



UNIVERSIDAD DE MÁLAGA



INGENIERÍA DEL SOFTWARE

STOCKMTS: Gestor de stock y localización de productos en
2D/3D

STOCKMTS: Stock manager and product localization in
2D/3D

Realizado por
Alberto Fernández Ramos

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO 2025



INGENIERÍA DEL SOFTWARE

STOCKMTS: Gestor de stock y localización de productos en
2D/3D

STOCKMTS: Stock manager and product localization in
2D/3D

Realizado por
Alberto Fernández Ramos

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO 2025

Resumen

En este trabajo fin de grado se ha desarrollado una aplicación web, denominada *stockmts*, destinada a la mejora de gestión y facilidad de localización de stock de productos dentro de los almacenes de las empresas que puedan tener un volumen del mismo elevado. El principal objetivo de esta aplicación es facilitar la trazabilidad de los productos que gestione la empresa y reducir el tiempo de búsqueda de un producto específico dentro de un almacén.

Para el desarrollo se ha seguido una aproximación full-stack, donde se han utilizado tecnologías muy utilizadas dentro del desarrollo de software, empleando como lenguaje de programación JavaScript. Más concretamente, se ha utilizado React para el front-end, Node.js, Express.js y SQL Server, **para el back-end**.

Dentro de las utilidades que nos proporciona *stockmts* podemos definir dos grandes áreas funcionales: la gestión de productos en estanterías del almacén y la trazabilidad de los productos mediante el tratamiento de las facturas.

En primer lugar, *stockmts* permite la creación de una representación visual del almacén que posea la empresa, a través de la creación y ubicación de las estanterías dentro del almacén, la ubicación de productos dentro de las estanterías, la visualización de las estanterías en el mapa del almacén, la de los productos en las estanterías o la búsqueda de productos dentro del mapa del almacén, indicando la ubicación de las estanterías que almacenan ese producto a buscar.

En segundo lugar, tenemos la gestión de facturas, que nos permite subirla en archivos pdf, extraer la información de las mismas y, con ello, llevar la trazabilidad de los productos con el fin de tener constancia de los productos que hay en stock, conocer qué productos quedan en el almacén, cuántos están colocados en las estanterías, cuál es su proveedor y qué clientes los han adquirido.

Palabras Clave: React, JavaScript, Node.js, gestión del stock, gestión de almacén.

Abstract

In this bachelor's thesis, a web application named *stockmts* has been developed with the aim of improving stock management and facilitating the location of inventory within company warehouses that handle large volumes of products. The primary objective of this application is to enhance product traceability and reduce the time required to locate specific items within a warehouse.

The development followed a full-stack approach, employing widely used technologies in modern software development, with JavaScript as the main programming language. Specifically, React was used for the front-end, while Node.js, Express.js, and SQL Server were used for the back-end.

Among the features provided by *stockmts*, two main functional areas can be defined: product management on warehouse shelves and product traceability through invoice processing.

Firstly, *stockmts* allows the creation of a visual representation of the company's warehouse, including the creation and placement of shelves within the warehouse, the allocation of products on those shelves, the visualization of shelves on the warehouse map, the display of products on each shelf, and the search for products within the warehouse map, which indicates the location of the shelves storing the searched item.

Secondly, there is invoice management, which allows PDF files to be uploaded, extracts their information, and use it to track product traceability. This make it possible to keep track of the products in stock, identify which products remain in the warehouse, how many are placed on the shelves, who their suppliers are, and which customers have acquired them.

Keywords: React, JavaScript, Node.js, Stock management, Warehouse management.

Índice

Contenido

Resumen	1
Abstract.....	1
Índice.....	1
Introducción	1
1.1 Motivación.....	1
1.2 Objetivos	2
1.3 Metodología.....	3
1.4 Estructura de la Memoria.....	4
Tecnologías y Herramientas.....	7
2.1 Lenguajes de Programación.....	7
2.1.1 JavaScript	7
2.1.2 HTML.....	8
2.1.3 SQL.....	9
2.1.4 Decisión de la elección	9
2.2 Tecnologías Front-End.....	10
2.2.1 React.....	10
2.2.2 Decisión de la elección	10
2.3 Tecnologías Back-End	10
2.3.1 Node.js	11
2.3.2 Express.js.....	11
2.3.3 Decisión de la elección	11
2.4 Base de Datos.....	12
2.4.1 SQL Server	12
2.4.2 Decisión de la Elección.....	12
2.5 Herramientas.....	13

2.5.1 Visual Studio Code	13
2.5.2 SQL Server Management Studio	13
2.5.3 GitHub	14
Especificación y Análisis	15
3.1 Requisitos funcionales	15
3.1.1 Requisitos funcionales del sistema	15
3.1.2 Requisitos Funcionales del usuario	16
3.2 Requisitos no funcionales	17
3.3 Casos de uso	18
3.3.1 Registrarse	18
3.3.2 Iniciar sesión	20
3.3.3 Cierre de sesión	21
3.3.4 Subir factura cliente	22
3.3.5 Crear Almacén	23
3.3.6 Editar Almacén	24
3.3.7 Eliminar Almacén	25
3.3.8 Crear estantería	25
3.3.9 Reubicar estantería	26
3.3.10 Eliminar estantería	27
3.3.11 Crear cliente o proveedor	28
3.3.12 Editar cliente o proveedor	29
3.3.13 Eliminar cliente o proveedor	30
3.3.14 Editar factura	32
3.3.15 Eliminar factura	32
3.3.16 Editar producto	33
3.3.17 Eliminar Producto	34
3.3.18 Añadir producto a estantería	35
3.3.19 Retirar producto de estantería	36
3.3.20 Transferir producto entre estantería	37
3.3.21 Subir imagen de producto	39
3.3.22 Cambiar imagen de producto	40

3.3.23 Eliminar imagen de producto	41
3.3.24 Ubicar producto en estantería	42
3.3.25 Buscar producto en estantería	43
3.4 Diagramas de Secuencia	44
3.4.1 Introducción.....	44
3.4.2 Subir Factura.....	44
3.4.3 Buscar producto en estantería	45
3.4.4 Transferir producto entre estantería	45
3.5 Diagrama de flujo.....	46
3.5.1 Introducción.....	46
3.5.2 Subir Factura.....	46
3.5.3 Buscar producto en estantería	47
3.5.4 Transferir producto entre estanterías.....	47
Diseño del Sistema.....	49
4.1 Introducción	49
4.2 Diseño arquitectónico de la aplicación.....	49
4.2.1 Vista general del sistema	49
4.2.2 Componentes y conectores	50
4.3 Diseño de la base de datos	51
4.4 Diseño visual de la aplicación.....	54
4.4.1 Página principal.....	54
4.4.2 Página registro.....	55
4.4.3 Página inicio sesión.....	56
4.4.4 Página productos	57
4.4.5 Página facturas	57
4.4.6 Página estanterías.....	58
4.4.7 Página edición mapa almacén	59
4.4.8 Página visión productos de las estanterías	59
Implementación y Pruebas	61
5.1 Introducción	61
5.2 Back-End	62

5.2.1 Configuración de conexión con SQL Server	62
5.2.2 Configuración e iniciación del servidor	63
5.2.3 Rutas	64
5.3 Front-End	65
5.3.1 Components	66
5.3.2 Context	68
5.3.3 Hooks	68
5.3.4 Models	69
5.3.5 Pages	69
5.3.6 Presenters	69
5.3.7 Services	70
5.3.8 Ui	70
5.3.9 Utils	72
5.3.10 Views	72
5.4 Pruebas	73
5.4.1 Introducción	73
5.4.2 Registro	73
5.4.3 Inicio sesión	74
5.4.4 Subir factura	76
5.4.5 Crear Almacén	77
5.4.6 Buscar producto en estantería	78
Conclusiones y Líneas Futuras	81
6.1 Conclusiones	81
6.2 Dificultades encontradas	82
6.3 Líneas futuras	82
Referencias	85
Manual de Usuario	89
A.1. Registro	89
A.2. Inicio de sesión	90
A.3. Pestaña principal y perfil	91
A.4. Carga de facturas	91

A.5. Operaciones en productos, facturas y estanterías	92
A.6. Operaciones en almacén	93
A.7. Organizar productos en estantería 2D	94
A.8. Búsqueda y sugerencias de productos.....	95
Manual de Despliegue.....	97
B.1. Despliegue del servidor	97
B.2. Despliegue del cliente	98

Introducción

1.1 Motivación

La gestión del almacenaje en pequeñas y medianas empresas suele presentar mayores desafíos de los que aparenta a simple vista, especialmente cuando se busca realizarla de forma eficiente y con un control adecuado. Estas dificultades se intensifican cuanto mayor sea el tamaño del almacén y el volumen de mercancías que maneje la empresa.

Además, para los nuevos trabajadores, el proceso de adaptación y memorización de las localizaciones de los artículos dentro del almacén puede ser un trabajo ocioso y que en ocasiones haya productos específicos que por no ser usados habitualmente o por un cambio de su ubicación, pueda hacer que ese trabajador desperdicie tiempo de su trabajo en buscar ese producto en vez de poder estar llevando a cabo otras acciones que sean beneficiosas para la empresa.

Por ello, y tomando de ejemplo a una empresa familiar de distribución de bebidas se planteó el desarrollo de esta aplicación para intentar facilitarles las acciones dirigidas a la gestión y almacenaje de productos dentro de sus almacenes.

1.2 Objetivos

Este proyecto busca crear una herramienta digital que facilite la gestión del stock dentro de un almacén de una manera intuitiva y clara, para conseguir disminuir el tiempo empleado en la búsqueda de productos dentro de un almacén. Para ello se ha creado una aplicación web full-stack.

En este modelo, el front-end se desarrolla utilizando React, siguiendo el patrón Modelo-Vista-Presentador(MVP) [1] , y encargándose únicamente de la interfaz de usuario y de recibir las acciones del usuario. El Presentador, implementado en el cliente mediante controladores y servicios, gestiona la lógica de interacción entre front y back-end. La parte back-end, construida con Node.js y Express, a su vez, se conecta a una base de datos SQL Server para almacenar, actualizar y consultar los datos.

La aplicación permitirá realizar un proceso completo relacionado con las facturas. Pudiendo subir una factura, visualizarla e introducirla dentro de la aplicación, reflejándose así en la cantidad de cajas en stock de los productos que contenga la factura. Además, permitirá distinguir si la factura es de un proveedor, lo que incrementa el número de stock en el almacén, o si es un cliente lo decrementa. Gracias a esto se podrá decir que se puede llevar una trazabilidad del stock que se ha movido entre entradas y salidas de mercancía.

Por otro lado, la aplicación también permitirá la creación de una representación visual de un almacén, pudiendo editarle las dimensiones (eso sí, deben de ser dimensiones NxM). En el almacén se podrá introducir estanterías, las cuales, cuando se crean, puede modificarse el número máximo de cajas de stock posibles de almacenar en esta. Además, se puede reubicar en el mapa del almacén o eliminarla (junto con la cantidad de productos que almacene, que pasarán a estar en el stock no ubicado en estanterías).

Siguiendo con las estanterías, al hacer clic en una de ellas se podrá desplegar una ventana en la que se mostrará una representación de la estantería, pudiendo visualizar y ubicar los diferentes productos en las baldas que se desee e indicarle cuantas baldas quiere que ocupe.

A su vez, en la visualización del mapa, a la derecha de la página principal, se encontrará un buscador, el cual, permitirá buscar el producto que se desee, y si se encuentra en stock marcará las estanterías que contengan ese producto de color verde, indicando que esa estantería posee ese producto y con ello no tener que gastar tiempo buscándolo e ir directamente a por el producto que se desee.

1.3 Metodología

Para el desarrollo de este trabajo se ha seguido una metodología ágil basada en Scrum [\[2\]](#), adaptada para su aplicación en un entorno de trabajo individual. Scrum es una metodología centrada en la elaboración continua para alcanzar los mejores resultados posibles sostenido en el tiempo. Esto se logra mediante la realización periódica de entregas parciales a lo largo del proyecto, convirtiéndola en una opción excelente en situaciones donde los requisitos pueden cambiar frecuentemente. Facilitando, además, la incorporación de nuevas ideas, permitiendo introducir cambios de forma ágil y creativa.

El desarrollo del proyecto se organizó en iteraciones de dos semanas, siguiendo una metodología basada en sprints. En el sprint inicial se definieron las tecnologías a emplear, se estudió sobre su funcionamiento y se preparó el entorno necesario para poner en marcha la aplicación. A continuación, se llevaron a cabo seis sprints adicionales centrados en la implementación y documentación progresiva de las funcionalidades. El sprint final se destinó a la revisión y mejora

de la memoria del proyecto, con el objetivo de lograr una presentación más clara y comprensible.

Durante la primera iteración, se realizó un análisis detallado de los beneficios y limitaciones de las distintas opciones tecnológicas evaluadas para el proyecto. A partir de los resultados obtenidos, se tomó la decisión final sobre las herramientas a utilizar, lo que permitió proceder con la configuración inicial y el despliegue de los componentes fundamentales del sistema. En la segunda iteración se desarrolló los CRUD necesarios para la parte de Productos, Facturas y Estanterías. En la tercera iteración se dedicó en su mayoría a las funcionalidades para la extracción de información de las facturas que se podían subir a la aplicación, junto con los respectivos CRUD del inicio de sesión y registro de la empresa que utilizase la aplicación desarrollada, denominada stockmts. En la cuarta iteración se desarrollaron las funcionalidades ligadas al mapa del almacén, la ubicación y visión de las estanterías, visión de los productos en las estanterías y buscador de productos en el mapa del almacén. En la quinta iteración se dedicó a resolución de incoherencias y puesta a punto mediante testeo y pruebas. La sexta iteración se dedicó a completar y perfeccionar la redacción de la memoria para lograr una mejor comprensión de la misma. A la vez, que se realizó una revisión global del proyecto.

1.4 Estructura de la Memoria

La estructura de esta memoria se compone de los siguientes capítulos.

- **Capítulo 1. Introducción.** Se presentará la motivación y la metodología seguida para la ejecución de este proyecto.
- **Capítulo 2. Tecnologías y Software.** Se explicará las tecnologías y software empleado durante el desarrollo del proyecto.

- **Capítulo 3. Especificación y Análisis.** Se indicarán los requisitos y casos de uso necesarios para la implementación del proyecto.
- **Capítulo 4. Diseño del Sistema.** Se explicará las condiciones que sirvieron de inspiración para el desarrollo y se visualizará el proyecto.
- **Capítulo 5. Implementación y Pruebas.** Se analizará el código del servidor y la aplicación del cliente. Además, se mostrarán las pruebas realizadas demostrando casos de uso.
- **Capítulo 6. Conclusiones y Líneas Futuras.** Se realizará una explicación final del proyecto, sus dificultades y se indicarán unas posibles mejoras para implementar en el futuro.

Para concluir, se tratará la bibliografía y referencias utilizadas en la memoria, a la vez que varios apéndices, como son el manual de usuario y el manual de despliegue.

Tecnologías y Herramientas

Tras haber presentado la motivación y los objetivos del proyecto en el capítulo anterior, este capítulo se centrará en describir las principales tecnologías empleadas para su desarrollo. Con el propósito de facilitar su comprensión, dichas tecnologías se han clasificado en tres categorías: herramientas destinadas al front-end, las correspondientes al back-end y otras complementarias que han apoyado el proceso de desarrollo.

2.1 Lenguajes de Programación

En este apartado se describirán los lenguajes de programación utilizados en el desarrollo de la aplicación. Con los cuales se ha construido tanto la parte visual con la que el usuario interactúa directamente, como los procesos encargados de manejar y almacenar los datos en la base de datos.

2.1.1 JavaScript

JavaScript es un lenguaje de programación de alto nivel ampliamente utilizado en el desarrollo web. Su principal objetivo es dotar de dinamismo e

interactividad a las páginas web, complementando tecnologías como HTML y CSS.

Tuvo su origen en mayo de 1995 por Brendan Eich y se estandarizaron los objetos en JavaScript en el 2000 gracias a la creación del documento JSON, desarrollado por Douglas Crockford [3]. La estandarización de JavaScript fue formalizada a través de ECMA International, bajo el nombre de ECMAScript (ES).



Figura 1. Logo JavaScript

JavaScript permite la programación de comportamientos dinámicos en sitios web, permitiendo destacar funcionalidades como manipulación y actualización del *Document Object Model* en tiempo real, gestión de eventos del usuario, desarrollo back-end a través de entornos de ejecución como Node.js, entre otros.

Actualmente JavaScript evoluciona bajo las directrices del comité TC39 de ECMA International, siendo la versión utilizada del estándar la ECMAScript 262 (ECMA-262) [4].

2.1.2 HTML

HTML (HyperText Markup Language) es un lenguaje de programación utilizado para la creación y estructuración de páginas web [5]. Su objetivo principal es definir la organización del contenido en la web mediante el uso de etiquetas, que permiten estructurar texto, imágenes, formularios y otros elementos multimedia.



Figura 2. Logo HTML

Tim Berners-Lee fue el desarrollador de HTML en 1991 y ha ido evolucionando continuamente bajo la gestión del grupo WHATWG. Actualmente estando en la versión HTML5, lanzada en 2014 [\[6\]](#).

2.1.3 SQL

SQL (Structured Query Language) es un lenguaje de programación especializado en la gestión y manipulación de bases de datos relacionales. Su principal objetivo es permitir la definición, consulta, modificación y control de la información almacenada por el sistema gestión de base de datos [\[7\]](#).

SQL fue desarrollado por Donald D. Chamberlin y Raymond F. Boyce en los años 70. El cual, ha ido evolucionando a lo largo de los años mediante los estándares como el SQL:2016 y variantes implementadas por sistemas como MySQL, PostgreSQL, Microsoft SQL Server y Oracle Database [\[8\]](#).



Figura 3. Logo SQL

2.1.4 Decisión de la elección

Se decidió elegir estos lenguajes de programación sobre otros teniendo en cuenta que se usaría el framework React, de ahí el uso de JavaScript y HTML.

En relación a la base de datos, se optó por una relacional facilitando la gestión eficiente de la información entre los objetos requeridos durante el desarrollo de la aplicación. Por ello se decantó por el uso de SQL a través de la herramienta SQL Server.

2.2 Tecnologías Front-End

En este apartado se describirá el framework que se ha utilizado para crear la parte visual de la aplicación, es decir, todo lo que el usuario ve y con lo que interactúa directamente.

2.2.1 React

React es una biblioteca de JavaScript de código abierto enfocada en la construcción de interfaces de usuario. Su principal objetivo es facilitar el desarrollo de componentes reutilizables que actualicen y rendericen eficientemente la vista en función del estado de la aplicación.

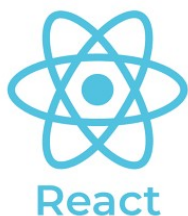


Figura 4. Logo React

React fue lanzado oficialmente en 2013. Desde entonces, ha evolucionado rápidamente hasta convertirse en una de las tecnologías más populares en el desarrollo front-end, mantenida actualmente por Meta [\[9\]](#).

Actualmente, React continúa su desarrollo activamente, incorporando características como Hooks, Context Api y Server Components, entre otros.

2.2.2 Decisión de la elección

Se optó por el uso de React frente a Angular, en primer lugar, debido al conocimiento y experiencia previa en esta tecnología. Posteriormente, tras evaluar los beneficios que ambos podían ofrecer, React destacó por su enfoque más declarativo en la construcción de interfaces y por la facilidad que brinda en la creación de aplicaciones complejas mediante el uso de componentes, propiedades y especialmente, la gestión de estados.

2.3 Tecnologías Back-End

En este apartado se tratarán las herramientas, lenguajes o frameworks que se han utilizado para crear la parte interna de la aplicación, encargándose de la

lógica, el procesamiento de datos y la conexión con la base de datos, aunque el usuario no la vea directamente.

2.3.1 Node.js

Node.js es un entorno de ejecución de JavaScript de código abierto que permite ejecutar código JavaScript fuera del navegador, enfocado en el servidor. Su principal objetivo es extender las capacidades de JavaScript al desarrollo back-end, facilitando la creación de aplicaciones web escalables y eficientes.

Node.js se creó en 2009 por Ryan Dahl [\[10\]](#). Su evolución ha sido continua hasta convertirse en una herramienta fundamental en el desarrollo web moderno, gracias en parte al apoyarse en su gestor de paquetes, npm.



Figura 5. Logo Node.js

2.3.2 Express.js

Express.js es un framework de Node.js que facilita la creación de servidores y APIs de manera rápida y estructurada.



Figura 6. Logo Express

Lanzado en 2010 por TJ Holowaychuk [\[11\]](#), Express proporciona herramientas esenciales para gestionar rutas y solicitudes HTTP, manteniendo un enfoque flexible y extensible.

Gracias a su sencillez y versatilidad, Express se ha convertido en uno de los pilares fundamentales en el desarrollo back-end con JavaScript.

2.3.3 Decisión de la elección

Se optó por la utilización de este stack, Node.js y Express.js, debido al uso de JavaScript, ya que de esta manera se podría usar tanto en el front-end como

en el back-end, lo que simplifica mucho el desarrollo y permite una mayor productividad. Además, se facilitaría en gran medida la integración con la base de datos SQL que se utiliza.

2.4 Base de Datos

En este apartado se tratará el sistema organizado que se encargará de almacenar, gestionar y recuperar la información de forma eficiente, permitiendo que la aplicación pueda guardar y consultar los datos de manera estructurada.

2.4.1 SQL Server

SQL Server es un sistema de gestión de bases de datos relacional desarrollado por Microsoft, diseñado para almacenar, administrar y recuperar datos de manera eficiente [\[12\]](#).

Se ha utilizado SQL Server 2022 con SQL Server Management Studio 20, teniendo como configuración de seguridad la opción SQL Server Authentication para poder vincularlo con el backend de la aplicación de forma segura.



Figura 7. Logo SQL Server

2.4.2 Decisión de la Elección

Se decantó por el uso de SQL Server 2022 y SQL Server Management Studio 20 debido a varios puntos fuertes que tiene esta combinación, como puede ser la manera en que se logra una integración completa del ecosistema Microsoft.

Individualmente, podemos comentar que SQL Server destaca por su fiabilidad y funcionalidades avanzadas, mientras que SQL Server Management Studio 20 proporciona una interfaz gráfica integral y versátil. Gracias a estas dos facetas, que al unir las nos proporcionan grandes facilidades, se optó por ellas para el desarrollo de la aplicación.

2.5 Herramientas

En este apartado se tratarán de las herramientas que se han utilizado para facilitar el desarrollo de la aplicación, abarcando tanto la edición de código como la administración de los datos y la realización de copias de seguridad.

2.5.1 Visual Studio Code

Visual Studio Code es un editor de código fuente gratuito y multiplataforma desarrollado por Microsoft, diseñado para programadores y con la capacidad de permitir la programación sobre una amplia variedad de lenguajes de programación como puede ser JavaScript, C++, HTML, entre otros [\[13\]](#).

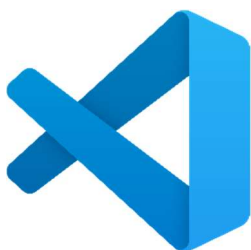


Figura 8. Logo Visual Studio Code

Además de lo mencionado anteriormente, se optó por este editor debido a su amplia gama de extensiones, por la integración nativa de git y control de versiones, y por las facilidades que ofrece para el desarrollo web y front-end, entre otros puntos favorables que tiene este editor.

2.5.2 SQL Server Management Studio

SQL Server Management Studio (SSMS) es una herramienta de software gratuita desarrollada por Microsoft que permite trabajar y administrar base de datos SQL Server de forma gráfica o mediante consultas, escritas en T-SQL [\[14\]](#). Ofrece la posibilidad de integrarse con otras herramientas desarrolladas de Microsoft como Azure SQL, SQL Server Reporting Services, entre otros.



Figura 9. Logo SSMS

Se priorizó esta herramienta sobre otras para gestionar la base de datos debido a la experiencia previa (en SSMS14) que se extrapoló a la versión actual SSMS 20, permitiendo un desempeño mayor y en menor tiempo de lo que se tardaría en familiarizarse con otra herramienta como DBeaver o HeidiSQL.

2.5.3 GitHub

GitHub es una plataforma en línea que permite el almacenaje, compartición y colaboración en proyectos de software por parte de la comunidad. Está basado en “el sistema de control de versiones Git” y fue lanzado en 2008, siendo actualmente propiedad de Microsoft [\[15\]](#).



Figura 10. Logo GitHub

Se contemplaba esta elección desde el principio del trabajo, ya que es una de las herramientas más útiles para gestionar el control de versiones mediante la creación de un repositorio en el que subir los cambios (*push*) implementados dentro de la aplicación a la finalización de cada sprint.

Especificación y Análisis

La aplicación se concibe para su uso de un único rol, usuario, que gestione las entradas y salidas de mercancía del almacén.

3.1 Requisitos funcionales

3.1.1 Requisitos funcionales del sistema

Identificador	Nombre	Descripción
RF1	Información del usuario.	El sistema deberá almacenar en la base de datos toda la información referente al usuario.
RF2	Información del almacén.	El sistema deberá almacenar en la base de datos toda la información referente al almacén.
RF3	Información de los productos.	El sistema deberá almacenar en la base de datos toda la información referente a los productos.

RF4	Información de las facturas.	El sistema deberá almacenar en la base de datos toda la información referente a las facturas.
RF5	Información de las estanterías.	El sistema deberá almacenar en la base de datos toda la información referente a las estanterías.

3.1.2 Requisitos Funcionales del usuario

Identificador	Nombre	Descripción
RF6	Registro de usuario.	Un usuario debe poder crear una cuenta a partir de un nombre, apellido, correo electrónico y contraseña válidos.
RF7	Inicio de sesión.	Un usuario debe poder iniciar sesión a partir de su correo electrónico y su contraseña.
RF8	Cierre de sesión.	Un usuario debe poder cerrar su sesión anteriormente iniciada.
RF9	Crear un almacén.	Un usuario debe poder crear un almacén con las dimensiones que desee.
RF10	Gestionar almacén.	Un usuario debe poder editar y eliminar el almacén, si este existe.
RF11	Subir factura.	Un usuario debe poder subir una factura si las condiciones son válidas.
RF12	Gestionar productos.	Un usuario debe poder editar y eliminar los productos que estén en stock.
RF13	Añadir clientes y proveedores.	Un usuario debe poder crear nuevos clientes y proveedores dentro de la aplicación.
RF14	Gestión de clientes y proveedores.	Un usuario debe poder editar y eliminar los clientes y

		proveedores que existan en la aplicación.
RF15	Crear estanterías.	Un usuario debe poder crear nuevas estanterías dentro de la aplicación.
RF16	Gestionar estanterías.	Un usuario debe poder editar y eliminar las estanterías que existan en la aplicación.
RF17	Gestionar productos en estanterías.	Un usuario debe poder introducir, eliminar o transferir productos de una estantería a otra existente.
RF18	Gestionar mapa del almacén.	Un usuario debe poder visualizar, colocar, reubicar y eliminar las estanterías dentro del mapa del almacén.
RF19	Visualizar información de las estanterías en el mapa del almacén.	Un usuario debe poder visualizar la información de los productos que contienen las estanterías mostradas en el mapa del almacén.
RF20	Gestionar disposición de las estanterías.	Un usuario debe poder visualizar, ubicar y redimensionar el espacio ocupado por un producto dentro de una estantería.
RF21	Consultar productos	Un usuario debe poder buscar el producto deseado y visualizar las estanterías que lo contengan.

3.2 Requisitos no funcionales

Identificador	Nombre	Descripción
RNF1	Respuesta de la base de datos.	La realización de las consultas a la base de datos no debe de superar los 4 segundos.
RNF2	Correos únicos.	El sistema debe de controlar que todos los correos

		electrónicos de los usuarios sean únicos dentro de la aplicación.
RNF3	Seguridad.	El sistema debe de controlar que cada usuario solo pueda acceder a la información relacionada con su cuenta.
RNF4	Número de la factura único.	El sistema debe de controlar que el número de cada factura debe ser único.
RNF5	Número de la estantería único.	El sistema debe de controlar que el número de cada estantería debe ser único.
RNF6	Interfaces organizables	El sistema debe de permitir la organización alfabética o numérica de los productos y las facturas.

3.3 Casos de uso

3.3.1 Registrarse

Título	Registro de usuario.
Actor	Usuario.
Descripción	El usuario crea una cuenta en la aplicación.
Pre-Condición	El usuario debe haber desplegado la aplicación y haber hecho clic en el botón “Registrarse”.
Post-Condición	El usuario habrá creado su cuenta en la aplicación.
Escenario Principal	1. El sistema muestra la pantalla con el formulario de registro. 2. El usuario rellena un nombre, apellido, correo electrónico y una contraseña. 3. El usuario pulsa el botón “REGISTRARSE”. 4. El sistema valida los datos. 5. El sistema comprueba que el correo electrónico no esté ya en uso.

	6. El sistema guarda los datos en la base de datos. 7. El sistema redirige al usuario a la pantalla de inicio de sesión.
Escenario Alternativo	2a. El usuario hace clic en “Iniciar sesión”. 2a.1 El sistema redirige a la página de inicio de sesión. 2b. El usuario hace clic en “Volver”. 2b.1 El sistema redirige a la página principal de la aplicación si haber iniciado sesión. 2c. El usuario introduce un correo electrónico ya en uso. 2c.1 El sistema marca en rojo el marco del formulario en el que se introduce el correo electrónico y se indica “Este email ya está registrado”. 2d. El usuario introduce la confirmación de la contraseña que no coincide. 2d.1 El sistema marca en rojo el marco del formulario en el que se introduce la confirmación de la contraseña y se indica “Las contraseñas no coinciden”. 4a. El sistema detecta el campo no válido. 4a.1 El sistema marca en rojo el marco del formulario en el que se introduce el nombre y se indica lo que debe de cumplir. 4b. El sistema detecta los campos no válidos. 4b.1 El sistema marca en rojo los marcos del formulario que son no válidos e indican lo que deben de cumplir.

3.3.2 Iniciar sesión

Título	Inicio de sesión.
Actor	Usuario.
Descripción	El usuario inicia sesión en la aplicación.
Pre-Condición	El usuario debe haberse registrado en la aplicación, y haber hecho clic en “INICIAR SESIÓN”
Post-Condición	El usuario entrará en la página principal de su cuenta dentro de la aplicación.
Escenario Principal	1. El sistema muestra la pantalla con el formulario de inicio de sesión. 2. El usuario rellena los campos de correo electrónico y la contraseña. 3. El usuario pulsa el botón “INICIAR SESIÓN”. 4. El sistema valida los datos. 5. El sistema redirige al usuario a la página principal de su cuenta.
Escenario Alternativo	2a. El usuario hace clic en “Registrarse”. 2a.1 El sistema redirige a la página de registro. 2b. El usuario hace clic en “Volver”. 2b.1 El sistema redirige a la página principal de la aplicación si haber iniciado sesión. 4a. El sistema reconoce el no haber introducido el domino del correo electrónico. 4a.1 El sistema despliega un mensaje de alerta indicando “Incluye un signo “@” en la dirección de correo electrónico. La dirección “correo introducido” no incluye el signo “@” ”. 4b. El sistema reconoce que el correo electrónico no se encuentra en la base de datos.

	4b.1 El sistema marca en rojo el marco del formulario en el que se introduce el correo electrónico y se indica “No existe ninguna cuenta con este correo electrónico”. 4c. El sistema reconoce que la contraseña del correo electrónico es incorrecta. 4c.1 El sistema marca en rojo el marco del formulario en el que se introduce la contraseña y se indica “Contraseña incorrecta”.
--	---

3.3.3 Cierre de sesión

Título	Cierre de sesión.
Actor	Usuario.
Descripción	El usuario cierra la sesión en la aplicación.
Pre-Condición	El usuario debe tener la sesión iniciada y haber hecho clic en el icono que aparece en la parte superior derecha, que es el perfil.
Post-Condición	El usuario estará en la página principal de la aplicación con la sesión sin iniciar.
Escenario Principal	1. El sistema muestra la pestaña con la información del perfil del usuario. 2. El usuario pulsa el botón “Cerrar sesión”. 3. El sistema valida la petición. 4. El sistema redirige al usuario a la página principal de la aplicación con la sesión cerrada.
Escenario Alternativo	2a. El usuario hace clic en fuera del recuadro del perfil. 2a.1 El sistema muestra la página principal con la sesión encendida.

3.3.4 Subir factura cliente

Título	Subir factura de un cliente.
Actor	Usuario.
Descripción	El usuario sube una factura de tipo cliente.
Pre-Condición	El usuario debe haber creado al cliente en la aplicación, haber hecho clic en “PROCESAR PDF” y hacer clic en “FACTURA DE CLIENTE”.
Post-Condición	El sistema mostrará en la pestaña facturas, dentro del cliente de la factura una nueva factura con los datos de la misma. Además, se actualizará la cantidad de cajas de los productos de la factura en stock y los colocados en las estanterías.
Escenario Principal	1. El sistema muestra una pestaña para que el usuario seleccione de su ordenador el archivo pdf de la factura. 2. El usuario selecciona el archivo pdf de la factura. 3. El usuario pulsa el botón “Abrir”. 4. El sistema carga la factura. 5. El sistema muestra los datos de la factura. 6. El usuario hace clic en “GUARDAR”. 7. EL sistema valida los datos. 8. El sistema muestra una alerta indicando “Factura guardada correctamente con (n · productos cargados) de (n · total de productos de la factura) productos. 9. El sistema redirige al usuario a la pestaña Facturas.
Escenario Alternativo	3a. El usuario pulsa el botón “cancelar”. 3a.1 El sistema cierra la pestaña y muestra la página de procesar facturas pdf. 6a. El usuario pulsa el botón “CANCELAR”.

	6a.1 El sistema redirige al usuario a la página principal de la aplicación, manteniendo la sesión iniciada. 7a El sistema detecta datos incorrectos en el cliente de la factura. 7a.1 El sistema muestra un mensaje indicando el error. 7a.2 El sistema redirige al usuario a la página principal de la aplicación, manteniendo la sesión iniciada.
--	---

3.3.5 Crear Almacén

Título	Crear Almacén
Actor	Usuario.
Descripción	El usuario crea un mapa del almacén.
Pre-Condición	El usuario debe estar en la página principal, haber hecho clic en “GESTIONAR” y en “CREAR NUEVO ALMACÉN”.
Post-Condición	El sistema mostrará la página principal con el mapa del almacén.
Escenario Principal	1. El sistema muestra una ventana en la que seleccionar el nombre y dimensiones. 2. El usuario indica el nombre del almacén. 3. El usuario indica las dimensiones del almacén. 4. El usuario hace clic en el botón “CREAR ALMACÉN” 5. El sistema muestra la página principal con el mapa del almacén.
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema muestra la página principal sin el mapa del almacén. 3a. El usuario arrastra el cuadrado naranja fuera del mapa del almacén y lo suelta.

	3a.1 El sistema procesa la posición, comprueba que está fuera de los límites. 3a.2 El sistema vuelve a mostrar el modo creación sin haber hecho ninguna modificación.
--	---

3.3.6 Editar Almacén

Título	Editar almacén
Actor	Usuario.
Descripción	El usuario edita el almacén.
Pre-Condición	El usuario debe haber creado el almacén, haber abierto el menú de gestión del almacén y hacer clic en “Editar Almacén”.
Post-Condición	El sistema mostrará la página principal con el mapa del almacén actualizado.
Escenario Principal	1. El sistema muestra una ventana para seleccionar el almacén. 2. El usuario selecciona el almacén a editar. 3. El usuario introduce las nuevas dimensiones y nombre del mapa del almacén. 4. El usuario hace clic en el botón “GUARDAR CAMBIOS” 5. El sistema verifica las dimensiones y el nombre. 6. El sistema actualiza el mapa del almacén. 7. El sistema muestra la pestaña principal.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.7 Eliminar Almacén

Título	Eliminar almacén
Actor	Usuario.
Descripción	El usuario elimina el almacén.
Pre-Condición	El usuario debe haber creado el almacén, haber abierto el menú de gestión del almacén y hacer clic en “Eliminar Almacén”.
Post-Condición	El sistema mostrará la página principal sin el almacén.
Escenario Principal	1. El sistema muestra una ventana para seleccionar el almacén. 2. El usuario selecciona el almacén a eliminar. 3. El usuario hace clic en el botón “ELIMINAR”. 4. El sistema muestra una ventana emergente. 5. EL usuario hace clic en el botón “CONFIRMAR ELIMINACIÓN”. 6. El sistema elimina el mapa del almacén. 7. El sistema muestra la pestaña principal.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas. 5a. El usuario pulsa el botón “CANCELAR”. 5a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.8 Crear estantería

Título	Crear Estantería
Actor	Usuario.
Descripción	El usuario crea una estantería colocándola en el mapa del almacén.

Pre-Condición	El usuario debe haber creado el almacén, estar en la página de edición del mapa del almacén y haber pulsado el botón “Crear Estantería”.
Post-Condición	El sistema mostrará el mapa del almacén con la nueva estantería en él.
Escenario Principal	1. El sistema muestra un recuadro naranja, siendo la estantería, la capacidad máxima y un botón de cancelar. 2. El usuario indica la cantidad máxima de cajas de productos que caben en la estantería. 3. El usuario arrastra el cuadrado naranja dentro del mapa del almacén y lo suelta en la celda que desee. 4. El sistema guarda la posición establecida. 5. El sistema muestra el mapa con la nueva estantería introducida.
Escenario Alternativo	1a. El usuario pulsa el botón “Cancelar Creación”. 1a.1 El sistema cierra el modo creación. 3a. El usuario arrastra el cuadrado naranja fuera del mapa del almacén y lo suelta. 3a.1 El sistema procesa la posición, comprueba que está fuera de los límites. 3a.2 El sistema vuelve a mostrar el modo creación sin haber hecho ninguna modificación.

3.3.9 Reubicar estantería

Título	Reubicar estantería
Actor	Usuario.
Descripción	El usuario reubica una estantería dentro del mapa del almacén.
Pre-Condición	El usuario debe haber creado el almacén, este contener mínimo una estantería, estar en la

	página de edición del mapa del almacén y haber pulsado el botón “Reubicar Estantería”.
Post-Condición	El sistema mostrará el mapa del almacén con la estantería en la nueva posición.
Escenario Principal	1. El sistema muestra el modo reubicar estantería activado. 2. El usuario selecciona una estantería del almacén. 3. El sistema guarda la estantería que se ha seleccionado. 4. El usuario arrastra esa estantería a otra celda dentro del mapa del almacén. 5. El sistema guarda la nueva posición de la estantería. 6. El sistema muestra el mapa con la nueva ubicación de la estantería.
Escenario Alternativo	1a. El usuario pulsa el botón “Cancelar Reubicación”. 1a.1 El sistema cierra el modo reubicar. 4a. El usuario arrastra el cuadrado azul fuera del mapa del almacén y lo suelta. 4a.1 El sistema procesa la posición, comprueba que está fuera de los límites. 4a.2 El sistema vuelve a mostrar el modo creación sin haber hecho ninguna modificación.

3.3.10 Eliminar estantería

Título	Eliminar estantería
Actor	Usuario.
Descripción	El usuario elimina una estantería dentro del mapa del almacén.
Pre-Condición	El usuario debe haber creado el almacén, este contener mínimo una estantería, estar en la página de edición del mapa del almacén y haber pulsado el botón “Eliminar Estantería”.

Post-Condición	El sistema mostrará el mapa del almacén sin la estantería seleccionada anteriormente.
Escenario Principal	1. El sistema muestra el modo eliminar estantería activado. 2. El usuario selecciona una estantería del almacén. 3. El sistema guarda la estantería que se ha seleccionado. 4. El usuario hace clic en el botón “Eliminar estantería” 5. El sistema muestra una ventana emergente. 6. EL usuario hace clic en el botón “ELIMINAR”. 7. El sistema muestra el mapa sin la estantería seleccionada anteriormente.
Escenario Alternativo	1a. El usuario pulsa el botón “Cancelar Eliminación”. 1a.1 El sistema cierra el modo eliminar. 6a. El usuario hace clic en el botón “CANCELAR”. 6a.2 El sistema vuelve a mostrar el modo eliminación sin haber hecho ninguna modificación y seleccionando aún la estantería a eliminar.

3.3.11 Crear cliente o proveedor

Título	Crear cliente o proveedor
Actor	Usuario.
Descripción	El usuario crea un cliente o proveedor de su almacén.
Pre-Condición	El usuario debe estar en la pestaña de facturas y haber hecho clic en el icono “+” ya sea de cliente o de proveedor.
Post-Condición	El sistema mostrará la página de facturas con el cliente o proveedor creado.

Escenario Principal	1. El sistema muestra una ventana con los campos a rellenar. 2. El usuario rellena los campos. 3. El usuario hace clic en “INSERTAR” 4. El sistema valida los datos ingresados en los campos de la ventana. 5. El sistema guarda el cliente o proveedor que se ha creado. 6. El sistema muestra la pestaña facturas.
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema cierra el listado de clientes o de proveedores. 4a. El sistema muestra error por campos vacíos. 4a.1 El usuario rellena los datos. 4b. El sistema muestra error en el campo teléfono por no tener 9 dígitos. 4b.1 El usuario rellena correctamente el teléfono.

3.3.12 Editar cliente o proveedor

Título	Editar cliente o proveedor
Actor	Usuario.
Descripción	El usuario editar un cliente o proveedor de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, haber pulsado el botón con el icono del lápiz.
Post-Condición	El sistema mostrará la página de facturas con el cliente o proveedor seleccionada anteriormente actualizado.
Escenario Principal	1. El sistema muestra un listado de clientes o proveedores.

	2. El usuario selecciona al cliente o proveedor a editar. 3. El usuario hace clic en el botón “EDITAR”. 4. El sistema guarda el cliente o proveedor que se ha seleccionado. 5. El sistema muestra los datos del cliente o proveedor seleccionado. 6. El usuario edita los datos del cliente o proveedor seleccionado. 7. El usuario hace clic en el botón “GUARDAR” 8. El sistema valida los datos introducidos. 9. El sistema edita al cliente o proveedor 10. El sistema muestra la pestaña facturas
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema cierra el listado de clientes o de proveedores. 8a. El sistema muestra error por campos vacíos. 8a.1 El usuario rellena los datos. 8b. El sistema muestra error en el campo teléfono por no tener 9 dígitos. 8b.1 El usuario rellena correctamente el teléfono.

3.3.13 Eliminar cliente o proveedor

Título	Eliminar cliente o proveedor
Actor	Usuario.
Descripción	El usuario elimina un cliente o proveedor de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, haber pulsado el botón con el icono de la papelera.
Post-Condición	El sistema mostrará la página de facturas sin el cliente o proveedor seleccionada anteriormente.

Escenario Principal	1. El sistema muestra un listado de clientes o proveedores. 2. El usuario selecciona al cliente o proveedor a eliminar. 3. El sistema guarda el cliente o proveedor que se ha seleccionado. 4. El usuario hace clic en el botón “Eliminar” 5. El sistema muestra una ventana emergente. 6. EL usuario hace clic en el botón “CONTINUAR”. 7. El sistema muestra una nueva ventana emergente. 8. EL usuario hace clic en el botón “ELIMINAR DEFINITIVAMENTE”. 9. El sistema elimina el cliente o proveedor 10. El sistema muestra la pestaña facturas
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema cierra el listado de clientes o de proveedores. 4a. El usuario pulsa el botón “CANCELAR”. 4a.1 El sistema cierra la ventana y muestra la pestaña facturas. 6a. El usuario pulsa el botón “CANCELAR”. 6a.1 El sistema cierra la ventana y muestra la pestaña facturas. 8a. El usuario pulsa el botón “CANCELAR”. 8a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.14 Editar factura

Título	Eliminar factura
Actor	Usuario.
Descripción	El usuario edita una factura de un cliente o proveedor de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, dentro de este haber subido una factura, seleccionar la factura a editar y haber pulsado el botón con el icono del lápiz.
Post-Condición	El sistema mostrará la página de facturas con la factura del cliente o proveedor seleccionada anteriormente actualizada.
Escenario Principal	1. El sistema guarda la factura del cliente o proveedor que se ha seleccionado. 2. El sistema muestra una ventana con los datos de la factura 3. El usuario edita los datos de la factura. 4. EL usuario hace clic en el botón “GUARDAR”. 5. El sistema actualiza la factura del cliente o proveedor. 6. El sistema muestra la pestaña facturas.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.15 Eliminar factura

Título	Eliminar factura
Actor	Usuario.
Descripción	El usuario elimina una factura de un cliente o proveedor de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, dentro de este haber subido una

	factura, seleccionar la factura a eliminar y haber pulsado el botón con el icono de la papelera.
Post-Condición	El sistema mostrará la página de facturas sin la factura del cliente o proveedor seleccionada anteriormente.
Escenario Principal	1. El sistema guarda la factura del cliente o proveedor que se ha seleccionado. 2. El sistema muestra una ventana emergente. 3. EL usuario hace clic en el botón “CONTINUAR”. 4. El sistema muestra una nueva ventana emergente. 5. EL usuario hace clic en el botón “ELIMINAR DEFINITIVAMENTE”. 6. El sistema elimina la factura del cliente o proveedor. 7. El sistema muestra la pestaña facturas.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas. 5a. El usuario pulsa el botón “CANCELAR”. 5a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.16 Editar producto

Título	Editar producto
Actor	Usuario.
Descripción	El usuario edita un producto de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, dentro de este haber subido una factura con productos, seleccionar en la pestaña productos el producto a editar y haber pulsado el botón con el icono del lápiz.

Post-Condición	El sistema mostrará la página de productos con el producto seleccionado anteriormente actualizado.
Escenario Principal	1. El sistema guarda el producto seleccionado. 2. El sistema muestra una ventana con los datos del producto. 3. El usuario edita los datos del producto. 4. EL usuario hace clic en el botón “EDITAR”. 5. El sistema actualiza el producto. 6. El sistema muestra la pestaña productos.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.17 Eliminar Producto

Título	Eliminar producto
Actor	Usuario.
Descripción	El usuario elimina un producto de su almacén.
Pre-Condición	El usuario debe haber creado al cliente o proveedor, dentro de este haber subido una factura con productos, seleccionar en la pestaña productos el producto a eliminar y haber pulsado el botón con el icono de la papelera.
Post-Condición	El sistema mostrará la página de productos sin el producto seleccionado anteriormente.
Escenario Principal	1. El sistema guarda el producto que se ha seleccionado. 2. El sistema muestra una ventana emergente. 3. EL usuario hace clic en el botón “CONTINUAR”. 4. El sistema muestra una nueva ventana emergente.

	5. EL usuario hace clic en el botón “ELIMINAR DEFINITIVAMENTE”. 6. El sistema elimina el producto. 7. El sistema muestra la pestaña productos.
Escenario Alternativo	3a. El usuario pulsa el botón “CANCELAR”. 3a.1 El sistema cierra la ventana y muestra la pestaña facturas. 5a. El usuario pulsa el botón “CANCELAR”. 5a.1 El sistema cierra la ventana y muestra la pestaña facturas.

3.3.18 Añadir producto a estantería

Título	Añadir producto de estantería
Actor	Usuario.
Descripción	El usuario añade un producto de una estantería.
Pre-Condición	El usuario debe haber creado el almacén, este contener mínimo una estantería, estar en la pestaña estanterías y haber pulsado el icono “+” de la estantería que se quiera añadir un producto.
Post-Condición	El sistema mostrará la pestaña estanterías y la estantería seleccionada contendrá las cajas indicadas que se añadieron del producto
Escenario Principal	1. El sistema muestra una ventana para elegir que producto desea añadir. 2. El usuario selecciona el producto a añadir. 3. El sistema muestra en la ventana un input para introducir la cantidad a añadir. 4. El usuario introduce la cantidad. 5. El usuario hace clic en el botón “INSERTAR PRODUCTO” 6. El sistema valida la cantidad. 7. El sistema muestra la pestaña estanterías con la estantería sin el producto que había sido seleccionado para ser retirado.

Escenario Alternativo	1a. El usuario pulsa el botón “CANCELAR”. 1a.1 El sistema cierra el modo eliminar.
------------------------------	--

3.3.19 Retirar producto de estantería

Título	Retirar producto de estantería
Actor	Usuario.
Descripción	El usuario retira un producto de una estantería.
Pre-Condición	El usuario debe haber creado el almacén, este contener mínimo una estantería, estar en la pestaña estanterías y haber pulsado el icono “-” de la estantería que se quiera retirar el producto.
Post-Condición	El sistema mostrará la pestaña estanterías y la estantería seleccionada no contendrá las cajas indicadas que se retiraron, o el producto en caso de retirar todo el stock de la estantería.
Escenario Principal	1. El sistema muestra una ventana para elegir que producto desea retirar. 2. El usuario selecciona el producto a retirar. 3. El sistema muestra en la ventana un input para introducir la cantidad a retirar. 4. El usuario introduce la cantidad. 5. El usuario hace clic en el botón “RETIRAR PRODUCTO” 6. El sistema valida la cantidad. 7. El sistema muestra la pestaña estanterías con la estantería sin el producto que había sido seleccionado para ser retirado.
Escenario Alternativo	1a. El usuario pulsa el botón “CANCELAR”. 1a.1 El sistema cierra el modo eliminar.

3.3.20 Transferir producto entre estantería

Título	Transferir producto entre estantería
Actor	Usuario.
Descripción	El usuario mueve un producto de una estantería a otra.
Pre-Condición	El usuario debe haber creado el almacén, este contener mínimo dos estanterías, estar en la pestaña estanterías y haber pulsado el botón “TRANSFERIR PRODUCTOS”
Post-Condición	El sistema mostrará la pestaña estanterías y las estanterías habrán actualizado el stock que contienen.
Escenario Principal	1. El sistema muestra una ventana para elegir la estantería origen. 2. El usuario selecciona la estantería origen mediante el botón “SELECCIONAR EN EL MAPA”. 3. El sistema muestra en la ventana con el mapa del almacén y sus estanterías. 4. El usuario hace clic en la estantería que tiene los productos a mover. 5. El sistema muestra el mapa del almacén con las estanterías y la estantería seleccionada como origen en color verde. 6. El usuario hace clic en la estantería a la que se deben mover los productos. 7. El sistema muestra una ventana con la estantería origen y destino seleccionadas. 8. El usuario hace clic en el botón “SIGUIENTE” 9. El sistema muestra una ventana con los productos que contiene la estantería. 10. El usuario hace clic en los productos que desea transferir. 11. El sistema permite introducir la cantidad de cajas que desea transferir. 12. El usuario hace clic en el botón “SIGUIENTE”.

	13. EL sistema muestra una ventana de resumen de transferencia. 14. El usuario hace clic en el botón “TRANSFERIR PRODUCTOS”. 15. El sistema actualiza los productos que contiene cada estantería. 16. El sistema muestra la pestaña estanterías con las estanterías actualizadas.
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema cierra el modo transferir. 4a. El usuario pulsa el botón “CANCELAR”. 4a.1 El sistema cierra el modo transferir. 5a. El usuario pulsa el botón “CANCELAR”. 5a.1 El sistema cierra el modo transferir. 8a. El usuario pulsa el botón “CANCELAR”. 8a.1 El sistema cierra el modo transferir. 10a. El usuario pulsa el botón “ATRÁS”. 10a.1 El sistema vuelve a la pestaña anterior. 10a.2 El usuario pulsa el botón “CANCELAR”. 10a.3 El sistema cierra el modo transferir. 14a. El usuario pulsa el botón “ATRÁS”. 14a.1 El sistema vuelve a la pestaña anterior. 14a.2 El usuario pulsa el botón “ATRÁS”. 14a.3 El sistema vuelve a la pestaña anterior. 14a.4 El usuario pulsa el botón “CANCELAR”. 14a.5 El sistema cierra el modo transferir.

3.3.21 Subir imagen de producto

Título	Subir imagen de producto
Actor	Usuario.
Descripción	El usuario sube la imagen de un producto.
Pre-Condición	El usuario debe haber subido una factura con al menos un producto, encontrarse en la pestaña productos y haber hecho clic en el icono de una cámara.
Post-Condición	El sistema mostrará la pestaña productos con una imagen del mismo.
Escenario Principal	1. El sistema muestra una ventana para subir una imagen del producto. 2. El usuario hace clic en el botón “SELECCIONAR IMAGEN”. 3. El sistema muestra una ventana para seleccionar la imagen desde sus archivos. 4. El usuario hace clic en “Abrir”. 5. El sistema muestra una ventana con la previsualización de la imagen que se ha seleccionado. 6. El usuario hace clic en el botón “SUBIR IMAGEN”. 7. El sistema valida la imagen. 8. El sistema muestra una alerta de confirmación de subida. 9. El sistema muestra la pestaña productos con la imagen del producto visible.
Escenario Alternativo	2a. El usuario pulsa el botón “CANCELAR”. 2a.1 El sistema cierra el modo eliminar. 4a. El usuario pulsa el botón “Cancelar”. 4a.1 El sistema cierra el modo eliminar. 6a. El usuario pulsa el botón “CANCELAR”. 6a.1 El sistema cierra el modo eliminar.

3.3.22 Cambiar imagen de producto

Título	Cambiar imagen de producto
Actor	Usuario.
Descripción	El usuario cambia la imagen de un producto.
Pre-Condición	El usuario debe haber subido una factura con al menos un producto, encontrarse en la pestaña productos, haber subido una imagen del producto y haber hecho clic en el icono de una cámara.
Post-Condición	El sistema mostrará la pestaña productos con la imagen actualizada.
Escenario Principal	1. El sistema muestra una ventana con la imagen del producto. 2. El usuario hace clic en el botón “CAMBIAR IMAGEN”. 3. El sistema muestra una ventana para subir una imagen del producto. 4. El usuario hace clic en el botón “SELECCIONAR IMAGEN”. 5. El sistema muestra una ventana para seleccionar la imagen desde sus archivos. 6. El usuario hace clic en “Abrir”. 7. El sistema muestra una ventana con la previsualización de la imagen que se ha seleccionado. 8. El usuario hace clic en el botón “SUBIR IMAGEN”. 9. El sistema valida la imagen. 10. El sistema muestra una alerta de confirmación de subida. 11. El sistema muestra la pestaña productos con la imagen del producto visible.
Escenario Alternativo	2a. El usuario pulsa el botón “CERRAR”. 2a.1 El sistema cierra el modo eliminar. 4a. El usuario pulsa el botón “Cancelar”. 4a.1 El sistema cierra el modo eliminar.

	6a. El usuario pulsa el botón “CANCELAR”. 6a.1 El sistema cierra el modo eliminar. 8a. El usuario pulsa el botón “CANCELAR”. 8a.1 El sistema cierra el modo eliminar.
--	---

3.3.23 Eliminar imagen de producto

Título	Eliminar imagen de producto
Actor	Usuario.
Descripción	El usuario elimina la imagen de un producto.
Pre-Condición	El usuario debe haber subido una factura con al menos un producto, encontrarse en la pestaña productos, haber subido una imagen del producto y haber hecho clic en el icono de una cámara.
Post-Condición	El sistema mostrará la pestaña productos sin una imagen como icono del producto.
Escenario Principal	1. El sistema muestra una ventana con la imagen del producto. 2. El usuario hace clic en el botón “ELIMINAR IMAGEN”. 3. El sistema muestra una ventana para eliminar una imagen del producto. 4. El usuario hace clic en el botón “ELIMINAR DEFINITIVAMENTE”. 5. El sistema valida la eliminación de la imagen. 6. El sistema muestra una alerta de confirmación de eliminación. 7. El sistema muestra la pestaña productos con la imagen del producto visible.
Escenario Alternativo	2a. El usuario pulsa el botón “CERRAR”. 2a.1 El sistema cierra el modo eliminar. 4a. El usuario pulsa el botón “CANCELAR”.

	4a.1 El sistema cierra el modo eliminar.
--	--

3.3.24 Ubicar producto en estantería

Título	Ubicar producto en estantería
Actor	Usuario.
Descripción	El usuario ubica un producto dentro de una estantería.
Pre-Condición	El usuario debe haber subido una factura con al menos un producto, haber creado un mapa del almacén, tener como mínimo una estantería, encontrarse en la pestaña principal, haber hecho clic en una estantería y hacer clic en la pestaña “VISTA 2D”.
Post-Condición	El sistema mostrará la pestaña “VISTA 2D” con la ubicación del producto actualizada.
Escenario Principal	1. El sistema muestra una ventana con la disposición de la estantería seleccionada. 2. El usuario arrastra el producto y lo coloca en la celda que desee. 3. El sistema guarda la nueva posición del producto. 4. El sistema muestra en la ventana la nueva colocación del producto en la estantería.
Escenario Alternativo	2a. El usuario pulsa el botón “CERRAR”. 2a.1 El sistema cierra la ventana. 2a.2 El sistema muestra la pestaña principal. 2b. El usuario pulsa “shift” más clic en un producto. 2b.1 El sistema muestra una ventana para cambiar el tamaño. 2b.2 El usuario configura el tamaño que ocupa el producto dentro de la estantería. 2b.3 El sistema valida los datos de dimensión del producto. 2b.4 El usuario hace clic en el botón “APLICAR”.

	2b.5 El sistema muestra la ventana de la estantería con el nuevo tamaño del producto. 2b.2.a El usuario hace clic en el botón “CANCELAR”. 2b.2.a.1 El sistema muestra la ventana de la estantería.
--	---

3.3.25 Buscar producto en estantería

Título	Buscar producto en estantería
Actor	Usuario.
Descripción	El usuario busca un producto mediante un buscador, y el mapa indica en que estanterías se encuentra ese producto.
Pre-Condición	El usuario debe haber subido una factura con al menos un producto, haber creado un mapa del almacén, tener como mínimo una estantería, encontrarse en la pestaña principal, haber hecho clic en el buscador de “Buscar Productos”
Post-Condición	El sistema mostrará en el mapa del almacén las estanterías que contienen ese producto en color verde.
Escenario Principal	1. El sistema muestra una un recuadro para introducir el producto a buscar. 2. El usuario introduce el producto que desea buscar. 3. El sistema comprueba qué estanterías contiene ese producto. 4. El sistema marca en el mapa en color verde las estanterías que contienen a ese producto. 5. El usuario hace clic en la cruz del recuadro para introducir la búsqueda del producto. 6. El sistema marca en el mapa con sus estanterías de forma predeterminada.

Escenario Alternativo	4a. El sistema no marca en el mapa en color verde ninguna estantería por no estar colocado ese producto en ninguna de estas.
------------------------------	--

3.4 Diagramas de Secuencia

3.4.1 Introducción

Teniendo en cuenta la cantidad de casos de uso desarrollados, para los diagramas de secuencias nos vamos a centrar en las tres acciones más relevantes de la aplicación.

El actor será siempre el usuario, el cual interactuará con el sistema para representar las secuencias que se realizarán en cada uno de los procesos vistos a continuación.

3.4.2 Subir Factura

El diagrama de secuencias que nos muestra la Figura 11 nos indica las interacciones que ocurren entre el usuario y el sistema durante el proceso de subir una factura a la aplicación.

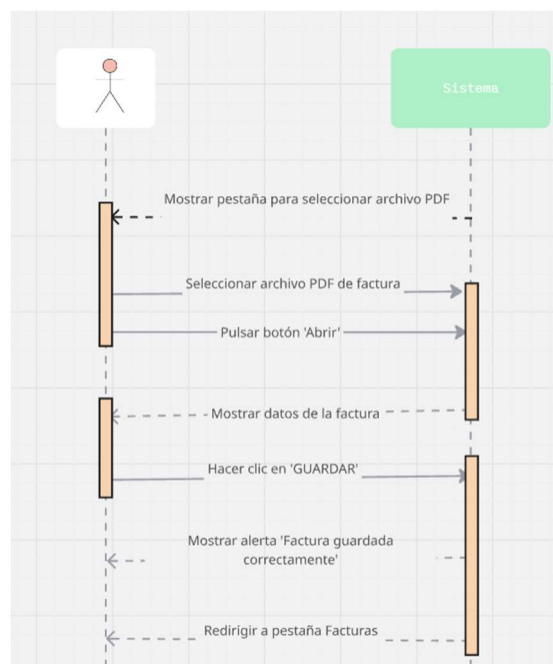


Figura 11. Diagrama de secuencias subir una factura

3.4.3 Buscar producto en estantería

En este diagrama de secuencias se muestra las interacciones desarrolladas durante el proceso de buscar un producto dentro de una estantería entre el usuario y el sistema.

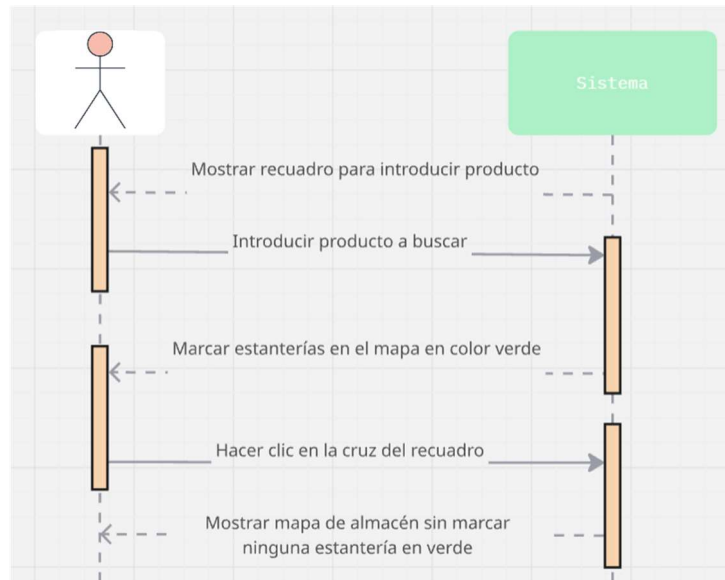


Figura 12. Diagrama secuencias buscar producto en estantería.

3.4.4 Transferir producto entre estantería

En este diagrama de secuencias se muestra las interacciones desarrolladas entre el usuario y el sistema que ocurren durante el proceso de transferir un producto colocado en una estantería a otra del almacén.

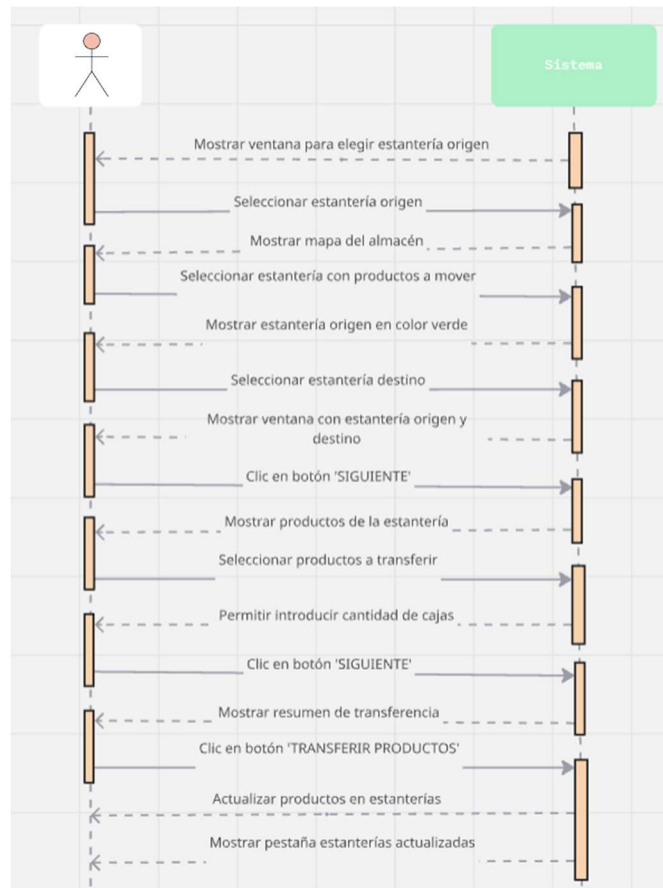


Figura 13. Diagrama de secuencias transferir productos de una estantería a otra.

3.5 Diagrama de flujo

3.5.1 Introducción

En este apartado se tratará los diagramas de flujo de los procesos explicados en el apartado anterior, permitiéndonos entender de una forma más visual el procedimiento a seguir para la realización de las tareas que se ejecutan en estos procesos.

3.5.2 Subir Factura

En el diagrama de flujo contenido en la Figura 14, se muestra visualmente los pasos que debe de seguir el usuario para realizar el proceso de subir una factura a la aplicación.

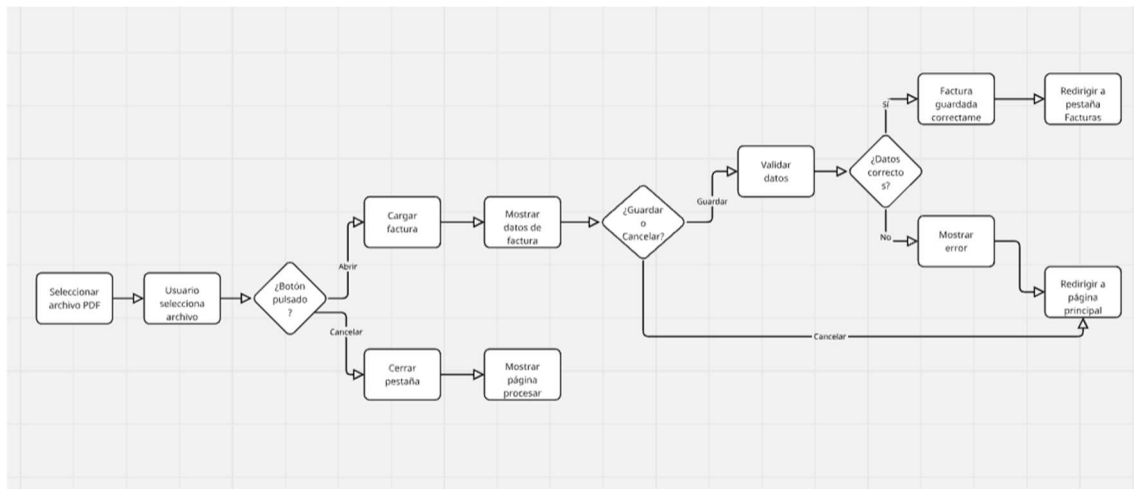


Figura 14. Diagrama de flujo subir factura

3.5.3 Buscar producto en estantería

En este diagrama de flujo se muestra visualmente los pasos que debe de seguir el usuario para realizar el proceso de buscar un producto dentro de las estanterías que se encuentran en el mapa del almacén.

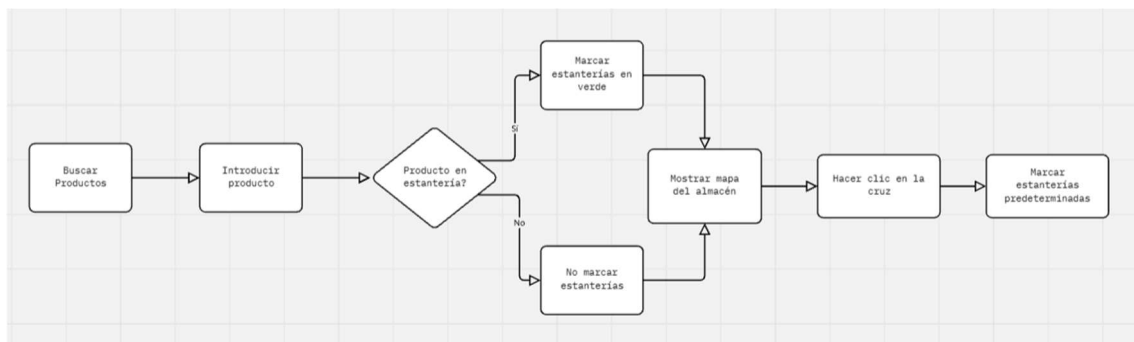


Figura 15. Diagrama de flujo buscar producto en estantería

3.5.4 Transferir producto entre estanterías

En este diagrama de flujo se muestra visualmente los pasos que debe de seguir el usuario para realizar el proceso de transferir productos que se encuentren en una estantería a otra diferente dentro del almacén.

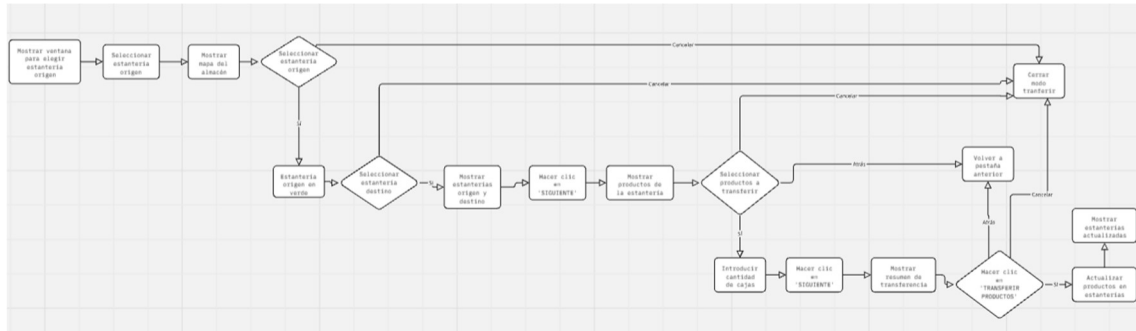


Figura 16. Diagrama de flujo transferir producto entre estanterías

Diseño del Sistema

4.1 Introducción

En este apartado se describe el diseño arquitectónico del sistema, así como el diseño visual de la aplicación, explicando las diversas pestañas pertenecientes a la aplicación. Indicando las funcionalidades que se pueden utilizar dentro de cada una de estas pestañas.

4.2 Diseño arquitectónico de la aplicación

4.2.1 Vista general del sistema

La aplicación desarrollada es un sistema web para la gestión de almacenes que permite a los usuarios registrar, visualizar y modificar información relacionada con productos, facturas, estanterías y ubicaciones en un entorno visual e interactivo.

Esta solución sigue una arquitectura full-stack, donde el front-end se basa a su vez en el patrón Modelo Vista Presentador (MVP), complementado con una arquitectura Modelo Controlador (MVC), para el back-end, separando claramente la lógica de negocio, la interfaz de usuario y el acceso a datos.

El estilo arquitectónico seguido en este proyecto es, por tanto, el de una arquitectura multicapas organizadas en tres niveles:

- Nivel de presentación: desarrollada en React, bajo el patrón MVP, que es responsable de mostrar la información al usuario y recibir sus interacciones. No contiene lógica de negocios.
- Nivel de lógica: actúa como intermediario entre el nivel de presentación y el nivel de datos, a través de un patrón MVC, gestionando la lógica de negocios, controlando los flujos de datos y coordinando las interacciones del usuario. Está implementado en Node.js con Express.
- Nivel de datos: representado mediante una base de datos SQL Server. Gestiona el acceso, almacenamiento y recuperación de datos del sistema.

4.2.2 Componentes y conectores

A continuación, se describen los principales componentes que forman el sistema y cómo interactúan entre ellos:

Frontend (React + MVP):

- Views: componentes visuales que representan las páginas (Login, Registro, Productos, Facturas, Estanterías, etc.).
- Presenters: clases encargadas de procesar la lógica de presentación, obteniendo los datos del modelo y actualizando la vista.
- Services: módulos que manejan las llamadas a la API REST usando Axios.
- Context y Hooks: para la gestión del estado global de la aplicación, como la sesión del usuario.
- UI/Utils: elementos comunes como botones, formularios, mensajes, validaciones, etc.

Backend (Node.js + Express):

- Rutas API (Controllers): definen los endpoints para operaciones como login, CRUD de productos, subida de facturas, etc.
- Lógica de negocio: incluida dentro de los controladores o servicios auxiliares para validación, cálculos o transformaciones necesarias.

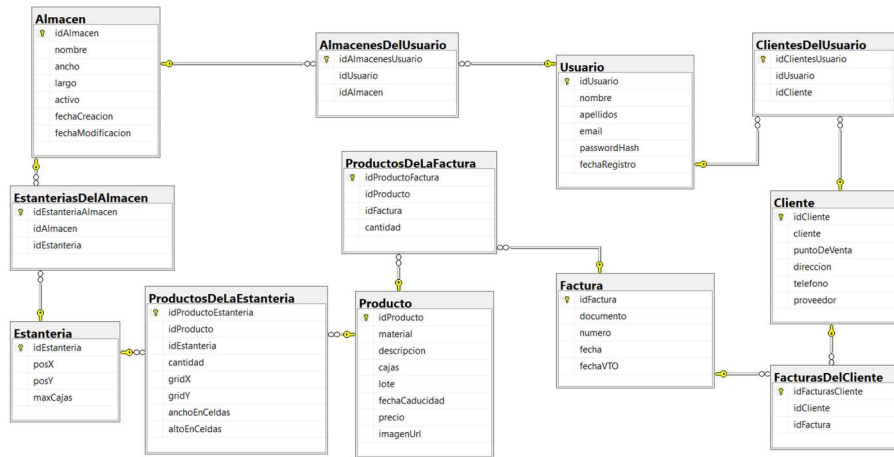
Base de datos (SQL Server):

- Almacena información estructurada relacionada con usuarios, productos, facturas, estanterías, almacenes y sus ubicaciones.
- Gestionada desde SQL Server Management Studio, con estructuras relacionales bien definidas y restricciones de integridad.

4.3 Diseño de la base de datos

En este apartado se explicará la estructura de la base de datos relacional utilizada, cuyo objetivo es almacenar la información dentro de la aplicación.

Se ha seguido un diseño en el que cada tabla contenga la información de una entidad específica. Para poder expresar las relaciones de MxM (o muchos a muchos) se han empleado tablas intermedias, las cuales solamente tengan los identificadores de las tablas que vinculan y algún atributo específico que se necesite de esa relación como se puede apreciar en la tabla “productosDeLaFactura” de la figura 17 con la utilización del atributo “cantidad”.



Las tablas encargadas de almacenar la información sobre las entidades que posteriormente se utilizan en la aplicación son las siguientes:

- **Almacén:** encargada de almacenar la información de las dimensiones, el nombre y las fechas de creación y modificación del almacén que tendrá el usuario de la aplicación.
- **Estantería:** encargada de almacenar la posición en la que se encuentra y la cantidad máxima de productos que pueden colocarse en una estantería.
- **Producto:** encargada de almacenar información relevante sobre un producto, como es la descripción, fecha de caducidad, precio del producto. Además, teniendo la posibilidad de almacenar la imagen del producto para que sea más fácil identificarlo.
- **Factura:** encargada de almacenar la información para identificar cada factura que se procese en la aplicación, mediante la columna *número*, e indicando que tipo de factura es mediante la columna *documento*.
- **Cliente:** encargada de almacenar la información relativa a los clientes o proveedores creados en la aplicación, obteniendo de estos su dirección, teléfono o punto de venta.

- **Usuario:** encargada de almacenar la información del usuario que utiliza la aplicación. Recogiendo su correo electrónico y contraseña para permitir su inicio de sesión y registro.

Las tablas encargadas de relacionar las tablas anteriores entre sí para lograr mantener la coherencia y la viabilidad de las funciones que se aplican en la aplicación son las siguientes:

- **EstanteriasDelAlmacen:** encargada de almacenar la relación que existe entre el almacén y las estanterías que pertenecen a ese almacén.
- **ProductosDeLaEstantería:** encargada de almacenar la relación que creamos entre una estantería y los productos que están colocados en dicha estantería. Almacenando a su vez, la cantidad colocada y posición ocupada de los productos en dicha estantería.
- **ProductosDeLaFactura:** encargada de almacenar la relación entre una factura y los productos asociados a ella, especificando la cantidad de cajas de cada producto correspondiente a dicha factura.
- **FacturasDelCliente:** encargada de almacenar la relación existente entre un cliente o proveedor y las facturas registradas hacia ese cliente o proveedor.
- **ClientesDelUsuario:** encargada de almacenar la relación entre el usuario que utiliza la aplicación y los clientes o proveedores que ha creado dentro de la misma.
- **AlmacenesDelUsuario:** encargada de almacenar la relación entre el usuario que utiliza la aplicación y el almacén (que debe de haber creado anteriormente).

4.4 Diseño visual de la aplicación

En este apartado se expondrá el diseño de las pantallas de la aplicación, junto con sus diversas funcionalidades.

Esta aplicación web está estructurada para el uso de un único rol, usuario, que gestione las entradas y salidas de mercancía del almacén. Diferenciando estas en: “entradas” como los productos que llegan al almacén mediante una factura de un proveedor, y “salidas” como los productos que salen del almacén mediante una factura de un cliente.

Este administrador tendrá las siguientes páginas para disponer del stock de su almacén.

4.4.1 Página principal

En esta página nos encontraremos la primera interacción posible con la aplicación. Cuando se entra en la aplicación web se mostrará la página mostrada en la figura 18. En esta, se indica que debe iniciar sesión para introducirse en la aplicación web. Teniendo la posibilidad de iniciar sesión o registrarse haciendo clic en la esquina superior derecha de la pantalla.



Figura 18. Página Home Sin inicio de sesión

Una vez iniciado sesión, se mostrará la siguiente página, como se puede apreciar en la figura 19. Teniendo la posibilidad de iniciar un procesamiento de facturas, gestionar el almacén, ver las facturas ya subidas, desplazarnos por la barra de navegación azul superior a otras pestañas, o visualizar el mapa con las estanterías, a la vez que poder buscar en ellas un producto.

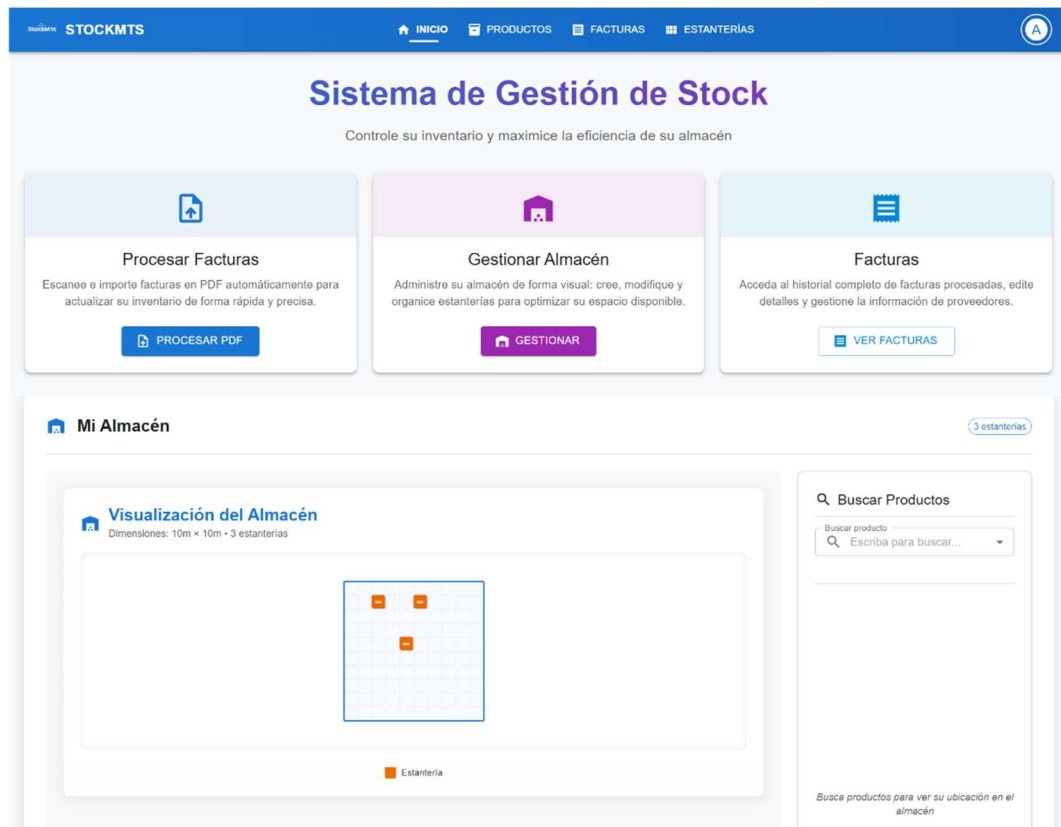


Figura 19. Página Home Iniciando Sesión

4.4.2 Página registro

En esta pantalla se realizará el proceso de registro en la aplicación web, teniendo que indicar la información correctamente en los campos que muestra la figura 20.

STOCKMTS REGISTRARSE INICIAR SESIÓN

Registro de usuario

Requisitos para registrarse:

- Todos los campos son obligatorios
- La contraseña debe cumplir con los criterios de seguridad
- El correo electrónico debe tener un formato válido

Nombre *
Al menos 2 caracteres

Apellidos *
Requerido

Correo electrónico *
Introduce un correo electrónico válido (ejemplo@dominio.com)

Contraseña *

- ☐ Al menos 6 caracteres
- ☐ Al menos una letra mayúscula
- ☐ Al menos una letra minúscula
- ☐ Al menos un número

Confirmar contraseña *

REGISTRARSE

Figura 20. Página registro

4.4.3 Página inicio sesión

En esta pestaña se encontrará el proceso inicio de sesión, necesario para introducirse en la aplicación web. Siendo esto posible tras haber realizado el proceso de registro en la misma. Teniendo que ingresar el correo y la contraseña con la que se registró.

STOCKMTS REGISTRARSE INICIAR SESIÓN

Iniciar sesión

Correo electrónico *

Contraseña *

INICIAR SESIÓN

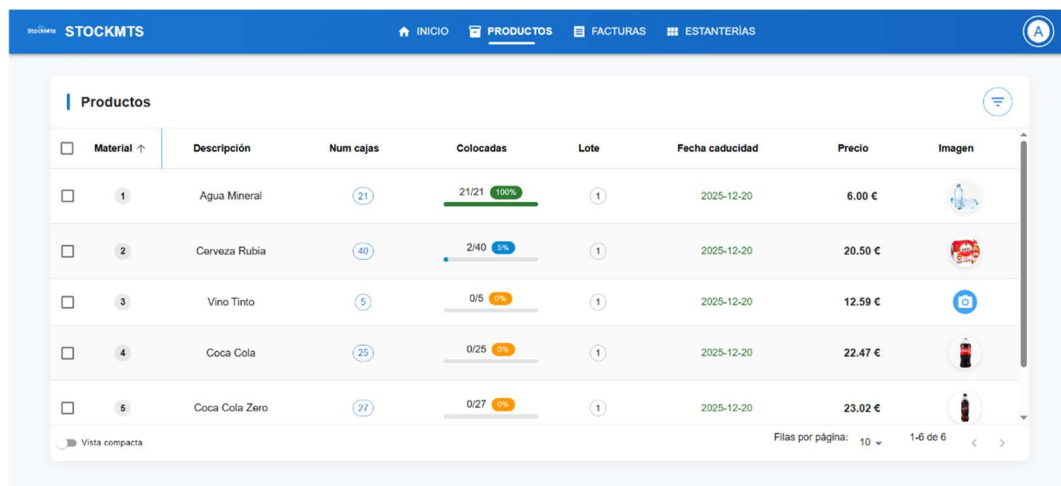
¿No tienes una cuenta? [Regístrate](#)

¿Quieres volver a la página principal? [Volver](#)

Figura 21. Página inicio de sesión

4.4.4 Página productos

En esta pestaña se muestran los productos que están en stock en el almacén en este momento, indicando la información sobre ellos, y destacando una barra que nos informa cuantas cajas hay colocadas en las estanterías de las cajas totales que hay de ese producto en stock. Pudiéndolos editar o eliminar, pero solo añadirlos mediante el procesamiento de facturas por parte de un proveedor válido.



☐ Material ↑	Descripción	Num cajas	Colocadas	Lote	Fecha caducidad	Precio	Imagen
☐ 1	Agua Mineral	21	21/21 100%	1	2025-12-20	6.00 €	
☐ 2	Cerveza Rubia	40	2/40 5%	1	2025-12-20	20.50 €	
☐ 3	Vino Tinto	5	0/5 0%	1	2025-12-20	12.59 €	
☐ 4	Coca Cola	25	0/25 0%	1	2025-12-20	22.47 €	
☐ 5	Coca Cola Zero	27	0/27 0%	1	2025-12-20	23.02 €	

Vista compacta Filas por página: 10 1-6 de 6

Figura 22. Página productos

4.4.5 Página facturas

En esta pestaña podemos obtener la información de los proveedores y clientes que tenemos en nuestro almacén, a la par que las facturas que tiene cada uno de estos. Pudiendo agregar nuevos proveedores y clientes, lo que permitirá subir sus facturas a la aplicación. A la vez que se podrá editar o eliminar tanto proveedores y clientes, como las facturas de los mismos.

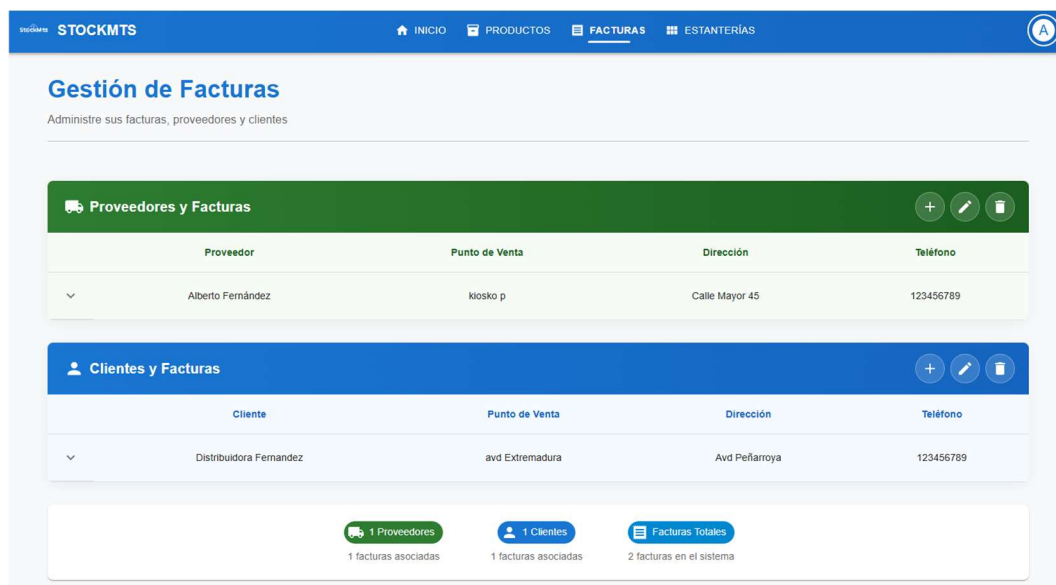


Figura 23. Página facturas

4.4.6 Página estanterías

En esta pestaña se podrá obtener información sobre las estanterías que se disponen en ese momento y sobre los productos que contiene, a la vez que se podrá colocar productos en las estanterías, eliminarlos de las mismas o trasferir la cantidad que se desee de ese producto, de una estantería origen a una estantería destino.

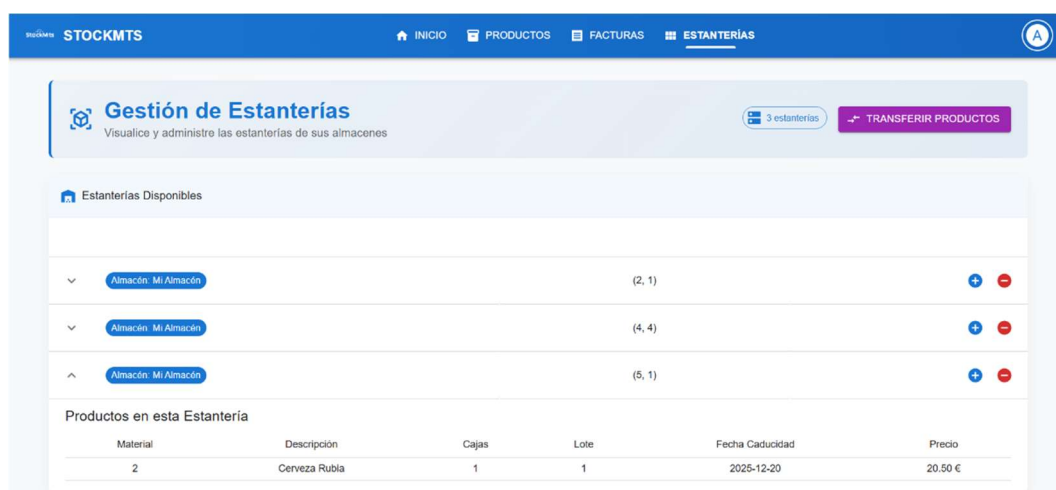


Figura 24. Página estanterías

4.4.7 Página edición mapa almacén

En esta página se podrá visualizar el almacén una vez creado, y se podrá insertar en él las estanterías, eligiendo la capacidad de las mismas de forma independiente. También se podrá reubicar una estantería de un punto a otro del almacén o eliminarla del mismo.

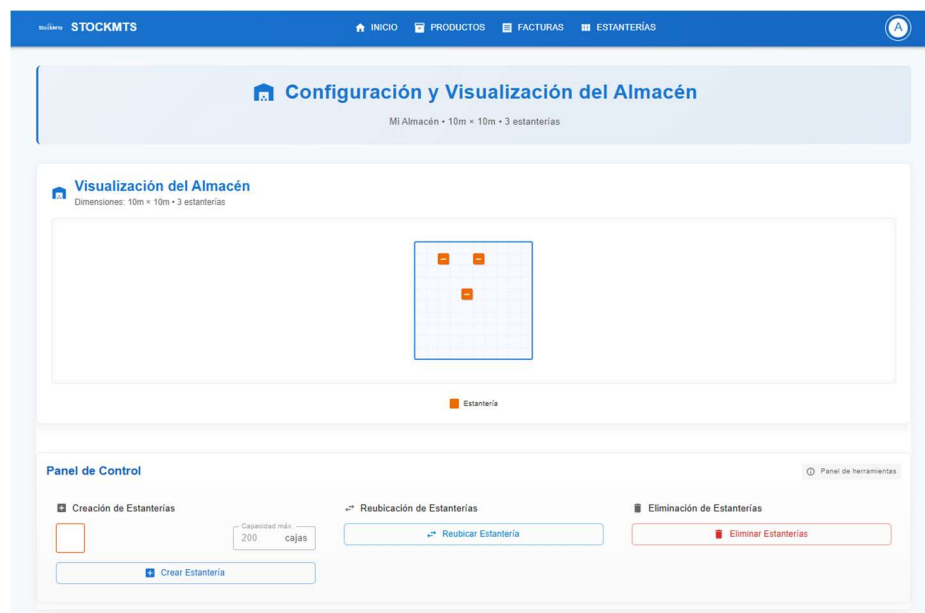


Figura 25. Página configuración y visualización del almacén

4.4.8 Página visión productos de las estanterías

En esta ventana, la cual emerge tras hacer clic en una estantería del mapa del almacén, se podrá observar la lista de los productos que contiene. Indicando a su vez, de cada producto, la cantidad disponible, ocupada y máxima como se puede apreciar en la figura 26. Además, se indica de cada producto las cajas exactas que están colocadas en la estantería.



Figura 26. Página detalles de información de estantería

Del mismo modo, en la pestaña “VISTA 2D” se despliega una representación de una posible estantería, en la que se puede colocar los productos, indicando las cajas y las baldas que se usarían en la estantería. Como detalle, se puede hacer zoom en el mapa para verlo mejor, a la vez que si se pulsa la tecla “H” se despliega una información para abrir un menú que te permite indicar si deseas que el producto utilice más, o menos, baldas de la estantería. Inicialmente ocupando únicamente una balda.

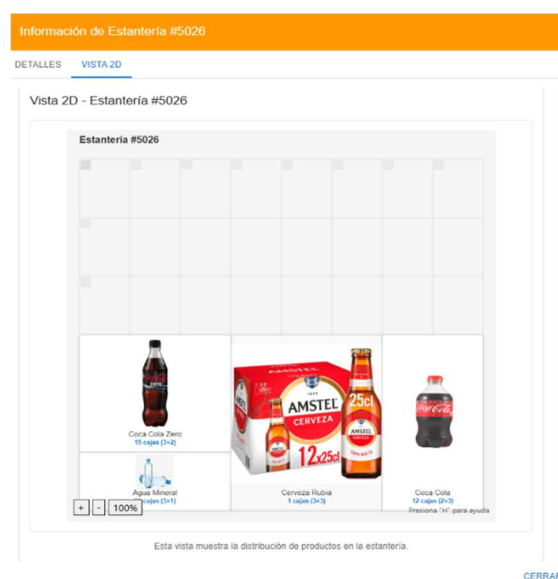


Figura 27. Página vista 2D de información de estantería

Implementación y Pruebas

5.1 Introducción

En este capítulo, tras el estudio sobre las tecnologías y haber analizado y especificado lo que se va a desarrollar en este proyecto, se procederá a explicar cómo se implementa el proyecto, teniendo como objetivo la estructura final del proyecto, como se observa en la figura 28.

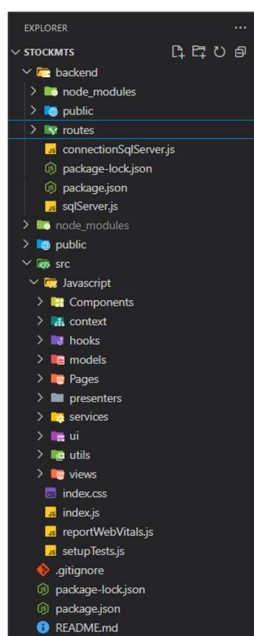


Figura 28. Estructura proyecto Stockmits

5.2 Back-End

En este apartado se explicará la parte del back-end desarrollada para el funcionamiento de este proyecto. Es decir, hablaremos sobre la parte de Express.js para hacer las diversas rutas dentro del back-end y sobre la conexión con la base de datos.

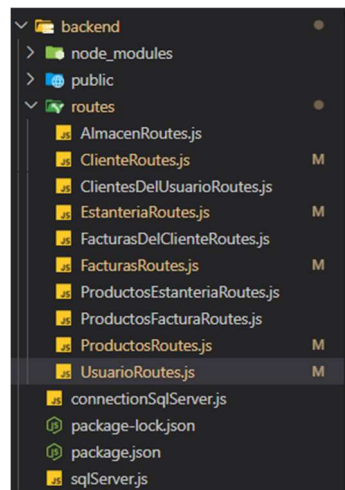


Figura 29. Esquema del back-end

5.2.1 Configuración de conexión con SQL Server

Comenzaremos hablando sobre la conexión con la base de datos, como se muestra en la figura 30. En esta conexión nos encargamos de indicarle:

- El nombre del servidor, en este caso usamos localhost.
- El nombre de la base de datos.
- El usuario y la contraseña de la base de datos. Esto se necesita ya que utilizamos la SQL Server Authentication. Cualquier cambio en esta autenticación en SQL Management Studio obliga a un cambio en el archivo “connectionSqlServer.js” para poder ejecutar correctamente la conexión con la base de datos.
- El puerto al que se conectará al servidor SQL. En este caso estamos usando el puerto predeterminado de SQL Server.

- `encrypt: true` nos indica que la conexión con la base de datos se hará de forma encriptada.
- `trustServerCertificate: true` nos indica que podemos confiar en el certificado del servidor.

Tras tener la configuración de la conexión, se realiza una función asíncrona, en la que se espera a que la conexión se realice correctamente. En caso contrario se lanza un error.

Finalmente, se exporta la función `getConnection` como el módulo `mssql`.

```

1  import mssql from 'mssql';
2
3  const connectionSettings = {
4    server: "localhost",
5    database: "stockmts",
6    user: "albertollfdez",
7    password: "1234",
8    port: 1433,
9    options: {
10     encrypt: true,
11     trustServerCertificate: true
12   }
13 };
14
15 export async function getConnection() {
16   try{
17     return await mssql.connect(connectionSettings);
18   }catch(err){
19     console.error(err);
20   }
21 }
22 export {mssql};

```

Figura 30. Conexión SQL

5.2.2 Configuración e iniciación del servidor

Para continuar se explicará el proceso de configuración del servidor back-end con Node.js y Express para el uso de una API REST. Se empleará la importación de CORS (Cross-Origin Resource Sharing) permitiendo el acceso entre dominios y de esta manera que el front-end pueda hacer peticiones al back-end sin ser bloqueado por políticas CORS. Además, se realiza la importación de `express` para crear el servidor web.

```

1  import express from 'express';
2  import cors from 'cors';

```

Figura 31. Importaciones para configuración

Se seguirá configurando el servidor creando una instancia de express permitiendo CORS, parsear cuerpos JSON y utilizar archivos estáticos como imágenes o archivos HTML desde la carpeta public.

```
14 const app = express();
15 app.use(cors());
16 app.use(express.json());
17 app.use(express.static('public'));
18
```

Figura 32. Configuración del servidor

Tras configurar el servidor y para tener una mayor legibilidad se definen las rutas para cada endpoint y se inicia el servidor en el puerto 3001. Para corroborar qué puerto es el que usamos, una vez desplegado el servidor, se mostrará por pantalla de la terminar el puerto en el que está alojado.

```
20 app.use("/api/productos", productosRoutes);
21 app.use("/api/facturas", facturasRoutes);
22 app.use("/api/estanteria", estanteriaRoutes);
23 app.use("/api/productosEstanteria", productosEstanteriaRoutes);
24 app.use("/api/productosFactura", productosFacturaRoutes);
25 app.use("/api/cliente", clienteRoutes);
26 app.use("/api/usuarios", usuarioRoutes);
27 app.use("/api/clientesusuario", clientesDelUsuarioRoutes);
28 app.use("/api/facturascliente", facturasDelCliente);
29 app.use("/api/almacen", Almacen);
30
31
32
33
34 const PORT = 3001;
35 app.listen(PORT, () => {
36   console.log(`Server running on port ${PORT}`);
37 });
38
39
```

Figura 33. Definición de rutas

5.2.3 Rutas

Tras haber definido en el archivo sqlServer.js las rutas, dentro de la carpeta routes se encuentran todos los archivos que realizan las consultas a la base de datos. Estructurados de tal manera que cada archivo tiene las consultas específicas que necesita esa tabla. Como se puede apreciar en la figura 34, dentro de AlmacenRoutes, primero se realizan las importaciones necesarias para usar funcionalidades:

- Router() o app.use() para gestionar las peticiones del backend.

- getConnection() para conectarse a la base de datos y mssql para declarar los parámetros SQL.
- const router = express.Router() para crear una instancia del enrutador de Express, que luego nos permitirá hacer los get, post, put y delete.

Cada uno de estos archivos están estructurados, de forma ordenada, en gets, posts, puts y deletes. De esta manera, se agiliza la búsqueda de cualquier consulta realizada, al poder tener una referencia de donde se podría encontrar dentro del código.

```

backend > routes > AlmacenRoutes.js
1  import express from 'express';
2  import { getConnection, mssql } from '../connectionSqlServer.js';
3  const router = express.Router();
4
5  // Obtener todos los almacenes de un usuario
6  router.get('/usuario/:idUsuario', async (req, res) => {
7    const { idUsuario } = req.params;
8    try {
9      const pool = await getConnection();
10     const result = await pool.request()
11       .input('idUsuario', mssql.Int, idUsuario)
12       .query(
13         `
14         SELECT a.*
15         FROM Almacen a
16         INNER JOIN AlmacenesDelUsuario au ON a.idAlmacen = au.idAlmacen
17         WHERE au.idUsuario = @idUsuario
18         `);
19     res.json(result.recordset);
20   } catch (err) {
21     console.error('Error al obtener almacenes por usuario:', err);
22     res.status(500).send(err.message);
23   }
24 });
25
26 // Crear un nuevo almacén
27 router.post('/', async (req, res) => {
28   const { nombre, ancho, largo, idUsuario } = req.body;
29   try {
30     if (!nombre || !ancho || !largo || !idUsuario) {
31       return res.status(400).send('Nombre, ancho, largo e idUsuario son requeridos');
32     }
33
34     const pool = await getConnection();
35
36     // Insertar el almacén y obtener su ID
37     const result = await pool.request()
38       .input('nombre', mssql.VarChar, nombre)
39       .input('ancho', mssql.Int, ancho)
40       .input('largo', mssql.Int, largo)

```

Figura 34. Rutas del almacén

5.3 Front-End

En este apartado trataremos la parte relacionada con el front-end de la aplicación. Es decir, hablaremos sobre la estructura que se ha seguido para la organización del código en carpetas especializadas, como se muestra a continuación.

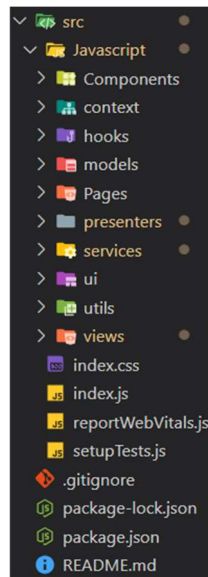


Figura 35. Estructura Front-end

Comenzaremos hablando del index.js, el cual se emplea para enrutar las diversas páginas que se muestran en la aplicación, pudiendo de esta manera seguir una lógica en las URLs utilizadas.

```

src > index.js > ...
16
17 root.render(
18   <React.StrictMode>
19     <AuthProvider>
20       <Router>
21         <Routes>
22           <Route path="/" element={<Home />} />
23           <Route path="/productos" element={<ProductosPage />} />
24           <Route path="/facturas" element={<FacturasPage />} />
25           <Route path="/estanterias" element={<EstanteriasPage />} />
26           <Route path="/register" element={<RegisterPage />} />
27           <Route path="/login" element={<LoginPage />} />
28           <Route path="/almacen" element={<AlmacenPage />} />
29         </Routes>
30       </Router>
31     </AuthProvider>
32   </React.StrictMode>
33 );
34
35 reportWebVitals();

```

Figura 36. Índice del Front-end

Dentro de la carpeta JavaScript encontramos una serie de subcarpetas, estructuradas de esta manera para una mayor legibilidad y organización del proyecto.

5.3.1 Components

En esta carpeta nos encontraremos los componentes reutilizables organizados por funcionalidad. Estos se encargan de la gestión, visualización y

manejo de cada una de las entidades que conforman la aplicación, como es el almacén, las estanterías, los productos o las facturas.

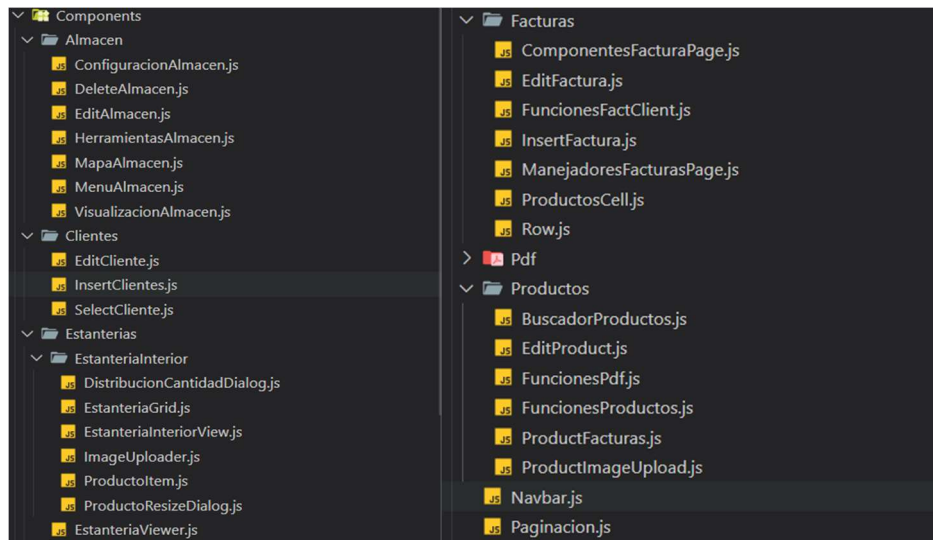


Figura 37. Estructura carpeta Components

Cabe destacar la carpeta Pdf, encargada de la carga, extracción y procesamiento de datos de las facturas en pdf que se suben de los clientes y proveedores del usuario.

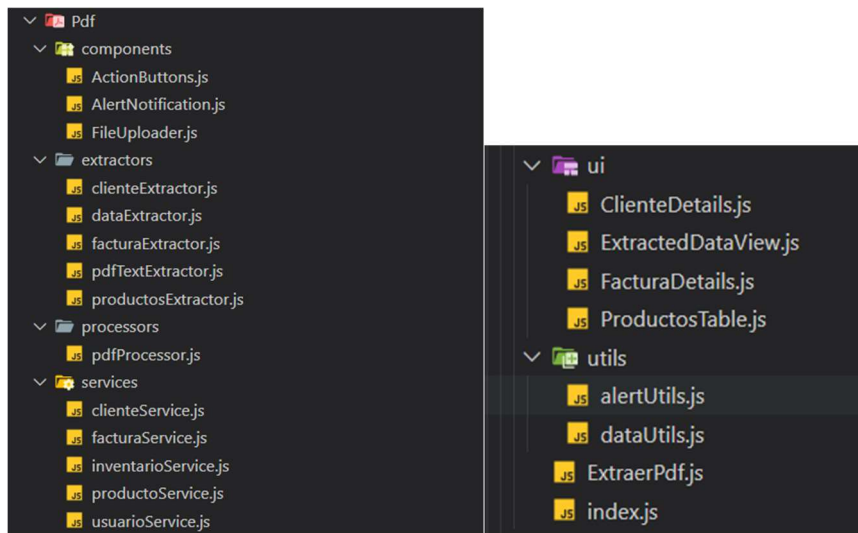


Figura 38. Estructura carpeta Pdf

5.3.2 Context

En la carpeta context, nos encontramos los contextos de gestión de estado global del proyecto. En la aplicación se utiliza para el AuthContext, el cual permite compartir datos entre los componentes sin pasar los props manualmente.

```
src > Javascript > context > AuthContext.js > ...
1 import React, { createContext, useState, useEffect, useContext } from 'react';
2 import usuarioService from '../services/usuarioService';
3
4 // Crear el contexto
5 const AuthContext = createContext();
6
7 // Proveedor del contexto
8 export const AuthProvider = ({ children }) => {
9   const [currentUser, setCurrentUser] = useState(null);
10   const [loading, setLoading] = useState(true);
11
12   // Cargar usuario del localStorage al iniciar
13   useEffect(() => {
14     const loadUser = () => {
15       try {
16         const user = usuarioService.getCurrentUser();
17         console.log("Usuario cargado desde localStorage:", user);
18         setCurrentUser(user);
19       } catch (error) {
20         console.error("Error cargando usuario:", error);
21       } finally {
22         setLoading(false);
23       }
24     };
25     loadUser();
26   }, []);
27
28   // Función para iniciar sesión
29   const login = (userData) => {
30     console.log("INICIO DE SESIÓN ÉXITOSO - Usuario:", userData);
31     console.log("Nombre:", userData.nombre);
32     console.log("Apellidos:", userData.apellidos);
33     console.log("Email:", userData.email);
34     console.log("Id:", userData.idUsuario);
35     usuarioService.saveUserToLocalStorage(userData);
36     setCurrentUser(userData);
37   };
38
39   // Función para cerrar sesión
40   const logout = () => {
41     usuarioService.logout();
42     setCurrentUser(null);
43   };
44
45   // Valor que proporcionará el contexto
46   const value = {
47     currentUser,
48     login,
49     logout,
50     loading
51   };
52
53   return (
54     <AuthContext.Provider value={value}>
55       {!loading ? children : <div>Cargando...</div>}
56     </AuthContext.Provider>
57   );
58 };
59
60 // Hook personalizado para usar el contexto
61 export const useAuth = () => {
62   return useContext(AuthContext);
63 };
64
65
```

Figura 39. Componente AuthContext

5.3.3 Hooks

En la carpeta Hooks se encapsula la lógica reutilizable, organizada por dominio de la aplicación. Dentro de esta encontramos las carpetas almacén, cliente, estantería y factura. En cada una de estas carpetas se implementa la lógica de la gestión de esas mismas entidades.

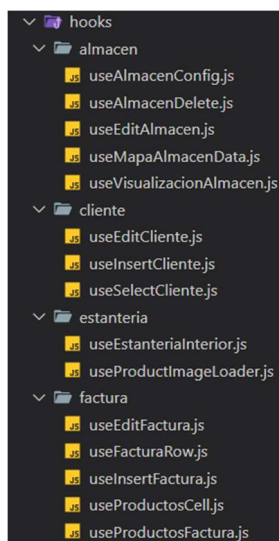


Figura 40. Estructura carpeta Hooks

5.3.4 Models

En la carpeta models encontramos los componentes utilizados para cada tipo de entidad, en el que definimos los modelos de datos y las validaciones de cada uno de estos. De esta manera se representan las entidades principales del sistema, tales como almacenes, estanterías, productos, etc.

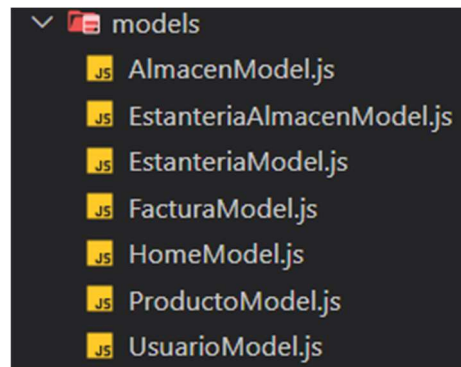


Figura 41. Estructura carpeta Models

5.3.5 Pages

Esta carpeta contiene los componentes de las página principales que tienen como función principal cargar los presenters correspondientes de cada página.

```
src > Javascript > Pages > AlmacenPage.js > default
1  import React, { useEffect } from 'react';
2  import AlmacenPresenter from '../presenters/AlmacenPresenter.js';
3  import { useAuth } from '../context/AuthContext';
4  import { Navigate } from 'react-router-dom';
5
6  function AlmacenPage() {
7    const { currentUser } = useAuth();
8
9    useEffect(() => {
10      console.log("AlmacenPage renderizado");
11      console.log("Usuario actual:", currentUser);
12    }, [currentUser]);
13
14    // Redirigir si no hay usuario logueado
15    if (!currentUser) {
16      console.log("No hay usuario, redirigiendo a login");
17      return <Navigate to="/login" />;
18    }
19
20    console.log("Renderizando AlmacenPresenter");
21    return <AlmacenPresenter />;
22  }
23
24  export default AlmacenPage;
```

Figura 42. Componente AlmacenPage.js

5.3.6 Presenters

Siguiendo el patrón de diseño arquitectónico Modelo-Vista-Presentador, en esta carpeta se implementa el presentador, que contiene en esta la lógica de

presentación y que conecta las vistas con los modelos y servicios. Además, se encarga de gestionar estados, procesar datos y controlar el flujo de la aplicación.

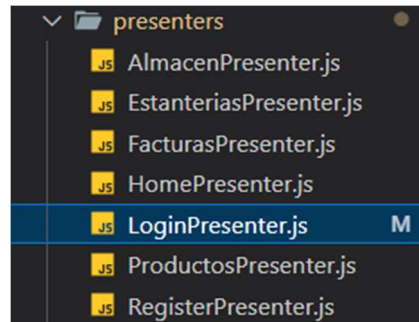


Figura 43. Estructura carpeta Presenters

5.3.7 Services

Esta carpeta contiene los servicios encargados de encapsular la comunicación con el backend a través de las llamadas a la API. Cada entidad principal tiene su propio servicio, como se puede apreciar en la figura 44.

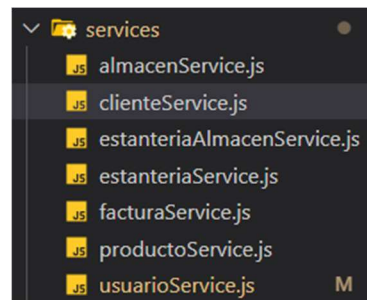


Figura 44. Estructura carpeta Services

5.3.8 Ui

En la carpeta Ui, encontramos componentes de interfaz de usuario, los cuales, están divididos en:

- a) comunes, siendo este los diálogos de confirmación.
- b) específicos, utilizados para secciones específicas de la aplicación dentro de las entidades principales de la misma.

```

30 const ConfirmDialog = ({
31   open,
32   onClose,
33   onConfirm,
34   title = "Confirmar acción",
35   message = "¿Está seguro que desea continuar?",
36   warningMessage = "",
37   cancelButtonText = "Cancelar",
38   confirmButtonText = "Confirmar",
39   severity = "question"
40 }) => {
41   // Determinar el color y el icono según la severidad
42   const getSeverityConfig = () => {
43     switch (severity) {
44       case "warning":
45         return {
46           color: '#ff9800',
47           icon: <WarningIcon fontSize="large" />
48         };
49       case "error":
50         return {
51           color: '#f44336',
52           icon: <ErrorOutlineIcon fontSize="large" />
53         };
54       case "info":
55         return {
56           color: '#2196f3',
57           icon: <InfoOutlinedIcon fontSize="large" />
58         };
59       case "question":
60       default:
61         return {
62           color: '#673ab7',
63           icon: <HelpOutlineIcon fontSize="large" />
64         };
65     }
66   };
67   const { color, icon } = getSeverityConfig();
68 }

```

Figura 45. Lógica componente ConfirmDialog.js

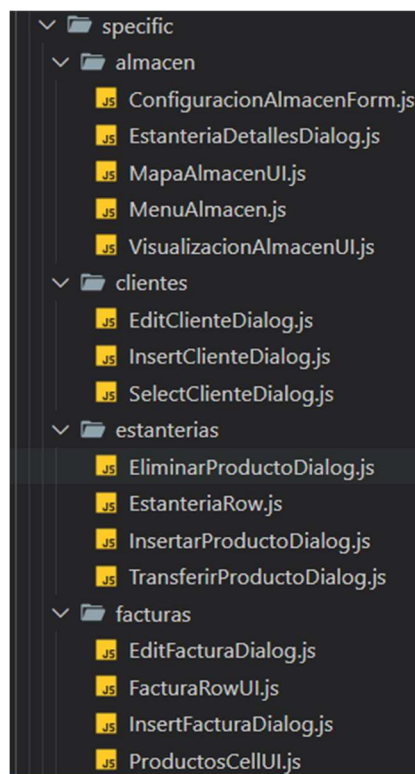


Figura 46. Estructura carpeta ui/specific

5.3.9 Utils

En esta carpeta encontramos funciones que proporcionan operaciones comunes reutilizables en la aplicación. Algunas de estas operaciones son el formateo de fechas o validaciones de datos, mostrados en la figura 48.

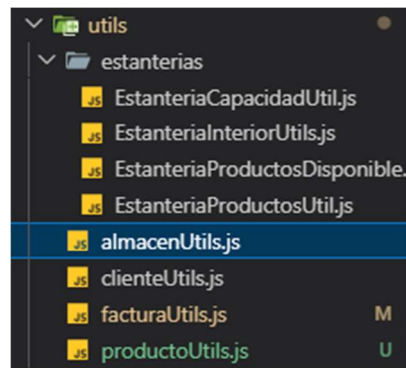


Figura 47. Estructura carpeta utils

```
10 export const validarDimensionesAlmacen = (campo, valor, dimensionesActuales, maxPos, estanterias) => {
11   const result = {
12     error: '',
13     dimensiones: { ...dimensionesActuales }
14   };
15
16   if (campo === 'ancho') {
17     const minAncho = estanterias.length > 0 ? maxPos.x + 1 : 1;
18
19     if (valor < minAncho) {
20       result.error = 'El ancho no puede ser menor que ${minAncho} metros debido a que hay estanterías ubicadas en esa posición';
21       result.dimensiones.ancho = minAncho;
22     } else {
23       result.dimensiones.ancho = valor;
24     }
25   } else if (campo === 'largo') {
26     const minLargo = estanterias.length > 0 ? maxPos.y + 1 : 1;
27
28     if (valor < minLargo) {
29       result.error = 'El largo no puede ser menor que ${minLargo} metros debido a que hay estanterías ubicadas en esa posición';
30       result.dimensiones.largo = minLargo;
31     } else {
32       result.dimensiones.largo = valor;
33     }
34   }
35
36   return result;
37 };
```

Figura 48. Componente almacenUtils.js

5.3.10 Views

En esta carpeta encontramos las interfaces visuales de la aplicación, siendo estas vistas responsables únicamente de mostrar los elementos visuales y recibir eventos del usuario, sin manejar la lógica de negocio. Cada entidad principal tiene su propia vista.

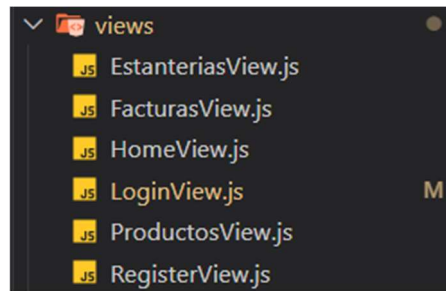


Figura 49. Estructura carpeta Views

5.4 Pruebas

5.4.1 Introducción

En este apartado trataremos los casos de prueba empleados para verificar el correcto funcionamiento de diversas partes del software. A partir de los casos de uso explicados en el capítulo 3 se crean estos casos de pruebas.

Se han tenido en consideración todos estos casos de uso, realizando sus pruebas pertinentes, pero en este apartado nos centraremos en mostrar los que se han considerados más relevantes.

5.4.2 Registro

Casos de prueba
Feature: Registro en la aplicación. Background Given: El usuario se llama David. Scenario: David se registra Given: David hace clic en el botón registrarse de la página principal. When: David rellena el formulario. When: David hace clic en el botón registrarse. Then: David se encuentra en la página de inicio de sesión. Scenario: David se registra con errores en la contraseña. Given: David hace clic en el botón registrarse de la página principal. When: David rellena el formulario sin indicar la contraseña. When: David hace clic en el botón registrarse.

Then: EL sistema muestra un error al registrarse indicando fallo en la contraseña.

Scenario: David se registra con errores en el email.

Given: David hace clic en el botón registrarse de la página principal.

When: David rellena el formulario sin indicar el email.

When: David hace clic en el botón registrarse.

Then: EL sistema muestra un error al registrarse indicando fallo en el email.

Siguiendo los pasos indicados en el escenario en el que se registra el usuario correctamente, podemos observar el comportamiento de la API, devolviendo una respuesta 201, indicando el correcto funcionamiento del proceso.

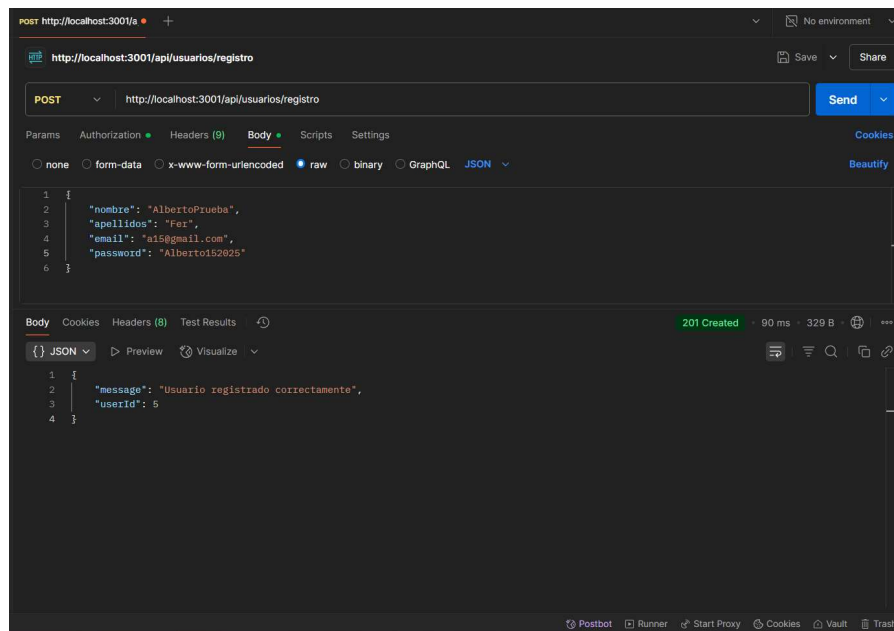


Figura 50. Postman método post registro

5.4.3 Inicio sesión

Casos de prueba

Feature: Inicio de sesión en la aplicación.

Background

Given: El usuario se llama David.

Scenario: David inicia sesión.

Given: David hace clic en inicio de sesión de la página principal.

When: David rellena el formulario.

When: David hace clic en iniciar sesión.

Then: David se encuentra en la página principal con la sesión iniciada.

Scenario: David inicia sesión con contraseña incorrecta.

Given: David hace clic en inicio de sesión de la página principal.

When: David rellena el formulario con error en la contraseña.

When: David hace clic en iniciar sesión.

Then: El sistema muestra un error al iniciar sesión indicando el fallo en la contraseña.

Scenario: David inicia sesión con email incorrecta.

Given: David hace clic en inicio de sesión de la página principal.

When: David rellena el formulario con error en el email.

When: David hace clic en iniciar sesión.

Then: El sistema muestra un error al iniciar sesión indicando el fallo en el email.

Siguiendo los pasos indicados en el escenario en el que se inicia sesión correctamente, si observamos el comportamiento de la API encontraremos el siguiente proceso, devolviendo una respuesta 200, indicando el correcto funcionamiento del proceso.

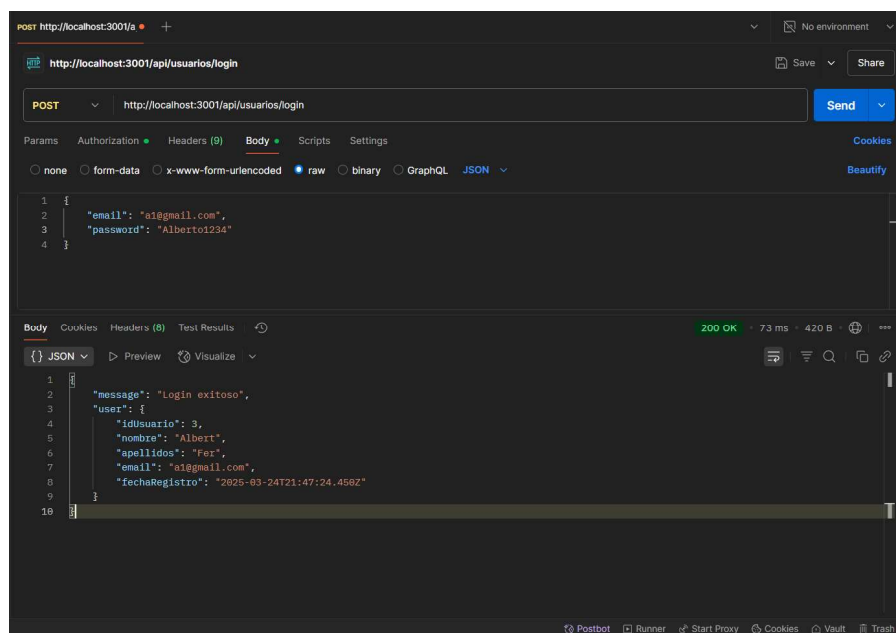


Figura 51. Postman método post para login

5.4.4 Subir factura

Casos de prueba

Feature: Subir una facture en la aplicación.

Background

Given: El usuario se llama David.

Scenario: David sube una factura en la aplicación.

Given: David ha hecho clic en procesar pdf y en factura cliente, seleccionando una factura en pdf.

When: David ha hecho clic en guardar.

When: El sistema ha cargado los productos de la factura.

Then: David se encuentra en la página facturas con la nueva factura subida.

Scenario: David sube una factura en la aplicación sin estar el cliente creado.

Given: David ha hecho clic en procesar pdf y en factura cliente, seleccionando una factura en pdf.

When: David ha hecho clic en guardar.

When: El sistema ha cargado los productos de la factura.

Then: El sistema muestra error al cargar la factura, el usuario vuelve a la página principal.

Siguiendo los pasos anteriormente presentados, podemos hacer una prueba de integración de las facturas en la API, comprobando como se puede apreciar en la figura 52 que se realizó la inserción de esa nueva factura correctamente.

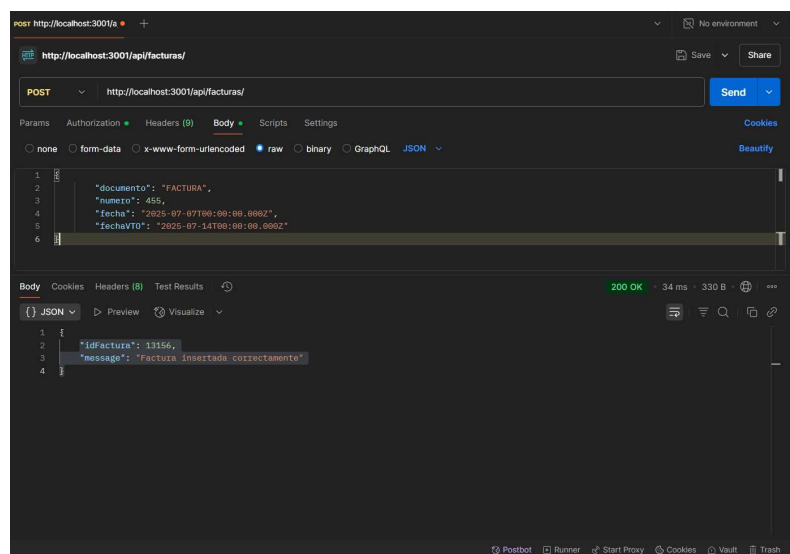


Figura 52. Postman método post facturas

5.4.5 Crear Almacén

Casos de prueba
Feature: Crear un almacén en la cuenta de un usuario de la aplicación.
Background
Given: El usuario se llama David.
Scenario: David crea un almacén.
Given: David ha hecho clic en gestionar y en crear almacén.
When: David rellena el nombre y las dimensiones del almacén.
Then: El sistema muestra el mapa del almacén en la página principal.

Una vez realizado este proceso, podremos comprobar que se ha realizado correctamente haciendo una prueba en la API, debiendo de obtener la información mostrada en la figura 53.

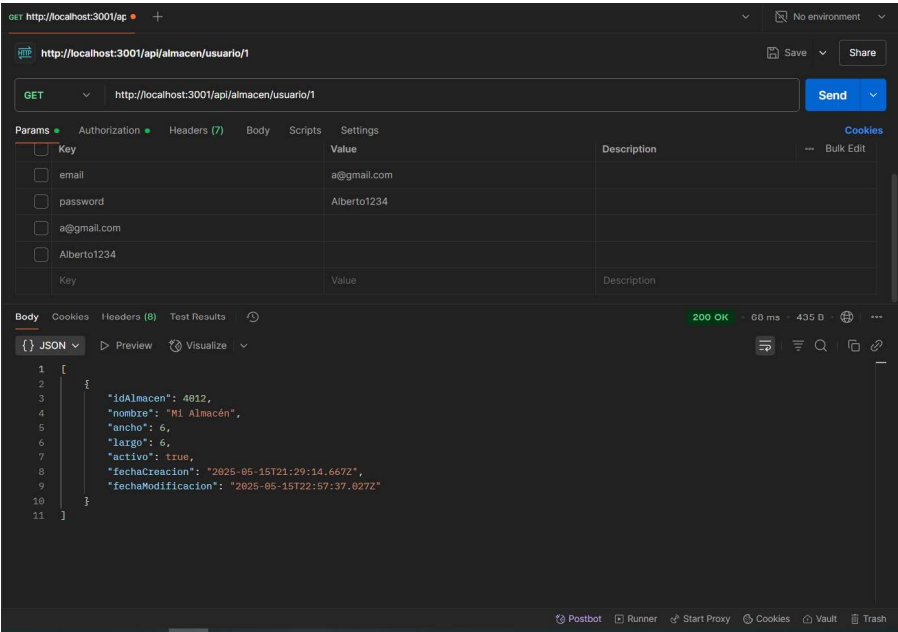


Figura 53. Postman método get almacén

5.4.6 Buscar producto en estantería

Casos de prueba

Feature: Buscar un producto dentro de una estantería ya creada.

Background

Given: El usuario se llama David.

Scenario: David busca un producto que se encuentra en stock y colocado en una estantería.

Given: David ha hecho clic en el buscador de productos.

When: David introduce el nombre del producto a buscar.
--

Then: El sistema muestra en el mapa, la estantería que contiene el producto de color verde.

Scenario: David busca un producto que se encuentra en stock, pero no colocado en una estantería.
--

Given: David ha hecho clic en el buscador de productos.

When: David introduce el nombre del producto a buscar.
--

Then: El sistema muestra en el mapa sin indicar ninguna estantería en verde.
--

Scenario: David busca un producto que no se encuentra en stock.

Given: David ha hecho clic en el buscador de productos.

When: David introduce el nombre del producto a buscar.
--

Then: El sistema muestra por pantalla un recuadro indicando que el producto no ha sido encontrado.
--

Una vez realizado este proceso, podremos comprobar que se ha realizado correctamente haciendo una prueba en la API. En caso de ser correcto, obtendremos como respuesta la información de que estanterías contienen ese producto buscado.

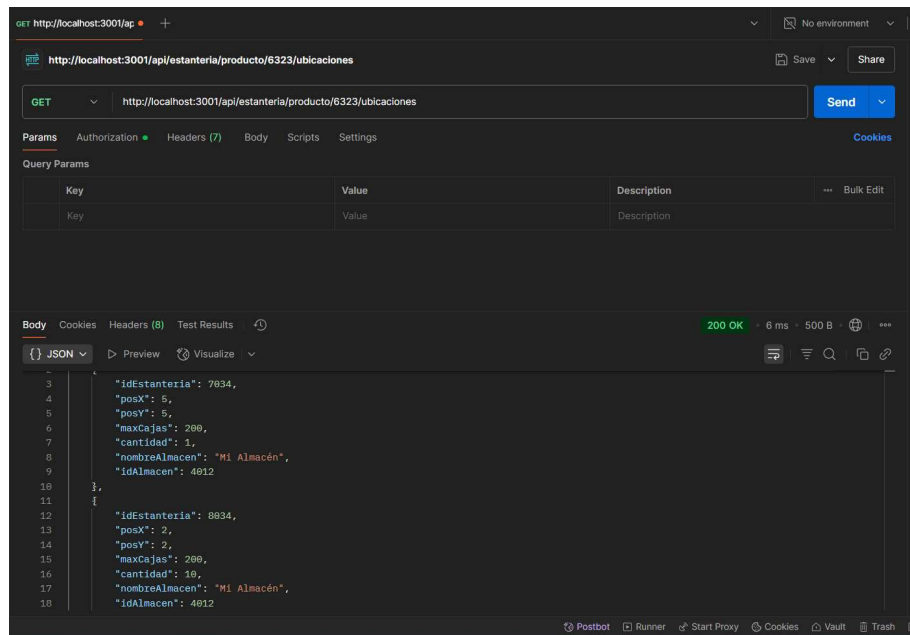


Figura 54. Postman método get buscar producto en estantería

Conclusiones y Líneas Futuras

En este capítulo, tras el desarrollo y documentación del proyecto, se indicarán las conclusiones obtenidas, las dificultades encontradas y las posibles líneas futuras, haciendo especial énfasis en las mejoras potencialmente aplicables.

6.1 Conclusiones

El propósito central de esta proyecto consistió en desarrollar una aplicación web fácil de usar, intuitiva y útil para el usuario, pudiendo permitir que empresas que necesiten una manera de contabilizar el stock de su almacén lo puedan lograr. Además, teniendo como objetivo secundario, permitir la reducción de tiempo empleado en la búsqueda de productos dentro del almacén.

Tras el desafío de aprender las tecnologías empleadas, como React, node.js o express.js, durante el desarrollo, se considera que los puntos marcados en el objetivo principal y secundario han sido logrados.

Además, el desarrollo realizado ha supuesto todo un reto al querer abarcar un gran arco de funcionalidades, en su mayoría relacionadas entre sí.

6.2 Dificultades encontradas

Durante la fase de implementación surgieron varios retos a resolver. Uno de los primeros fue adaptarse al entorno de trabajo con React, comprendiendo cómo estructurar un proyecto utilizando las múltiples funcionalidades que ofrece este framework, así como familiarizarse con la librería MUI, la cual proporcionó herramientas como iconos, tablas y alertas que facilitaron una mayor personalización del sistema.

Además, los constantes ajustes en la estructura de datos durante el primer sprint, especialmente en el diseño de tablas intermedias para representar relaciones de muchos a muchos, implicaron un análisis detallado para determinar la solución más adecuada que favoreciera un desarrollo más eficiente en fases posteriores.

6.3 Líneas futuras

El proyecto presenta diversas oportunidades de ampliación, entre ellas una autenticación y un registro más completo, que incluyan opciones como el acceso mediante Google o mecanismos para restablecer la clave de acceso mediante notificación por correo electrónico.

Por otro lado, una mejora que se podría implementar es en la extracción de información de las facturas. Actualmente, se lee un único tipo de modelo de factura, por lo que para cada tipo de factura nuevo que necesites se tendría que modificar y añadir nuevas funciones que extrajeran, de una manera correcta, la información de ese tipo de factura.

Además, se podría desarrollar más la parte del perfil del usuario, incluyendo una página con sus datos y foto de perfil.

Por último, se barajó que la vista de las estanterías en 2D se viese en 3D, pero por sobrecarga de trabajo y falta de tiempo no se pudo ejecutar esta

funcionalidad. Para una posible mejora futura, se podría cambiar la visión 2D, hecha mediante una matriz, a un modelado 3D de cada estantería.

Referencias

- [1] Microsoft. (s.f). Patrón de diseño Modelo-Vista-Presentador (MVP).
[https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649571\(v=pandp.10\)](https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649571(v=pandp.10))
- [2] Qué es SCRUM. (2021, 20 septiembre). Proyectos Ágiles.
<https://proyectosagiles.org/que-es-scrum/>
- [3] Casero, A. (2024, 25 octubre). La historia de JavaScript.
<https://keepcoding.io/blog/la-historia-de-javascript/>
- [4] JavaScript | MDN. (s. f.). MDN Web Docs.
<https://developer.mozilla.org/es/docs/Web/JavaScript>
- [5] HTML: Lenguaje de etiquetas de hipertexto | MDN. (s. f.). MDN Web Docs.
<https://developer.mozilla.org/es/docs/Web/HTML>
- [6] Casero, A. (2025, 27 febrero). Historia de HTML: Desde su origen hasta 2025.
<https://keepcoding.io/blog/historia-de-html/>
- [7] ¿Qué es SQL? - Explicación de lenguaje de consulta estructurado (SQL) - AWS. (s. f.). Amazon Web Services. <https://aws.amazon.com/es/what-is/sql/>

- [8] De Expertos En Ciencia y Tecnología, E. (2023, 6 noviembre). Lenguaje SQL, historia y conceptos básicos. VIU Universidad Online. <https://www.universidadviu.com/es/actualidad/nuestros-expertos/lenguaje-sql-historia-y-conceptos-basicos>
- [9] React: framework multi-propósito - Aplyca. (2022, 19 septiembre). Aplyca Tecnología SAS. <https://www.aplyca.com/blog/react-biblioteca-javascript-multiproposito>
- [10] Llamas, L. (2024, 15 abril). Breve historia y evolución de Node.js. Luis Llamas. <https://www.luisllamas.es/historia-de-nodejs/>
- [11] Parker, R. N. (2023b, septiembre 30). The Notable History of Express.js - Ryan Neil Parker - Medium. Medium. <https://medium.com/@ryanneilparker/the-notable-history-of-express-js-6869ba4fcd25>
- [12] MikeRayMSFT. (s. f.). Novedades de SQL Server 2022 - SQL Server. Microsoft Learn. <https://learn.microsoft.com/es-es/sql/sql-server/what-s-new-in-sql-server-2022?view=sql-server-ver16>
- [13] Arsys. (s. f.). ¿Qué es Visual Studio Code y cuáles son sus ventajas? <https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas>
- [14] Erinstellato-Ms. (s. f.). Descarga de SQL Server Management Studio (SSMS). Microsoft Learn. <https://learn.microsoft.com/es-es/ssms/download-sql-server-management-studio-ssms>

[15] Acerca de GitHub y Git - documentación de GitHub. (s. f.). GitHub Docs.
<https://docs.github.com/es/get-started/start-your-journey/about-github-and-git>

Apéndice A

Manual de Usuario

A continuación, se detallarán los pasos a seguir para obtener el correcto funcionamiento por parte del usuario que utilice Stockmts. Detallando estos pasos según las distintas funcionalidades que tenga cada proceso.

A.1. Registro

Al abrir la aplicación nos mostrará la pestaña principal, únicamente indicándonos que estamos en la aplicación “Stockmts” y los botones “Registrarse” e “INICIO DE SESIÓN” (Figura 55).



Figura 55. Pestaña principal Stockmts sin iniciar sesión

Se debe hacer clic en el botón “REGISTRARSE” y tras ello rellenar los datos correctamente para crearnos una cuenta, como se muestra en la Figura 56. Por último, debemos hacer clic en “Registrarse”.

Registro de usuario

① Requisitos para registrarse:
 • Todos los campos son obligatorios • La contraseña debe cumplir con los criterios de seguridad • El correo electrónico debe tener un formato válido

Nombre * Al menos 2 caracteres

Apellidos * Requerido

Correo electrónico * Introduce un correo electrónico válido (ejemplo@dominio.com)

Contraseña *

- ☐ Al menos 6 caracteres
- ☐ Al menos una letra mayúscula
- ☐ Al menos una letra minúscula
- ☐ Al menos un número

Confirmar contraseña *

REGISTRARSE

[¿Ya tienes una cuenta? Inicia sesión](#)
[¿Quieres volver a la página principal? Volver](#)

Figura 56. Pestaña de registro

A.2. Inicio de sesión

Se debe pulsar en el botón “INICIO DE SESIÓN” y completar la información solicitada en el formulario correspondiente mostrado en la Figura 57. Posteriormente, se hará clic en el botón “INICIAR SESIÓN”.

Iniciar sesión

Correo electrónico *

Contraseña * 👁

INICIAR SESIÓN

[¿No tienes una cuenta? Regístrate](#)
[¿Quieres volver a la página principal? Volver](#)

Figura 57. Pestaña de inicio de sesión

A.3. Pestaña principal y perfil

Al cargar la pestaña principal tras haber iniciado sesión veremos la página principal como nos muestra la Figura 58.

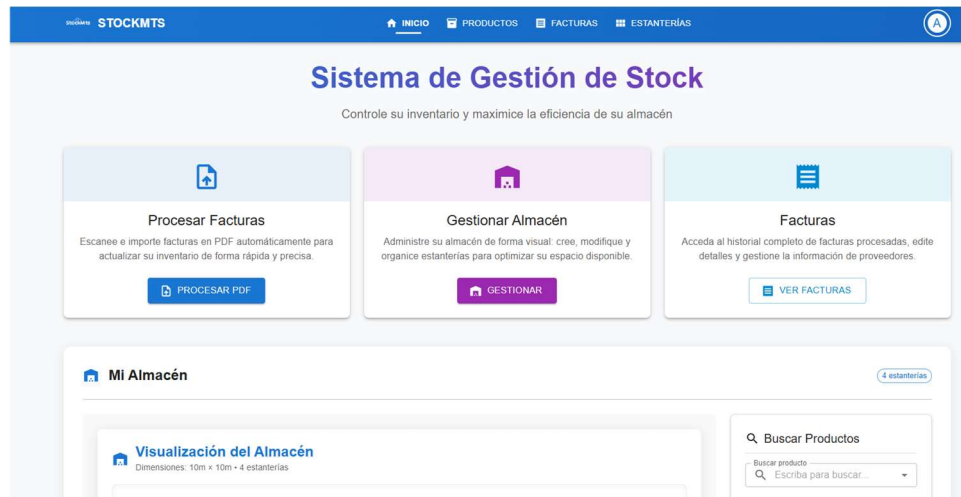


Figura 58. Pestaña principal Stockmts habiendo iniciado sesión

Una vez en esta página nos podemos desplazar a la pestaña Productos, Facturas o Estanterías mediante la barra azul superior.

A.4. Carga de facturas

Cuando deseemos subir una factura, estando en la página principal y estando con la sesión iniciada, debemos hacer clic en “PROCESAR PDF”. Una vez dentro, debemos seleccionar si es una factura de cliente o de proveedor, y tener creado en la pestaña Facturas a dicho cliente o proveedor (Figura 59).

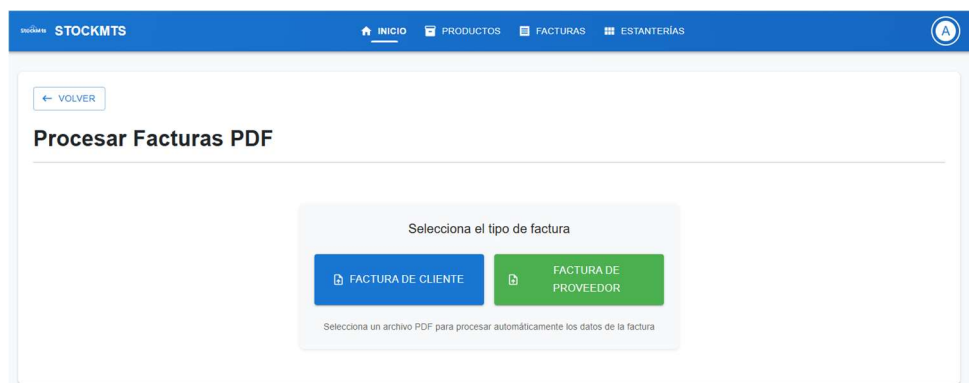


Figura 59. Página Procesar Facturas Pdf

Tras hacer esto, podemos seleccionar la factura en pdf que corresponda con el cliente o proveedor del que queramos subir la factura y se mostrará la información como la Figura 60.

Procesar Facturas PDF

Datos extraídos del PDF

Datos de Factura

Documento FACTURA	Número 45
Fecha 08/05/2025	Fecha Vencimiento 15/05/2025

Datos de Cliente

ID Cliente 1009	Nombre Distribuidora Fernandez
Punto de Venta Avd Extremadura	Dirección Avd Peñaroya
Teléfono 123456789	

Productos

Material	Descripción	Cantidad	Lote	Fecha Caducidad	Precio
1	Agua Mineral	10	1	20/12/2025	7.00 €

CANCELAR GUARDAR

Figura 60. Información al procesar factura

Por último, tras comprobar que es correcta la información que nos muestra, se debe hacer clic en el botón “GUARDAR”.

A.5. Operaciones en productos, facturas y estanterías

En cada una de las páginas indicadas al seleccionar uno o más productos nos muestra las operaciones de editar y eliminar como las mostradas en la figura 61.

Seleccionados 1

Material	Descripción	Num cajas	Colocadas	Lote	Fecha caducidad	Precio	Imagen
☒ 1	Agua Mineral	11	11/11 100%	1	2025-12-20	6.00 €	
☐ 2	Cerveza Rubia	40	2/40 5%	1	2025-12-20	20.50 €	
☐ 3	Vino Tinto	5	0/5 0%	1	2025-12-20	12.59 €	
☐ 4	Coca Cola	25	24/25 96%	1	2025-12-20	22.47 €	
☐ 5	Coca Cola Zero	27	15/27 56%	1	2025-12-20	23.02 €	
☐ 6	Fanta Naranja	22	0/22 0%	1	2025-12-20	20.23 €	

Vista compacta Filas por página: 10 1-6 de 6

Figura 61. Visualización de productos con operaciones

Seleccionando un único elemento, nos despliega una ventana en la que podremos editar la información, y posteriormente, haciendo clic en guardar se actualizará.

Editar Producto

Material
1

Descripción
Agua Mineral

Número de Cajas
11

Lote
1

Precio
6

Fecha de Caducidad
12/20/2025

CANCELAR EDITAR

Figura 62. Ventana editar producto

También se puede seleccionar uno o muchos elementos y eliminarlos, teniendo que confirmar la acción de eliminar el mismo.

Confirmar eliminación

¿Estás seguro de eliminar este producto?

Esta acción modificará tu inventario.

CANCELAR CONTINUAR

Confirmación final

¿Está ABSOLUTAMENTE SEGURO de querer eliminar este producto?

Esta es una acción IRREVERSIBLE. Una vez eliminados, no podrá recuperar estos datos de su inventario.

CANCELAR ELIMINAR DEFINITIVAMENTE

Figura 63. Ventanas de eliminación y confirmación

A.6. Operaciones en almacén

En primer lugar, se puede crear, editar o eliminar el almacén dentro de “Gestionar almacén”, mostrando el menú que encontramos en la figura 64. Una vez creado el almacén, no se permitirá crear otro.

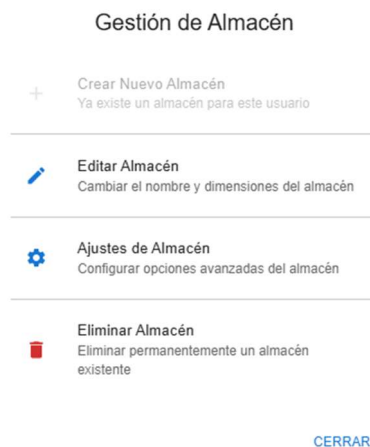


Figura 64. Menú gestión de almacén

Dentro de la opción “Ajustes de Almacén” encontramos las operaciones para poder introducir, reubicar y eliminar las estanterías dentro del mapa del almacén ya creado.

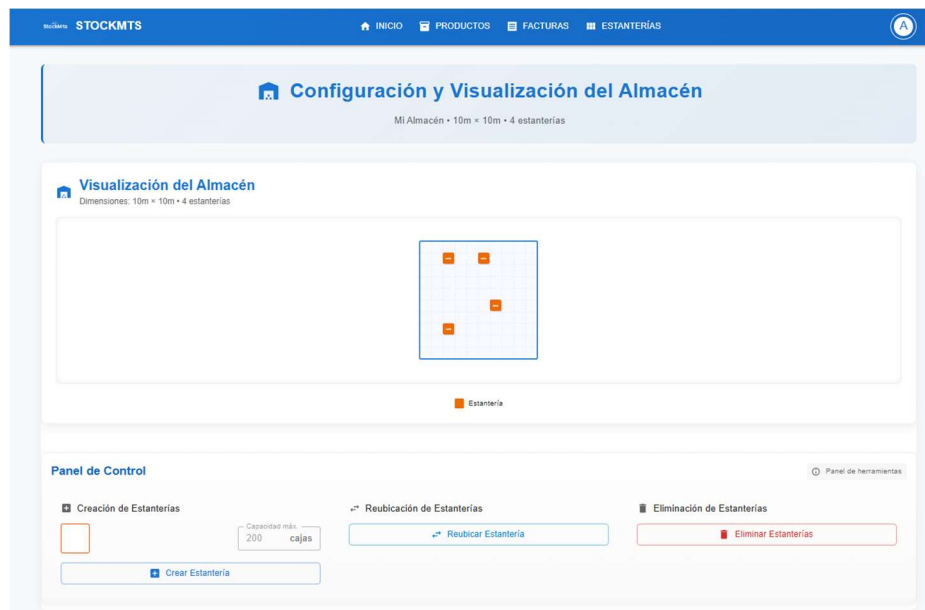


Figura 65. Ventana visualización y configuración del almacén

A.7. Organizar productos en estantería 2D

Una vez creado el mapa y, como mínimo, una estantería, podremos hacer clic sobre ella y dirigirnos a la pestaña 2D, dentro de la cual podremos organizar los productos por baldas, representando cómo se colocarían en una estantería real, como nos muestra la figura 66.

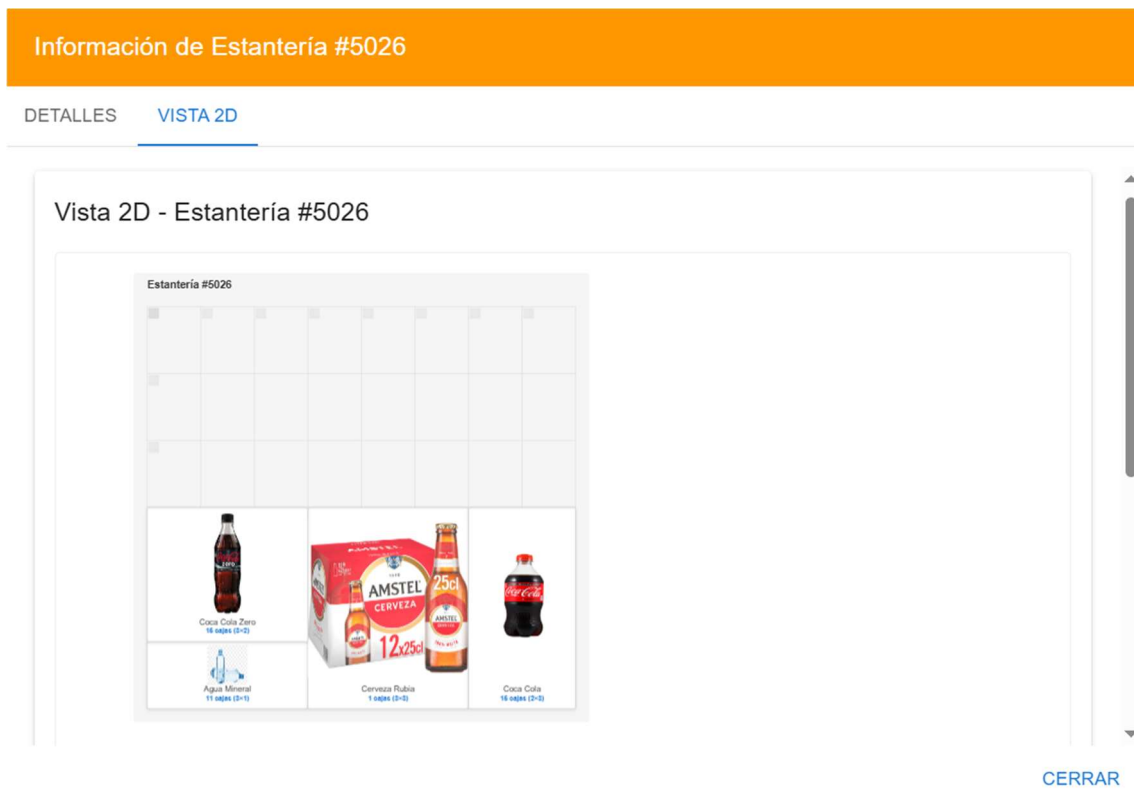


Figura 66. Información de estantería 2D

A.8. Búsqueda y sugerencias de productos

Situándonos en la página principal podemos encontrar el buscador de productos dentro de las estanterías del almacén. Para ello se debe introducir el producto a buscar, y si se encuentra dentro del stock del almacén, mostrará las estanterías que contengan dicho producto.

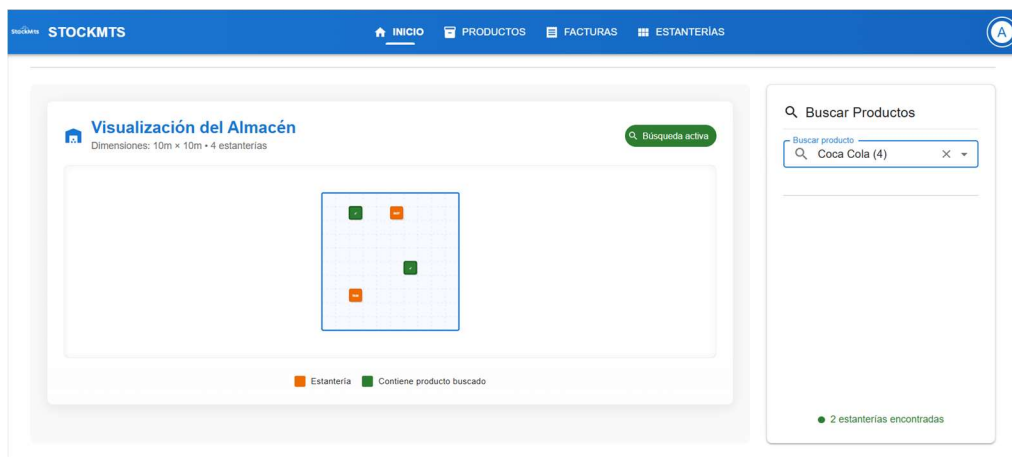


Figura 67. Búsqueda de productos

Apéndice B

Manual de Despliegue

A continuación, se detallarán las acciones necesarias para poner en marcha tanto el servidor como el cliente.

B.1. Despliegue del servidor

Para iniciar el servidor, primero es necesario que la conexión con la base de datos esté activa. Para ello se utilizó SQL Management Studio 20 con el método de autenticación SQL Server Authentication, tal y como se a continuación.

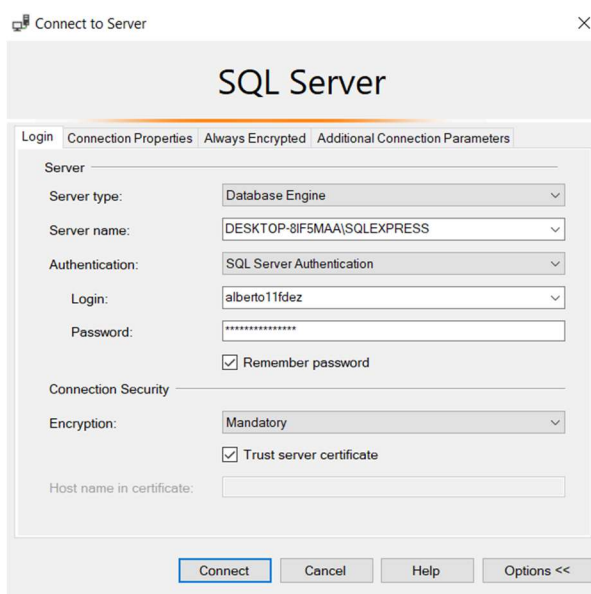
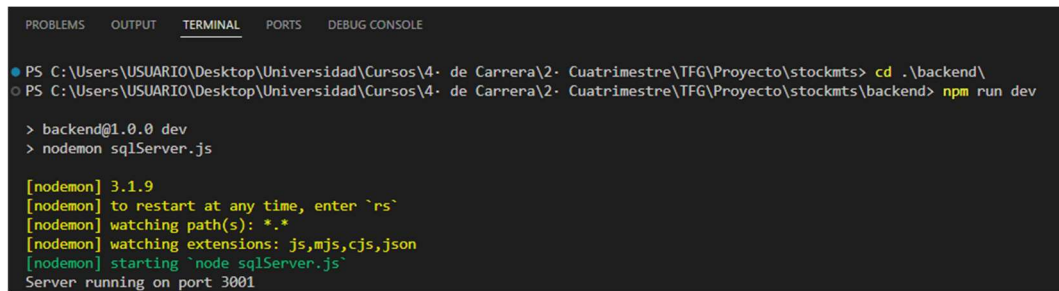


Figura 68. Conexión SQL Server

En segundo lugar, tras haber conectado con la base de datos, se ejecutará la terminal. Direccionándonos dentro del proyecto a la carpeta backend, mediante “cd .\backend\”, se introducirá el comando “npm run dev” y con ello, como se muestra en la figura 69, se desplegará el servidor, escuchando el puerto 3001.



```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
PS C:\Users\USUARIO\Desktop\Universidad\Cursos\4- de Carrera\2- Cuatrimestre\TFG\Proyecto\stockmts> cd .\backend\
PS C:\Users\USUARIO\Desktop\Universidad\Cursos\4- de Carrera\2- Cuatrimestre\TFG\Proyecto\stockmts\backend> npm run dev

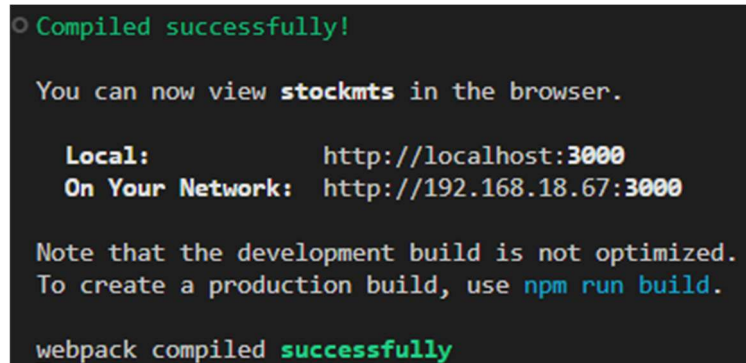
> backend@1.0.0 dev
> nodemon sqlServer.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node sqlServer.js`
Server running on port 3001
```

Figura 69. Despliegue de servidor

B.2. Despliegue del cliente

Para iniciar el cliente, debemos abrir una segunda terminal dentro del proyecto y ejecutar el comando “npm start”, mediante el cual se desplegará el cliente en el puerto 3000, como nos muestra la figura 70.



```
Compiled successfully!

You can now view stockmts in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.18.67:3000

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

webpack compiled successfully
```

Figura 70. Despliegue de cliente



UNIVERSIDAD
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga