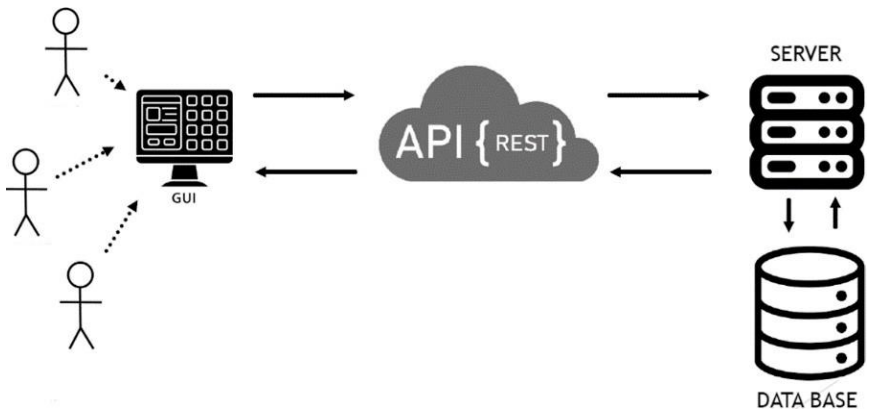

Ilustración 1: Idea propuesta

Además, durante esta fase inicial, se llevan a cabo varias reuniones importantes con el cliente para discutir y perfilar la idea del sistema. Estas reuniones son cruciales para comprender a fondo los requisitos y expectativas del cliente, así como para definir el alcance real del proyecto. El diálogo continuo con el cliente permite una alineación efectiva de las expectativas y asegura que el proyecto se orientara hacia la consecución de los objetivos establecidos.

3.1.2. Fase de análisis

La fase de análisis se centra en la comprensión detallada de los requisitos y necesidades del sistema. Durante esta etapa, se recopilan y documentan los requisitos funcionales y no funcionales del sistema; también los requisitos de información, así como las restricciones y limitaciones que deberían tenerse en cuenta en el diseño y desarrollo.

En las entrevistas con el cliente se llevan a cabo sesiones de trabajo para identificar y priorizar los requisitos del sistema. Esto permite una comprensión profunda de los procesos comerciales y logísticos del propio supermercado, así como de las áreas donde se necesitaba mejorar la eficiencia y la automatización. Pero esta fase no solo se limita a recopilar los requisitos del sistema, sino que también implica una comprensión profunda del entorno en el que operará la solución propuesta. Para ello, se llevará a cabo un análisis exhaustivo del usuario objetivo, identificando sus necesidades, habilidades y expectativas en relación con el sistema. Esto garantizará que el diseño y desarrollo del sistema estén alineados con las expectativas del usuario final, lo que a su vez mejorará la usabilidad y la satisfacción del cliente.

También, se realizará una documentación exhaustiva del entorno, centrándose en el contexto del mercado minorista de supermercados. Este proceso implica recopilar información relevante sobre los etiquetados que se utilizan, como se codifican y sus desafíos operativos. Es crucial comprender el funcionamiento interno de dicho supermercado y los procesos logísticos existentes para identificar áreas de mejora y oportunidades para la optimización. Una parte integral de esta fase de análisis será el estudio del sistema actual utilizado en el supermercados. Se examinarán las limitaciones y deficiencias del sistema heredado, así como los desafíos asociados con su uso en la gestión diaria. Este análisis proporcionará una visión completa de las áreas problemáticas que el nuevo sistema debe abordar y mejorar.

Además del análisis del usuario objetivo y del entorno del mercado, se llevará a cabo un estudio presupuestario completo. Este estudio detallará los costes asociados con el desarrollo, implementación y mantenimiento del nuevo sistema. Proporcionará una estimación precisa de los recursos financieros necesarios para completar el proyecto y permitirá al cliente evaluar el retorno esperado de la inversión.

3.1.3. Fase de desarrollo

La siguiente fase es la de desarrollo, que representa el corazón del proyecto, donde se lleva a cabo la construcción del sistema de información según los requisitos y especificaciones establecidos en las fases anteriores. Esta fase se divide en dos grandes etapas clave que guiarán el proceso de

construcción, son el diseño y la implementación. Empezando claro está por el diseño, en el primer paso se establecerá la arquitectura general del sistema, definiendo los componentes principales, los módulos funcionales y las tecnologías a emplear. Se diseñará una arquitectura robusta y escalable que cumpla con los objetivos del proyecto y permita la integración fluida de los distintos componentes. Esta fase sentará las bases para el desarrollo eficiente y ordenado del sistema.

Una vez definida la arquitectura del sistema, se procederá al diseño detallado de la base de datos. Se modelarán las entidades, relaciones y atributos del sistema, asegurando la integridad y consistencia de los datos. Se establecerán las restricciones de integridad y las reglas de negocio que guiarán la estructura de la base de datos, garantizando su eficiencia y fiabilidad.

La última etapa de la fase de diseño estará centrada en la creación de las interfaces de usuario. Se crearán unas interfaces gráficas y funcionales que permitirán a los usuarios interactuar con el sistema de manera intuitiva y eficiente. Se definirán los elementos necesarios que deben tener, el flujo de navegación y la disposición de los elementos en pantalla, asegurando una experiencia de usuario coherente y satisfactoria. Finalmente, la fase de desarrollo también abarca la implementación del programa, donde se lleva a cabo la codificación del sistema de acuerdo con los diseños y especificaciones previamente establecidos. Se desarrollan los componentes del sistema, se integran los distintos módulos y se realizan las adaptaciones necesarias para asegurar el funcionamiento correcto del sistema en su entorno de producción.

Cabe destacar que en este proyecto se apuesta por un enfoque flexible y adaptable principalmente en este proceso de desarrollo en el que en ocasiones hay que volver atrás y rehacer o cambiar algo. En lugar de seguir una enfoque *waterfall* [8] donde el proceso es lineal en sus fases y la vuelta atrás para cambiar algo es costosa y lenta como se ve en la Ilustración 2; se escoge un enfoque *agile* [9][10] donde a partir de los requerimientos se entra en un ciclo donde es rápido pasar del diseño al despliegue y volver a empezar. No se trata de la metodología agile estricta ya que requiere el aspecto colaborativo y yo en este caso trabajo solo; pero sí extraer los principios de priorizar un software en funcionamiento frente a documentación exhaustiva, participación activa del cliente y capacidad de respuesta a cambios e imprevistos.

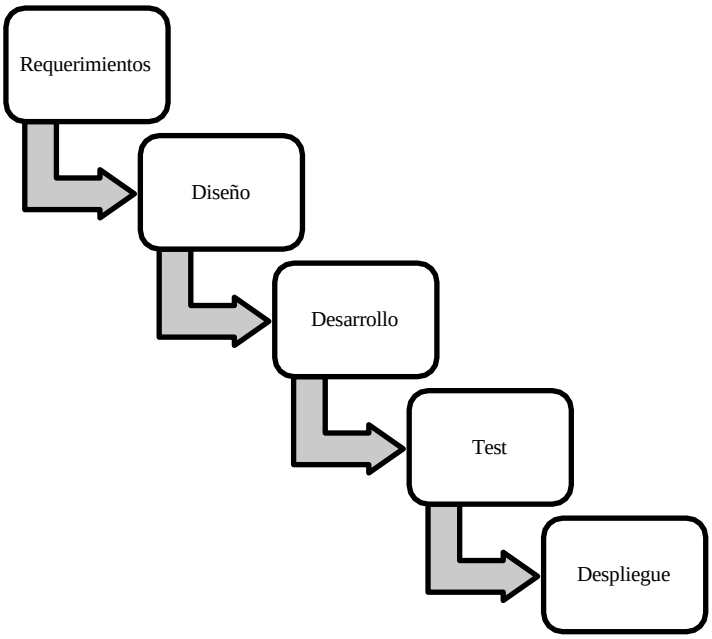

Ilustración 2: Esquema enfoque de diseño Waterfall

Valoro la entrega temprana de funcionalidades y la retroalimentación continua del cliente para marcar el camino de sus preferencias. Esta aproximación me permite ajustar y mejorar el producto de forma iterativa, respondiendo eficientemente a los cambios a medida que surgen. Incido en que no sigo una metodología *Agile* formal ya que no cuento con roles definidos y las reuniones específicas del equipo, más bien mi enfoque se orienta hacia la creación de valor de forma iterativa en el proceso y la capacidad de respuesta ante los cambios, lo que me permite trabajar de manera eficaz y centrada en el cliente.

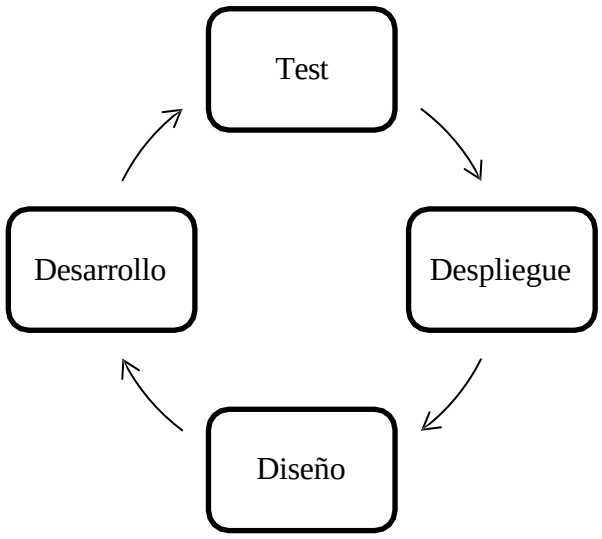

Ilustración 3: Esquema enfoque de diseño Agile

3.1.4. Fase de prueba y correcciones

Otra etapa distinta aunque algo peculiar es la de probar y corregir aquello que se ha desarrollado. Peculiar porque no se trata de una fase que empieza cuando acaba la anterior. A lo largo del proceso de desarrollo, se llevan a cabo pruebas y correcciones de forma iterativa cada vez que se crea una funcionalidad significativa. Estas pruebas no se limitan a una fase única al final del desarrollo, sino que se realizan de manera continua para garantizar la funcionalidad, fiabilidad y rendimiento del sistema en cada momento. Una vez que se implementa una nueva funcionalidad, se somete a rigurosas pruebas funcionales para validar que cumple con los requisitos especificados. Se realizan pruebas exhaustivas de extremo a extremo, simulando situaciones reales de uso para verificar que todas las funcionalidades operan correctamente. Cualquier fallo o incidencia identificado se detecta y se corrige de inmediato.

Además de las pruebas funcionales, también se realizan pruebas de rendimiento y seguridad. Se evalúa la capacidad del sistema para manejar cargas de trabajo elevadas y proteger la integridad de los datos. Se simulan escenarios de carga máxima y se optimiza el rendimiento del sistema para garantizar una experiencia fluida para el usuario.

Durante todo este proceso, se abordan los errores y deficiencias identificados de manera continua, realizando las correcciones necesarias para garantizar el correcto funcionamiento del sistema. Se lleva a cabo un proceso iterativo de identificación, corrección y verificación de errores en cada fase del desarrollo. Una vez que se han completado todas las pruebas y correcciones, se presenta el sistema al cliente para su validación final. El cliente realiza pruebas de aceptación para verificar que el sistema cumple con sus expectativas y requisitos. Se recopilan y documentan los comentarios y sugerencias del cliente para futuras mejoras, asegurando una entrega final del sistema que satisfaga plenamente sus necesidades.

3.2. Elaboración de la memoria

Para proporcionar una visión más detallada de la planificación de este proyecto, se ha elaborado un diagrama de Gantt que muestra la estimación del tiempo requerido para cada tarea y la secuencia en la que se llevarán a cabo. Este diagrama presenta una representación visual de las actividades planificadas y su duración prevista, lo que permite una mejor estimación del proyecto completo para el cliente. El diagrama de Gantt incluye todas las fases del proyecto, desde la conceptualización inicial hasta la entrega final del sistema. Cada tarea se ha desglosado en actividades específicas y se ha asignado una duración estimada. Además, se han identificado las dependencias entre las tareas para garantizar un flujo de

trabajo coherente y eficiente.

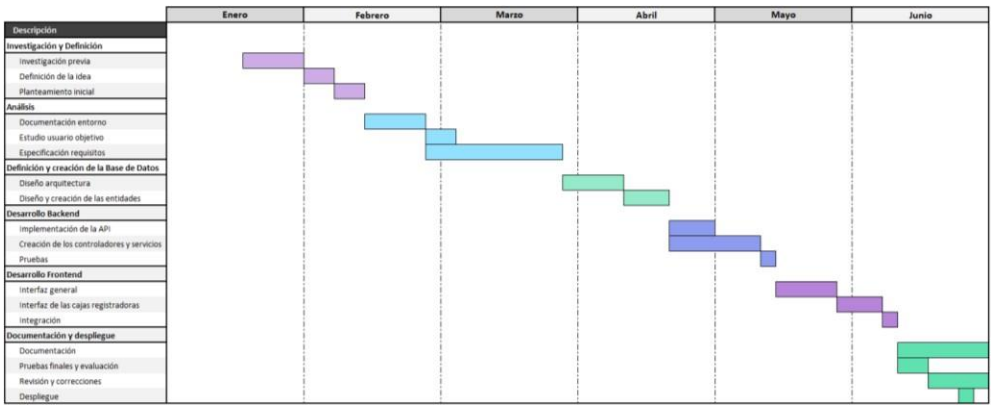

Ilustración 4: Diagrama de Gantt (Previsión)

Es importante destacar que el diagrama de Gantt refleja una estimación inicial y que la planificación real puede variar a medida que avanza el proyecto y se enfrenta a diferentes desafíos y circunstancias. Una vez finalizado el proyecto, se realizará una enfrentamiento del diagrama de Gantt de la previsión y del real para comparar la planificación inicial con el tiempo real empleado en cada tarea, lo que proporcionará información valiosa para futuros proyectos y mejoras en la gestión del tiempo.

3.3. Estudio presupuestario

En esta sección se detallan las distintas partidas que generan gasto y que son necesarias para poder acometer el proyecto. Resultando el presupuesto de la Tabla 6.

3.3.1. Licencias de uso

Con el fin de ofrecer un presupuesto fiel y realista se deben incluir también aquellas licencias de uso de programas o recursos utilizados durante el desarrollo del proyecto. La primera de todas sería la del editor de textos Word en el que se ha redactado la memoria. Para disfrutar este programa hace falta adquirir el paquete Microsoft Office completo y como la duración del proyecto son 4 meses se opta por el plan mensual de 7,00 €/mes [11].

El entorno de programación utilizado es IntelliJ IDEA Ultimate del paquete de desarrollo JetBrains IDE's el cual se puede disfrutar por un plan mensual de 16,90€/mes más IVA, (20,45€/mes) [12].

Concepto	Coste mensual	Coste total (4 meses)
Paquete Microsoft Office	7,00€	28,00€
Entorno IntelliJ IDEA Ultimate	20,45€	81,80€
TOTAL	**27,45€**	**109,08€**

Tabla 1: Costes asociados a las licencias de uso

3.3.2. Recursos materiales

En este apartado debemos incluir el equipo informático usado para desarrollar el proyecto, detallado en Anexo C Recursos Informáticos. Para este tipo de equipos la vida útil se estima en 4 años de media por lo que para calcular la amortización anual deberíamos aplicar la siguiente formula:

$$Amortización\ anual\ = \frac{Coste\ inicial}{Vida\ útil}$$

Por tanto, considerando que el ordenador portátil tuvo un coste inicial de 1.164,54€ tenemos que la amortización anual es de 291,14€. Por último, para calcular la amortización relativa al proyecto debemos ajustar está a la duración del mismo, 4 meses calculándose como:

$$Amortización\ del\ proyecto = Amortiz.anual \times \frac{Meses\ del\ proyecto}{12\ meses}$$

Lo que nos deja un coste total de amortización técnica del proyecto de 97,05€.

Concepto	Amortiz. Anual	Amortiz. Proyecto (4 meses)
Amortización técnica equipos informáticos	291,14€	97,05€
TOTAL	**291,14€**	**97,05€**

Tabla 2: Costes asociados a la amortización técnica

3.3.3. Costes de personal

Para los recursos humanos he tenido en cuenta un único trabajador, un ingeniero informático. Tomando como partida los datos propios de la Seguridad Social para este año 2024 [13], la base mínima sería de

1.459,20€/mes y la base máxima sería de 4.720,50€/mes.

Tipo de cotización	Empresa	Trabajador
Común	23,60%	4,70%

Tabla 3: Tipos de cotización

Concepto	Coste mensual	Coste total (4 meses)
Base mínima de cotización	1.459,20€	5.836,80€
Cotización empresa (base mínima)	344,37€	1.377,48€
Cotización trabajador (base mínima)	68,58€	274,32€
TOTAL (Base mínima)	**1.872.15€**	**7.488,60€**
Base máxima de cotización	4.720,50€	18.882,00€
Cotización empresa (base máxima)	1.114,04€	4.456,16€
Cotización trabajador (base máxima)	221,86€	887,44€
TOTAL (Base máxima)	**6.056,40€**	**24.225,60€**

Tabla 4: Cálculo del coste mínimo y máximo asociado al personal

Haciendo una media entre el mínimo y el máximo nos quedaría la siguiente base media, lo que nos deja un total (incluyendo las cotizaciones) de 15.857,11€ para todo el proyecto.

Concepto	Coste mensual	Coste total (4 meses)
Base media de cotización	3.089,85€	12.359,40€
Cotización empresa (base media)	729,20€	2.916,82€
Cotización trabajador (base media)	145,22€	580,89€
TOTAL (Base media)	**3.964,28€**	**15.857,11€**

Tabla 5: Cálculo del coste medio asociado al personal

3.3.4. Otros costes

En este apartado se incluyen gastos como por ejemplo el material de oficina, la conexión a internet, la electricidad o los de desplazamiento para las reuniones entre otros. Lo más común es que este tipo de gastos se estimen a partir de un porcentaje del gasto principal, en mi caso el de personal. Se ha estimado que estos gastos representan un 10% del gasto antes comentado por lo que quedaría como:

$$15.857{,}11€ \times 0{,}10 = 1.585{,}71€$$

Este coste adicional será de 1.585,71€ que se sumarán al presupuesto total.

3.3.5. Presupuesto total

En la siguiente tabla se muestra y detalla el coste total del proyecto.

Concepto	**Coste mensual**	**Coste total (4 meses)**
Licencias de uso	27,45€	109,08€
Amortización equipos informáticos	291,14€	97,05€
Salario base	3.089,85€	12.359,40€
Cotizaciones	874,42€	3.497,68€
Otros costes	396,73€	1.585,71€
TOTAL	**4.679,59€**	**17.648,92€**

Tabla 6: Cálculo del presupuesto total

Capítulo 4
Análisis

Este capítulo se centra en el análisis exhaustivo que fundamenta las bases del proyecto, abordando meticulosamente cada aspecto clave para asegurar que la solución propuesta esté perfectamente alineada con las necesidades del usuario objetivo y el entorno en el que se desplegará. Iniciamos con un detallado análisis del usuario objetivo, identificando sus necesidades, preferencias y comportamientos, lo cual es esencial para definir el enfoque y los requisitos del proyecto (véase apartado 4.1).

Acto seguido se incluye la documentación del entorno, necesaria para entender los aspectos que rodean al funcionamiento de un supermercado y también el sistema actual que se está utilizando. Este análisis asegura que el proyecto se desarrolle en consonancia con las condiciones externas y las limitaciones existentes, permitiendo anticipar posibles desafíos (véase apartado 4.2). Se aborda también la especificación de requisitos tanto los funcionales como los no funcionales, lo que establece una guía definitiva para las fases de diseño y desarrollo posteriores (véase apartado 4.3).

Finalmente, el capítulo también incluye la elaboración de escenarios de uso, que describen las interacciones entre los usuarios y el sistema (véase apartado 4.4) y el modelo de datos, esencial para estructurar la información que el sistema manejará (véase apartado 4.5).

4.1. Análisis del usuario objetivo

Este primer paso dentro de la fase de análisis es fundamental en el diseño y desarrollo de cualquier sistema de información. En el caso de este proyecto, el usuario objetivo principal son los trabajadores de supermercado, quienes utilizarán el sistema en su día a día para realizar diversas tareas relacionadas con la gestión logística y de inventario.

Los usuarios de este sistema no necesariamente poseen una formación técnica o especializada en informática. Por lo tanto, es crucial diseñar una interfaz de usuario intuitiva y fácil de usar que permita a estos usuarios realizar sus tareas de manera eficiente, sin la necesidad de una capacitación extensa. El fin último es que los usuarios utilicen el sistema para realizar acciones como el registro de productos en el inventario, la gestión de pedidos y entregas, la búsqueda y consulta de información sobre productos, entre otras actividades relacionadas con la operación diaria de un supermercado.

Es importante tener en cuenta que estos usuarios pueden tener diferentes niveles de experiencia y habilidades con respecto al uso de tecnología. Por lo tanto, el diseño del sistema debe ser inclusivo y adaptable para satisfacer las necesidades de todos los usuarios, independientemente de su nivel de habilidad técnica.

Además, es crucial recopilar comentarios y retroalimentación de los usuarios durante el proceso de desarrollo para identificar posibles áreas de mejora y garantizar que el sistema satisfaga realmente sus necesidades y expectativas.

4.2. Documentación del entorno

En este apartado se aborda la exhaustiva investigación necesaria para comprender el entorno que rodea al proyecto. Comprender cómo funciona el “mundillo” de los supermercados, el sistema del etiquetado y la forma de trabajar que se sigue en un supermercado como este, viendo cómo es el programa actual que utilizan. Para comenzar hablaremos del básico y común “código de barras” [14] que todo producto lleva. El estándar EAN-13, cuyas siglas EAN provienen de *European Article Numbering Association*, antiguo nombre de la organización encargada de gestionar estos estándares actualmente conocida como GS1 [15] y el 13 es referente al número de dígitos que incluye. Este estándar es el más utilizado, está pensado para unidades de consumo las cuales están destinadas a un cierto punto de venta.

Quiero destacar que, en este proyecto, no nos adentraremos en los

procesos de creación de estos códigos, ni en los pasos para dar de alta un nuevo producto. Para ello hay que seguir un procedimiento con organismos como AECOC (Asociación Española de Codificación Comercial) dándose de alta como empresa y registrando los productos que se desean vender. Dado que los supermercados de nuestro caso "Carrefour Express" [16] son franquicias que forman parte de una matriz más grande, Carrefour, los códigos de los productos ya están asignados y gestionados centralmente y nos son proporcionados. Por lo tanto, nuestra atención se centrará en la implementación de un sistema de gestión que opere con la información que estos códigos y estándares incluyen.

Ilustración 5: Ejemplo EAN-13

Este estándar está compuesto por dos componentes, la primera de ellas es el GTIN, la secuencia de números que identifica de forma inequívoca un producto y la segunda el propio EAN, las barras, que son una representación gráfica de la secuencia numérica inferior. Los códigos GTIN *(Global Trade Identification Number)* tienen una estructura específica y un propósito concreto. Estos códigos identifican artículos comerciales y en concreto nuestro caso, el GTIN-13 [17], está pensado para unidades comerciales destinadas al punto de venta detallista, como por ejemplo las que se venden en un supermercado directamente al consumidor final, nuestro caso.

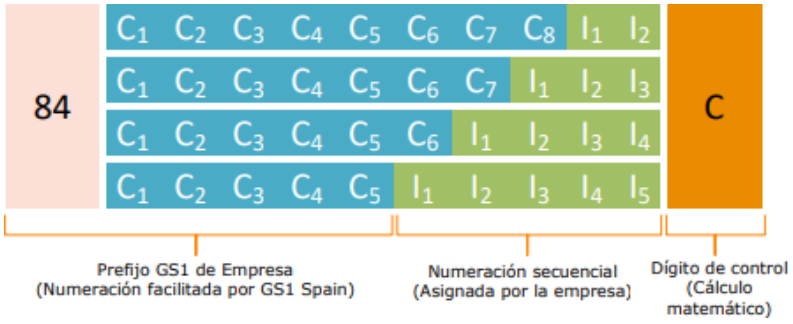

Ilustración 6: Estructura GTIN-13

Su estructura es la siguiente; los dos primeros dígitos indican el país de procedencia del artículo en cuestión [18] asignado por GS1, los siguientes cinco u ocho dígitos indican la empresa que lo produce, los próximos dos o cinco dígitos se asignan por la propia empresa y representan el código del producto concreto y por último el dígito final es el de control y es el resultado de aplicar un cálculo algorítmico a los doce anteriores.

Continuando con la documentación del etiquetado, habiendo hablado de los productos ahora es turno de hablar de los palés. Estos utilizan otro estándar, similar al anterior en forma, pero distinto en cuanto a la información que incluyen. El estándar que se utiliza es el GS1-128 (o también denominado EAN-128) [19].

Ilustración 7: Estructura GS1-128

Ilustración 8: Ejemplo código GS1-128

Este tiene una estructura concreta donde las primeras tres barras indican el Juego de Caracteres que sería algo así como indicar que tipo de información se va a representar (números, letras mayúsculas o letras minúsculas), las siguientes tres barras son el carácter FNC1 [20] que se incluye para la correcta estructura del código y también se usa como separador. Esta primera parte es protocolaria, no nos centraremos mucho en ella. En la parte siguiente iría la información representada siempre con una estructura de IA más datos. Los identificadores de aplicación (IA) [21] son prefijos numéricos que indican el formato y el tipo de los datos que se incluyen. Estos van entre paréntesis, aunque en el código de barras los paréntesis no se representan. Se pueden incluir varios, es más; es lo lógico para incluir más información siguiendo siempre ciertas indicaciones como por ejemplo que una línea de código de barras GS1-128 puede contener como máximo 48 datos (incluyendo el IA y el FNC1) por lo que en ocasiones será necesario estructurar la información en varios códigos de barras que componen una misma etiqueta. Algunos de los Identificadores de Aplicación más comunes se muestran en la Ilustración 9.

IA	Descripción	Formato	Necesita FNC1 (separador
00	Código Seriado de la Unidad de Envío (SSCC)	n2 + n18	No
01	Código del artículo/agrupación	n2 + n14	No
02	Código del artículo/agrupación contenido	n2 + n14	No
37	Cantidades (acompañando al 02)	n2 + n..8	Sí
10	Número de lote	n2 + an...20	Sí
11	Fecha de fabricación	n2 + n6	No
13	Fecha de envasado	n2 + n6	No
15	Fecha de consumir preferentemente	n2 + n6	No
17	Fecha de caducidad	n2 + n6	No
310X	Peso neto en kilos	n4 + n6	No
330X	Peso bruto en kilos	n4 + n6	No

Ilustración 9: Identificadores de Aplicación más comunes

Algunas particularidades de estos son por ejemplo que el (00) que es el propio código que identifica el envío que estamos etiquetando siempre irá en la última línea del código de barras si hay varias, es decir en la de abajo y en la primera posición de esta. El (01) y el (02) irían siempre en la primera línea del código de barras y al principio. En nuestro caso también nos interesaría fijarnos en códigos de la familia de las fechas como la de fabricación, caducidad o consumo preferente ya que en el sistema actual estos no se incluyen y esto genera el problema que queremos solventar.

Ilustración 10: Ejemplo de uso Identificador de Aplicación

En la ilustración 10 se observa un ejemplo de cómo se usan estos Identificadores de Aplicación junto con sus datos. Podemos ver cómo esta etiqueta sería la propia de un palé que incluye artículos cuyo GTIN es 0845678956807, cuya fecha de consumo preferente es el 8 de abril de 2023 y el número de lote es el 89B23.

Es ahora cuando, ya sabemos que información se incluye en las etiquetas y cómo se codifica dicha información cuando podemos trabajar con nuestro sistema para que sea capaz de utilizar esta información.

Dicho esto, en esta documentación del entorno en el que nos vamos a desarrollar, para entender cómo trabajan en el supermercado que tenemos cómo cliente es necesario también conocer el sistema que se nos propone

sustituir. Partimos de la base que las tareas de gestión del inventario se realizan con este sistema y puesto que el único que sabe manejarlo es el gerente, es el único que puede realizar estas tareas. La ilustración 11 muestra la primera toma de contacto con el programa, la pantalla de inicio de sesión.

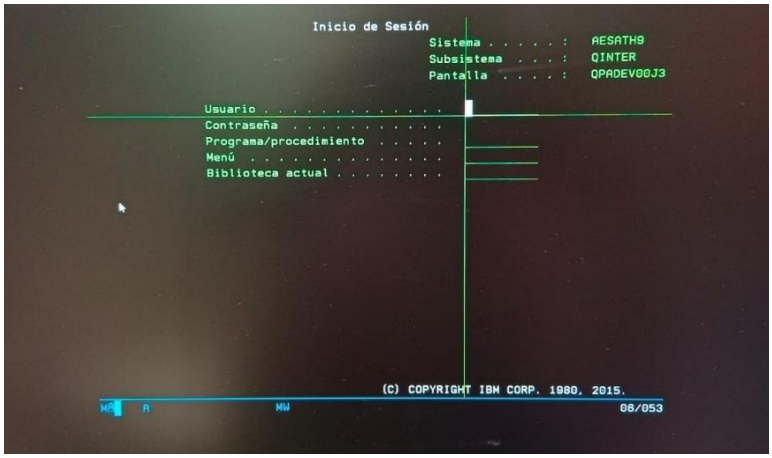

Ilustración 11: Captura de pantalla del Sistema Actual (Inicio de Sesión)

Una vez que el gerente de la tienda introduce las credenciales presionando una combinación de teclas especifica se accede al sistema. Esta interfaz es simple y funcional pero rudimentaria al mismo tiempo, más adelante veremos cómo se va complicando la cosa. Una vez se accede al sistema se nos da la posibilidad de "viajar" por el menú que se muestra en la Ilustración 12, y digo "viajar" porque la manera de entrar en alguna opción concreta es introducir su número y seguidamente una tecla de función específica o por ejemplo para cerrar nuestra sesión deberíamos introducir *'90'* más la tecla de función ; todo esto hace muy incómodo su uso y hace que sea poco intuitivo.

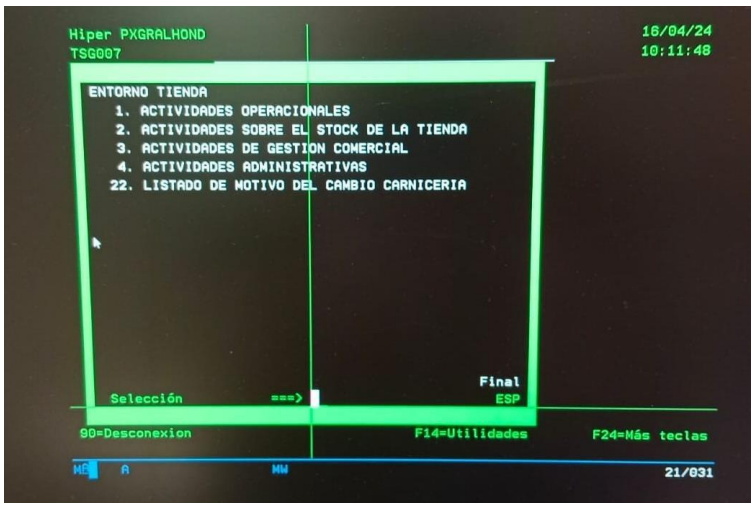

Ilustración 12: Captura del Sistema Actual (Menú de opciones)

Una de las funciones principales de este sistema es la gestión sobre un producto concreto, como por ejemplo se muestra en la Ilustración 13, cabe destacar que para llegar hasta este *display* de información el usuario debe navegar por varios menús y posteriormente reconocer el código GTIN del producto que quiere consultar. Volvemos a lo mismo, la experiencia necesaria para realizar una tarea tan simple como consultar los precios de un producto es muy alta y esto es consecuencia de la complejidad y obsolescencia del sistema utilizado. La Ilustración 14 sigue el mismo estilo que las anteriores en este caso se consultan los datos del stock de un producto en un mes concreto. La forma de gestionar la cantidad disponible de cada producto es almacenar una variable con el número de unidades de las que se disponen e ir restando a esta las unidades vendidas o las mermas así como sumando las unidades que se piden al proveedor.

La ausencia total de una interfaz gráfica intuitiva y actualizada a las posibilidades actuales y de imágenes para reconocer los productos, la cantidad de combinaciones que se necesitan para viajar por las opciones de los menús hacen que el trabajo diario con este sistema sea una ardua tarea.

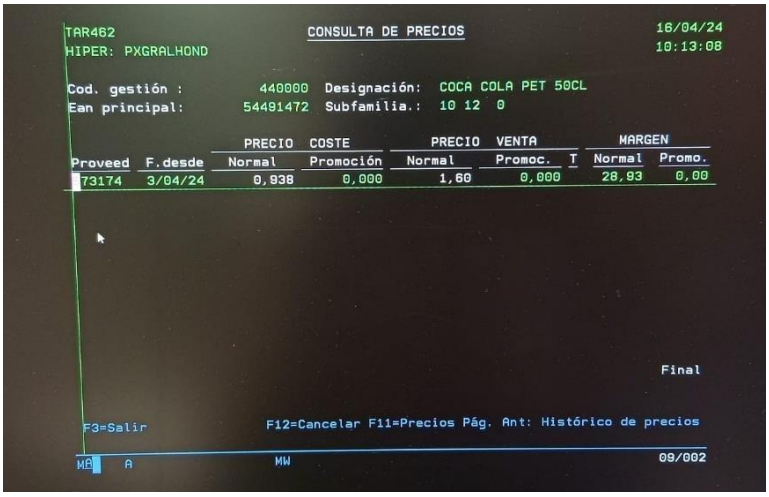

Ilustración 13: Captura del Sistema Actual (Consulta de precios)

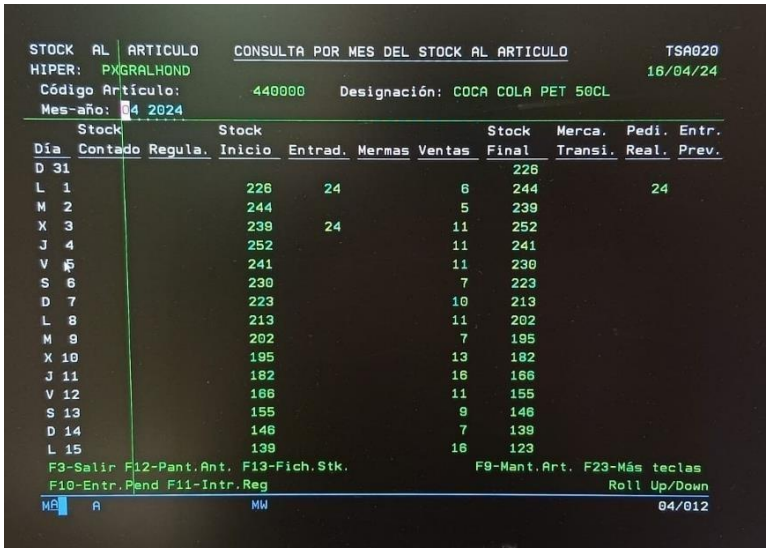

Ilustración 14: Captura del Sistema Actual (Consulta por mes del stock)

El fin de mostrar estas capturas de pantalla es reflejar fielmente la situación actual en la que se encuentra el cliente, cómo trabaja con lo que tiene y aquello que le permite hacer el sistema del que dispone así como aquello que no le permite hacer. Esto en relación a la manera de trabajar en cuanto a la gestión del inventario pero también sería justo hablar de la manera de proceder en las tareas de logística. Entendiendo como tareas de logística las de recibir un camión con un pedido al proveedor y almacenar los productos o también la tarea de realizar una entrega de una compra a un domicilio. La primera de ellas primero necesita que el gerente desde el sistema haya realizado un pedido al proveedor, de esta operación se obtienen datos como el número de pedido, numero de envío o el SSCC *(Serial Shipping Container Code)* haciendo especial hincapié en este último ya que este es el número utilizado pada identificar una unidad logística y está escrito en el estándar GS-128 del que hablamos anteriormente.

Cuando el camión llega un trabajador del supermercado chequea que los datos del pedido que él tiene coinciden con los del albarán del repartidor y si es así lee la etiqueta del palé, una como la que se muestra en la Ilustración 15. Al leer la etiqueta con el lector de códigos de barras el sistema carga en el stock del almacén de la tienda los productos que contiene el pedido, este proceso tarda de 5 a 10 minutos hasta que se actualiza con la nueva información. Una vez hecho esto solo queda guardar los productos en el almacén y posteriormente colocarlos en los expositores de la tienda.

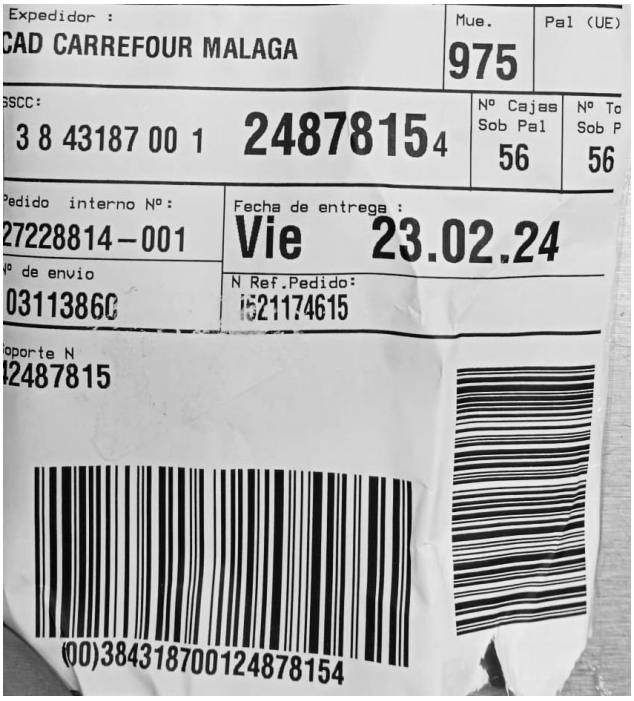

Ilustración 15: Etiqueta logística de un palé de un pedido al proveedor

El proceso de entrega a domicilio de las ventas que necesiten este servicio se realiza de la siguiente forma; cuando un cliente hace su compra indica en la caja que quiere que se lo lleven a casa, da sus datos y abona el servicio (*el envío es gratuito si la compra es superior a 40€*) entonces dependiendo de la hora en la que se realiza la venta; se realiza la entrega en el momento o el supermercado guarda la compra en refrigeradores y se realiza la entrega por la tarde. Es un proceso nada informatizado, todo manual; llevado a cabo por un trabajador de la tienda.

4.3. Especificación de requisitos

Para una mejor organización y comprensión de los requisitos del sistema, se han dividido en subsistemas internos que corresponden a las principales funcionalidades del sistema. Cada subsistema está diseñado para abordar un conjunto específico de tareas relacionadas con la gestión integral del supermercado. En este sentido, el subsistema de Almacén se centra en todas las operaciones relacionadas con el inventario de productos y el control de estos, incluyendo la gestión de estocaje y descuentos. Por otro lado, el subsistema de Pedidos se enfoca en las interacciones con los proveedores, como la creación, modificación y seguimiento de pedidos a la empresa matriz para abastecer el almacén. Finalmente, el subsistema de Ventas se encarga de las transacciones en caja y el subsistema de Clientes con los propios clientes y con la entrega de los pedidos a domicilio a ellos.

4.3.1. Requisitos funcionales

4.3.1.1. Generales

ID	RF 1.1.1
Nombre	Inicio de sesión
Descripción	Las tiendas dadas de alta pueden acceder al sistema introduciendo su usuario y contraseña. El sistema debe garantizar la seguridad tanto del acceso como del almacenaje de las contraseñas
Prioridad	Alta
Estado	Completado

Tabla 7: Requisito funcional 1.1.1: Inicio de sesión

ID	RF 1.1.2
Nombre	Registro de tienda
Descripción	Nuevas tiendas podrán ser dadas de alta en la plataforma.
Prioridad	Alta
Estado	Completado

Tabla 8: Requisito funcional 1.1.2: Registro de usuarios

4.3.1.2. Subsistema de almacén

- **Gestión de productos**

ID	RF 2.1.1
Nombre	Añadir productos
Descripción	El sistema podrá añadir nuevos productos al inventario del almacén, incluyendo información como una descripción, precio, variables sobre el stock y subcategoría
Prioridad	Alta
Estado	Completado

Tabla 9: Requisito funcional 2.1.1: Añadir productos

ID	RF 2.1.2
Nombre	Modificar producto
Descripción	Debe permitir modificar la información de los productos existentes, como el precio, la descripción, las variables de stock o la subcategoría
Prioridad	Alta
Estado	Completado

Tabla 10: Requisito funcional 2.1.2: Modificar producto

ID	RF 2.1.3
Nombre	Eliminar producto

Descripción	El sistema debe permitir la eliminación de productos que ya no se encuentren en el inventario.
Prioridad	Alta
Estado	Completado

Tabla 11: Requisito funcional 2.1.3: Eliminar producto

ID	RF 2.1.4
Nombre	Mostrar producto
Descripción	Se podrá mostrar la información de un producto concreto
Prioridad	Alta
Estado	Completado

Tabla 12: Requisito funcional 2.1.4: Mostrar producto

ID	RF 2.1.5
Nombre	Listar productos
Descripción	Debe permitir listar los diferentes productos que hay en el inventario filtrando por subcategorías.
Prioridad	Alta
Estado	Completado

Tabla 13: Requisito funcional 2.1.5: Listar productos

ID	RF 2.1.6
Nombre	Sacar producto a la venta
Descripción	Debe permitir registrar que un producto que estaba en el almacén se ha puesto en un stand de la tienda para su posible venta
Prioridad	Alta
Estado	Completado

Tabla 14: Requisito funcional 2.1.6: Sacar producto a la venta

ID	RF 2.1.7
Nombre	Aviso reponer stock
Descripción	El sistema debe avisar si el stock disponible para la venta de un producto está a cero para poder reponerlo
Prioridad	Alta
Estado	Completado

Tabla 15: Requisito funcional 2.1.7: Aviso reponer stock

- **Gestión de Stock**

ID	RF 2.2.1
Nombre	Registrar stock
Descripción	Debe permitir registrar la información detallada de cada stock, incluyendo la fecha de caducidad, los productos asociados, la cantidad y si tiene algún descuento asociado.
Prioridad	Alta
Estado	Completado

Tabla 16: Requisito funcional 2.2.1: Registrar stock

ID	RF 2.2.2
Nombre	Mostrar stock
Descripción	Debe permitir listar los diferentes productos que hay en el inventario que pertenecen a un mismo stock.
Prioridad	Alta
Estado	Completado

Tabla 17: Requisito funcional 2.2.2: Mostrar stock

ID	RF 2.2.3
Nombre	Alertas caducidad stock
Descripción	El sistema debe proporcionar alertas cuando un producto esté próximo a vencer.
Prioridad	Alta
Estado	Completado

Tabla 18: Requisito funcional 2.2.3: Alertas caducidad stock

ID	RF 2.2.4
Nombre	Modificar stock
Descripción	Debe permitir modificar la información de los productos de stock existentes, como su fecha de caducidad
Prioridad	Media
Estado	Completado

Tabla 19: Requisito funcional 2.2.4: Modificar stock

ID	RF 2.2.5
Nombre	Eliminar stock
Descripción	El sistema debe permitir la eliminación de stock que ya no se encuentren en el inventario.
Prioridad	Media
Estado	Completado

Tabla 20: Requisito funcional 2.2.5: Eliminar stock

- **Gestión de Descuentos**

ID	RF 2.3.1
Nombre	Crear descuento
Descripción	Se debe permitir la creación de descuentos aplicables al stock de los productos
Prioridad	Alta
Estado	Completado

Tabla 21: Requisito funcional 2.3.1: Crear descuento

ID	RF 2.3.2
Nombre	Establecimiento duración de descuento
Descripción	Debe ser posible establecer la duración del descuento

Prioridad	Media
Estado	Completado

Tabla 22: Requisito funcional 2.3.2: Establecimiento duración de descuento

ID	RF 2.3.3
Nombre	Mostrar descuentos activos
Descripción	El sistema permitirá mostrar todos los descuentos que se tienen
Prioridad	Media
Estado	Completado

Tabla 23: Requisito funcional 2.3.3: Mostrar descuentos activos

ID	RF 2.3.4
Nombre	Modificar descuento
Descripción	Se podrá modificar la información relacionada con un descuento, como el porcentaje aplicado.
Prioridad	Media
Estado	Completado

Tabla 24: Requisito funcional 2.3.4: Modificar descuento

4.3.1.3. Subsistema de pedidos

ID	RF 3.1.1
Nombre	Crear pedido
Descripción	El sistema debe permitir crear nuevos pedidos realizados al proveedor, incluyendo los distintos productos y la cantidad. Indicando la fecha y la tienda que lo realiza.
Prioridad	Alta
Estado	Completado

Tabla 25: Requisito funcional 3.1.1: Crear pedido

ID	RF 3.1.2
Nombre	Modificar pedido
Descripción	Debe permitir modificar la información del pedido antes de que este se solicite si se refiere a los productos y la cantidad que incluye. También se podrá modificar el estado o la fecha de entrega.
Prioridad	Media
Estado	Completado

Tabla 26: Requisito funcional 3.1.2: Modificar pedido

ID	RF 3.1.3
Nombre	Anular pedido
Descripción	El sistema debe permitir la anulación de pedidos pendientes de entrega, indicando el motivo de la anulación.
Prioridad	Media
Estado	Completado

Tabla 27: Requisito funcional 3.1.3: Anular pedido

ID	RF 3.1.4
Nombre	Seguimiento pedido
Descripción	Se debe proporcionar la capacidad de realizar un seguimiento del estado de los pedidos, desde su creación hasta su entrega
Prioridad	Alta
Estado	Completado

Tabla 28: Requisito funcional 3.1.4: Seguimiento pedido

ID	RF 3.1.5
Nombre	Listar productos
Descripción	Debe permitir listar los diferentes pedidos que hay en el sistema filtrando por estado actual o fecha.
Prioridad	Media
Estado	Completado

Tabla 29: Requisito funcional 3.1.5: Listar productos

ID	RF 3.1.6
Nombre	Registro de entrega
Descripción	El sistema debe permitir registrar las entregas recibidas por los proveedores, actualizando automáticamente el inventario con los productos que contiene.
Prioridad	Alta
Estado	Completado

Tabla 30: Requisito funcional 3.1.6: Registro de entrega

ID	RF 3.1.7
Nombre	Pedido automático
Descripción	El sistema debe realizar pedidos automáticos cuando el stock de un producto alcance el mínimo si así está establecido. Avisará al usuario.
Prioridad	Alta
Estado	Completado

Tabla 31: Requisito funcional 3.1.7: Pedido automático

4.3.1.4. Subsistema de ventas

ID	RF 4.1.1
Nombre	Registro de ventas
Descripción	El sistema deberá permitir registrar las ventas realizadas en cada tienda, incluyendo los productos vendidos, cantidades y precios. También se podrán eliminar productos de la venta.
Prioridad	Alta
Estado	Completado

Tabla 32: Requisito funcional 4.1.1: Registro de ventas

ID	RF 4.1.2

Nombre	Generación de ticket
Descripción	Deberá permitir generar facturas para cada venta realizada, incluyendo los detalles de la transacción
Prioridad	Alta
Estado	Completado

Tabla 33: Requisito funcional 4.1.2: Generación de ticket

ID	RF 4.1.3
Nombre	Proceso de pago
Descripción	El sistema debe permitir procesar el pago de las ventas realizadas, ofreciendo diferentes métodos de pago como efectivo o tarjeta.
Prioridad	Alta
Estado	Completado

Tabla 34: Requisito funcional 4.1.3: Proceso de pago

4.3.1.5. Subsistema de clientes

ID	RF 4.2.1
Nombre	Incluir entrega para cliente
Descripción	Se dará la opción de incluir una entrega a domicilio de la venta si el cliente así lo desea. Incluirá los datos de la dirección de entrega
Prioridad	Alta
Estado	Completado

Tabla 35: Requisito funcional 4.2.1: Incluir entrega para cliente

ID	RF 4.2.2
Nombre	Modificar datos entrega
Descripción	El sistema permitirá modificar los datos de la entrega
Prioridad	Media
Estado	Completado

Tabla 36: Requisito funcional 4.2.2: Modificar datos entrega

ID	RF 4.2.3
Nombre	Modificar estado entrega
Descripción	El sistema podrá cambiar el estado de la entrega así como finalizarla
Prioridad	Alta
Estado	Completado

Tabla 37: Requisito funcional 4.2.3: Modificar estado entrega

4.3.2. Requisitos no funcionales

ID	RNF 01
Nombre	Usabilidad
Descripción	El sistema deberá ser fácil de usar e intuitivo para los usuarios
Prioridad	Alta
Estado	Completado

Tabla 38: Requisito no funcional 01: Usabilidad

ID	RNF 02
Nombre	Rendimiento
Descripción	El sistema deberá ser capaz de manejar alto volumen de transacciones sin experimentar demoras significativas en las respuestas
Prioridad	Alta
Estado	Completado

Tabla 39: Requisito no funcional 02: Rendimiento

ID	RNF 03
Nombre	Seguridad
Descripción	El sistema debe implementar medidas de seguridad robustas para proteger la integridad y confidencialidad de los datos del cliente y la empresa
Prioridad	Alta
Estado	Completado

Tabla 40: Requisito no funcional 03: Seguridad

ID	RNF 04
Nombre	Fiabilidad
Descripción	El sistema debe estar disponible en todo momento, minimizando los tiempos de inactividad.
Prioridad	Alta
Estado	Completado

Tabla 41: Requisito no funcional 04: Fiabilidad

ID	RNF 05
Nombre	Mantenibilidad
Descripción	El sistema debe ser fácil de mantener y actualizar, permitiendo la incorporación de nuevas funcionalidades y la corrección de errores de manera eficiente
Prioridad	Media
Estado	Completado

Tabla 42: Requisito no funcional 05: Mantenibilidad

ID	RNF 06
Nombre	Escalabilidad
Descripción	Este sistema debe ser escalable, permitiendo la adición de nuevos usuarios y funcionalidades sin afectar negativamente al rendimiento general
Prioridad	Alta
Estado	Completado

Tabla 43: Requisito no funcional 06: Escalabilidad

ID	RNF 07
Nombre	Compatibilidad
Descripción	Este sistema debe ser compatible con los navegadores web más comunes como Chrome, Firefox o Edge
Prioridad	Media
Estado	Completado

Tabla 44: Requisito no funcional 07: Compatibilidad

4.3.3. Requisitos de información

ID	RI-01
Nombre	Tienda
Descripción	Datos esenciales para identificar una tienda
Contenido	Información como país, ciudad, dirección y una descripción del establecimiento.

Tabla 45: Requisito de Información 01: Tienda

ID	RI-02
Nombre	Usuario
Descripción	Un usuario es una persona que trabaja en una tienda concreta
Contenido	Cada usuario tendrá su nombre de usuario y su contraseña para iniciar sesión. Nombre completo con apellidos, el rol que desempeña en la tienda así como la tienda en la que trabaja.

Tabla 46: Requisito de Información 02: Usuario

ID	RI-03
Nombre	Producto
Descripción	Información identificativa de forma inequívoca y referente al estocaje en el almacén.
Contenido	Datos identificativos como el GTIN, una descripción del propio producto, una imagen, el precio en euros de coste y de venta; y la subcategoría a la que pertenece. También información relativa al almacén como el stock total y el disponible a la venta, el stock

	mínimo que debe haber y la cantidad que se pide automáticamente si se alcanzara el stock mínimo.

Tabla 47: Requisito de Información 03: Producto

ID	RI-04
Nombre	Venta
Descripción	Información relativa al momento y lugar en el que se realizó la venta e información detallada de cada producto que contiene la venta. Información de la entrega a domicilio y su estado actual.
Contenido	Para cada producto que incluya la venta se incluirá la cantidad y el precio unitario y el stock al que pertenece. Además del precio total de la venta se incluirá también la fecha, la hora y la tienda en la que se realizó la venta. Si el pedido lo necesitase se añadirá información relativa a la entrega a domicilio como el nombre del cliente, su localidad, código postal y dirección; además del estado actual en el que se encuentra la entrega y quien la realiza.

Tabla 48: Requisito de Información 04: Venta

ID	RI-05
Nombre	Pedido
Descripción	Información relativa al momento en el que se realizó el pedido, así como la tienda que lo hizo. También el estado actual y los productos con sus respectivas cantidades.
Contenido	Para cada producto que incluya el pedido se incluirá la cantidad pedida.

Tabla 49: Requisito de Información 05: Pedido

ID	RI-06
Nombre	Stock
Descripción	Información sobre los productos que contiene y del descuento asociado.
Contenido	Cada stock incluirá la fecha de caducidad de los productos que contiene, así como la cantidad de estos productos. También se incluye información del descuento asociado como una descripción y el porcentaje aplicado.

Tabla 50: Requisito de Información 06: Stock

4.4. Modelos de caso de uso y escenarios de uso

4.4.1. Actores del sistema

Este sistema no destaca por involucrar la interacción de muchos actores, pero no por ello vamos a dejar de detallar aquellos que desempeñan

roles específicos dentro del entorno operativo. Los actores identificados en el sistema son los siguientes:

- **Trabajadores de Supermercado**: Este grupo constituye el actor principal del sistema. Su interacción se centra en la gestión diaria de inventario, pedidos y ventas dentro del supermercado. Los trabajadores utilizan el sistema para registrar nuevas llegadas de productos, gestionar el stock, realizar pedidos al proveedor y registrar las ventas.

- **Administrador del Sistema**: Este usuario tiene el mayor nivel de control dentro del propio sistema. Se encarga de dar de alta las nuevas tiendas, así como de modificar su información, configurar parámetros clave del sistema y supervisar el funcionamiento general. El administrador también es responsable de realizar copias de seguridad de la base de datos y garantizar la seguridad y la integridad de la información.

- **Proveedor**: El papel que desempeña básicamente es el de suministrador. En el proceso de gestión de pedidos es el encargado de introducir los nuevos productos en el catálogo de los posibles para pedir. Los proveedores al ser los responsables de suministrar los productos al supermercado pueden reciben notificaciones del sistema cuando se requiere reabastecer inventario por parte de una de las tiendas.

Estos actores desempeñan roles distintos pero complementarios dentro del sistema de gestión logística del supermercado, asegurando que las operaciones se realicen de manera eficiente y fluida. La Ilustración 8: Diagrama de caso de uso del Sistema es un ejemplo de cómo estos actores interactúan con el sistema en diferentes operaciones

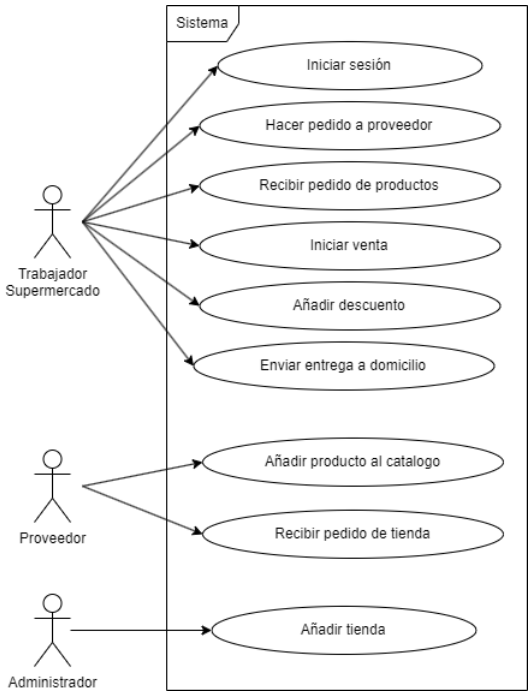

Ilustración 16: Diagrama de Caso de Uso del Sistema

4.4.2. Escenarios de uso

En esta sección, se procede a examinar los escenarios de uso que involucran a los diversos usuarios que interactúan con la plataforma. Cada escenario será ilustrado mediante tablas que proporcionan una visión clara de las interacciones entre los usuarios y el sistema. Los escenarios se enfocan en los aspectos críticos de la plataforma, poniendo énfasis en aquellos factores que resultan más relevantes y distintivos. Se ha tomado la decisión de excluir escenarios de uso comunes y demasiado intuitivos, como el inicio de sesión, registro o cierre de sesión, con el fin de centrarse en aquellos que aportan una comprensión más profunda de las funcionalidades específicas y la interacción del usuario con la plataforma.

ID	EU-01
Nombre	Añadir producto al catálogo
Actor	Proveedor
Descripción	El proveedor da de alta un nuevo producto en el sistema para que las tiendas puedan pedirlo para venderlo.
Precondición	- El proveedor ha iniciado sesión - El producto no existe en el sistema
Postcondición	- El producto se incluye en el sistema
Flujo	1. El proveedor accede al apartado de alta de productos 2. Se abre un formulario con los datos del producto 3. El proveedor rellena los datos del producto 4. Envía los datos al sistema 5. Recibe confirmación

Tabla 51: Escenario de Uso 01: Añadir producto al catálogo

ID	EU-02
Nombre	Realizar pedido
Actor	Trabajador Supermercado
Descripción	Un trabajador de una tienda concreta realiza un pedido al proveedor para abastecer su tienda de productos.
Precondición	- El trabajador ha iniciado sesión como tienda
Postcondición	- Se crea un nuevo pedido en el sistema
Flujo	1. El trabajador accede al apartado de realizar pedidos 2. Selecciona los productos deseados y la cantidad 3. El sistema crea el pedido 4. El trabajador confirma el pedido

Tabla 52: Escenario de Uso 02: Realizar pedido

ID	EU-03
Nombre	Enviar pedido
Actor	Proveedor
Descripción	El proveedor envía el pedido que ha hecho un trabajador para su propia tienda
Precondición	- El proveedor ha iniciado sesión - Un trabajador de una tienda ha realizado un pedido
Postcondición	- El pedido cambia de estado
Flujo	1. El proveedor accede al apartado de pedidos pendientes 2. Consulta la información del pedido 3. Prepara el pedido físicamente 4. Envía el pedido con un camión

Tabla 53: Escenario de Uso 03: Enviar pedido

ID	EU-04
Nombre	Recibir camión
Actor	Trabajador supermercado
Descripción	El trabajador del supermercado descarga la mercancía del camión y la almacena en el supermercado
Precondición	- El trabajador ha iniciado sesión - El pedido ha sido enviado por el proveedor
Postcondición	- Los productos del pedido son añadidos al stock del almacén - El pedido cambia de estado
Flujo	1. El trabajador descarga el pedido del camión 2. Lee la etiqueta del palé con un lector 3. El sistema añade los productos al stock del almacén 4. El trabajador coloca los productos en la tienda 5. El sistema registra las unidades que están a la venta

Tabla 54: Escenario de Uso 04: Recibir camión

ID	EU-05
Nombre	Hacer venta
Actor	Trabajador Supermercado
Descripción	Se realiza una venta de uno o varios productos en una tienda
Precondición	- El trabajador ha iniciado sesión en la caja
Postcondición	- Una nueva venta se ha registrado en el sistema - El stock del almacén disminuye
Flujo	1. El trabajador dese la interfaz de la caja crea una nueva venta 2. Pasa el/los producto/s por el lector de códigos de barras 3. El sistema calcula el precio de la venta 4. Se realiza el pago 5. El sistema imprime el ticket 6. El sistema disminuye del stock del almacén las unidades vendidas

Tabla 55: Escenario de Uso 05: Hacer venta

ID	EU-06
Nombre	Añadir descuento
Actor	Trabajador Supermercado
Descripción	Un trabajador aplica cierto descuento sobre algún stock
Precondición	- El trabajador ha iniciado sesión - El lote sobre el que quiere aplicar el descuento existe - El descuento es aprobado por el gerente y la matriz
Postcondición	- Un nuevo descuento se ha añadido al sistema
Flujo	1. El trabajador accede al módulo de descuentos 2. El sistema le muestra un formulario 3. El trabajador añade los datos del descuento y selecciona el stock asociado 4. Confirma el descuento

Tabla 56: Escenario de Uso 06: Añadir descuento

4.4.3. Restricciones y reglas de negocio

Estas restricciones y reglas se establecen para garantizar el correcto funcionamiento, de forma coherente y segura del sistema; protegiendo los intereses de la empresa, así como de sus clientes.

Restricciones de Acceso

El acceso al sistema se limita según el rol que se tenga dentro del propio sistema, con garantías de seguridad y eficiencia en las operaciones.

- **Trabajador**: Dispone de acceso a casi la totalidad de las

herramientas; como la gestión del inventario, el pedido de productos a la matriz y el de la venta a los clientes en caja.

- **Proveedor**: Tiene acceso al sistema para acciones básicas como incluir nuevos productos en el catálogo o consultar los pedidos que les hacen las tiendas.

- **Administrador**: Cuenta con privilegios de acceso totales para la gestión y configuración del sistema

Reglas de Validación

- Los pedidos deben contener información completa y válida de los productos solicitados. Incluyendo siempre la fecha de caducidad de los stocks.

- Los descuentos aplicados deben cumplir con las condiciones establecidas por la matriz y no podrán exceder los limites definidos por esta.

Reglas de Negocio Adicionales

- Si la lectura del código de barras de un producto falla o no se puede leer la etiqueta, se podrá introducir manualmente el código.

4.5. Modelo de Datos y Diagrama Entidad-Relación

El modelado de datos se erige como un pilar esencial en el diseño y desarrollo de sistemas informáticos, proporcionando una guía clara sobre cómo las diferentes entidades se organizan y se relacionan dentro de la estructura de la base de datos. En el Capítulo 5 Diseño se profundizará mucho más en este aspecto. En este proyecto, hemos identificado una serie de entidades cruciales que desempeñan roles fundamentales en la operación y gestión del sistema. Desde la categorización de productos hasta la ejecución de ventas y entregas. Estas entidades interactúan de manera interdependiente, conformando el entramado operativo del sistema.

- **Categoría**: Representa las diferentes grandes familias distintas de productos que hay en el sistema.

- **Subcategoría**: Es una subdivisión de las categorías principales y proporciona una estructura jerárquica para una clasificación más detallada de cada producto.

- **Producto**: Diferentes artículos que estás disponibles para la venta dentro del supermercado.

- **Stock**: Representa el estocaje de productos que tiene una tienda.

- **Descuento**: Hace alusión a las diferentes promociones o reducciones de precio que pueden aplicarse a los productos durante una venta.

- **Venta**: Registra la adquisición de uno o varios productos por parte del cliente en una tienda.

- **DetalleVenta**: Representa la asociación de un producto a una venta.

- **Entrega**: Distribución física de una venta a una dirección concreta.

- **Pedido**: Solicitud de productos del catálogo al proveedor por parte de una tienda concreta.

- **DetallePedido**: Representa la asociación de un producto a un pedido.

- **Tienda**: Representa los distintos puntos de venta físicos donde se adquieren los productos. Cada una tiene su propio inventario y realiza ventas independientes.

- **Usuario**: Cada uno de los trabajadores asociados a una tienda que acceden al sistema.

- **EstadoPedido**: Las diferentes etapas por las que pasa un pedido desde que este es encargado por la tienda al proveedor hasta que lo recibe.

- **EstadoEntrega:** Las diferentes etapas por las que pasa una entrega desde que esta es solicitada por el cliente al hacer la venta hasta que un trabajador la lleva a su domicilio.

El Diagrama Entidad-Relación (ER) proporciona una representación visual detallada de estas interacciones, destacando los vínculos esenciales entre las entidades y su funcionalidad específica dentro del ecosistema del sistema. Este enfoque visual permite comprender de manera integral cómo estas entidades se conectan y colaboran entre sí para cumplir con los objetivos establecidos.

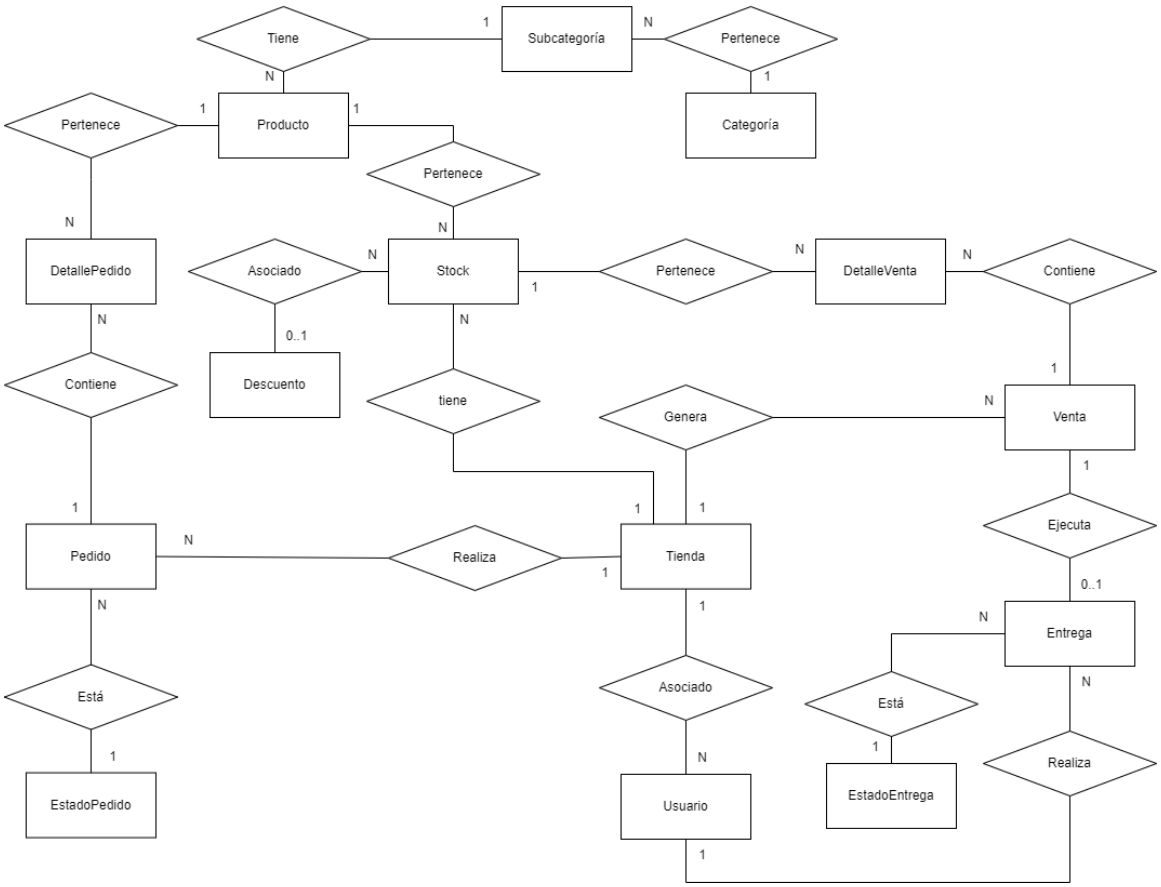

Ilustración 17: Diagrama Entidad - Relación

Capítulo 5
Diseño

Este capítulo está dedicado a detallar tanto la arquitectura como el diseño del sistema que sustentará la solución propuesta, presentando un análisis detallado de cada uno de los componentes clave que conforman la infraestructura tecnológica del proyecto. Comenzamos con la arquitectura del sistema, describiendo su estructura general y los principios de diseño que garantizan la escalabilidad, la seguridad y la eficiencia en el manejo de datos y procesos (véase apartado 5.1).

También nos enfocamos en comentar el diseño de la base de datos, crucial para el soporte y la gestión efectiva de la información. Este apartado contiene la explicación de las tablas que compondrán la base de datos, asegurando que cada una esté optimizada para las operaciones requeridas y que en conjunto proporcionen una integración cohesiva y funcional (véase apartado 5.2). Posteriormente, abordamos el diseño de las interfaces de usuario, que juegan un papel esencial en la interacción efectiva entre el sistema y sus usuarios (véase apartado 5.3).

5.1. Modelo y Arquitectura del Sistema

Para entender el modelo y arquitectura del sistema, debemos recordar que esta es una aplicación web a la que los usuarios acceden. El modelo seguido, en este caso es uno en capas. Este enfoque permite organizar el código en diferentes niveles de abstracción, facilitando la

mantenibilidad, escalabilidad y modularidad de la aplicación. La arquitectura del sistema se divide en varias capas, cada una con responsabilidades específicas.

La primera de ellas es Capa de Presentación, está compuesta por el frontend, básicamente es la interfaz de usuario del sistema. Es la parte visible y accesible para los usuarios finales, donde interactúan con la aplicación a través de navegadores web. Su funcionabilidad es recibir las acciones del usuario, las procesa mediante scripts y envía solicitudes HTTP a la siguiente capa. Luego, interpreta y muestra las respuestas recibidas.

La siguiente capa es La Capa de Lógica de Negocio. Está compuesta por los Controladores, los Servicios y los Mappers (componentes que se detallan más adelante). Esta capa lo que hace es recibir las solicitudes HTTP de la capa anterior a través de los Controladores, las procesa según la lógica de negocio definida en los Servicios y usa los Mappers para adaptar los datos al formato necesario; luego interactúa con la base de datos a través de la capa siguiente. Finalmente, devuelve la respuesta adecuada al frontend.

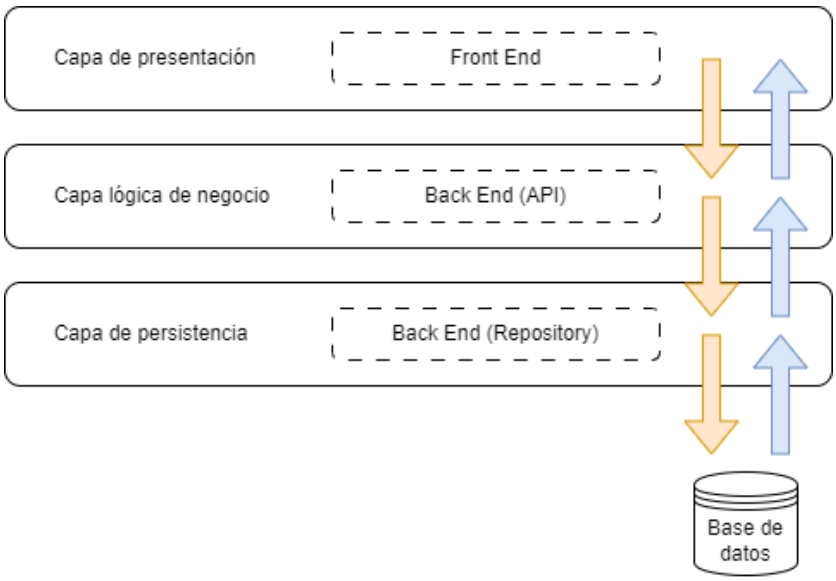

Ilustración 18: Arquitectura por capas de la Aplicación

La Capa de Persistencia maneja la interacción directa con la base de datos. Proporciona una abstracción sobre el acceso a los propios datos, facilitando las operaciones CRUD. Los repositorios se encargan de ejecutar consultas a la base de datos y de devolver los resultados a la capa anterior para su procesamiento.

Finalmente nos encontramos con la Base de Datos, que para algunos autores es considerada una capa distinta, sería La Capa de Datos que almacena toda la información que maneja el sistema. Es la fuente de datos a

la que acceden las capas superiores para realizar operaciones de lectura y escritura. Esta almacena permanentemente los datos del sistema, permitiendo su recuperación y manipulación a través de consultas SQL ejecutadas por los repositorios.

Como hemos visto diferenciamos dos grandes partes, el *Frontend* y el *Backend*. La primera de ellas es la que interactúa con el usuario que usa el sistema, esta interacción se lleva a cabo usando tecnologías básicas como lo son páginas HTML, no existe un *framework* adicional en esta parte que nos dé soporte. La interacción de estas páginas HTML entre ellas como con el propio usuario que está navegando por la aplicación web, se hace con JavaScript y el estilo se le da con CSS. Estilo de diseño que es *Web Responsive* para asegurar la correcta visualización en los distintos dispositivos. Al interactuar con estas páginas lo que está haciendo el usuario es generar peticiones HTTP, que son captadas por nuestro *Backend* y generan una respuesta que es devuelta, interpretada y mostrada al usuario, de esta forma se obtiene la información y se interactúa con el sistema.

Adentrándonos en esta segunda parte, encontramos el *Backend*. El cual ha sido desarrollado utilizando un *framework* de soporte como Spring Framework [22][23]. En el capítulo siguiente Capitulo 6 Implementación se detallan más a fondo los términos sobre la programación y codificación, en el presente nos centraremos en el esquema de la arquitectura. Este *Backend* es el encargado de captar las peticiones HTTP gracias a los *endpoints* de la API REST que los administra y redirige hasta la base de datos, con la que mantiene una conexión ininterrumpida. La base del proyecto es una simple clase que lanza un comando de SpringBoot que arranca la propia aplicación, básicamente lo que hace es crear un contexto de aplicación adecuado, teniendo en cuenta la configuración dada.

Con la Ilustración 19 podemos observar la arquitectura de la que se está hablando. Dentro del dominio de nuestra aplicación para cada entidad de las que disponemos tenemos los siguientes elementos:

- **Entidad**: Como su propio nombre indica es la representación de la propia entidad y sus atributos. Representa cada una de las tablas de nuestra base de datos. En ella se definen la clave primaria o las relaciones entre entidades, entre otros aspectos.

- **Controlador**: Encargado de gestionar las interacciones del usuario, partiendo con los datos que este introduce o solicita; devuelve una respuesta adecuada. Cada método dentro del controlador maneja una ruta de acceso específica y un tipo de solicitud HTTP concreta, los denominados *endpoints*. Este llama al servicio para procesar los datos y devuelve los resultados obtenidos al cliente.

- **Servicio**: Contienen la lógica esencial de negocio, es decir, la forma de actuar con los datos. Es llamado desde el controlador. En él se realizan las operaciones básicas de procesamiento de datos para ello, actúa de intermediario entre el controlador y el repositorio.

- **Mapper**: Es un tipo muy concreto de servicio. Este permite transformar una instancia de la entidad en un DTO *(Data Transfer Object)*, transformación necesaria sobre el modelo de datos para hacer que la petición HTTP y la base de datos se entiendan.

- **Repositorio**: Este componente es el que maneja directamente la persistencia de la entidad en la base de datos. Proporciona los métodos CRUD para hacer las consultas a la propia base de datos. Abstrae la logia de acceso a datos del resto de la aplicación.

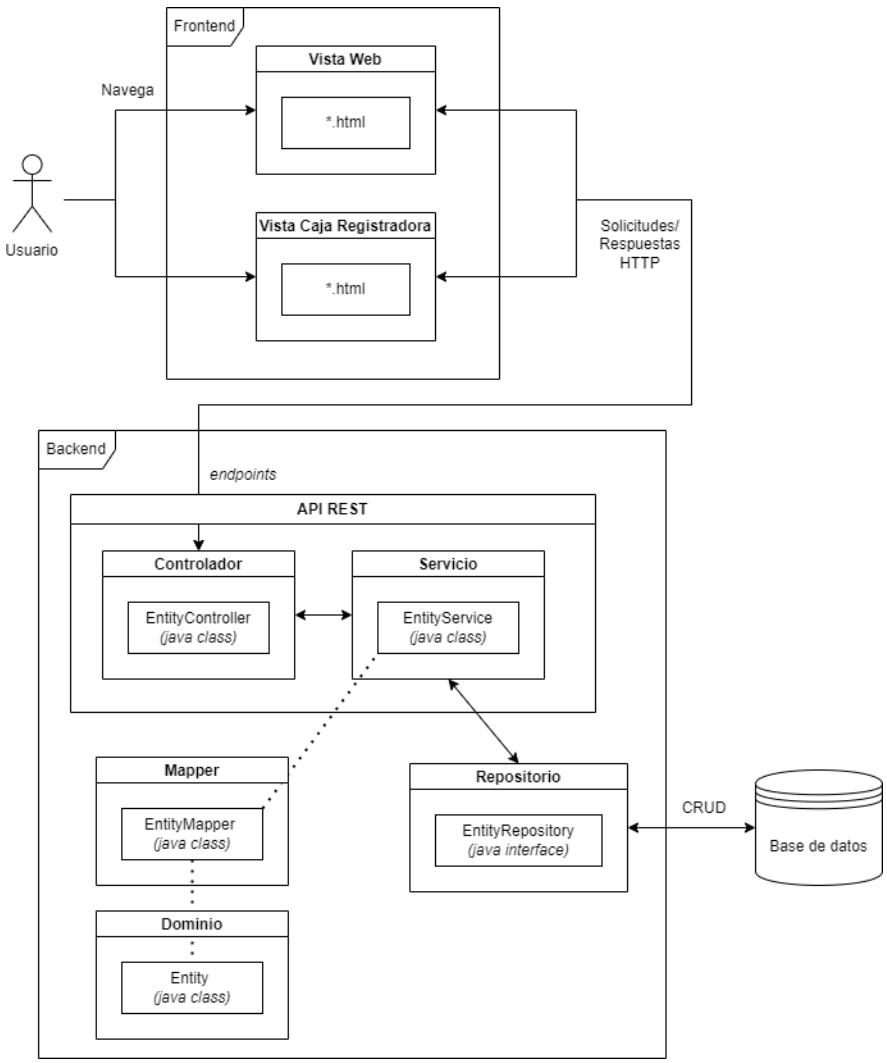

Ilustración 19: Esquema arquitectura del proyecto

5.2. Diseño de Base de Datos

En este apartado del presente capítulo se comenta cada tabla explicando sus atributos y el porqué de su diseño. Así como el diseño final de la base de datos observable en la Ilustración 33, resultado del trabajo previo de análisis y diseño. Estas tablas contienen la información necesaria y se relacionan unas con otras de forma que la información sea accesible.

Algunos de los aspectos que se han tenido en cuenta a la hora de realizar el diseño son; en primer lugar la normalización como bandera ya que es uno de los principios fundamentales a la hora de estructurar la información para perseguir minimizar la redundancia y poder así maximizar la eficiencia. Por ello las tablas han sido llevadas hasta su tercera forma normal, en la que cada atributo no clave depende únicamente de la clave primaria y no de otros atributos no clave. Otro aspecto clave perseguido ha sido el de tener las relaciones entre entidades bien definidas ya que conseguir una buena red de relaciones facilita la consulta de los datos y claro está evita los ciclos. El uso de los buenos índices en las tablas es esencial, saber cuándo conviene usar una clave primaria autogenerada o cuando necesitamos aportar más información y proporcionarlos nosotros mismos.

Por ello, podríamos concluir en que la solución propuesta y que se comenta a continuación es una base de datos robusta y diseñada para crecer siendo algo más que un simple conjunto de información.

5.2.1. Tablas

Parece casi obvio que la primera entidad a comentar, hablando de un supermercado, fuera la de Producto. Cada artículo distinto de la tienda se identifica con una única tupla de esta tabla, es decir, el artículo “Lata Coca-Cola Zero 33 cl.” es una sola entrada en esta tabla; no hay una entrada por cada unidad de este artículo que tengamos. Dicho esto, como atributos encontramos una *descripcion* del producto, que nos hace reconocer de qué producto se trata, suele incluir la marca, o la cantidad que contiene. Para ayudar al reconocimiento del artículo también se incluye como atributo una *imagen* de este, codificada en Base64. Relativo a los precios del producto se incluyen dos atributos distintos; *precio_coste* representa el precio en euros que ha pagado el supermercado al proveedor por el artículo mientras que *precio_venta* es el precio en euros que el supermercado establece para sus clientes. Por último pero no por ello menos importante, el atributo principal que hace la función de clave primaria es *id_producto*, para ello se usa el GTIN del propio producto, ya que este es único.

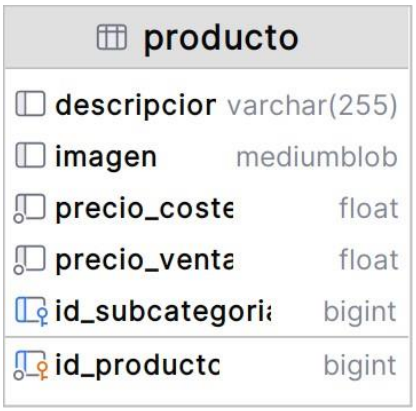

Ilustración 20: Tabla "producto"

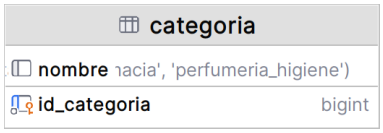

Ilustración 21: Tabla "categoría"

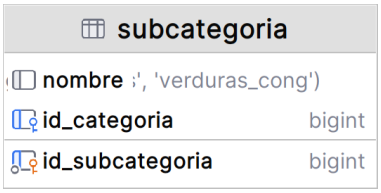

Ilustración 22: Tabla "subcategoria"

En la tabla anterior mencionada además de lo comentado, encontramos una clave externa a otra tabla, *id_subcategoria*, ya que todo producto está incluido en subcategoría específica, y esta a su vez, pertenece a una categoría principal (de ahí que en la tabla *Subcategoría* se use la clave externa *id_categoria*). La estructura de estas dos tablas *Categoría* y *Subcategoría* es muy similar, incluyen un id único y el nombre identificativo, estos serán uno de entre una lista cerrada de opciones.

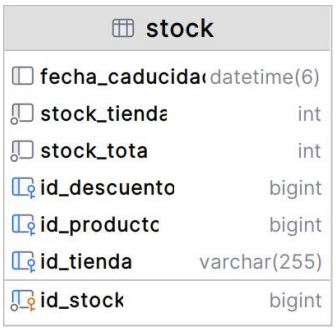

Ilustración 23: Tabla "stock"

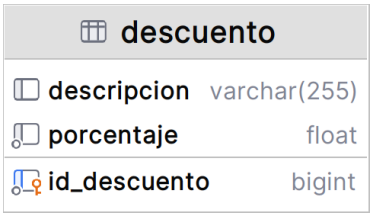

Ilustración 24: Tabla "descuento"

Como estamos en un supermercado, será muy frecuente que nos encontremos dos unidades de un mismo producto pero que tienen distinta fecha de caducidad, esto es porque pertenecen a distintas producciones. De esta idea nacen nuestras siguientes dos tablas, la primera de ellas *Stock* que además de su clave primaria *id_stock* tiene atributos que almacenan la fecha de caducidad en la que vencen dichos productos, *fecha_caducidad*, la clave externa de los productos que incluye *id_producto*. Las unidades que tiene el supermercado de dicho producto se almacenan en el atributo *stock_total*, atributo que guarda relación con *stock_tienda* siendo este último las unidades de las anteriores que están colocadas en la tienda (no guardadas en el almacén) disponibles para la venta.

Con la idea de dar pronta salida a los productos que tengan una fecha de caducidad cercana, cada lote puede tener asociado un descuento a través de la clave externa *id_descuento*. La segunda tabla, *Descuento*, incluye un atributo *descripcion* para identificar el tipo de descuento del que se trata así como el *porcentaje* de descuento a aplicar sobre el precio del producto asociado.

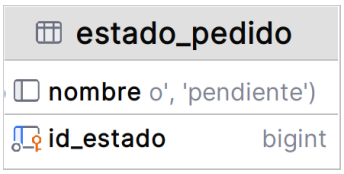

Ilustración 25: Tabla "estado_pedido"

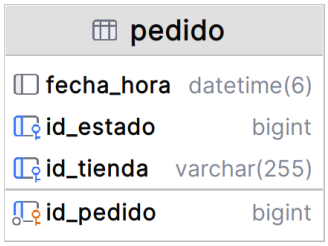

Ilustración 26: Tabla "pedido"

Ilustración 27: Tabla "detalle_pedido"

De la acción de solicitar productos al proveedor para poder venderlos en nuestro supermercado nace la necesidad de tener una tabla *Pedido*, que incluye el atributo *fecha_hora* para almacenar el momento exacto en el que se realizó dicho pedido ya que incluye tanto la fecha como la hora con una alta precisión. También incluye dos claves externas como son *id_estado* y *id_tienda* ya que cada pedido en todo momento tiene asociado un estado concreto que refleja su situación actual y también tiene asociado una tienda, que es la que lo ha realizado al proveedor. Podríamos pensar que esta tabla está incompleta ya que no contiene información de los productos que contiene el pedido, pero aquí es donde entra en juego la siguiente tabla. Debido a que la relación entre "Producto" y "Pedido" es del tipo muchos a muchos, ya que un pedido puede contener varios productos distintos y al mismo tiempo un producto puede estar en varios pedidos distintos. De dicha relación surge la tabla intermedia *Detalle_pedido* que almacena como claves externas el pedido, *id_pedido* y el producto, *id_producto*. Además guarda en un atributo *cantidad* las unidades que incluye dicho encargo del producto en cuestión. De esta forma almacenamos toda la información de una forma elegante y sencilla para que pueda ser accesible por el sistema.

Ilustración 28: Tabla "tienda"

Ilustración 29: Tabla "usuario"

Como ya hemos adelantado antes, los pedidos al proveedor los realiza una tienda, entre otras funciones; por lo que parece obvio almacenar una tabla *Tienda*, cuyos atributos son todos cadenas de texto descriptivas de dicha tienda, como son *ciudad*, *pais*, *direccion* y una breve *descripcion*. La tabla *Usuario* es la encargada de almacenar la información de cada trabajador asociado a una tienda. De la misma forma que antes, la mayoría de sus atributos son meramente descriptivos el *nombre* y *apellidos* del trabajador, su *rol* en la tienda, una clave externa que indica la tienda a la que pertenece y además necesita credenciales para iniciar sesión; por lo que se almacenan tanto su *username* como clave primaria como su contraseña *password* claro está encriptada. Se ahondará más en el aspecto de la encriptación y seguridad en el siguiente capitulo, Capitulo 5 Implementación.

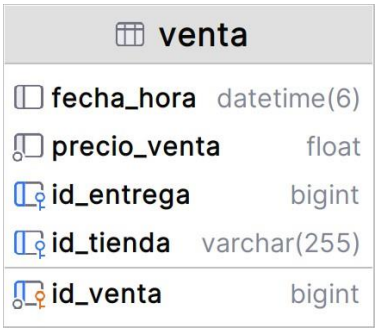

Ilustración 30: Tabla "venta"

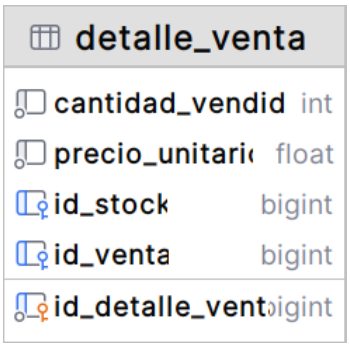

Ilustración 31: Tabla "detalle_venta"

Una de las funciones principales de un supermercado, por no decir la principal, es la de vender productos a sus clientes para ello en nuestro sistema contamos con la tabla *Venta.* Esta representa una compra por parte de un cliente que se lleva a cabo en el establecimiento, de ahí usar una clave

externa *id_tienda*; en un momento concreto almacenado en *fecha_hora*. Se almacena también el precio que el cliente ha pagado por esta compra en el atributo *precio_venta*. Como el supermercado ofrece la posibilidad de llevar la compra del cliente a su casa, se incluye también una clave externa *id_entrega* que hace referencia a dicha entrega.

Antes de ello, comentar que de igual manera que sucedía con la relación entre "Producto" y "Pedido", sucede ahora entre "Stock" y "Venta", por eso se ha resuelto de la misma forma, introduciendo una tabla intermedia *Detalle_venta*. En ella se incluyen las referencias tanto al stock del producto vendido como a la venta; *id_stock* e *id_venta* respectivamente. Además se almacena la cantidad vendida de ese producto *cantidad_vendida* y el precio que tiene una unidad de dicho artículo *precio_unitario*.

Ilustración 32: Tabla "entrega"

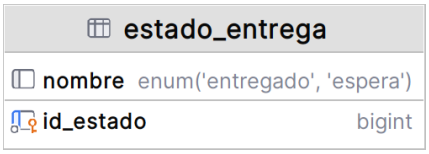

Ilustración 33: Tabla "estado_entrega"

Como adelantábamos, las ventas a los clientes pueden llevar asociada una entrega por lo que nuestras ultimas tablas son *Entrega* la cual almacena atributos descriptivos como son el nombre del cliente *cliente*, su código postal *cod_postal*, una cadena de texto con la dirección completa *direccion* así como su localidad *localidad*. Por último esta tabla cuenta con dos claves externas, una hacia el estado en el que se encuentra la entrega en el momento actual *estado* y también una hacia el trabajador que realizó o va a realizar la entrega *repartidor*. La tabla *Estado_entrega* es una simple tabla que almacena la información del nombre del estado.

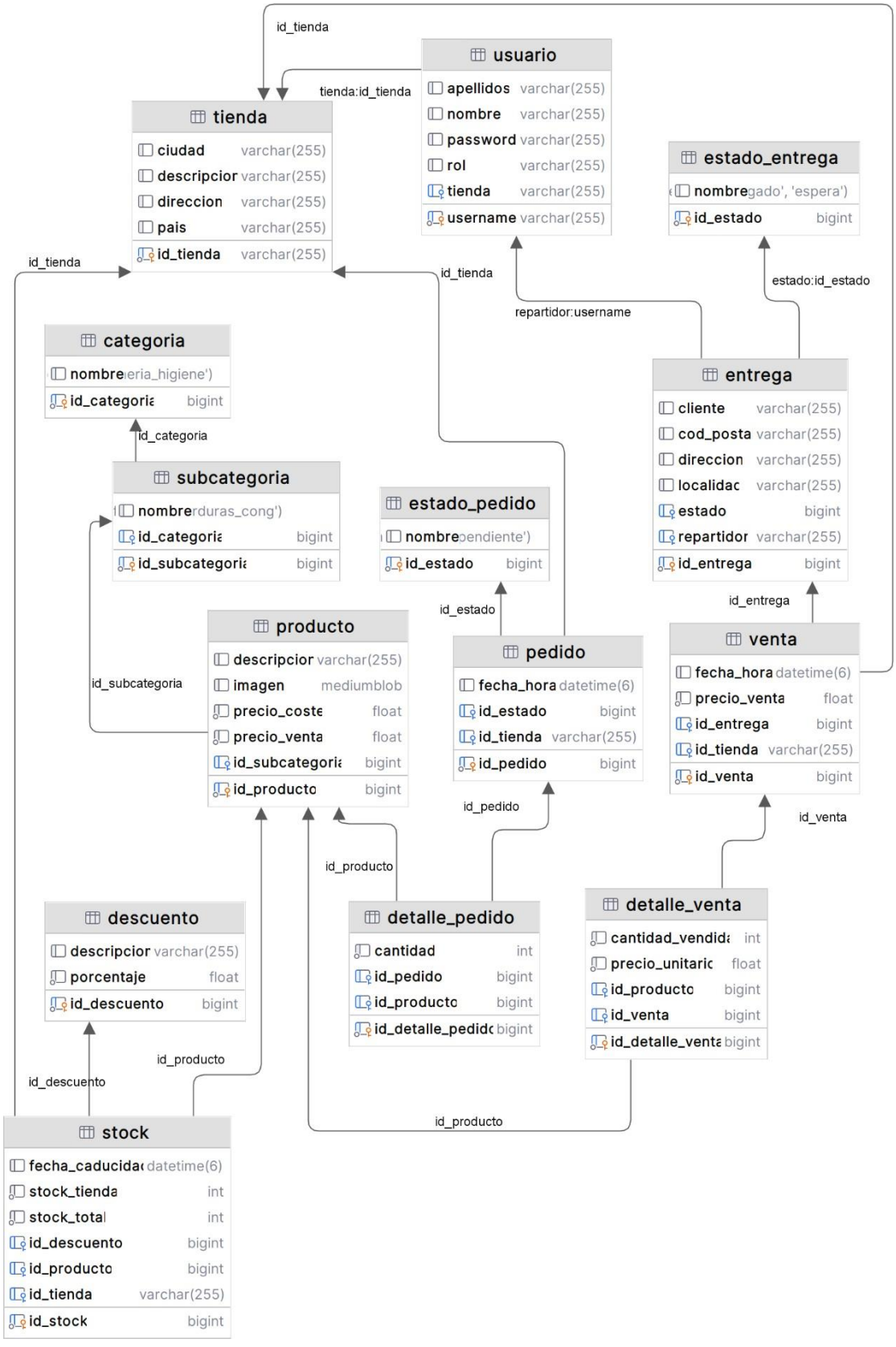

Ilustración 34: Diagrama de clases

5.3. Diseño de las interfaces

En esta sección se incluyen y comentan los mockups de las diferentes páginas que tendrá el sistema. Todos ellos tratan de ser una guía de aquello que requiere el cliente incluyendo las funcionalidades mínimas que desea ver en cada página.

5.3.1. Mockups de las páginas web

En primer lugar como todo sistema que cuente con usuarios se necesita una página de inicio de sesión para poder acceder al mismo.

Ilustración 35: Mockup inicio sesión

Una vez iniciada la sesión, la página principal es aquella que muestra tanto la información del usuario que se ha registrado como la de la tienda a la que pertenece.

Ilustración 36: Mockup página home

Una vez registrado el usuario, para viajar de una página a otra, este cuenta con un menú vertical situado en la parte izquierda. Acompañado también de un menú horizontal en la parte superior que incluye un botón para poder cerrar la sesión. Ambos aparecerán en todas las páginas para hacer la navegación más amena. La siguiente página es la que usará el usuario para pedir productos al proveedor y que se los envíen a su tienda. Se incluye también un filtro de búsqueda para los productos.

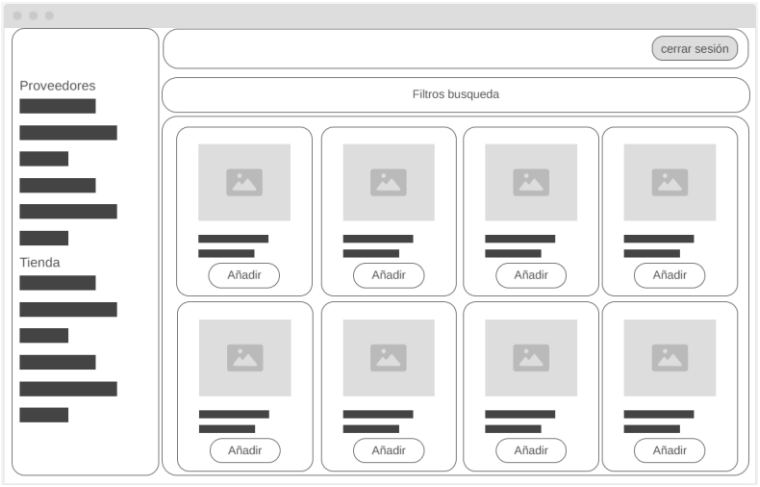

Ilustración 37: Mockup página catálogo

El sistema cuenta también con una página en la que los trabajadores pueden consultar el stock del almacén de su tienda, ver las unidades que tienen disponibles en tienda, cuando caducan…

Ilustración 38: Mockup página almacén

Es también necesaria una página donde un trabajador de una tienda pueda visualizar los pedidos que ha hecho al proveedor, tanto aquellos que están pendientes como los que están entregados. También de una forma similar es posible visualizar las ventas que una tienda ha hecho

Ilustración 39: Mockup página mis pedidos

Ilustración 40: Mockup página mis ventas

No nos podemos olvidar que estamos en un supermercado y que es necesario tener una página para las cajas registradoras que asisten una venta. Básicamente estas se componen de muchos botones que sirven de atajos para las funciones más comunes que se utilizan en una venta.

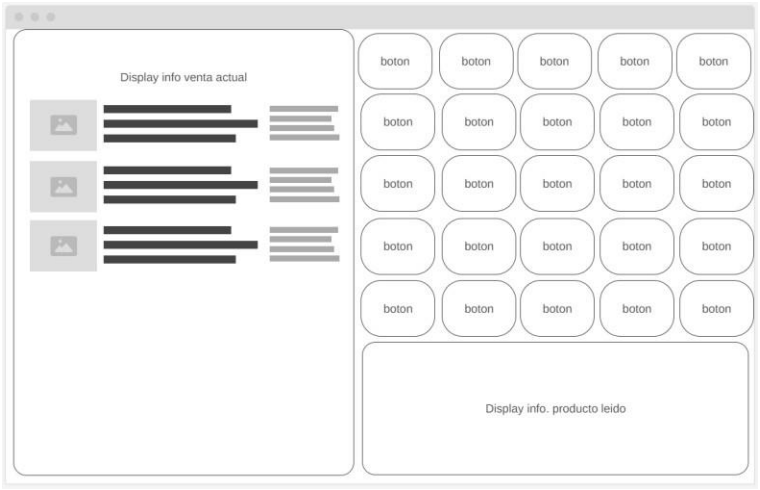

Ilustración 41: Mockup página caja registradora

Capítulo 6
Implementación

El presente capítulo se enfoca en describir de manera detallada la implementación de la solución propuesta, describiendo las herramientas y tecnologías utilizadas. Se examina minuciosamente la estructura y el entorno de desarrollo empleado, desglosando la implementación y comentando los diferentes componentes que conforman el sistema.

Comenzamos con una explicación de las herramientas y el entorno de desarrollo utilizado, distinguiendo entre Backend y Frontend, ya que cada uno presenta particularidades y herramientas específicas (véase apartado 6.1). Se detalla la estructura de los paquetes y clases fundamentales que componen el Backend, desde la configuración inicial hasta los controladores, servicios y repositorios, proporcionando fragmentos de código para ilustrar su funcionamiento. El siguiente apartado se dedica al Frontend, que se centra en la parte gráfica y visual con la que interactúa el usuario. Aquí, se describe la organización modular del código y las herramientas empleadas para garantizar una interfaz de usuario eficiente y amigable. Se da cierre al capítulo hablando específicamente de los lenguajes utilizados (véase apartado 6.2).

Este análisis exhaustivo permite comprender cómo se ha implementado cada parte del sistema, asegurando que todos los componentes trabajan de manera cohesiva y eficiente para proporcionar una solución tecnológica robusta y escalable. Para una revisión más detallada del código fuente completo, se puede consultar el Anexo A.

6.1. Entorno y Framework

Para comentar tanto el entorno de desarrollo utilizado como el *framework* usado para dar soporte a la implementación del código haremos una distinción entre el Backend y el Frontend, algo obvio teniendo en cuenta que son distintos en cada caso

6.1.1. Backend

Comenzando por el desarrollo del Backend, para este proyecto se ha utilizado Spring Boot, un framework ampliamente reconocido en la comunidad de desarrollo Java por su capacidad para simplificar la creación de aplicaciones web robustas y escalables, que es básicamente lo que perseguíamos en este caso. Como entorno de programación para esta parte he utilizado IntelliJ IDEA [24], la razón de esta decisión es que se trata de una herramienta extremadamente completa además si cabe específicamente para proyectos construidos con Java Spring. El manejo tan sencillo que hace de las dependencias, así como del acceso a la base de datos que tiene; hace que la versión Ultimate de este entorno sea la idónea para la implementación del Backend de este proyecto.

El esquema interno que se ha seguido es estructurar el código en diferentes paquetes lo que facilita el mantenimiento y la extensibilidad del código. Los paquetes que incluyen la información principal de esta parte del proyecto son:

- **`com.tfg.config`**. Paquete que incluye las clases de configuración del proyecto, que establecen los parámetros y ajustes necesarios para la correcta ejecución de la aplicación. Como pueden ser la admisión de peticiones HTTP desde otro dominio o el uso de tokens y encriptación de las contraseñas para la seguridad.

- **`com.tfg.domain`**. Paquete que contiene en su interior varios paquetes, en concreto uno por cada entidad que tenemos. Son las clases que representan los datos y la lógica del negocio. Podríamos decir que es el paquete más importante del proyecto ya que contiene la mayor parte del código. Cada entidad cuenta con distintos componentes clave que son:

    - **`Entity`**. Es la clase que representa la propia entidad de dominio con sus atributos y relaciones con las otras entidades, mapeada a una tabla en la base de datos mediante *Hibernate* [25], que se encarga de generar las tablas en la primera ejecución del código. Usando *ProductoEntity* como ejemplo la implementación sería

algo así:

```
package com.tfg.domain.producto;

import com.tfg.domain.detalle_pedido.DetallePedidoEntity;
import com.tfg.domain.subcategoria.SubcategoriaEntity;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.List;
@Entity
@Table(name = "producto")
@Getter
@Setter
public class ProductoEntity {
    @Id
    private Long id_producto;
    private String descripcion;
    private float precio_venta;
    private float precio_coste;

    @Lob
    @Column(name = "imagen", length = 200000)
    private byte[] imagen;

    @ManyToOne
    @JoinColumn(name = "id_subcategoria")
    private SubcategoriaEntity subcategoria;

    @OneToMany(mappedBy = "producto", cascade = CascadeType.ALL)
    private List<DetallePedidoEntity> detallesPedido;

    }
```

- **`Controller`**. Se avecina necesario que si vamos a contar con una API que gestione peticiones, cada entidad cuente con un controlador propio que sea responsable de manejar las solicitudes HTTP entrantes y definir los *endpoints* relacionados con dicha entidad. La implementación del controlador sigue el siguiente proceso. En primer lugar se documentan los que serán los futuros métodos del propio controlador, los llamados *endpoints*, cada uno con su ruta única. Este es un extracto del archivo *openapi.yml* del proyecto, que contiene toda esta documentación:

```
tags:
  - name: Producto
    description: Operaciones con ProductoEntity
```

```yaml

paths:
 /producto/{id_producto}:
    get:
      summary: "Devuelve un producto por su id"
      operationId: getProductoById
      tags:
        - Producto
      parameters:
        - name: id_producto
          in: path
          required: true
          schema:
            type: integer
            format: int64
      responses:
        '200':
          description: "Operacion correcta"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ProductoInfo'

components:
  schemas:
    ProductoInfo:
      properties:
        id_producto:
          type: integer
          format: int64
        descripcion:
          type: string
        precio_venta:
          type: number
          format: float
        precio_coste:
          type: number
          format: float
        imagen:
          type: string
          format: byte
          description: "Datos de la imagen del producto en
formato byte[]"
        subcategoria:
          $ref: '#/components/schemas/SubcategoriaInfo'
```

En el código anterior podemos ver como se define la ruta y la especificación de un *endpoint* para que dado el identificador único de un producto se devuelva la información de dicho producto. Así como también se incluye la información para generar la clase DTO *(Data Transfers Objects)* de esta entidad, que básicamente es una clase customizada que indica la

estructura del objeto de entrada/salida en las solicitudes. Una vez hecho esto, gracias a las dependencias y plugins de OpenAPI incluidas en el proyecto se generan las clases *ProductoAPI*, que usaremos a continuación y *ProductoInfoDTO* necesarias más adelante. Por último el paso final es crear la clase controladora que implementa la anterior generada.

```
1  package com.tfg.domain.producto;
2
3  import com.tfg.code.api.ProductoApi;
4  import com.tfg.code.model.ProductoInfoDto;
5  import lombok.RequiredArgsConstructor;
6  import org.springframework.http.MediaType;
7  import org.springframework.http.ResponseEntity;
8  import org.springframework.web.bind.annotation.RestController;
9
10 import java.util.List;
11
12 @RestController
13 @RequiredArgsConstructor
14 public class ProductoController implements ProductoApi {
15
16     private final ProductoService productoService;
17
18     @Override
19     public ResponseEntity<ProductoInfoDto> getProductoById(Long idProducto) {
20         ProductoInfoDto result = productoService.getProductoById(idProducto);
21         return ResponseEntity.ok(result);
22     }
23
24     @Override
25     public ResponseEntity<List<ProductoInfoDto>> getProductos() {
26         List<ProductoInfoDto> result = productoService.getProductos();
27         return ResponseEntity.ok(result);
28     }
29 }
```

En el código anterior, siendo este un extracto de la clase *ProductoController*, podemos ver estos métodos implementados. El primero es para obtener un determinado producto a partir de su *id* y el segundo es para obtener todos los productos existentes en la base de datos. Básicamente lo que hacen estos es llamar a una clase servicio que realizará la acción pertinente. Para ello cada clase controladora tiene su par con una clase servicio.

- **`Service`**. Una clase servicio es aquella que implementa la lógica real del negocio, operaciones

básicas como crear, leer, actualizar… operar con los datos que vienen desde la base de datos para mostrarlos al usuario o viceversa. Siguiendo con el esquema de mostrar parte del código para ilustrar esta explicación de la implementación ahora se muestra un extracto de la clase *ProductoService:*

```
1  package com.tfg.domain.producto;
2
3  import com.tfg.code.model.ProductoInfoDto;
4  import com.tfg.domain.subcategoria.SubcategoriaService;
5  import jakarta.transaction.Transactional;
6  import lombok.RequiredArgsConstructor;
7  import org.springframework.stereotype.Service;
8
9  import java.util.List;
10
11 @Service
12 @Transactional
13 @RequiredArgsConstructor
14 public class ProductoService {
15     private final ProductoRepository productoRepository;
16     private final ProductoMapper productoMapper;
17
18     private ProductoInfoDto createInfoDto(ProductoEntity productoEntity){
19         return productoMapper.createInfoDto(productoEntity);
20     }
21
22     public ProductoInfoDto getProductoById(Long idProducto) {
23         return createInfoDto(loadProducto(idProducto));
24     }
25
26     public List<ProductoInfoDto> getProductos() {
27         return
28 productoRepository.findAll().stream().map(this::createInfoDto).toList();
29     }
30 }
```

Cada método en un controlador, tiene su método homólogo en el servicio. Además de estos métodos también cabe destacar que este tipo de clases incluye uno muy necesario que es el que dado una instancia de la entidad te devuelve una instancia del DTO del que hablábamos anteriormente. Especial mención a los atributos de esta clase, una instancia de la clase mapeadora *ProductoMapper* y una instancia de la clase repositorio *ProductoRepository*; comentadas ambas a continuación.

- **`Repository`**. En este caso se trata de una interfaz que extiende *JpaRepository* para interactuar directamente

con la base de datos mediante los métodos que esta incluye. Se nos brinda la posibilidad también de incluir aquellos métodos necesarios para la lógica de nuestro negocio pero no incluidos por defecto para interactuar con las tablas.

```
1  package com.tfg.domain.producto;
2
3  import com.tfg.domain.subcategoria.SubcategoriaEntity;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import java.util.List;
6
7  public interface ProductoRepository extends JpaRepository<ProductoEntity,Long>{
8      List<ProductoEntity> findAllByDescripcionContaining(String key);
9      List<ProductoEntity> findAllBySubcategoria(SubcategoriaEntity
10                                                 subcategoriaEntity);
11 }
```

Como vemos en el código anterior, interfaz completa de *ProductoRepository* se extiende *JpaRepository* [26] indicando que la tabla con la que vamos a interactuar contiene entidades que son productos y que la clave primaria que usamos en dicha tabla es de tipo *Long*. Dicho eso a esta clase se le han añadido dos métodos que requería la lógica de negocio como son el devolver todos aquellos productos que contengan cierta cadena de texto en su descripción, como el de devolver todos aquellos productos que pertenezcan a una subcategoría concreta. *JpaRepository* se encarga de solicitar a la base de datos los datos pertinentes y devolvérnoslos.

- **Mapper**. Por último pero no por ello menos importante, cada entidad de nuestro programa tiene una clase mapeadora que como adelantábamos es la encargada de 'traducir' una entidad a un DTO, ya que la base de datos trabaja con entidades y el frontend utiliza el segundo tipo. Un ejemplo de este tipo de clases es el siguiente:

```
1
2   package com.tfg.domain.producto;
3
4   import com.tfg.code.model.ProductoInfoDto;
5   import com.tfg.domain.subcategoria.SubcategoriaMapper;
6   import lombok.RequiredArgsConstructor;
7   import org.springframework.stereotype.Service;
8
9   @Service
10  @RequiredArgsConstructor
```

```
11  public class ProductoMapper{
12
13      private final SubcategoriaMapper subcategoriaMapper;
14
15      public ProductoInfoDto createInfoDto(ProductoEntity
16  productoEntity){
17          ProductoInfoDto dto = new ProductoInfoDto();
18          dto.setIdProducto(productoEntity.getId_producto());
19          dto.setDescripcion(productoEntity.getDescripcion());
20          dto.setPrecioCoste(productoEntity.getPrecio_coste());
21          dto.setPrecioVenta(productoEntity.getPrecio_venta());
22          dto.setImagen(productoEntity.getImagen());
23          dto.setSubcategoria(subcategoriaMapper.createInfoDto
24  (productoEntity.getSubcategoria()));
25          return dto;
26      }
    }
```

Esta clase únicamente tiene un método y como decíamos su finalidad es crear una instancia con los mismos datos que la pasada como parámetro. Si fuera necesario, como es este caso de *ProductoMapper*, se incluirán como atributo las clases mapeadoras de otras entidades si esta incluye atributos cuyo tipo es otra de las entidades.

- **`com.tfg.exceptions`**. La finalidad de este paquete es contener las clases que manejan las excepciones que pudieran ocurrir durante la ejecución de nuestro programa, que recordemos que se trata de una aplicación web que está corriendo continuamente y para la que la aparición de una excepción que abortase la ejecución podría ser fatal.

- **`com.tfg.init`**. Paquete sencillo a la par que necesario. Este incluye una única clase y la finalidad es la de imitar (de ahí el nombre de la clase, *MockDataInitializer*) datos reales para rellenar la base de datos y así poder primero probar y seguidamente hacer funcionar la aplicación.

- **`com.tfg.security`**. Último paquete pero quizás uno de los más importantes ya que contiene todo los aspectos de seguridad que protegen nuestra aplicación web. El principal objetivo de este es asegurar que solo los usuarios autenticados y autorizados puedan acceder a ciertos recursos del sistema. Este paquete contiene varias clases que trabajan de forma conjunta para proporcionar funcionalidades de autenticación y autorización. Para ilustrar el sistema de seguridad implementado, se va a comentar el proceso paso a paso que sigue un usuario para acceder al sistema y cómo este es

autenticado y autorizado a usarlo. En el primer paso este introduce sus credenciales (nombre de usuario y contraseña) en la página inicial. Esta información se envía al Backend en un DTO, *LoginRequestBody* a través de una solicitud concreta a la API que devuelve otro DTO diferente *UsuarioResult*. Esto es así ya que sería contraproducente una vez chequeado que las credenciales de inicio de sesión son correctas las devolviésemos, exponiéndolas a posibles ataques. En lugar de esto, una vez comprobado que son correctas se crea un token de acceso y este es el que se devuelve al frontend junto con información no sensible para el usuario.

```
LoginRequestBody:
  properties:
    username:
      type: string
    password:
      type: string

  /usuario/login:
    post:
      summary:  Crea un nuevo token de seguridad
      operationId: createSecurityToken
      tags:
        - Usuario
      requestBody:
        description: "Successful operation"
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/LoginRequestBody'
      responses:
        '201':
          description: "Successful operation"
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/UsuarioResult'

    UsuarioResult:
      properties:
        accessToken:
          type: string
        id_tienda:
          type: string
```

El proceso de creación del token corre a cargo de la clase *TokenManager*. La cual genera tokens únicos para los usuarios y los mantiene en una estructura de pares que junta un token con su usuario asociado. Esto permite validar y obtener el nombre de usuario asociado a un token en el futuro

proceso de autenticación.

```java
package com.tfg.security;

import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

@Service
public class TokenManager {
    private final Map<String, String> usernames = new HashMap<>();
    public String createTokenByUsername(String username){
        String token = UUID.randomUUID().toString();
        usernames.put(token,username);
        return token;
    }
    public String getUsernameByToken(String token){
        return usernames.get(token);
    }
}
```

El proceso de creación de un token no siempre se lleva a cabo, es decir, todo *endpoint* que llega a nuestro sistema ya sea este de solicitud de ingreso o algo tan sencillo como pedir los datos de un producto pasa primero un filtro; en concreto por la clase *TokenUserFilter*. Este filtro primero comprueba si la solicitud contiene un token valido, si no es así no se le permite el acceso al sistema, a menos que la solicitud sea para obtener ese preciso token. En este caso se comprueban sus credenciales y si son correctas se le envía al usuario para que este tenga acceso.

```java
package com.tfg.security;
// imports necessaries

@RequiredArgsConstructor
public class TokenUserFilter extends OncePerRequestFilter {

    private final TokenManager tokenManager;
    private final SecurityUserService securityUserService;
    @Override
    protected void doFilterInternal(HttpServletRequest request,
    HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
        String authHeader = "";
        try{
            authHeader = request.getHeader("Authorization");
            String token = authHeader.substring(7);  // Bearer + token
            String username = tokenManager.getUsernameByToken(token);
            UserDetails userDetails =
                        securityUserService.loadUserByUsername(username);
```

```
            UsernamePasswordAuthenticationToken authenticationToken = new
                    UsernamePasswordAuthenticationToken(
                    userDetails, null, userDetails.getAuthorities());
            authenticationToken.setDetails(new WebAuthenticationDetailsSource()
                                                .buildDetails(request));
           SecurityContextHolder.getContext()
                                .setAuthentication(authenticationToken);
        } catch (Exception e){
            logger.warn("Authorization header empty request to obtain it");
            logger.warn(e);
        }
        filterChain.doFilter(request, response);
    }
}
```

Este filtro hace uso de una clase concreta que es *SecurityUserService*, que implementa *UserDetailsService* [27] de Spring Security. Su responsabilidad principal, al tratarse de un servicio como los vistos con anterioridad, es cargar los detalles del usuario desde la base de datos haciendo uso de la clase repositorio pertinente.

```
package com.tfg.security;

import com.tfg.domain.usuario.UsuarioEntity;
import com.tfg.domain.usuario.UsuarioRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class SecurityUserService implements UserDetailsService {

    private final UsuarioRepository usuarioRepository;
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        UsuarioEntity userEntity =
usuarioRepository.findByUsername(username).orElseThrow();
        CustomUserDetails customUserDetails = new CustomUserDetails();
        customUserDetails.setUsername(userEntity.getUsername());
        customUserDetails.setPassword(userEntity.getPassword());
        customUserDetails.setRol(userEntity.getRol());
        return customUserDetails;
    }
}
```

Esta clase anterior convierte la entidad *UsuarioEntity*, manejada por la base de datos, en una instancia de una clase

que Spring Security entienda, como es *CustomUserDetails*, clase que debemos crear implementando *UserDetails* [28] también de Spring Security, encapsulando la información del usuario que se va a usar para su autenticación y autorización. El método que implementa obtiene el nombre de usuario, su contraseña y el rol asociado que tiene.

```
package com.tfg.security;

import lombok.Getter;
import lombok.Setter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import java.util.Collection;
import java.util.List;

@Getter
@Setter
public class CustomUserDetails implements UserDetails {
    private String username;
    private String password;
    private String rol;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        SimpleGrantedAuthority authority= new
SimpleGrantedAuthority("ROLE_" + rol);
        return List.of(authority);
    }
}
```

Este es el proceso de autenticación que siguen las peticiones que se hacen a nuestro sistema. Al hablar de seguridad especial mención a la encriptación y desencriptación al consultar la contraseña en este proceso como en todos los demás, ya en la base de datos se encuentra almacenada de forma segura y no en texto plano. Esto se hace posible gracias al uso de capas de seguridad como la que proporciona *BCryptPasswordEncoder*, configuración incluida junto con la necesaria para el uso de los tokens comentados anteriormente.

```
    @Bean
    public PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }

    @Bean
    public TokenUserFilter tokenUserFilter(TokenManager tokenManager,
    SecurityUserService securityUserService){
        return new TokenUserFilter(tokenManager, securityUserService);
```

```
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity,
    TokenUserFilter filter) throws Exception {
        httpSecurity.cors()
                .and().csrf().disable()
                .exceptionHandling()
                .and().sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and().authorizeRequests()
                .requestMatchers("/usuario/login").permitAll()
                .anyRequest().authenticated();
        httpSecurity.addFilterBefore(filter,
                    UsernamePasswordAuthenticationFilter.class);
        return httpSecurity.build();
    }
```

- **App.java**. Por último, para terminar de hablar sobre la estructura e implementación del Backend del proyecto, aunque lo hagamos de manera anecdótica es preciso comentar que todo programa necesita una clase principal que es la que se ejecuta y hace que todo empiece. En nuestro caso esta clase es muy sencilla y se encarga de iniciar la aplicación de Spring Boot.

```
package com.tfg;

import lombok.extern.log4j.Log4j2;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@Log4j2
public class App {
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
        log.info("Running");
    }
}
```

Dicho todo esto, el código anteriormente incluido es una breve parte de la totalidad, incluido con el mero fin de poder ilustrar la explicación. Para consultas más específicas ver el código fuente completo en el Anexo A Código Fuente.

6.1.2. Frontend

Es turno ahora de hablar del Frontend, de la parte gráfica y visual, aquella con la que interactúa el usuario. Esta parte del proyecto está organizada de manera modular y clara, para así facilitar tanto el desarrollo

como el mantenimiento. A diferencia de la anterior, esta no se sustenta en ningún framework sino que el frontend es básicamente un conjunto de archivos HTML, JavaScript y CSS. La implementación comenzó con la búsqueda y selección de una plantilla web [29][30] que proporcionase un esqueleto básico en cuanto al diseño y tomar esto como punto de partida para adaptarla a las necesidades del proyecto. Esta proporciona una estructura básica de los estilos CSS además de pequeñas animaciones que se han mantenido para la mejor interacción del usuario. Todo el diseño e implementación del frontend han sido adaptados y personalizados completamente a medida para las necesidades que requería este proyecto.

La organización de este se estructura de la siguiente forma:

- **`pages`**. Directorio que contiene las páginas HTML por las que viajará el usuario en su navegación por la interfaz, cada una representa una vista específica con información muy concreta y para una funcionalidad. Entre otras, encontramos:
  - **`home.html`**. Es la página principal, la que visualiza el usuario una vez que accede con sus credenciales. Contiene información básica tanto de usuario como de su tienda.
  - **`almacen.html`**. Página que muestra la información sobre el stock de productos de la tienda de una manera sencilla y clara.
  - **`catalogo.html`**. Página que se utilizará para hacer los pedidos al proveedor, añadiendo productos al carrito y confirmando para añadirlos.
  - **`pedidos.html`**. Esta vista muestra la información de los pedidos de la tienda, tanto de los que ya están completados como de los que no lo están todavía.
- **`assets`**. Directorio que incluye los recursos a utilizar en nuestra interfaz. En concreto a destacar:
  - **`js`**. Subdirectorio que contiene los archivos JavaScript que marcan el funcionamiento de nuestra interfaz. A destacar los directorios **`action`** que contienen archivos JavaScript, uno por cada página HTML que tiene nuestra interfaz, con las funciones que dicha página tiene; y el directorio **`logic`** que contiene un archivo JavaScript distinto para cada entidad que tiene nuestro Backend, con las funciones que hacen las peticiones HTTP y que se usarán para obtener los datos.

- **`css`**. Subdirectorio con las hojas de estilos CSS que le dan el aspecto a nuestra interfaz.

- **`index.html`**. Página inicial que verá el usuario al conectarse a nuestro sistema. Consiste en una pantalla inicio de sesión en la que se introducen las credenciales y se da inicio a la navegación por la web.

Para desplegar la aplicación localmente durante el desarrollo, se ha utilizado Fenix Web Server [31]. Esta herramienta permite servir los archivos del frontend en un entorno localhost, facilitando las pruebas y el desarrollo continuo de la aplicación. Podríamos decir que la estructura del frontend está diseñada para ser intuitiva y eficiente, con un enfoque modular que facilita tanto el desarrollo como el mantenimiento.

A la hora de implementar el frontend de este proyecto siempre he tenido presente los principios de un diseño *responsive*, garantizando una experiencia de usuario óptima en una amplia variedad de dispositivos, desde móviles y tabletas hasta monitores de escritorio de distinto tamaño. Para lograr esta adaptabilidad, se han utilizado diversas técnicas y herramientas. En primer lugar, se hace uso de una rejilla fluida que permite que los elementos de la página web se dimensionen proporcionalmente en función del tamaño de la pantalla ya que usan unidades relativas en lugar de medidas fijas. Además, las imágenes flexibles aseguran que se ajusten automáticamente al contenedor en el que se encuentran, evitando desbordamientos y manteniendo la proporción adecuada. El uso de *media queries* en CSS ha sido crucial para aplicar estilos específicos según el tamaño del dispositivo. Estas *media queries* permiten modificar el diseño y la disposición de los elementos para garantizar una visualización coherente y atractiva en diferentes resoluciones de pantalla.

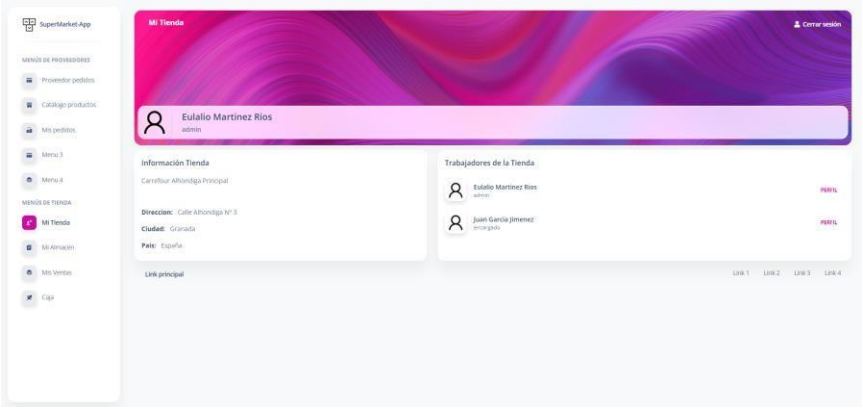

Ilustración 42: Vista Pagina Web en monitor estándar

En la ilustración 42 podemos ver cómo se visualiza la página de información de nuestra tienda en un monitor estándar de sobremesa con una resolución de 1920x1080. Al tener ser una pantalla amplia se muestran

todos los elementos que la página contiene. Sin embargo si nos fijamos en la Ilustración 43 y la Ilustración 44, pertenecientes ambas a dispositivos con distintos tamaños de pantalla, el diseño se adapta para mostrar la información de una manera elegante. El menú lateral se esconde, haciendo aparecer un botón en la parte superior derecha que lo hace aparecer para navegar por las distintas páginas.

Ilustración 43: Vista Página Web en dispositivo móvil Samsung Galaxy S20

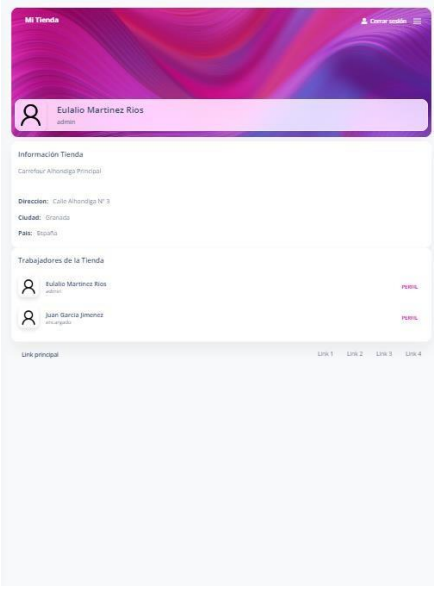

Ilustración 44: Vista Página Web en iPad Pro posición vertical

La Ilustración 45 pertenece al mismo dispositivo que la ilustración anterior pero al estar este en posición horizontal, hay más espacio visual que el diseño detecta y hace aparecer automáticamente el menú desplegable.

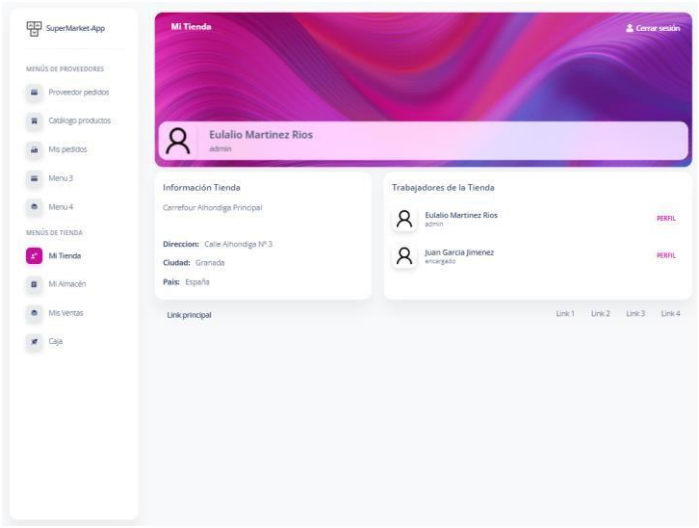

Ilustración 45: Vista Página Web en iPad Pro posición horizontal

6.2. Sistema Gestor de Base de Datos

En este apartado se comentará y describirá el Sistema Gestor de Base de Datos (SGBD) utilizado en el proyecto, sus características, ventajas y cómo se integra con la arquitectura e implementación general de la aplicación. Como ya sabemos, aunque no está de mal recordar, un Sistema Gestor de Base de Datos es un conjunto de programas que permiten la creación, gestión y manipulación de bases de datos. Estos sistemas proporcionan una manera eficiente de almacenar, recuperar y gestionar datos, garantizando la integridad y seguridad de la información.

En este proyecto, se ha utilizado MySQL [32] como el SGBD principal. MySQL es uno de los sistemas de bases de datos relacionales más populares y ampliamente utilizados, conocido por su robustez, facilidad de uso y rendimiento eficiente. En el contexto de este proyecto, MySQL se integra de la siguiente manera. Primero durante el desarrollo local, la aplicación Spring se conecta a una instancia de MySQL que se ejecuta en el entorno de *localhost*. La configuración de conexión a la base de datos se realiza mediante el archivo `application.properties` del proyecto, donde se especifican los detalles de la conexión como la URL, el usuario, o la contraseña. Una vez que el proyecto está listo para ser desplegado, se utilizará *Google Cloud App Engine* para alojar la aplicación. La base de datos MySQL es migrada a Google Cloud SQL, un servicio de base de datos gestionado que facilita la configuración, mantenimiento y escalabilidad de las bases de datos en la nube. La conexión entre la aplicación desplegada en App Engine y Google Cloud SQL se configura mediante parámetros de conexión seguros, asegurando un rendimiento óptimo y una alta

disponibilidad. Para más detalles sobre el despliegue leer el apartado 6.4 Despliegue del presente capítulo.

MySQL proporciona al proyecto un entorno fiable y escalable para el almacenamiento de datos, mientras que el uso de JPA Repository simplifica la interacción con la base de datos, permitiéndome como desarrollador centrarme en la lógica de negocio sin preocuparse por los detalles de las consultas SQL. Esta combinación asegura una integración fluida y una gestión eficiente de los datos en el sistema.

6.3. Lenguajes utilizados

El desarrollo de este proyecto ha requerido el uso de varios lenguajes de programación, como ya adelantábamos en el apartado anterior, para cubrir tanto el Backend como el Frontend, asegurando así una aplicación robusta y funcional. A continuación, se describen los lenguajes utilizados y su papel en el desarrollo del proyecto. Para el Backend de la aplicación, se ha utilizado Java, en concreto su versión 17 [33]. Este es un lenguaje de programación robusto y ampliamente utilizado en el desarrollo de aplicaciones empresariales debido a su escalabilidad, seguridad y gran comunidad de soporte. En este proyecto, Java ha sido fundamental para la implementación de la lógica de negocio, la gestión de datos y la interacción con la base de datos mediante el framework Spring Boot.

Si pasamos a hablar del Frontend de la aplicación, este ha sido desarrollado utilizando HTML *(HyperText Markup Language)* , CSS *(Cascading Style Sheets)* y JavaScript. Estos lenguajes son los pilares fundamentales del desarrollo web, proporcionando la estructura, el estilo y la interactividad necesarios para una experiencia de usuario atractiva y funcional. El primero de ellos ha utilizado para definir la estructura y el contenido de las páginas web. HTML permite organizar el contenido de manera semántica, facilitando tanto la lectura como el mantenimiento del código. El aspecto de diseñar la apariencia visual de la aplicación se hace a través de CSS, se han definido los estilos, colores, fuentes y disposición de los elementos en la página, garantizando una experiencia de usuario coherente y atractiva. Y por último JavaScript, lenguaje de programación que ha sido esencial para añadir interactividad y dinamismo a la aplicación. JavaScript nos permite la manipulación del DOM *(Document Object Model)*, la validación de formularios en el cliente, y la realización de peticiones HTTP asíncronas a la API del Backend, mejorando la capacidad de respuesta y la fluidez de la aplicación. Anecdóticamente comentar que se incluye también algo de SCSS *(Sassy CSS)*, un preprocesador de CSS que facilita la escritura de estilos más organizados y reutilizables.

Para dimensionar el proyecto quiero incluir algunos datos a modo de estadísticas que reflejen el tamaño del presente proyecto de implementación.

Se presentan, pues, métricas del código, proporcionando una visión cuantitativa de su magnitud y complejidad. Estas estadísticas incluyen el número total de líneas de código y la distribución de este entre Backend y Frontend. Conteo realizado con la extensión *VS Code Counter* [34].

En el Backend nos encontramos ante un proyecto que consta de 284 archivos. Estos contienen más de 18.000 líneas de código, casi todas ellas pertenecientes al lenguaje principal de este módulo, lenguaje Java. El resto de líneas de código son casi anecdóticas ya que pertenecen a pocos archivos de documentación para las clases generadas, para rellenar la base de datos inicialmente o para la configuración del proyecto y del despliegue.

```
Directory: c:\Users\Eulalio\TFG\backend
Total: 284 files, 16320 codes, 918 comments, 1735 blanks, all 18973 lines
```

language	files	code	comment	blank	total
Java	260	9,423	918	1,502	11,843
YAML	3	2,960	0	198	3,158
JSON	2	2,778	0	2	2,780
XML	15	1,141	0	22	1,163
Java Properties	3	11	0	3	14
Markdown	1	7	0	8	15

Si pasamos a hablar del frontend estaríamos manejando las cifras de un proyecto de 242 archivos con un total de 44.100 líneas de código. Heredadas la mayoría de la plantilla de estilo utilizada para este proyecto; reflejada en la cantidad de líneas de código de CSS y de SCSS. Punto importante los casi 50 archivos HTML y JavaScript que componen el grueso de este proyecto.

```
Directory: c:\Users\Eulalio\TFG\frontend
Total: 242 files, 44169 codes, 2633 comments, 8531 blanks, all 55333 lines
```

language	files	code	comment	blank	total
CSS	4	21,881	261	5,407	27,549
SCSS	169	14,105	1,805	2,556	18,466
JavaScript	36	4,367	333	503	5,203
HTML	12	3,174	230	59	3,463
XML	19	630	0	2	632
YAML	1	11	4	4	19
Markdown	1	1	0	0	1

Pese a que el número de archivos y líneas de código no refleja si un proyecto es de calidad o no. Me parece interesante incluir este breve análisis de las estadísticas del código para proporcionar una visión clara de la

magnitud y complejidad del proyecto. Con un total de algo más de 74.000 líneas de código.

6.4. Despliegue

Una vez comentada toda la parte de la implementación el siguiente paso es la apertura al público de la aplicación es decir, subirla a un servidor para que los usuarios puedan acceder a ella. Estamos hablando del despliegue de la aplicación. Se ha realizado en la plataforma Google Cloud Platform (GCP) [35]. Se ha elegido Google Cloud Platform para el despliegue de esta aplicación por varias razones. GCP ofrece una infraestructura escalable y servicios gestionados que permiten una administración eficiente y segura de los recursos. Además, proporciona herramientas de monitoreo y seguridad avanzadas, que son esenciales para mantener la disponibilidad y el rendimiento del sistema en producción. Hecho importante y clave en esta decisión es que la integración de servicios como App Engine y Cloud SQL simplifica considerablemente el desarrollo y despliegue de aplicaciones, permitiéndome como desarrollador centrarme más en el código y menos en la infraestructura. La forma de proceder ha sido desplegar el sistema en dos proyectos independientes para una gestión más eficiente y segura de los recursos. El proyecto *"supermercado-app"* alberga el componente Backend y la Base de Datos, mientras que el proyecto *"supermercado-ui"* se encarga del despliegue del Frontend.

A continuación, se detalla el proceso de despliegue y configuración de cada proyecto, aprovechando la infraestructura escalable y servicios gestionados de GCP para garantizar la disponibilidad y rendimiento del sistema en producción.

6.4.1. Proyecto Backend y BD

El componente Backend de la aplicación junto con la base de datos relacional se han desplegado en un proyecto GCP denominado *"supermercado-app"*, este proyecto sirve de contenedor principal para todos los recursos relacionados con la parte posterior del sistema. El Backend se ha alojado en el servicio App Engine [36], que proporciona un entorno de ejecución gestionado para aplicaciones web y APIs, encargándose de la escalabilidad automática y balanceo de carga. El primer paso, y no se nos puede olvidar es crear la instancia de Cloud SQL, nuestra base de datos, ya que las credenciales de acceso a esta las tendremos que incluir en el archivo de configuración de nuestro proyecto antes de desplegarlo. Para este proyecto opté por una con 10 GB de almacenamiento, 16 GB de RAM y 2 CPU’s virtuales. En este instante estamos olvidándonos de nuestro almacenaje local para pasar a uno semejante pero en la nube.

Una vez hecho esto se aseguraron las dependencias, se empaquetó de la forma correcta y se incluyó el archivo `app.yaml` pertinente especificando el entorno y lenguaje en el que se ejecutará la aplicación así como la instancia que se utilizará.

```
1 runtime: java17
2 instance_class: F2
```

He optado por una instancia de tipo F2 ya que proporciona más recursos en comparación con las instancias básicas, lo que mejora el rendimiento de la aplicación. Una vez hecho esto el siguiente paso es instalar Google Cloud SDK [37] en nuestra maquina local, herramienta de línea de comandos que nos permitirá subir nuestro proyecto local a la nube. Una vez instalado, en la consola y desde el directorio raíz; con un simple comando como `gcloud app deploy` el proyecto se despliega en la nube. De este despliegue obtenemos un enlace para acceder a nuestro proyecto que ya se encuentra corriendo, enlace que por sí solo, ahora mismo, no tiene mas utilidad que probar que el despliegue ha sido exitoso, haciendo peticiones a la API con una herramienta como POSTMAN [38].

6.4.2. Proyecto Frontend

El despliegue del proyecto Frontend comienza por modificar las URL's de las peticiones que se hacen a la Backend ya que las dejamos en `http://localhost:8080` , estas estarían apuntando a nuestro proyecto local. Es en este momento cuando usamos la URL que obtuvimos en el despliegue anterior, quedando por ejemplo de la siguiente forma:

```
1 const baseUrl = 'https://supermercado-app-425815.ew.r.appspot.com';
2
3 export function getPedido(pedidoId) {
4     console.log("See pedido by id: " + pedidoId);
5     return fetch(baseUrl + '/pedido/' + pedidoId, {
6         method: 'GET',
7         headers: {
8             "Authorization": "Bearer " + localStorage.getItem("auth_token")
9         }
10     })
11         .then((response) => {
12             if (!response.ok) {
13                 throw new Error('Network response was not ok');
14             }
15             return response.json();
16         })
17         .then(json => {
18             return json; // Devuelve el JSON obtenido
19         })
20         .catch(error => {
21             console.error('There has been a problem with your fetch operation:', error);
22             throw error;
```

```
23         });
24 }
```

Una vez hemos redirigido las peticiones modificando los archivos JavaScript pertinentes, de forma similar a como hicimos en el proyecto anterior es turno de crear un archivo `app.yaml` para especificar tanto el lenguaje que usará App Engine como la forma en la que se servirán las distintas rutas de nuestro proyecto. Esto lo hacemos incluyendo en este archivo distintos manejadores para las distintas rutas:

```
 1 runtime: python39
 2
 3 handlers:
 4   # Handler para servir archivos estáticos desde el directorio 'assets'
 5   - url: /assets
 6     static_dir: assets
 7
 8   # Handler para servir archivos estáticos desde el directorio 'pages'
 9   - url: /pages
10     static_dir: pages
11
12   # Handler para la ruta por defecto, que servirá 'index.html'
13   - url: /
14     static_files: index.html
15     upload: index.html
16
17   # Handler por defecto para todas las demás rutas
18   - url: /.*
19     script: auto
```

Esta parte frontal y visible del proyecto se ha desplegado en un proyecto GCP independiente denominado *"supermercado-ui"*, incluido en otro servicio App Engine similar al anterior. Una vez hechos estos pasos podemos decir que el proyecto ha sido desplegado y que podemos acceder al sistema accediendo a la URL que se nos facilita al desplegar el proyecto frontend.

Capítulo 7
Pruebas y Validación

La etapa de pruebas y validación es crucial en el desarrollo de software, ya que garantiza que el sistema funciona como se espera y cumple con los requisitos especificados. Cabe destacar que como se mencionó en el Capítulo 3 Planificación la fase de pruebas y correcciones es algo extraña ya que no es estrictamente una etapa que se inicia cuando acaba la anterior sino que al seguir una metodología en la que se prueba aquello que se implementa; esta etapa está un poco presente en todas. Por ello en este capítulo se describen las pruebas funcionales desarrolladas durante la fase de implementación (véase apartado 7.1), como las distintas pruebas que se llevaron a cabo, estas sí, una vez que el sistema es desplegado (véase apartados 7.2 y 7.3).

7.1. Pruebas de Funcionalidad

Este primer apartado se centra en las diferentes pruebas de funcionalidad realizadas para poder garantizar el correcto funcionamiento y la calidad del sistema. Estas pruebas incluyen diferentes tipos, ya que cada una de ellas está enfocada a validar aspectos específicos del sistema.

7.1.1. Pruebas Unitarias

El grueso de nuestro sistema es el Backend y este en concreto no cuenta con una lógica muy particular, es decir, el tratamiento que se hace

con los datos es sencillo. Se almacenan datos, y estos son consultados y editados. Esto no quiere decir que no se deban hacer test ni pruebas, todo lo contrario. Durante la implementación de esta parte del proyecto en Spring Boot, se desarrollaron pruebas unitarias para cada componente que forma el grueso de nuestra aplicación, el dominio; incluyendo entidades, controladores, mappers, servicios y repositorios. Estas pruebas, realizadas con la anotación `@SpringBootTest`, permitieron verificar el comportamiento individual de cada componente de forma aislada, asegurando que cumplen con los requisitos funcionales y no introducen errores en el sistema.

El objetivo de incluir estas pruebas es alcanzar el 100% de cobertura en los componentes mencionados, para validar la funcionalidad de los componentes individuales del sistema. Un ejemplo sería el siguiente método que se incluye en esta clase `TiendaControllerTest.java` que testea el método concreto de crear una tienda, de la clase `TiendaController.java`. Se procede de igual forma con cada método distinto.

```java
package com.tfg.domain.tienda;

import com.tfg.code.model.TiendaInfoDto;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;

@SpringBootTest
public class TiendaControllerTest {
    @Autowired
    private TiendaController tiendaController;

    @Test
    public void testCreateTienda() {
        TiendaInfoDto tiendaInfoDto = new TiendaInfoDto();
        tiendaInfoDto.setIdTienda("id de Ejemplo");
        tiendaInfoDto.setDescripcion("Tienda de ejemplo");
        tiendaInfoDto.setDireccion("Calle Ejemplo, 123");
        tiendaInfoDto.setCiudad("Ciudad Ejemplo");
        tiendaInfoDto.setPais("Pais Ejemplo");

        ResponseEntity<TiendaInfoDto> responseEntity =
                                    tiendaController.createTienda(tiendaInfoDto);

        assertEquals(HttpStatus.CREATED, responseEntity.getStatusCode());
        assertNotNull(responseEntity.getBody());
        assertEquals("Tienda de ejemplo", responseEntity.getBody().getDescripcion());
        assertEquals("Calle Ejemplo, 123", responseEntity.getBody().getDireccion());
        assertEquals("Ciudad Ejemplo", responseEntity.getBody().getCiudad());
```

```
34          assertEquals("Pais Ejemplo", responseEntity.getBody().getPais());
35          tiendaController.deteleTienda("id de Ejemplo");
36      }
```

7.1.2. Pruebas de Integración

Otro tipo de pruebas que se han llevado a cabo son las de integración. Se han realizaron pruebas de integración para garantizar la correcta interacción entre los distintos componentes que forman el sistema. Se verificó la comunicación entre módulos, servicios y bases de datos, así como las llamadas a la APIs, utilizando como principal herramienta Postman para identificar y corregir posibles errores de integración.

7.1.3. Pruebas Manuales

Continuando con otras pruebas diferentes las siguientes son las pruebas manuales, que fueron esenciales para validar la experiencia del usuario y asegurar que la funcionalidad de la aplicación cumpliera con las expectativas. Se simularon interacciones reales con las distintas interfaces y en diferentes dispositivos, para ver cómo se mostraban en cada uno así como en navegadores distintos, buscando identificar posibles problemas de usabilidad o errores no detectados en pruebas automatizadas.

7.1.4. Pruebas de Navegación y Flujo del Usuario

Por último, en cuanto a las pruebas funcionales, se evaluó la navegación por la propia aplicación, verificando que los flujos de usuario fueran intuitivos y coherentes. Se ha prestado especial atención a la facilidad de uso y a la eficiencia en la realización de tareas comunes, ya que es uno de los objetivos que perseguía este proyecto, buscando reducir la curva de aprendizaje del usuario en todo momento.

Podría concluir diciendo que estas pruebas permitieron validar la funcionalidad y usabilidad de la aplicación, asegurando que el sistema cumple con los requisitos establecidos y ofrece una experiencia satisfactoria a los usuarios.

7.2. Pruebas de Rendimiento

Una de las pruebas más importantes tras el despliegue es la de rendimiento ya que demuestra la capacidad que tiene el sistema para manejar cargas de trabajo elevadas. Se ha intentado simular un escenario de prueba en el que muchos usuarios acceden al sistema, para realizar distintas operaciones en él, estresándolo y viendo su comportamiento con distintas métricas. Para ello se ha empleado Apache JMeter [39], herramienta idónea para este caso ya que simula múltiples solicitudes al servidor con el fin de simular el escenario que buscado.

Los parámetros de configuración de la prueba se establecieron para ir incrementando gradualmente el numero de usuarios virtuales y así poder generar una carga más significativa.

- Número total de peticiones : 100.000
- Usuarios concurrentes (N.º de hilos): 1.000
- Periodo de subida en rampa: 10 segundos
- Duración de la prueba: 5 minutos

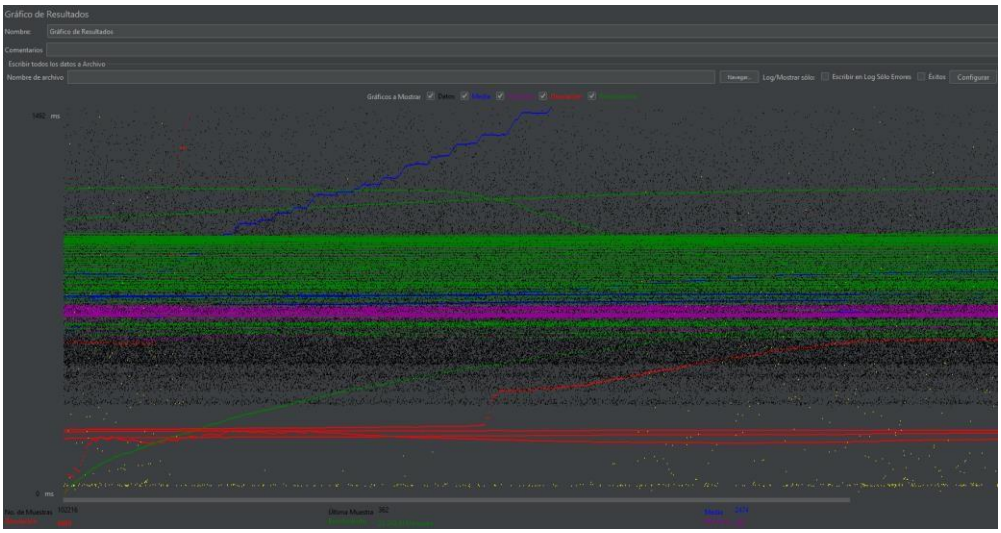

Ilustración 46: Resultado prueba rendimiento

Tras configurar la prueba y lanzarla los resultados que esta mostró fueron que el sistema desplegado es capaz de manejar un throughput de 23.742 peticiones por minuto o lo que es lo mismo 395 peticiones por segundo, nuestro sistema es capaz de manejar una carga bastante considerable. Comentando los demás resultados tenemos que la media del tiempo de respuesta es de 2474 ms, es decir, en promedio cada transacción tarda alrededor de 2,4 segundos en completarse; lo que nos confirma que el

sistema cumple con algo que buscaba tener, y es un alto rendimiento con tiempos de respuesta bajos como se especificaba en el FNR 02. Terminando de comentar los valores arrojados, la desviación estándar se sitúa en 6608 ms, lo que nos resulta como una desviación en los tiempos de respuesta de 6 segundos algo que no sale de los parámetros esperados y que como se refleja en la siguiente ilustración, los tiempos de respuesta están en un rango de entre 1 y 6 segundos.

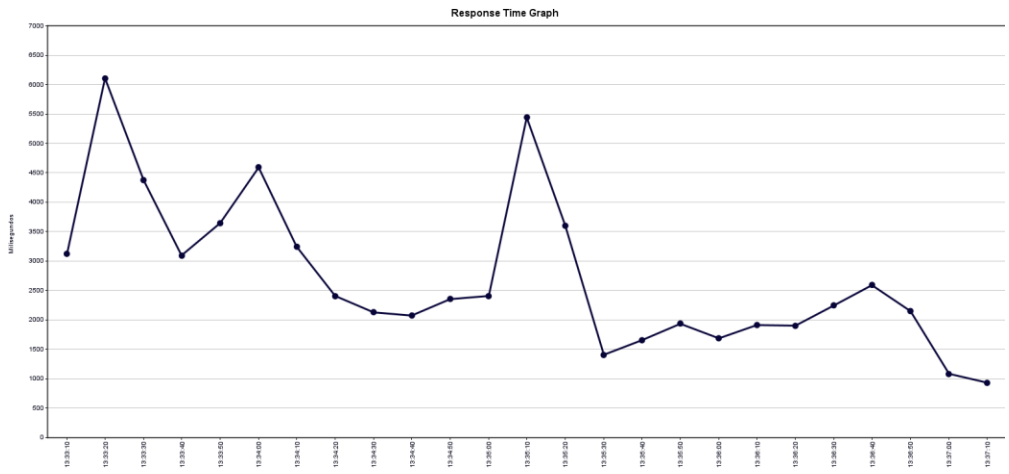

Ilustración 47: Gráfico tiempos de respuesta en prueba de rendimiento

Estos resultados recogidos reflejan que el sistema es robusto y que es capaz de manejar cargas de usuarios elevadas. Algo que se buscaba al escoger Google App Engine para el despliegue, ya que una de sus principales características es que ofrece robustez, escalabilidad y alto rendimiento, permitiendo que la aplicación maneje eficientemente grandes volúmenes de tráfico y garantizando así una experiencia de usuario óptima incluso bajo condiciones de carga intensa.

7.3. Pruebas de Usabilidad

Una vez desplegado, para evaluar adecuadamente el sistema implementado, se han llevado a cabo pruebas de usabilidad con un diverso grupo de trabajadores del supermercado, con distintos niveles en cuanto al manejo de la tecnología se refiere. Estos empleados representan a los usuarios finales del sistema y proporcionan una valiosa retroalimentación sobre la funcionalidad y facilidad de uso de nuestro sistema desarrollado. Antes de comenzar las pruebas, se proporcionó a cada participante una copia del Manual de Usuario, incluido y detallado en el Anexo B Manual de Usuario. Este documento les ofreció una guía clara sobre cómo interactuar con el sistema, incluyendo capturas de pantalla y descripciones de las funcionalidades principales. Una vez terminada esta tarea previa, el siguiente paso era asignarles a cada trabajador tareas sencillas y

representativas de las operaciones diarias que se hacen en el establecimiento. Estas tareas incluyeron la comprobación de stock, solicitar un nuevo pedido al proveedor, registrar dicho pedido, creación de una nueva venta. El objetivo era evaluar si los usuarios podían completar estas tareas con éxito utilizando el sistema.

Durante estas sesiones de prueba, se observó a los participantes para poder identificar cualquier dificultad o problema que les surgiera. Se registraron métricas como el tiempo necesario para completar cada tarea y los errores cometidos. Además, se realizaron entrevistas individuales posteriores a las pruebas para recopilar opiniones subjetivas sobre la facilidad de uso y la eficiencia del sistema. Los resultados de las pruebas de usabilidad fueron muy positivos en general. Los trabajadores lograron completar las tareas asignadas sin mayores inconvenientes. El tiempo requerido para realizar cada tarea fue diverso entre los participantes pero se mantuvo dentro de los parámetros esperados, indicando que el sistema es intuitivo y fácil de usar.

Participante	**Edad (años)**	**Nivel tecn.**	**Tarea**	**Tiempo (min:seg)**
Trabajador 1	22	medio	Comprobación stock	1:57
			Hacer pedido	5:23
			Registrar pedido	9:12
			Nueva venta	4:46
Trabajador 2	47	bajo	Comprobación stock	2:49
			Hacer pedido	7:12
			Registrar pedido	12:32
			Nueva venta	4:05
Trabajador 3	33	medio	Comprobación stock	2:02
			Hacer pedido	5:45
			Registrar pedido	7:55
			Nueva venta	5:23
Encargado	25	alto	Comprobación stock	1:27
			Hacer pedido	3:04
			Registrar pedido	7:33
			Nueva venta	4:34

Tabla 57: Resultados pruebas usabilidad

Tras estas pruebas, a modo de comparación se les pidió a los sujetos que intentaran realizar estas tareas con el sistema anterior, el que el supermercado usa. El resultado fue que solamente el encargado supo navegar por el propio sistema y avanzar más allá de la pantalla inicial. Esta es una clara representación de la mejora en usabilidad que este sistema ofrece. Una de las sugerencias más recurrentes fue la recomendación de incorporar tutoriales interactivos, un entorno donde practicar en un entorno de pruebas; dentro del sistema para ayudar a los nuevos usuarios a familiarizarse con las funcionalidades más rápidamente. Podríamos decir entonces que las pruebas de usabilidad demostraron que el sistema es efectivo y accesible para los trabajadores del supermercado, tengan el nivel de manejo en la tecnología que tengan. Además los comentarios y observaciones obtenidos proporcionaron información valiosa que se utilizará para realizar mejoras futuras en el sistema, asegurando que satisfaga las necesidades de sus usuarios finales.

Capítulo 8
Conclusiones

En este capítulo final se presentan las conclusiones y reflexiones finales del proyecto. Aquí se revisan los objetivos planteados al inicio para comentar si se han logrado o no (véase apartado 8.1), también se reflexiona sobre el aprendizaje y desarrollo personal obtenido a lo largo del proyecto (véase apartado 8.2), y finalmente se exploran posibles futuras acciones para mejorar y expandir las capacidades del sistema (véase apartado 8.3). Este capítulo ofrece una visión completa de los logros alcanzados, las lecciones aprendidas y las oportunidades que se presentan para el futuro.

8.1. Objetivos alcanzados

Es momento de volver la vista atrás y recordar los objetivos planteados que se presentaron en el Capítulo 1. Una vez hecho el trabajo debemos comprobar si se ha conseguido aquello que se propuso. El primero y más importante de todos aquellos objetivos era el de integrar y optimizar un sistema en una plataforma web. Puedo decir que este se ha alcanzado con éxito ya que en estos momentos se tiene un sistema funcional que cumple las especificaciones iniciales; integrado en una plataforma web desplegado en la nube y accesible por el usuario.

En siguiente lugar nos encontramos la meta de lograr la automatización en la gestión del inventario y de las fechas de caducidad. Este era uno de los grandes problemas que tenía el sistema anterior y para lograrlo nuestro sistema cuenta con una lógica que trata las fechas de caducidad y las muestra para que el usuario sea consciente de la situación.

Esta automatización reduce significativamente los errores humanos y mejora la precisión en la gestión del propio inventario. El siguiente objetivo cumplido, es, y cito textualmente al cliente del producto tras ver el resultado del proyecto, “uno de los más importantes y que más importante le parecía” ya que le libera a él personalmente de muchísimo trabajo. Se trata de la mejora de la usabilidad y de la experiencia de usuario. Yo coincido completamente con él, ya que el salto dado es abismal. La interfaz de usuario se diseñó para ser intuitiva a la vez que amigable, reduciendo la curva de aprendizaje a niveles muy básicos. Las pruebas realizadas entre algunos trabajadores de la tienda reflejan una experiencia de usuario más fluida y agradable, cumpliendo así con el objetivo marcado.

El siguiente objetivo que se marcó fue el de implementar medidas de seguridad, y creo firmemente que se han establecido protocolos robustos de autenticación y autorización así como de encriptación de los datos sensibles que brindan la consecución de este objetivo. Por último no nos podemos olvidar del objetivo de diseñar un sistema escalable y adaptable, principio que se ha tenido en cuenta en todo momento. Empleando para ello tecnologías y arquitecturas que permiten un fácil mantenimiento del sistema a largo plazo así como la adición de futuros módulos y funcionalidades. Objetivo cumplido que quedó demostrado y argumentado en las pruebas de rendimiento. La adaptabilidad del sistema a distintos navegadores web y tipos de pantallas también se ha conseguido gracias al diseño responsive. Hilando un poco con este tema, en cuanto a posibles acciones futuras, a continuación, en el apartado “6.3 Futuras acciones” se comentan algunas de ellas.

8.2. Aprendizaje y reflexión personal

Desde el primer momento de la concepción inicial del proyecto hasta ahora, que ya ha pasado la implementación técnica y las pruebas finales, pienso firmemente que cada etapa ha contribuido significativamente a mi desarrollo profesional. La experiencia de trabajar en el desarrollo web de una aplicación ha sido enriquecedora y formativa, ofreciéndome una serie de aprendizajes muy valiosos.

Trabajar en un entorno, lo más real posible ya que se trata de una simulación al fin y al cabo, me ha proporcionado un punto de vista diferente que lo vivido en el ámbito académico durante estos años de estudio. La oportunidad de abordar un problema real, que existe a día de hoy y aportar una idea propia que nace desde mi punto de vista me ha permitido desarrollarme tanto profesional como personalmente. He fortalecido mi comprensión sobre cómo llevar soluciones tecnológicas desde la planificación hasta una implementación funcional y real.

Si hablamos del aspecto académico, este proyecto me ha servido como un puente para relacionar diversos campos de la ingeniería informática; arquitectura de software, experiencia del usuario, metodología de programación y de trabajo… Me enorgullece decir que he ganado una visión más clara de cómo se pueden fusionar diferentes disciplinas para producir soluciones efectivas y prácticas. Además, creo que he aprendido a valorar la importancia de la interdisciplinariedad en el desarrollo de aplicaciones complejas, algo que quizás al principio de mis estudio o durante los mismos dejaba de lado.

La metodología de trabajo seguida en este proyecto fue una lección valiosa para mí. Me ha enseñado la importancia de la adaptabilidad y la agilidad en un entorno tecnológico que está en constante cambio. La retroalimentación continua con el cliente y los ajustes constantes fueron cruciales para superar los desafíos que surgieron. Este enfoque me ha permitido mejorar mi capacidad para gestionar cambios y responder de manera eficiente a las necesidades del proyecto. No hay prueba que refleje más esto que hacer una comparación del Diagrama de Gantt que se presentó en capítulos anteriores en la planificación y un Diagrama de Gantt realizado a posteriori.

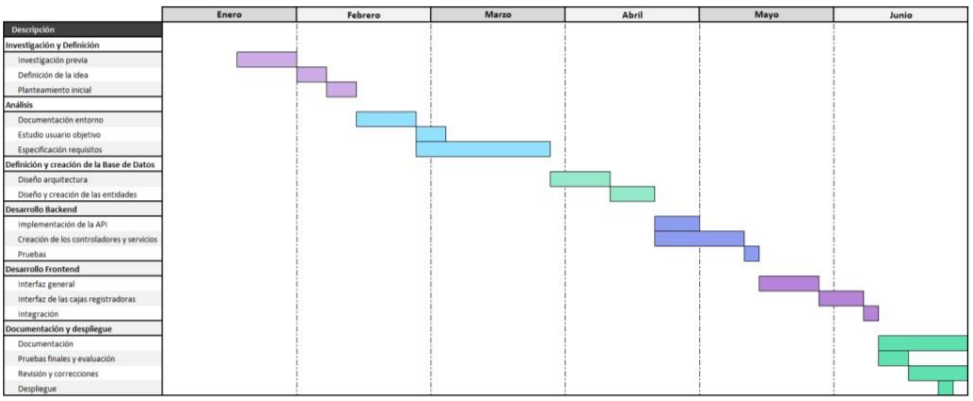

Ilustración 48: Diagrama de Gantt (Previsión)

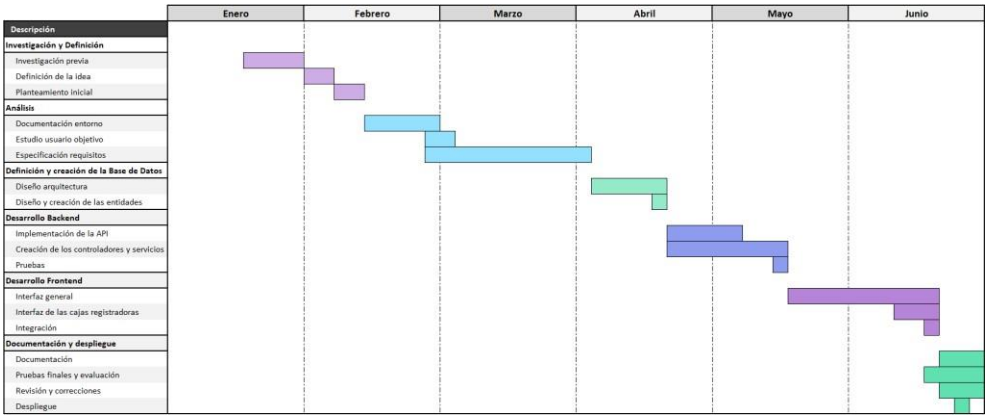

Ilustración 49: Diagrama de Gantt (Real)

Esta comparativa destaca las diferencias entre la planificación y la realidad del desarrollo del proyecto. Como podemos ver ambos diagramas son similares, pero ofrecen sutiles diferencias que es de recibo comentar y analizar. La clave entre una y otra está en la diferencia de duración que algunas tareas han tenido; debido a la complejidad no anticipada y la falta de bagaje en un entorno como este varias de ellas no se han completado en el tiempo planificado. Personalmente destacaría la implementación de las interfaces, ya que es la tarea que más alejada tiene su previsión con respecto a su duración real. Cabe destacar que esta experiencia me ha enseñado la importancia de la flexibilidad así como de realizar una buena planificación y la necesidad de estar preparado para ajustar las expectativas y los plazos conforme se desarrolla el proyecto.

8.3. Futuras acciones

El presente proyecto, pese a ser uno completo, puede ser la primera piedra algo mayor y más ambicioso. A medida que desarrollaba este sistema web para la gestión de supermercado, he sido testigo de cómo surgían diversas oportunidades y áreas de mejora que pueden ser abordadas en el futuro. Creo fielmente que estas futuras acciones tienen el potencial de ampliar las capacidades actuales del propio sistema, mejorar la eficiencia del trabajo que se realiza en los establecimientos y ofrecer una mayor variedad de herramientas útiles para los usuarios.

Una de las principales áreas de posible expansión es la incorporación de funcionalidades para la gestión económica del supermercado, especialmente en relación con los sueldos y salarios de los empleados; así como de los impuestos. Sería interesante estudiar el desarrollar un módulo de nóminas que permitiera a los encargados del supermercado gestionar todos los aspectos relacionados con el pago de las nóminas de los empleados, incluyendo el cálculo automático de estas, deducciones fiscales,

impuestos o beneficios varios. Esta funcionalidad no solo mejoraría la eficiencia operativa sino que también centralizaría la gestión financiera en una única plataforma ya que actualmente esta función se lleva a cabo con otro software distinto y esto ayudaría a simplificar la situación. También creo que el sistema desarrollado podría beneficiarse de la adición de más funcionalidades específicas propias del día a día de un supermercado, como puede ser la implementación de un sistema de fidelización de clientes ya que esto podría aumentar la satisfacción y la lealtad del cliente, proporcionando incentivos y recompensas basados en las compras y la frecuencia en sus visitas.

Otra futura acción clave sería la integración de herramientas avanzadas de estadísticas y análisis de datos. Estas funcionalidades permitirían a los propietarios generar informes detallados sobre diversas métricas, como el rendimiento de ventas, las mermas, los márgenes, la rotación de inventarios y las tendencias actuales de compra de los clientes. Ser capaces de aprovechar y analizar estos datos de manera efectiva ayudaría a tomar decisiones más informadas y estratégicas, optimizando las operaciones del supermercado.

La implementación de tecnologías de inteligencia artificial, tan a la orden del día actualmente, podría dar un gran salto de calidad a este sistema. Por contraparte habría que buscar la funcionalidad idónea para ello y en el que la simbiosis de estos dos elementos (nuestro sistema y una inteligencia artificial) fuera provechosa. Pese a que se han implementado medidas de seguridad robustas, siempre hay espacio para mejorar. Futuras acciones podrían incluir la actualización continua de los protocolos de seguridad para proteger contra nuevas amenazas y así blindar más nuestra aplicación. Además, se podría enfocar en mejorar la privacidad de los datos, asegurando que toda la información sensible sea manejada de acuerdo con las regulaciones más estrictas y las mejores prácticas del sector.

Bibliografía

[1] Davenport, T. H., & Harris, J. G. (2007). *Competing on Analytics: The New Science of Winning*. Harvard Business Review Press.

[2] Monk, E. F., & Wagner, B. J. (2012). *Concepts in Enterprise Resource Planning*. Cengage Learning

[3] Velte, T., Velte, A., & Elsenpeter, R. (2009). *Cloud Computing: A Practical Approach*. McGraw-Hill.

[4] Chopra, S., & Meindl, P. (2013). *Supply Chain Management: Strategy, Planning, and Operation*. Pearson.

[5] SAP. *Software para la industria minorista*. SAP. Accedido el 10 de junio de 2024, de https://www.sap.com/spain/industries/retail.html

[6] Microsoft. *Business Central*. Microsoft. Accedido el 10 de junio de 2024, de https://www.microsoft.com/es-es/dynamics-365/products/business-central

[7] Oracle. *Oracle Retail*. Oracle. Accedido el 10 de junio de 2024, de https://www.oracle.com/es/retail/

[8] Royce, W. W. (1970). *Managing the Development of Large Software Systems*. Proceedings of IEEE WESCON.

[9] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). *Manifesto for Agile Software Development*. Agile Alliance. Recuperado de https://agilemanifesto.org/

[10] Schwaber, K., & Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall.

[11] Microsoft. *Precios de Microsoft Office 365*. Microsoft. Accedido el 13 de marzo de 2024, de https://www.microsoft.com/es-es/microsoft-365/buy/compare-all-microsoft-365-products

[12] JetBrains. *JetBrains IntelliJ IDEA*: *Compra la suscripción mensual para uso personal*. Accedido el 13 de marzo de 2024, de https://www.jetbrains.com/es-es/idea/buy/?section=personal&billing=monthly

[13] Seguridad Social. *Información sobre cotización y recaudación para trabajadores*. Accedido el 13 de marzo de 2024, de https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537

[14] GS1 España. *Qué significa y para qué sirve el Código EAN*. GS1: EAN. Accedido el 18 de marzo de 2024, de https://www.gs1es.org/capturar-codigo-de-barras-gs1/ean-upc-simbologia/

[15] GS1. *What we do*?. Accedido el 18 de marzo de 2024, de https://www.gs1.org/about/what-we-do

[16] Carrefour. *Supermercados Carrefour Express*. Accedido el 18 de marzo de 2024, de https://www.carrefour.es/tiendas-carrefour/supermercados/carrefour-express/

[17] GS1 España. *Estándares GS1 para identificación: GTIN*. Accedido el 18 de marzo de 2024, de https://www.gs1es.org/estandares-gs1-identificar/gtin/

[18] Wikipedia. *Anexo de Código GS1 por países*. Accedido el 18 de marzo de 2024, de

https://es.wikipedia.org/wiki/Anexo:Prefijos_de_C%C3%B3digo_GS1_por_pa%C3%ADses

[19] GS1 España. *Estándar GS1-128.* Accedido el 19 de marzo de 2024, de https://www.gs1es.org/capturar-codigo-de-barras-gs1/estandar-gs1-128/

[20] GS1 España. *¿Qué es el carácter FNC1?.* Accedido el 19 de marzo de 2024, de https://www.gs1es.org/capturar-codigo-de-barras-gs1/fnc1/#:~:text=Qu%C3%A9%20es%20el%20car%C3%A1cter%20FNC1,128%20o%20el%20GS1%20DataMatrix.

[21] GS1 España. *GS: Identificadores de Aplicación.* Accedido el 19 de marzo de 2024, de https://www.gs1es.org/capturar-codigo-de-barras-gs1/gs1-ia/

[22] Rod Johnson, J., Hoeller, J., Cergol, R., & Schaefer, A. (2021). *Professional Java Development with the Spring Framework* (2nd ed.). Wrox.

[23] Spring Framework. *Spring Framework Documentation*. Accedido el 5 de Mayo de 2024, de https://spring.io/projects/spring-framework

[24] JetBrains. *Spring Framework con IntelliJ IDEA*. Accedido el 8 de mayo de 2024, de https://www.jetbrains.com/es-es/idea/spring/

[25] JetBrains. *Hibernate en IntelliJ IDEA*. Accedido el 8 de mayo de 2024, de https://www.jetbrains.com/help/idea/hibernate.html

[26] Spring. *Spring Data JPA - Reference Documentation*. Accedido el 9 de mayo de 2024, de https://docs.spring.io/spring-data/jpa/reference/index.html

[27] Spring. *UserDetailsService Documentation (Spring Security 6.3.0 API).* Accedido el 9 de mayo de 2024, de https://docs.spring.io/spring-security/site/docs/6.3.0/api/org/springframework/security/core/userdetails/UserDetailsService.html

[28] Spring. *UserDetails Documentation (Spring Security 6.3.0 API)*. Accedido el 9 de mayo de 2024, de https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html

[29] Creative Tim. *Soft UI Dashboard*. Accedido el 15 de abril de 2024, de https://www.creative-tim.com/product/soft-ui-dashboard

[30] Creative Tim. *Soft UI Dashboard - Documentation.* Accedido el 15 abril de 2024, de https://www.creative-tim.com/learning-lab/bootstrap/overview/soft-ui-dashboard

[31] Fenix Web Server. *Fenix Web Server Download and Documentation.* Accedido el 17 de abril de 2024, de http://fenixwebserver.com/

[32] Spring. *Acceso a datos con MySQL en Spring*. Accedido el 18 de abril de 2024, de https://spring.io/guides/gs/accessing-data-mysql

[33] Oracle. *Documentación de la API de Java SE 17*. Accedido el 19 de marzo de 2024, de https://docs.oracle.com/en/java/javase/17/docs/api/index.html

[34] Visual Studio Code Marketplace. *Extension: VS Code-Counter*. Accedido el 10 de junio de 2024, de https://marketplace.visualstudio.com/items?itemName=uctakeoff.vscode-counter

[35] Google Cloud Platform. *Documentación de Google Cloud*. Accedido el 17 de junio de 2024, de https://cloud.google.com/docs?hl=es-419

[36] Google Cloud. *Documentación de Google App Engine*. Accedido el 17 de junio de 2024, de https://cloud.google.com/appengine/docs?hl=es-419

[37] Google Cloud. *Google Cloud SDK.* Accedido el 17 de junio de 2024, de https://cloud.google.com/sdk?hl=es

[38] Postman. *POSTMAN Doc*. Accedido el 17 de junio de 2024, de https://learning.postman.com/docs/introduction/overview/

[39] Apache JMeter. *User Manual*. Accedido el 17 de junio de 2024, de https://jmeter.apache.org/usermanual/index.html

Anexo A
Código fuente

El código fuente del proyecto, tanto para el Backend como para el Frontend, ha sido desarrollado y organizado en repositorios separados siguiendo una metodología profesional.

En caso de que se desee consultar el código fuente del proyecto, se puede solicitar acceso a través de los medios proporcionado a continuación:

Email: martinezrioseulalio@gmail.com | eulalio@correo.ugr.es

Anexo B
Manual de Usuario

Desde el siguiente enlace se puede acceder al sistema: https://supermercado-ui.ew.r.appspot.com/ . En el presente texto se presentarán las distintas vistas así como una breve explicación del modo de funcionar que tienen estas. La primera toma de contacto con el sistema es la pantalla de inicio de sesión que se muestra a continuación en la Ilustración 50, donde el usuario deberá introducir sus credenciales (usuario y contraseña) y pulsar el botón 'ENVIAR' que aparece en la pantalla o en su defecto pulsar la tecla 'ENTER' de su teclado.

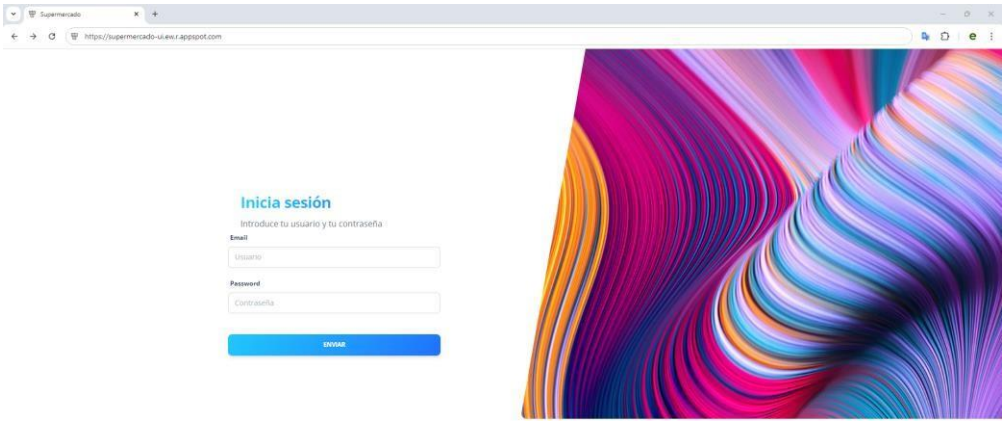

Ilustración 50: Pantalla de Inicio de sesión

Puede ser que el usuario no introduzca correctamente dichas

credenciales por lo que el sistema le advertirá de ello mostrándole el mensaje que se muestra en la Ilustración 51 y no le dejará ingresar.

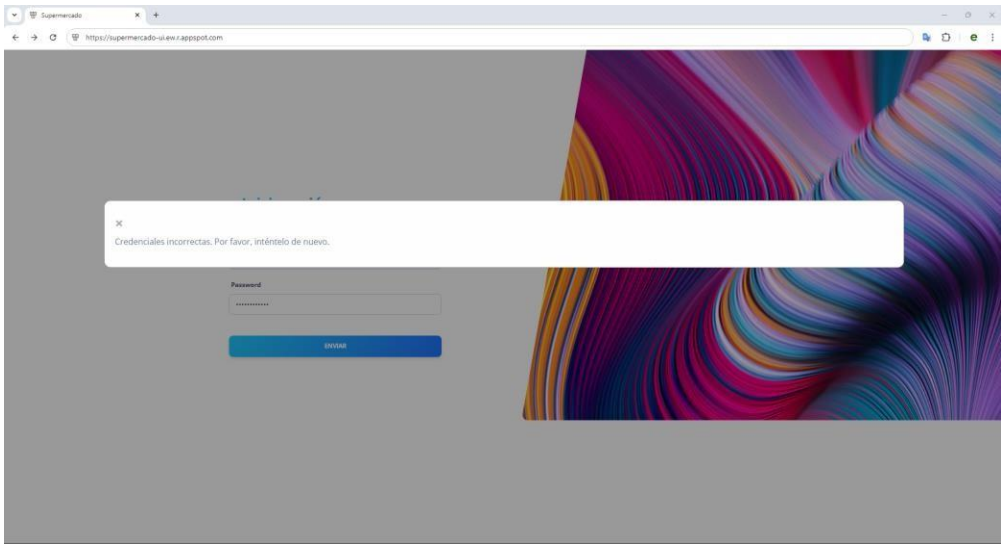

Ilustración 51: Mensaje de error credenciales incorrectas

Una vez las credenciales correctas han sido introducidas y el sistema así lo ha validado, se le dará acceso al sistema. El diseño de las distintas páginas es similar al mostrado en la Ilustración 52. En la parte izquierda encontramos un menú vertical, con las distintas páginas que podemos visitar y en la parte superior derecha encontramos el botón para poder cerrar sesión y salir del sistema. La Ilustración 52 muestra la página de inicio, que muestra su información personal así como datos generales de la tienda en la que trabaja; la dirección, la ciudad, los distintos trabajadores que tiene…

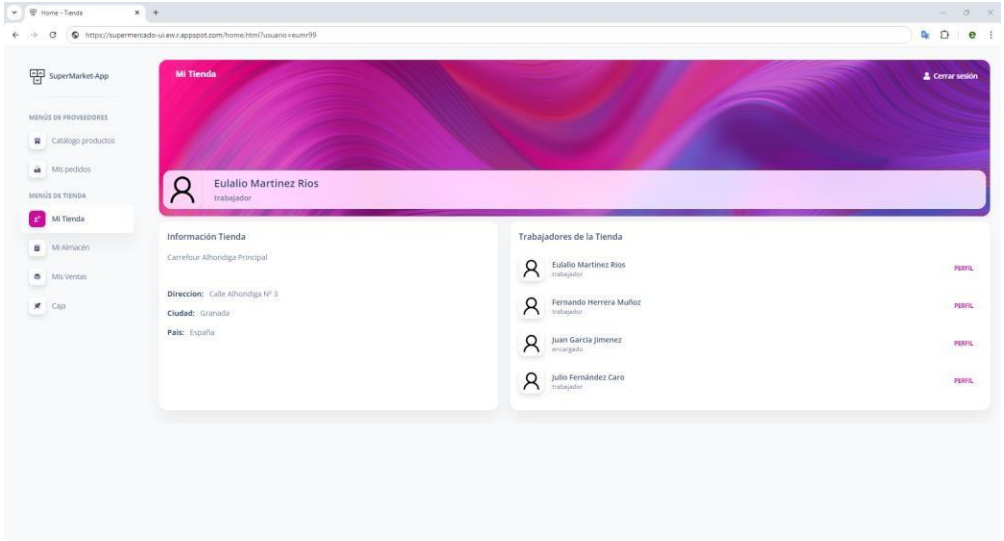

Ilustración 52: Menú Mi Tienda

Una de las opciones más importantes es la de gestionar los productos que tenemos en nuestra tienda. Para ello si pulsamos en la opción ‘Mi Almacén’ del menú lateral viajaremos hasta la página que se muestra en la Ilustración 53. En la que podremos hacer un seguimiento de los productos que tenemos tanto en el almacén de nuestra tienda como en los propios expositores, sus fechas de caducidad, el descuento que tienen aplicado o el precio que tienen asociado.

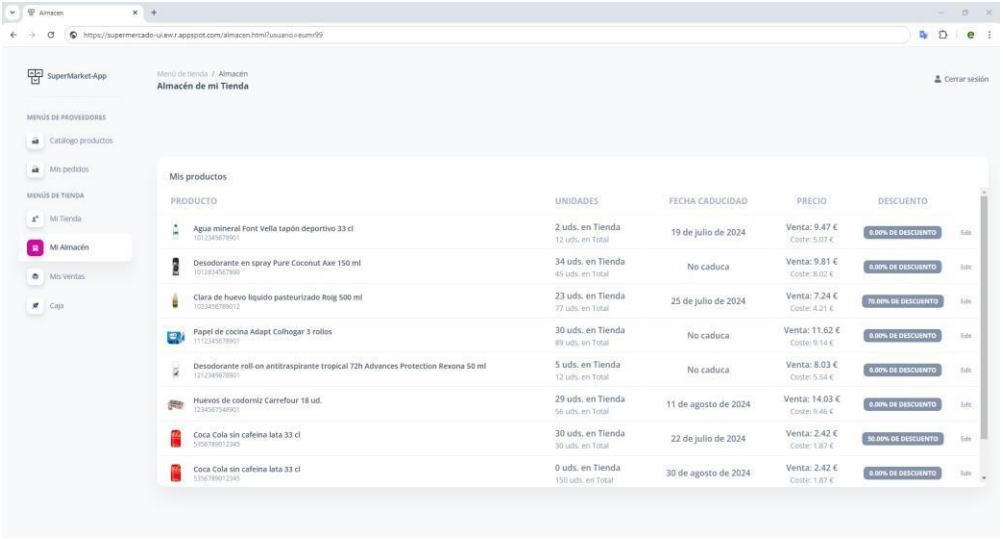

Ilustración 53: Menú Mi Almacén

Otra función principal en este sistema es la de permitir a los trabajadores de las tiendas realizar pedidos al proveedor para poder abastecer su almacén. Para ello al pulsar en la opción ‘Catálogo productos’ el sistema nos llevará a una página como la mostrada en la Ilustración 54, donde podremos encontrar todos los productos que el proveedor pone a la disposición de la tienda.

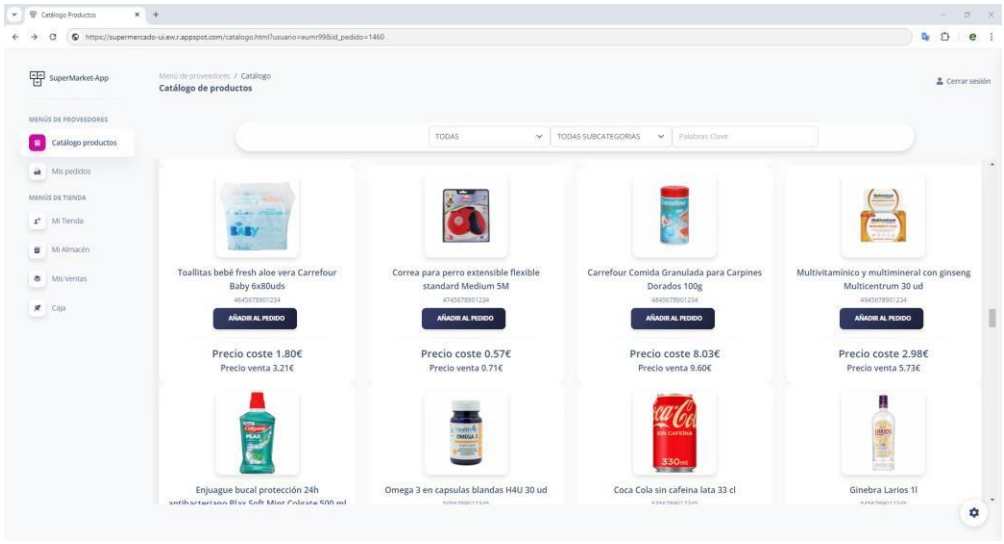

Ilustración 54: Menú Catálogo de productos

El sistema al tener los productos organizados por categorías y subcategorías, permite filtrar y buscar para comodidad del usuario. En la Ilustración 55 se muestra cómo se haciendo uso del filtro por categorías el sistema solo muestra los productos de la categoría ‘Bebidas’ y de la subcategoría ‘Refrescos’.

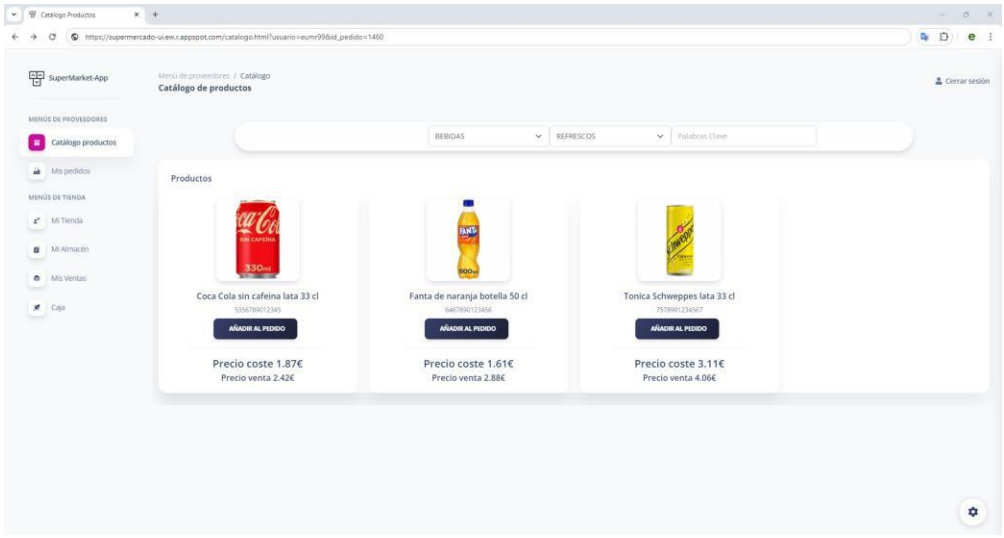

Ilustración 55: Menú Catálogo filtrado por subcategoria

El sistema también permite hacer una búsqueda por palabras clave, como se muestra en la Ilustración 56. Cada producto muestra tanto el precio de coste que tendría la unidad de dicho producto para la tienda como el precio base de venta que la matriz proveedora ha marcado.

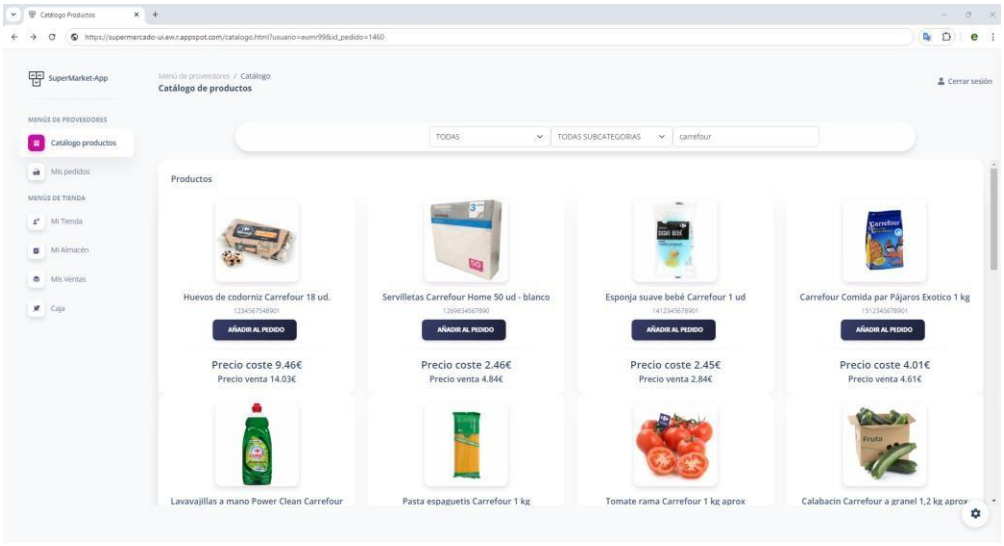

Ilustración 56: Menú Catálogo filtrado por palabras clave

Una vez que el usuario ha encontrado el producto deseado, al pulsar sobre su botón 'AÑADIR AL PEDIDO' se desplegará un menú donde el usuario deberá introducir la cantidad que desea pedir de dicho producto. Al pulsar en el botón 'AÑADIR' del menú desplegable, el producto se añadirá al carrito de su pedido. Carrito que se puede consultar en todo momento pulsando en el botón que aparece en la parte inferior derecha en la pantalla, lo que provocará que se muestre un menú desplegable en la parte derecha de la pantalla que contiene toda la información del carrito de su pedido actual como muestra la Ilustración 57.

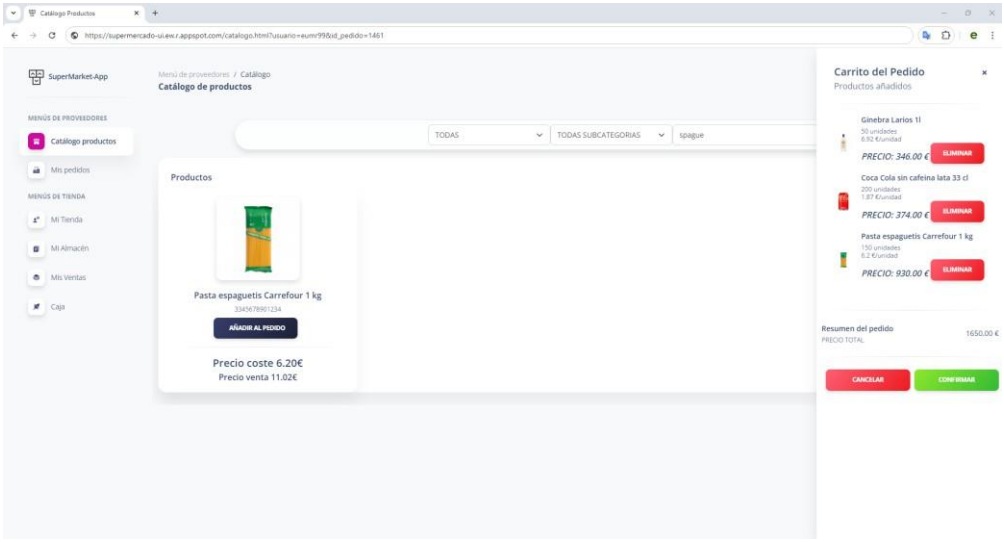

Ilustración 57: Menú Catálogo ver carrito de la compra

El carrito cuenta con la opción de eliminar del pedido algún producto que esté incluido en él. Una vez todos los productos deseados hayan dicho incluidos se pulsará el botón 'CONFIRMAR' para dar por

finalizado el pedido y que el proveedor reciba la información para poder prepararlo.

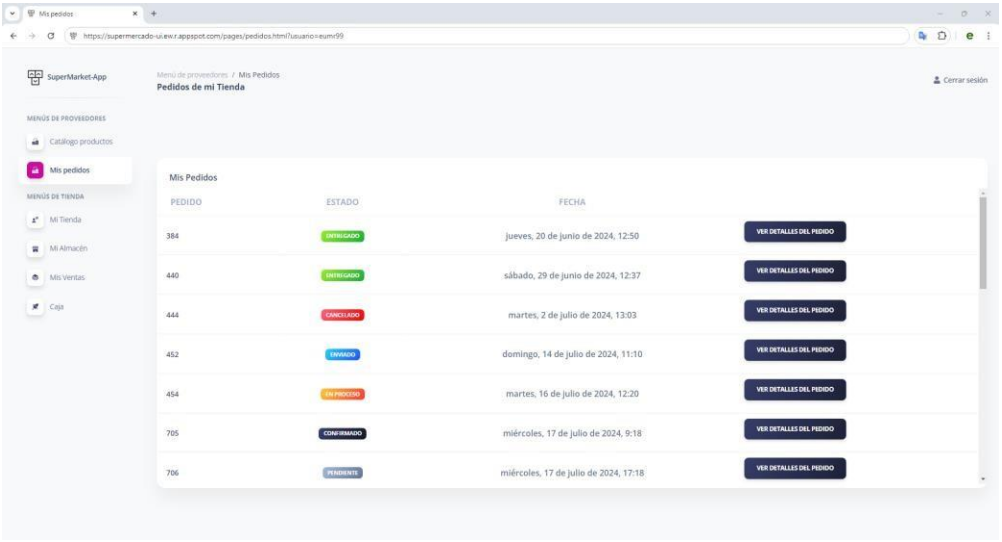

Ilustración 58: Menú mis pedidos

Al pulsar en la opción ‘Mis Pedidos’ del menú lateral se mostrará al usuario una pantalla similar a la mostrada en la Ilustración 58, en la que podrá consultar tanto el estado de los pedidos de su tienda, la fecha de realización. Para consultar los detalles de un pedido pulsar en el botón ‘VER DETALLES DEL PEDIDO’ correspondiente al pedido deseado. Al hacerlo el sistema nos mostrará tanto el precio total del pedido, como los productos que contiene, así como los detalles individuales de cada uno de ellos. Desde esta pantalla será desde la cual el trabajador irá escaneando los códigos de barras del albarán de entrega, cuando el pedido llegue a su tienda para que los datos de estos productos se carguen en el sistema de su almacén.

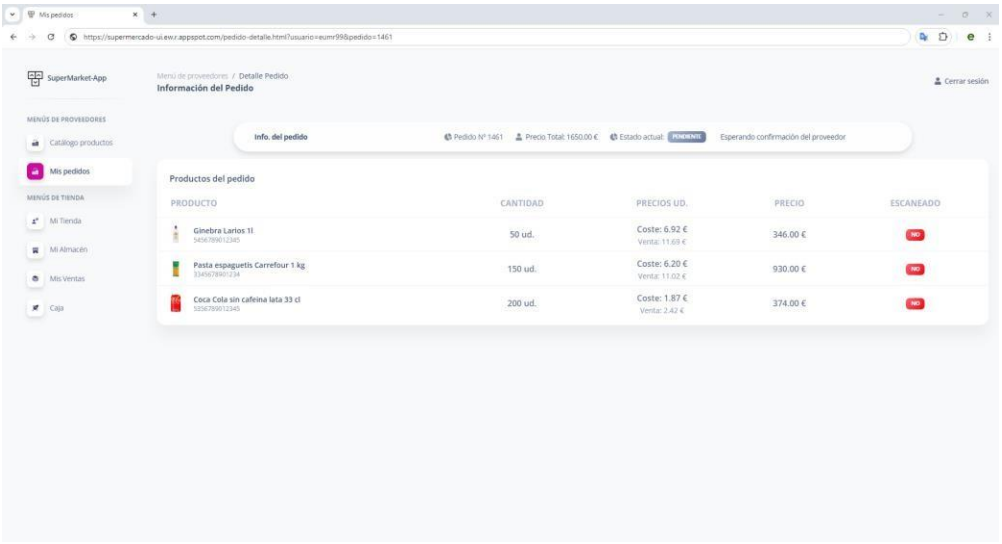

Ilustración 59: Menú detalles del pedido

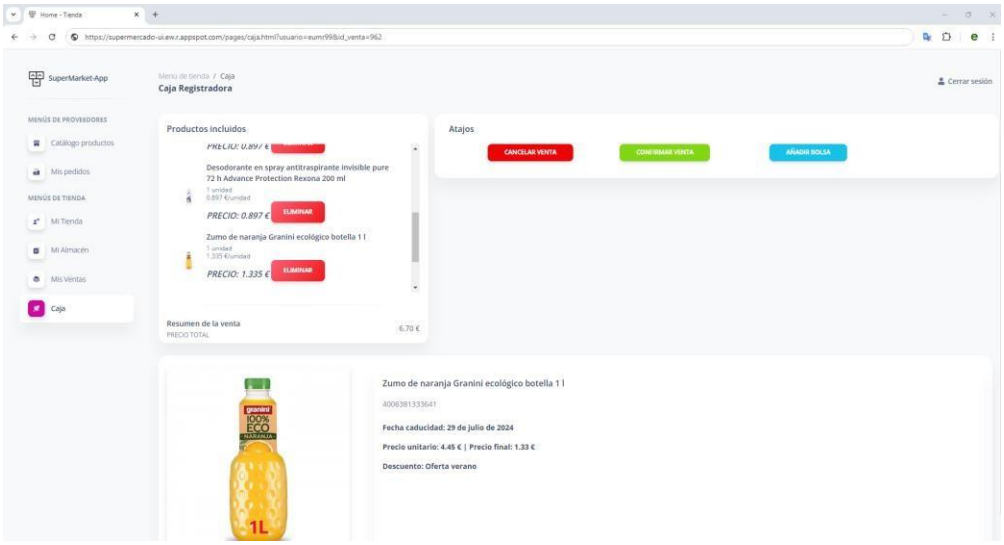

Ilustración 60: Menú Caja registradora

Para la función de caja registradora el sistema cuenta con la opción ‘Caja’ del menú lateral izquierdo, tal y como muestra la Ilustración 60. Desde esta podremos visualizar la información de cada producto del que se escanee su código de barras para añadirlo a una venta. Desde esta pantalla el trabajador podrá realizar todas las tareas relacionadas con las ventas.

Al pulsar en la opción ‘Mis Ventas’ del menú lateral se podrán visualizar todas las ventas realizadas en la tienda, como se muestra en la Ilustración 61. Información como el precio, la fecha o si requieren o no entrega a domicilio por parte de la tienda. Al pulsar en el botón ‘VER DETALLES DE LA VENTA’ se mostrará la información detallada de dicha venta, como los productos que contiene o sus precios.

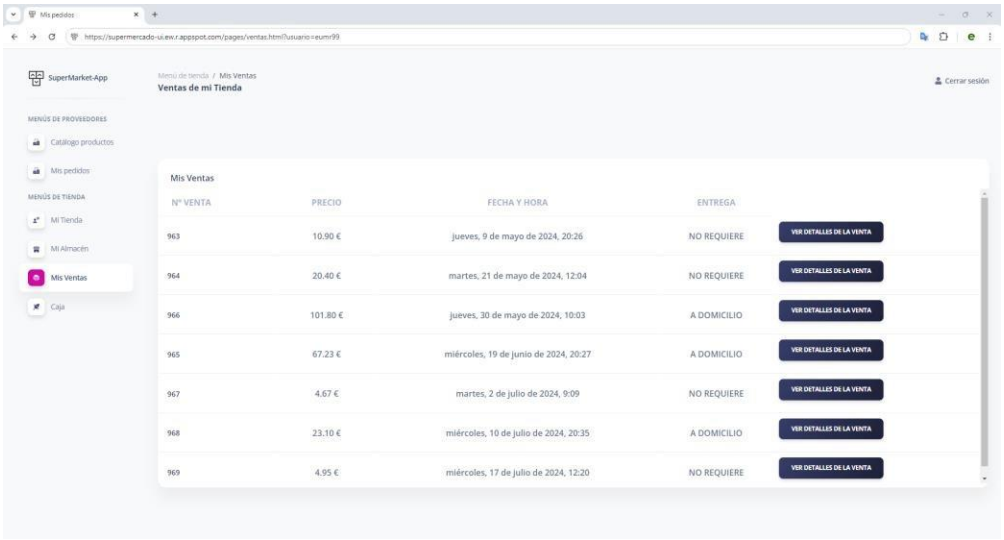

Ilustración 61: Menú Mis ventas

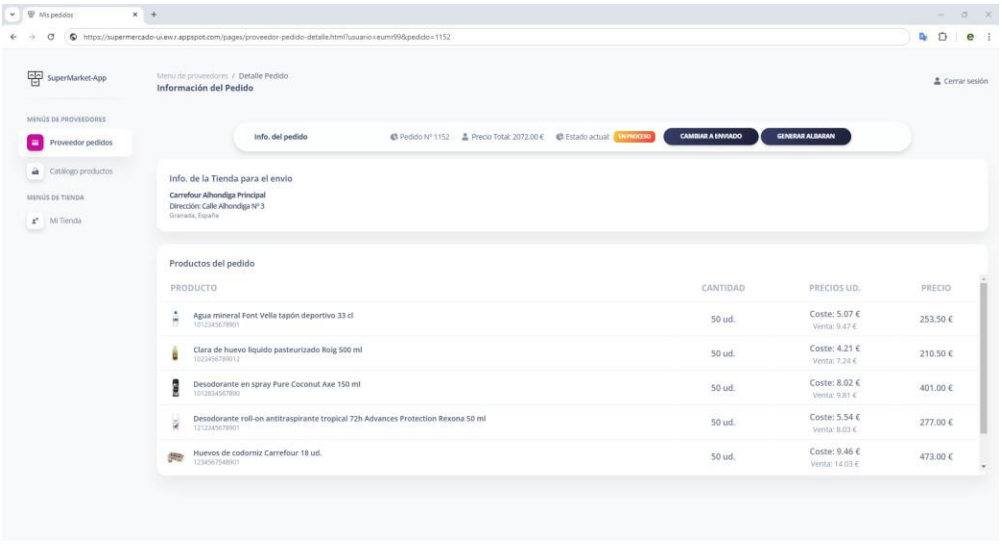

Ilustración 62: Página proveedor gestión de pedidos

De un modo similar al anterior si, el usuario que accede al sistema es el proveedor, este tiene en su menú lateral izquierdo otras opciones. 'Catalogo productos' para incluir productos nuevos, 'Mi Tienda' para ver la información de la matriz y la más importante 'Proveedor pedidos' en la que gestionará los pedidos que las tiendas le hacen. En esta opción el sistema le mostrará una lista con los pedidos de las tiendas y en la que al pulsar el botón 'VER DETALLES PEDIDO' se mostrará una vista como la mostrada en la Ilustración 62, en la que el proveedor puede cambiar el estado del pedido conforme este vaya evolucionando así como generar el albarán necesario para el reparto y la entrega del pedido pulsando el botón 'GENERAR ALBARÁN' lo que generará un archivo PDF que será automáticamente descargado por el navegador web.

Anexo C
Recursos Informáticos

En este anexo se describen los recursos informáticos empleados durante el desarrollo del proyecto, tanto en términos de hardware como de software. Recursos que fueron fundamentales para la implementación, pruebas y documentación del proyecto o aquellos que se usaron brevemente.

Hardware

Ordenador portátil: Dell XPS 13 9370

- Memoria: 8 GB RAM
- Procesador: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 1992 Mhz, 4 procesadores principales, 8 procesadores lógicos
- Almacenamiento: 256 GB

Software

Sistema Operativo: Windows 11 Home 64 bits

Procesador de textos: Word (Microsoft Office Profesional Plus 2019)

Entornos de programación:

- IntelliJ IDEA Ultimate Edition 2023.3.2.2
- Visual Studio Code

Lenguajes de programación:

- Java 19.0.2
- HTML 5 (HyperText Markup Language)
- JavaScript
- CSS (Cascading Style Sheets)
- SCSS (Sassy CSS)

- Postman
- Fenix Web Server
- MySQL Workbech 8.0
- Swagger Editor
- GitHub
- Draw.io
- Google Colab
- Google Cloud Platform
- Google App Engine
- SQL Cloud
- Apache JMeter