UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduada en Ingeniería Informática

Click2Order: Plataforma web para la gestión eficiente de pedidos de una empresa de distribución de bebidas

Click2Order: A web platform for the efficient management of orders in a beverage distribution company

Realizado por
Ana Isabel Ruiz González

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre de 2025

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA EN INGENIERÍA INFORMÁTICA

Click2Order: Plataforma web para la gestión eficiente de pedidos de una empresa de distribución de bebidas

Click2Order: A web platform for the efficient management of orders in a beverage distribution company

Realizado por
Ana Isabel Ruiz González

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE 2025

Fecha defensa: septiembre de 2025

Resumen

En las pequeñas y medianas empresas de distribución de bebidas, la comunicación entre proveedores y establecimientos comerciales como bares o restaurantes suelen gestionar sus pedidos mediante comunicaciones informales como llamadas telefónicas, mensajes de Whatsapp, audios o fotografías de listas escritas a mano. Sin un proceso tecnológico estructurado que facilite esta gestión, la interpretación de los pedidos es frecuentemente errónea, generando malentendidos en cantidades, marcas o precios.

Este Trabajo de Fin de Grado desarrolla Click2Order, una plataforma web especializada en optimizar la comunicación entre los distribuidores mayoristas y los establecimientos del sector hostelero. A diferencia del e-commerce convencional, esta solución pretende agilizar los procesos de gestión de pedidos entre pequeñas y medianas empresas del sector, eliminando las típicas barreras comunicativas.

La aplicación proporciona a los clientes de la empresa de distribución una interfaz estructurada simple y con las funcionalidades necesarias para consultar un catálogo, realizar y repetir pedidos o guardar productos. Por su lado, el distribuidor accederá a un sistema centralizado que procesa la información y los pedidos de los clientes, además de la posibilidad de calcular la ruta óptima de reparto mediante integración con Mapbox, calculando trayectorias eficientes que reducen tiempos de entrega y optimizan recursos logísticos.

Click2Order digitaliza la comunicación comercial en el sector de la distribución de bebidas, sustituyendo métodos informales propensos a errores por un canal estructurado y eficiente que beneficia tanto a distribuidores como a establecimientos comerciales.

Palabras clave: gestión, catálogo, pedidos, rutas, eficiencia.

Abstract

In small and medium-sized beverage distribution companies, communication between suppliers and commercial establishments such as bars or restaurants often involves managing orders through informal communications such as phone calls, WhatsApp messages, audio messages, or handwritten notes. Without a structured technological process to facilitate this management, orders are frequently misinterpreted, leading to misunderstandings about quantities, brands, or prices.

This Final Degree Project develops Click2Order, a web platform specialized in optimizing communication between wholesale distributors and establishments in the hospitality sector. Unlike conventional e-commerce, this solution aims to streamline order management processes between small and medium-sized companies in the sector, eliminating typical communication barriers.

The application provides the distribution company's customers with a simple, structured interface and the necessary functionalities to consult a catalog, place and repeat orders, or save products. For its part, the distributor will have access to a centralized system that processes customer information and orders, as well as the ability to calculate the optimal delivery route through integration with Mapbox, calculating efficient routes that reduce delivery times and optimize logistics resources.

Click2Order digitizes commercial communication in the beverage distribution sector, replacing informal methods prone to errors with a structured and efficient channel that benefits both distributors and commercial establishments.

Keywords: management, catalog, orders, routes, efficiency.

Índice

Índice de Figuras

Índice de Tablas

1

Introducción

Este capítulo define los pilares fundamentales del proyecto Click2Order, abordando las motivaciones que impulsaron el desarrollo de esta propuesta, los objetivos planteados para su realización, la estructuración que seguirá la memoria para registrar el proceso y la metodología aplicada durante la implementación.

1.1. Motivación

La gestión de pedidos en muchas pequeñas y medianas empresas del sector de la distribución de bebidas continúa realizándose de forma manual y desestructurada. En el caso particular que motiva a este proyecto, los encargos de productos por parte de bares, restaurantes o establecimientos de alimentación que venden bebidas llegan al administrador de la empresa distribuidora a través de llamadas telefónicas, mensajes de texto o de voz, o incluso mediante fotos de listas escritas a mano. Este sistema, aunque funcional en el día a día, es propenso a errores, genera retrasos, dificulta el control de pedidos y obstaculiza una planificación eficiente de las rutas de reparto.

La problemática específica se manifiesta de múltiples formas en la rutina diaria. Es frecuente que un cliente solicite por teléfono “la leche que suele pedir” sin especificar marca, tamaño o cantidad, obligando al distribuidor de la empresa a buscar en registros anteriores o confiar en la memoria para determinar qué productos entregar. De manera similar, cuando un cliente solicita mediante audio o texto “10 aguas de 1,5 litros” sin especificar si se refiere a Bezoya, Lanjarón u otra

marca disponible, el administrador debe interrumpir su flujo de trabajo para contactar nuevamente con el cliente y aclarar esos detalles, generando demoras y posibles errores en el proceso.

Otro problema común surge cuando los clientes envían solicitudes en las que no incluyen las cantidades específicas en sus comunicaciones o envían un audio y texto escrito a mano poco legible, lo que obliga al administrador a estar constantemente recurriendo al cliente para aclarar las especificaciones y esperar respuestas para poder procesar correctamente el pedido. Esta falta de información puede causar el movimiento de mercancía incorrecta e innecesaria y la necesidad de rectificar los pedidos en un futuro.

Adicionalmente, la planificación de las rutas de reparto se convierte en un proceso ineficiente cuando el administrador debe calcular manualmente las trayectorias óptimas sin herramientas tecnológicas de apoyo, dando lugar a rutas o trayectorias ineficientes que aumentan tanto el tiempo de recorrido como el consumo de recursos logísticos.

Este proyecto surge de la experiencia real de una empresa familiar que presenta precisamente estos mismos problemas operativos diariamente. La necesidad de una solución tecnológica que organice y automatice estos procesos ha motivado la creación de una aplicación web que permita a los clientes realizar sus pedidos de forma estructurada y al distribuidor gestionarlos con una mayor eficiencia.

La propuesta no pretende ser una tienda virtual convencional, sino una herramienta especializada que beneficia tanto a los clientes como al distribuidor para agilizar específicamente el proceso de pedidos y gestión comercial. La interfaz de usuario se ha diseñado intencionadamente simple para que sea lo más sencilla, rápida y cómoda posible para ambas partes, eliminando complejidades innecesarias que puedan obstaculizar la adopción de la tecnología.

Además de mejorar la comunicación entre ambas partes, esta solución pretende optimizar la logística diaria mediante el cálculo de rutas de entrega basadas en la ubicación de los clientes, así como proporcionar herramientas administrativas específicas para la gestión de productos, clientes y pedidos. El objetivo final no es solo digitalizar el proceso, sino también simplificar el trabajo cotidiano, reducir los errores y aumentar la eficiencia global del servicio.

Si bien existen sistemas ERP comerciales como SAP, Odoo u otras plataformas de gestión que podrían abordar parcialmente esta problemática, se ha optado por desarrollar otro sistema específico de este estilo por varias razones.

En primer lugar, el conocimiento directo de las necesidades reales por la experiencia de la empresa familiar que ha motivado el desarrollo de este proyecto, permite crear funcionalidades personalizadas y adaptadas a los procesos específicos del sector de distribución de bebidas, evitando una complejidad innecesaria o los costes generales de las típicas licencias comerciales. Además, los sistemas comerciales habituales suelen incluir módulos o funcionalidades que no son relevantes para este tipo de negocio, mientras carecen de otras características útiles como la optimización de rutas de reparto o la gestión particular que requiere la distribución de bebidas en este sector.

Por otro lado, desde la perspectiva académica y crecimiento personal, el desarrollo de un sistema completo desde cero proporciona una experiencia de aprendizaje mucho más enriquecedora que la configuración de un sistema ya existente. La decisión de emprender este proyecto de forma individual responde también a la motivación personal de desarrollar por primera vez una aplicación completa de manera autónoma, ya que los proyectos realizados durante la carrera han sido considerablemente más pequeños en alcance o se han desarrollado en equipo. De este modo, se permite aplicar los conocimientos adquiridos durante la carrera en las distintas fases del desarrollo del software, desde el análisis y diseño hasta la implementación y despliegue, constituyendo un proyecto de crecimiento personal que ha permitido aprender y aplicar múltiples tecnologías que no habían sido utilizadas anteriormente en ningún proyecto de características similares.

1.2. Objetivos

Atendiendo a la problemática descrita en el apartado anterior, el objetivo de este Trabajo de Fin de Grado es el desarrollo de Click2Order, una plataforma web destinada principalmente a optimizar la comunicación comercial entre la empresa distribuidora de bebidas y sus clientes del sector hostelero y alimentación. La aplicación pretende solucionar los problemas de interpretación errónea de pedidos, y la ineficiencia en la gestión logística que caracteriza actualmente este sector.

Para lograr estos objetivos, se diseñará una solución basada en una arquitectura cliente-servidor moderna, con un frontend desarrollado en React que consumirá una API REST implementada en Node.js y Express y una base de datos MongoDB alojada en la nube. Lo cual permitirá una mayor escalabilidad y facilitará el mantenimiento del sistema.

Los objetivos específicos que pretende abordar la plataforma incluyen:

Eliminar ambigüedades en los pedidos: Cada negocio o restauración podrá acceder a un catálogo interactivo filtrable por categorías, donde podrá consultar cada producto con especificaciones claras, añadirlos al carrito de compra con cantidades exactas o marcarlos como favoritos. Esta funcionalidad resuelve directamente el problema de las solicitudes imprecisas como “la leche que suele pedir” o “10 aguas de 1,5 litros” sin especificar la marca.

Facilitar la repetición de pedidos habituales: Será posible consultar pedidos anteriores y repetir pedidos completos con un solo clic, garantizando así que cada establecimiento pueda replicar sus patrones de consumo habituales sin necesidad de especificar nuevamente cada detalle, eliminando las consultas constantes del administrador sobre “qué marca suele pedir” o “qué cantidad necesita habitualmente” ese establecimiento.

Centralizar la gestión administrativa: El administrador de la empresa de distribución dispondrá de un panel de control centralizado donde podrá gestionar los productos, las categorías, los clientes y los pedidos de la aplicación, facilitando así al administrador la gestión de pedidos por parte de los cliente de una manera más clara y organizada para así evitar perder conversaciones, llamadas o imágenes donde se les solicitaba el pedido.

Automatizar las notificaciones: El sistema gestionará el registro y perfil de los clientes comerciales, permitiendo al administrador consultar todos los pedidos entrantes de los distintos comercios registrados con notificación automática por correo electrónico cada vez que se realice un nuevo pedido, el cual podrá descargar en formato PDF para la gestión logística de carga para su distribución.

Optimizar la planificación logística: Con la integración de la API de Mapbox, la aplicación calculará rutas de reparto óptimas según las direcciones de los clientes con pedidos realizados, resolviendo el problema de las trayectorias poco eficiente y permitiendo al administrador establecer recorridos óptimos y directos, acelerando la planificación logística y mejorando la eficiencia global en la gestión de pedidos.

La solución pretende transformar un proceso caracterizado por interrupciones constantes, aclaraciones telefónicas y gestión manual en un flujo de trabajo estructurado, automatizado y eficiente que beneficie tanto a la empresa distribuidora como a sus clientes.

1.3. Estructura de la memoria

El presente documento se estructura mediante los siguientes capítulos:

- **Capítulo 1. Introducción:** Se explican las motivaciones que originan el proyecto, los objetivos generales que se persiguen con el desarrollo de Click2Order y además se presenta la estructura general del documento.

- **Capítulo 2. Tecnologías y herramientas:** Se detallan todas las tecnologías utilizadas durante el desarrollo de la aplicación, incluyendo lenguajes de programación, frameworks, bases de datos, herramientas de diseño, documentación y control del proyecto.

- **Capítulo 3. Análisis:** Esta sección presenta los requisitos funcionales y no funcionales del sistema, estableciendo distinciones entre las capacidades dirigidas al cliente y al administrador. También se especifican los casos de usoque orientan el comportamiento del sistema.

- **Capítulo 4. Diseño:** Se incluye el prototipo inicial de la interfaz con maquetas realizadas a mano y aspectos clave como el modelo de datos de cada colección.

- **Capítulo 5. Desarrollo e implementación:** Esta sección analiza el proceso de desarrollo de la aplicación, las decisiones tecnológicas tomadas y la manera en que se han implementado las diferentes prestaciones del backend y del frontend.

- **Capítulo 6. Pruebas**: Se presenta un conjunto de pruebas realizadas con Postman, que han permitido validar las funcionalidades de la aplicación.
- **Capítulo 7. Conclusiones y líneas futuras:** Se presentan las conclusiones extraídas tras finalizar el desarrollo de Click2Order y se plantean posibles mejoras y ampliaciones para el sistema en el futuro.

Además, al final de la memoria se incluyen varios anexos con información complementaria como son:

- **Manual de usuario**: Explica paso a paso cómo utilizar la aplicación desde el punto de vista del cliente y del administrador, orientado a usuarios sin conocimientos técnicos.
- **Manual de instalación**: Detalla los requisitos, herramientas necesarias y pasos para desplegar y poner en marcha la aplicación en un entorno local.

1.4 Metodología

En este proyecto se ha optado por emplear una metodología ágil, concretamente la metodología Kanban [1]. Su principal característica es que no establece fechas de entrega fijas ni ciclos cerrados, como ocurre en Scrum, sino que ésta permite un flujo continuo y flexible, adaptándose fácilmente a cambios en los requisitos o prioridades. Este enfoque facilita la mejora de forma continua y mantener un ritmo constante de trabajo a lo largo del proyecto.

La implementación se ha llevado a cabo mediante la aplicación de Trello, una herramienta de gestión de proyectos basada en tableros, listas y tarjetas. Trello permite representar visualmente las tareas y su progreso y organizarlas de forma colaborativa o individual. Las tarjetas se desplazan entre columnas que representan los distintos estados del trabajo, ofreciendo una visión clara y actualizada del proyecto.

En este caso, el tablero está formado por las columnas *Backlog,* para las tareas pendientes de empezar; *En progreso*, con las tareas en desarrollo; *En pruebas*, para las funcionalidades terminadas que necesitan ser probadas y validadas, y *Completadas*, donde se agrupan las tareas finalizadas.

Este sistema facilita la organización del desarrollo en las distintas fases del proyecto (análisis, diseño, implementación, pruebas y documentación) y permite mantener una visión clara y actualizada del estado de cada tarea en todo momento.

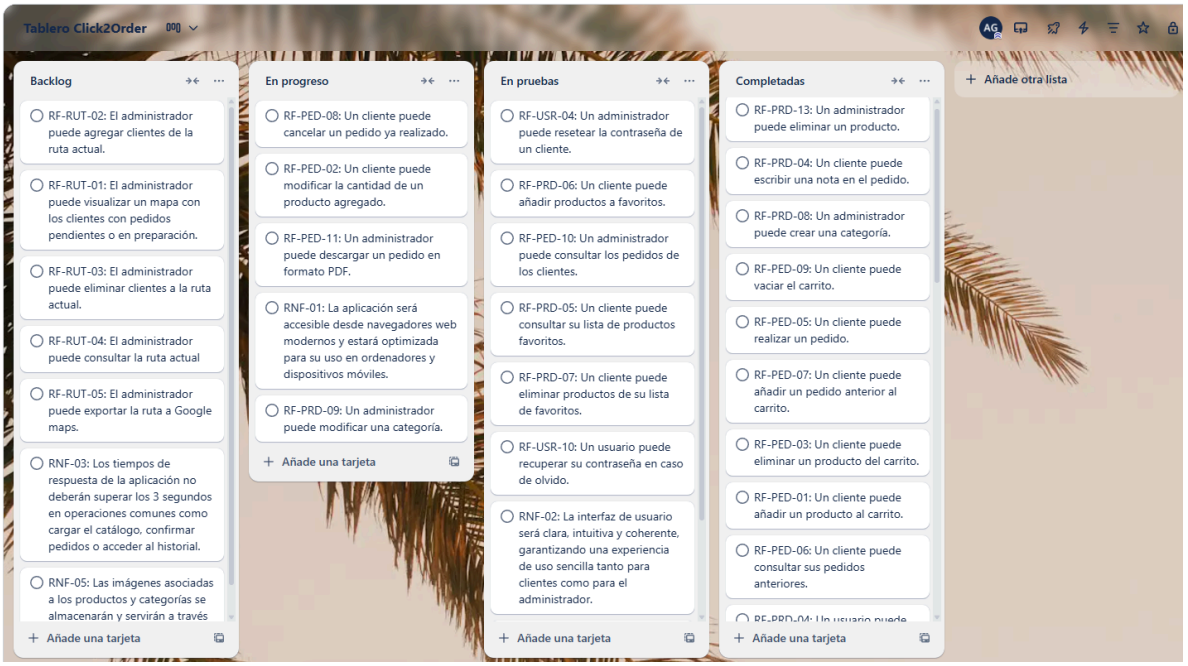

Figura 1. Tablero de planificación Kanban

2

Tecnologías y herramientas

En este capítulo se describirán las tecnologías utilizadas a la hora de realizar este Trabajo Fin de Grado. De cada una de ellas, se hará una pequeña introducción y se indicará en qué parte del proyecto se usa.

2.1. Lenguaje de programación

2.1.1. JavaScript

Javascript [2] se trata de un lenguaje de programación interpretado, dinámico y orientado a objetos, normalmente utilizado para el desarrollo web. Permite crear aplicaciones interactivas tanto en el lado del cliente como en el servidor.

Figura 2. Logo JavaScript

En este proyecto se ha empleado como lenguaje principal para el desarrollo de la plataforma, unificando la lógica del frontend y el backend en una sola tecnología, lo que simplifica el desarrollo y mantenimiento del proyecto frente a usar múltiples lenguajes.

2.2. Tecnologías para el frontend

2.2.1. React

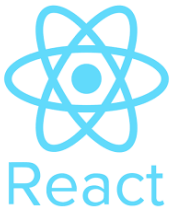

Figura 3. Logo React

React [3] es una herramienta de desarrollo de JavaScript empleada para construir interfaces de usuario de forma estructurada y eficiente. Su principal característica es el uso de componentes reutilizables, lo que permite dividir la interfaz en pequeñas piezas independientes, facilitando el mantenimiento y la escalabilidad del código

Se ha utilizado para desarrollar la interfaz tanto del cliente como del administrador. Se ha elegido esta opción frente a Vue.js O Angular debido a su curva de aprendizaje más gradual y su amplia documentación.

2.2.2. Vite

Vite [4] es una herramienta de desarrollo que permite crear aplicaciones web de forma ágil y eficiente sobre todo está pensada para agilizar el desarrollo de aplicaciones con JavaScript y TypeScript. Destaca por ofrecer una experiencia de desarrollo muy ágil, gracias a su servidor que aprovecha los módulos nativos del navegador para cargar y actualizar el código de forma inmediata.

Figura 4. Logo Vite

En este proyecto se ha utilizado como herramienta principal para la construcción del frontend, sobre otros frameworks como Create React App, ya que ofrece tiempos de compilación más rápidos y una experiencia de desarrollo más ágil.

2.2.3. Tailwind CSS

Figura 5. Logo Tailwind CSS

Tailwind CSS [5] es un framework de utilidades para CSS que permite aplicar estilos directamente en el HTML mediante clases predefinidas. Evita la necesidad de escribir reglas CSS personalizadas, facilitando un desarrollo rápido, flexible y con un diseño visualmente coherente. Además, su sistema modular permite construir componentes reutilizables de forma eficiente.

En este proyecto se ha utilizado Tailwind CSS para aplicar estilos de forma directa y ordenada, garantizando un diseño responsive que se adapta automáticamente a diferentes tamaños de pantalla, consistente y fácilmente escalable.

2.2.4. Accessible Web Color Contrast Checker

Accessible Web Color Contrast Checker [6] es una herramienta que verifica si el contraste entre colores de texto y fondo cumple con los estándares de accesibilidad WCAG (Web Content Accessibility Guidelines), un conjunto de pautas internacionales que garantizan la legibilidad del contenido incluso para usuarios con discapacidad visual.

Figura 6. Logo Accessible Web

En este proyecto se ha utilizado para verificar combinaciones de color en la interfaz, asegurando una correcta visibilidad del contenido.

2.2.5. React Context

Figura 7. Logo React Context

React Context [7] es una API nativa de React que permite compartir datos entre componentes facilitando así la gestión del estado global creando un contexto que envuelve los componentes y proporciona acceso directo a los datos compartidos desde cualquier nivel de la jerarquía.

En la capa frontend de la aplicación se ha optado por el uso de React Context para centralizar el estado y el uso compartido de las operaciones sobre él

2.3. Tecnologías para el backend

2.3.1. Node.js y Express

Node.js [8] es un entorno de ejecución de JavaScript que se ejecuta en el lado del servidor y permite construir aplicaciones rápidas y escalables. Es de código abierto y multiplataforma, ideal para aplicaciones de red en tiempo real.

Figura 8. Logo Node.js y Express

Sobre Node.js se ha integrado Express [9], un framework que simplifica la creación de aplicaciones y APIs. Proporciona un sistema sencillo para gestionar rutas, peticiones y middleware, lo que permite estructurar el código del servidor de forma clara y modular.

En este proyecto, ambas se han utilizado conjuntamente para implementar la API que conecta el frontend con la base de datos. Se ha elegido Node.js para mantener JavaScript en toda la aplicación, y Express se ha elegido por su simplicidad frente a otros frameworks más complejos.

2.3.2. Zod

Zod [10] es una librería de validación de esquemas para TypeScript y JavaScript que permite definir y validar la estructura de datos de forma declarativa. Proporciona verificación y genera mensajes de error cuando los datos no cumplen con el esquema definido.

Figura 9. Logo Zod

Se ha empleado en este proyecto para validar los datos de entrada de la API, asegurando que las peticiones cumplan con el formato y reglas establecidas. Se ha elegido Zod frente a otras librerías como Joi o Yuo por su integración nativa con TypeScript y su sintaxis más intuitiva.

2.3.3. Nodemailer

Figura 10. Logo Nodemailer

Nodemailer [11] es una librería para Node.js que facilita el envío de emails desde aplicaciones web. Permite configurar distintos proveedores como Gmail, Outlook u otros y soporta envío de emails mediante texto, HTML y archivos adjuntos.

Se ha empleado para automatizar el envío de notificaciones por correo electrónico en distintas situaciones. Se ha optado por Nodemailer frente a otras opciones de envío de emails por ser una solución gratuita y por su flexibilidad para usar cuentas de Gmail existentes.

2.4. Tecnologías de Bases de datos

2.4.1. MongoDB

MongoDB [12] es un gestor de bases de datos NoSQL que organiza la información en documentos con formato BSON, similar a JSON. Se caracteriza por su flexibilidad para manejar datos sin un esquema fijo y por su capacidad para escalar horizontalmente, lo que lo hace ideal para aplicaciones que requieren manejar grandes volúmenes de información.

Figura 11. Logo MongoDB

Se ha utilizado como base de datos principal para almacenar la información de la aplicación. Se ha elegido MongoDB sobre bases de datos relacionales como MySQL por la flexibilidad de su esquema NoSQL y su mejor integración con Node.js.

2.4.2. MongoDB Atlas

MongoDB Atlas [13] es el servicio en la nube administrado por MongoDB que facilita el despliegue, la gestión y la escalabilidad de bases de datos. Incluye funciones como copias de seguridad automáticas, alta disponibilidad, balanceo de carga y herramientas de monitorización.

Figura 12. Logo MongoDB Atlas

La base de datos está alojada en la nube mediante MongoDB Atlas. Se ha optado por esta solución frente a una instalación local para facilitar la gestión, el acceso remoto y la escalabilidad del sistema.

2.4.3. MongoDB Compass

MongoDB Compass [14] es una herramienta gráfica que permite explorar y trabajar con datos en MongoDB de forma visual. Facilita la ejecución de consultas, análisis de esquemas y validación de colecciones, sin necesidad de usar comandos.

Figura 13. Logo MongoDB Compass

Durante el desarrollo se ha usado MongoDB Compass para la administración y visualización de los datos almacenados en MongoDB Atlas, permitiendo inspeccionar colecciones y realizar consultas gráficas.

2.5. Herramientas de desarrollo

2.5.1. Visual Studio Code

Figura 14. Logo Visual Studio Code

Visual Studio Code [15] es un editor de código multiplataforma desarrollado por Microsoft, ligero pero muy potente, con soporte para múltiples lenguajes de programación y una gran variedad de extensiones. Se trata de una herramienta muy completa para desarrolladores.

Se ha utilizado como entorno principal para editar el código tanto del frontend como del backend. Se ha elegido por su facilidad de uso, integración con Git y amplio ecosistema de extensiones para JavaScript.

2.5.2. Postman

Postman [16] es una herramienta para probar APIs de manera visual e interactiva. Permite enviar peticiones HTTP, analizar respuestas, gestionar colecciones de pruebas y documentar endpoints, agilizando el desarrollo y depuración de servicios web.

Figura 15. Logo Postman

Se ha utilizado principalmente para comprobar el correcto funcionamiento de los endpoints del backend y validar que los datos se intercambian correctamente entre el cliente y el servidor.

2.5.3. Cloudinary

Figura 16. Logo Cloudinary

Cloudinary [17] es una plataforma en la nube para gestionar contenido multimedia. Facilita la carga, almacenamiento, transformación y entrega de imágenes y vídeos en aplicaciones web o móviles, todo ello optimizado automáticamente para distintos dispositivos y tamaños.

Dentro de este proyecto, se ha empleado para almacenar las imágenes asociadas a las categorías y productos, permitiendo una gestión centralizada y eficiente. Se ha optado por Cloudinary frente al almacenamiento local porque ofrece optimización automática de imágenes adaptándolas al dispositivo del usuario y entrega a través de una red global de servidores.

2.5.4. Mapbox

Mapbox [18] es una plataforma de mapas y servicios de localización que proporciona APIs con mapas interactivos y servicios de geocodificación en aplicaciones web. Proporciona herramientas para visualizar mapas personalizables y optimizar rutas.

Figura 17. Logo Mapbox

En este proyecto se ha usado para la visualización de mapas dónde los usuarios seleccionan su ubicación y para el cálculo de rutas optimizadas de reparto. Se ha integrado específicamente la Geocoding API V6 para obtener las direcciones textuales a partir de unas coordenadas y la Optimization API V1 para la planificación eficiente de rutas de entrega.

2.6. Herramientas de control y gestión del proyecto

2.6.1. GitHub

GitHub [19] es una plataforma para el control de versiones y la colaboración en el desarrollo de software, basada en Git. Permite almacenar el código en repositorios remotos, realizar seguimiento de los cambios y trabajar de forma colaborativa mediante ramas y solicitudes de fusión.

Figura 18. Logo GitHub

En este proyecto se ha utilizado para mantener el control de versiones del código fuente y registrar el historial de cambios, facilitando la organización del desarrollo y la trazabilidad del proyecto.

2.6.2. Trello

Figura 19. Logo Trello

Trello [20] es una herramienta de gestión de tareas visual basada en tableros, listas y tarjetas. Está orientada a la organización del trabajo mediante metodologías ágiles como Kanban, permitiendo asignar tareas, establecer prioridades y monitorizar el progreso de manera clara.

Durante el desarrollo del trabajo se ha empleado para planificar las fases del proyecto, dividir las tareas por entregables y controlar el avance en tiempo real. Se ha optado por Trello frente a herramientas más complejas como Jira por su interfaz visual intuitiva para la metodología Kanban.

2.6.3. Google Meet

Figura 20. Logo Google Meet

Google Meet [21] es una herramienta de comunicación en línea que permite realizar reuniones virtuales de forma sencilla y segura, accesible desde navegadores web o aplicaciones móviles. Facilita la interacción entre varios participantes mediante videollamadas, con funciones como compartir pantalla, activar subtítulos, etc.

En este proyecto se ha empleado para llevar a cabo sesiones de seguimiento y coordinación con el tutor. Se ha elegido por ser una herramienta accesible y gratuita que facilita las videollamadas desde cualquier dispositivo.

2.7 Herramientas de documentación y diseño

2.7.1. Google Docs

Google Docs [22] es una aplicación de procesamiento de texto en la nube que permite la edición colaborativa en tiempo real. Ofrece funciones como comentarios, control de cambios, historial de revisiones y compatibilidad con diversos formatos de exportación.

Se ha utilizado como herramienta principal para redactar la memoria del proyecto y realizar anotaciones sobre el desarrollo de la aplicación.

Figura 21. Logo Google Docs

2.7.2. GoodNotes

Figura 22. Logo Goodnotes

GoodNotes [23] es una aplicación de notas digitales diseñada para la escritura a mano y la organización de documentos. Permite crear esquemas, bocetos y anotaciones con alta precisión, integrando funciones de búsqueda, etiquetas y gestión de cuadernos.

En este proyecto se ha utilizado para planificar la estructura inicial, tomar notas y realizar esquemas visuales de diseño. Se ha elegido por ser una aplicación especializada en escritura a mano que permite crear bocetos con alta precisión en tablet.

3

Análisis

3.1. Requisitos funcionales

Este punto detalla los requisitos funcionales de la plataforma, organizados por áreas clave del sistema y diferenciando las acciones disponibles para el cliente y para el administrador.

Requisitos relativos a la gestión de usuarios

- RF-USR-01: Un administrador puede registrar clientes en la aplicación.
- RF-USR-02: Un administrador puede editar la información de un cliente.
- RF-USR-03: Un administrador puede eliminar clientes registrados.
- RF-USR-04: Un administrador puede resetear la contraseña de un cliente.
- RF-USR-05: Un administrador puede consultar el listado de los clientes registrados.
- RF-USR-06: Un usuario puede iniciar sesión en la aplicación.
- RF-USR-07: Un usuario puede consultar la información de su perfil.
- RF-USR-08: Un usuario puede modificar la información de su perfil.
- RF-USR-09: Un usuario puede cambiar la contraseña desde su perfil.
- RF-USR-10: Un usuario puede recuperar su contraseña en caso de olvido.
- RF-USR-11: Un usuario puede cerrar sesión.

Requisitos relativos a los productos

- RF-PRD-01: Un usuario puede consultar el catálogo de productos.

- RF-PRD-02: Un usuario puede realizar un filtrado de los productos.
- RF-PRD-03: Un usuario puede realizar una búsqueda de productos.
- RF-PRD-04: Un usuario puede ordenar la lista de productos.
- RF-PRD-05: Un cliente puede consultar su lista de productos favoritos.
- RF-PRD-06: Un cliente puede añadir productos a favoritos.
- RF-PRD-07: Un cliente puede eliminar productos de su lista de favoritos.
- RF-PRD-08: Un administrador puede crear una categoría.
- RF-PRD-09: Un administrador puede modificar una categoría.
- RF-PRD-10: Un administrador puede eliminar una categoría.
- RF-PRD-11: Un administrador puede crear nuevos productos.
- RF-PRD-12: Un administrador puede modificar un producto.
- RF-PRD-13: Un administrador puede eliminar un producto.

Requisitos relativos al carrito y los pedidos

- RF-PED-01: Un cliente puede añadir un producto al carrito.
- RF-PED-02: Un cliente puede modificar la cantidad de un producto agregado.
- RF-PED-03: Un cliente puede eliminar un producto del carrito.
- RF-PED-04: Un cliente puede escribir una nota en el pedido.
- RF-PED-05: Un cliente puede realizar un pedido.
- RF-PED-06: Un cliente puede consultar sus pedidos anteriores.
- RF-PED-07: Un cliente puede añadir un pedido anterior al carrito.
- RF-PED-08: Un cliente puede cancelar un pedido ya realizado.
- RF-PED-09: Un cliente puede vaciar el carrito.
- RF-PED-10: Un administrador puede consultar los pedidos de los clientes.
- RF-PED-11: Un administrador puede filtrar los pedidos de la lista.
- RF-PED-12: Un administrador puede ordenar los pedidos de la lista.
- RF-PED-13: Un administrador puede cambiar el estado de uno o varios pedidos.
- RF-PED-14: Un administrador puede descargar la lista de carga en PDF en uno o varios pedidos.

Requisitos relativos a la planificación de rutas

- ➢ RF-RUT-01: El administrador puede consultar un mapa con los clientes con pedidos realizados.
- ➢ RF-RUT-02: El administrador puede consultar la ruta de reparto óptima con los pedidos seleccionados.
- ➢ RF-RUT-03: El administrador puede exportar la ruta a Google Maps.

3.2. Requisitos no funcionales

Este apartado recoge las características técnicas, de rendimiento y calidad que debe cumplir la aplicación, independientemente de su funcionalidad.

- ➢ RNF-01: La aplicación será accesible desde navegadores web modernos y estará optimizada para su uso en ordenadores y dispositivos móviles.
- ➢ RNF-02: La interfaz de usuario será clara, intuitiva y coherente, garantizando una experiencia de uso sencilla tanto para clientes como para el administrador.
- ➢ RNF-03: Los tiempos de respuesta de la aplicación no deberán superar los 3 segundos en operaciones comunes como cargar el catálogo, confirmar pedidos o acceder al historial.
- ➢ RNF-04: La base de datos utilizada será NoSQL, concretamente MongoDB, y estará alojada en la nube mediante el servicio MongoDB Atlas.
- ➢ RNF-05: Las imágenes asociadas a los productos y categorías se almacenarán y servirán a través de la plataforma Cloudinary.

3.3. Casos de uso

A continuación se presentan los casos de uso derivados de los requisitos funcionales. Cada uno describe cómo interactúan los distintos tipos de usuario con el sistema, incluyendo su identificación, descripción, condiciones previas y posteriores, el requisito asociado, y los pasos principales y alternativos del proceso.

Tabla 1: CU01 - Registro de cliente

Nombre	Descripción
Caso de uso	CU01 - Registro de un cliente
Participantes	Administrador
Finalidad	El administrador da de alta un nuevo cliente en el sistema para que éste pueda acceder a la aplicación
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	El cliente queda registrado en la base de datos y puede acceder a la aplicación con sus credenciales
Requisito asociado	RF-USR-01
Secuencia principal	1. El administrador accede a la página de “Clientes”. 2. El administrador selecciona la opción de “Nuevo cliente”. 3. El sistema muestra el formulario de creación de cliente. 4. El administrador introduce los datos del cliente. 5. El administrador selecciona la opción de “Crear cliente”. 6. El sistema comprueba la cumplimentación de todos los campos. 7. El sistema guarda los datos del cliente en la aplicación.
Secuencia alternativa	6. El sistema detecta campos inválidos o incompletos. 7. Muestra un mensaje de error y no permite continuar. Retorno al paso 4 de la secuencia principal.

Tabla 2: CU02 - Modificación de un cliente

Nombre	Descripción
Caso de uso	CU02 - Modificación de un cliente
Participantes	Administrador
Finalidad	El administrador modifica la información de un cliente ya registrado
Precondición	El administrador ha iniciado sesión correctamente

Postcondición	La información actualizada se guarda en la base de datos
Requisito asociado	RF-USR-02
Secuencia principal	1. El administrador accede a la página de “Clientes”. 2. El administrador selecciona el cliente que desea editar. 3. El administrador selecciona la opción de “Editar cliente”. 4. El administrador modifica los datos necesarios. 5. El administrador selecciona la opción de “Guardar cambios”. 6. El sistema comprueba la cumplimentación de los campos. 7. El sistema guarda los datos del cliente.
Secuencia alternativa	6. El sistema detecta campos inválidos o incompletos. 7. Muestra un mensaje de error y no permite continuar hasta corregirlos. Retorno al paso 4 de la secuencia principal.

Tabla 3: CU03 - Eliminación de un cliente

Nombre	**Descripción**
Caso de uso	CU03 - Eliminación de un cliente
Participantes	Administrador
Finalidad	El administrador elimina un cliente del sistema
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	El cliente es eliminado de la base de datos
Requisito asociado	RF-USR-03
Secuencia principal	1. El administrador accede a la página de “Clientes”. 2. El administrador selecciona la opción de “Eliminar” sobre el cliente deseado. 3. El sistema muestra un modal de confirmación de eliminación. 4. El administrador selecciona la opción de “Aceptar”.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. Retorno al paso 1 de la secuencia principal.

Tabla 4: CU04 - Reseteo de contraseña de un cliente

Nombre	Descripción
Caso de uso	CU04 - Reseteo de contraseña de un cliente
Participantes	Administrador
Finalidad	El administrador resetea la contraseña de un cliente
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	Se le envía un email de reseteo de contraseña al cliente
Requisito asociado	RF-USR-04
Secuencia principal	1. El administrador accede a la página de “Clientes”. 2. El administrador selecciona el cliente que desea editar. 3. El administrador selecciona la opción de “Editar cliente”. 4. El administrador selecciona la opción de “Enviar nueva contraseña” en el panel de “Generar nueva contraseña”. 5. El sistema envía una contraseña temporal por email al usuario, indicando que la cambie.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. Retorno al paso 1 de la secuencia principal.

Tabla 5: CU05 - Consulta del listado de clientes

Nombre	Descripción
Caso de uso	CU05 - Consulta del listado de clientes
Participantes	Administrador
Finalidad	El administrador consulta todos los clientes registrados en el sistema
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	El sistema muestra el listado de clientes registrados
Requisito asociado	RF-USR-05
Secuencia principal	1. El administrador accede a la página de “Clientes”.

	2. El sistema muestra el listado de los clientes registrados.
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay clientes registrados.

Tabla 6: CU06 - Inicio de sesión de un usuario

Nombre	Descripción
Caso de uso	CU06 - Inicio de sesión de un usuario
Participantes	Cliente y Administrador
Finalidad	El usuario accede a la aplicación introduciendo sus credenciales
Precondición	El usuario debe estar registrado en la plataforma
Postcondición	El usuario accede a la aplicación con su sesión iniciada
Requisito asociado	RF-USR-06
Secuencia principal	1. El usuario accede a la página de “Acceso a la plataforma”. 2. El usuario introduce su “CIF” y “Contraseña”. 3. El usuario selecciona la opción de “Entrar”. 4. El sistema comprueba la cumplimentación de los campos. 5. El usuario accede a la página principal de la aplicación.
Secuencia alternativa	4. El sistema detecta credenciales incorrectas. 5. El sistema muestra un mensaje de “Credenciales inválidas” e impide el acceso. Retorno al paso 2 de la secuencia principal.

Tabla 7: CU07 - Consulta de perfil de un usuario

Nombre	Descripción
Caso de uso	CU07 - Consulta de perfil de un usuario
Participantes	Cliente y Administrador

Finalidad	El usuario consulta la información de su propio perfil
Precondición	El usuario inicia sesión en la aplicación
Postcondición	El usuario consulta la información de su propio perfil
Requisito asociado	RF-USR-07
Secuencia principal	1. El usuario accede a la página de “Perfil”. 2. El usuario puede consultar sus datos personales.
Secuencia alternativa	No existe ninguna secuencia alternativa.

Tabla 8: CU08 - Modificación de perfil del usuario

Nombre	**Descripción**
Caso de uso	CU08 - Modificación de perfil del usuario
Participantes	Cliente y Administrador
Finalidad	El usuario modifica los datos de su perfil
Precondición	El usuario inicia sesión en la aplicación
Postcondición	Se guardan correctamente las modificaciones del perfil del usuario
Requisito asociado	RF-USR-08
Secuencia principal	1. El usuario accede a la página de “Perfil”. 2. El usuario modifica la información que desee. 3. El usuario selecciona la opción “Guardar cambios”. 4. El sistema comprueba la cumplimentación de los campos modificados.
Secuencia alternativa	4. El sistema detecta algún dato inválido. 5. El sistema muestra un mensaje de error y no permite guardar los cambios. Retorno al paso 2 de la secuencia principal.

Tabla 9: CU09 - Modificación de contraseña del usuario

Nombre	Descripción
Caso de uso	CU09 - Modificación de contraseña del usuario
Participantes	Cliente y Administrador
Finalidad	El usuario modifica la contraseña de su propio perfil
Precondición	El usuario inicia sesión en la aplicación
Postcondición	Se guardan correctamente la nueva contraseña del usuario
Requisito asociado	RF-USR-09
Secuencia principal	1. El usuario accede a la página de “Perfil”. 2. El usuario selecciona el botón de “Mostrar” en el apartado de “Cambiar contraseña”. 3. El usuario introduce la contraseña actual y la nueva. 4. El usuario selecciona la opción de “Cambiar contraseña” 5. El sistema comprueba la cumplimentación de los datos. 6. Se guarda la nueva contraseña en la plataforma.
Secuencia alternativa	5. El sistema detecta algún dato inválido. 6. El sistema muestra un mensaje de error y no permite guardar los cambios. Retorno al paso 3 de la secuencia principal.

Tabla 10: CU10 - Recuperación de contraseña del usuario

Nombre	Descripción
Caso de uso	CU10 - Recuperación de contraseña del usuario
Participantes	Cliente y Administrador
Finalidad	El usuario recupera la contraseña en caso de olvido
Precondición	El usuario está registrado en la aplicación
Postcondición	Se envía un email con una contraseña temporal para poder acceder a la aplicación
Requisito asociado	RF-USR-10

Secuencia principal	1. El usuario accede a la página de inicio de sesión de la aplicación. 2. El usuario selecciona la opción de “¿Has olvidado tu contraseña?”. 3. El sistema muestra una ventana emergente para que introduzca su CIF y su Email. 4. El usuario introduce sus datos y selecciona la opción de “Enviar”. 5. El sistema comprueba la cumplimentación de los datos y envía el email de recuperación de contraseña.
Secuencia alternativa	5. El sistema detecta que no existe esa combinación de CIF y Email en el sistema. 6. El sistema muestra un mensaje de error y no envía el email de recuperación de contraseña. Retorno al paso 4 del de la secuencia principal.

Tabla 11: CU11 - Cierre de sesión del usuario

Nombre	Descripción
Caso de uso	CU11 - Cierre de sesión del usuario
Participantes	Cliente y Administrador
Finalidad	El usuario cierra la sesión de la aplicación
Precondición	El usuario inicia sesión en la aplicación
Postcondición	La sesión el usuario queda cerrada y navega a la página de inicio de sesión
Requisito asociado	RF-USR-11
Secuencia principal	1. El usuario selecciona la opción de “Cerrar sesión” de la barra de navegación. 2. El sistema cierra la sesión y navega a la pantalla de inicio de sesión.
Secuencia alternativa	No se contempla ninguna secuencia alternativa.

Tabla 12: CU12 - Consulta del catálogo de productos

Nombre	Descripción
Caso de uso	CU12 - Consulta del catálogo de productos
Participantes	Cliente y Administrador
Finalidad	Un usuario accede al catálogo para consultar los productos
Precondición	El usuario ha iniciado sesión correctamente
Postcondición	El sistema muestra el catálogo de productos
Requisito asociado	RF-PRD-01
Secuencia principal	1. El usuario accede a la página de inicio donde se encuentra el catálogo con las categorías de productos. 2. El sistema carga y muestra el catálogo con los productos registrados en el sistema.
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay productos registrados.

Tabla 13: CU13 - Filtrado de productos

Nombre	Descripción
Caso de uso	CU13 - Filtrado de productos
Participantes	Cliente y Administrador
Finalidad	El usuario aplica filtros en los productos
Precondición	El usuario ha iniciado sesión en la aplicación
Postcondición	El sistema muestra los productos con el filtro aplicado
Requisito asociado	RF-PRD-02
Secuencia principal	1. El usuario accede a la página de “Catálogo”. 2. El sistema muestra las categorías de productos existentes. 3. El usuario selecciona la opción de “Ver todos los productos”. 4. El sistema muestra todos los productos existentes.

	5. El usuario selecciona la categoría o subcategoría deseada desde el apartado de “Filtrar por categoría”. 6. El sistema muestra todos los productos con el filtro aplicado.
Secuencia alternativa	4a. El sistema muestra un mensaje indicando que no hay productos registrados.
	3b. El usuario selecciona una categoría. 4b. El sistema muestra los productos existentes en esa categoría.
	6c. El sistema muestra un mensaje indicando que no hay productos registrados..

Tabla 14: CU14 - Búsqueda de productos

Nombre	Descripción
Caso de uso	CU14 - Búsqueda de productos
Participantes	Cliente y Administrador
Finalidad	El usuario realiza una búsqueda de productos
Precondición	El usuario ha iniciado sesión en la aplicación
Postcondición	El sistema muestra los productos resultantes
Requisito asociado	RF-PRD-03
Secuencia principal	1. El usuario introduce el nombre del producto en la barra de “Buscar productos” de la barra de navegación. 2. El sistema muestra los productos cuyo nombre coincide con la búsqueda del usuario.
Secuencia alternativa	2. No se encuentra ninguna coincidencia con la búsqueda realizada y muestra un mensaje indicándolo.
	2. El sistema muestra un mensaje indicando que no hay productos registrados.

Tabla 15: CU15 - Ordenación de la lista de productos

Nombre	Descripción
Caso de uso	CU15 - Ordenación de la lista de productos
Participantes	Cliente y Administrador
Finalidad	El usuario puede ordenar la lista de productos
Precondición	El usuario ha iniciado sesión correctamente
Postcondición	La lista de productos mostrados aparece ordenada según el filtro deseado
Requisito asociado	RF-PRD-04
Secuencia principal	1. El usuario accede a la página de “Catálogo”. 2. El sistema muestra el catálogo de productos. 3. El usuario selecciona una categoría o la opción de “Ver todos los productos”. 4. El sistema muestra los productos pertenecientes a la opción seleccionada. 5. El usuario selecciona el desplegable de “Ordenar por” y elige la ordenación deseada. 6. El sistema muestra el listado con los filtros anteriores y ordenados según la selección.
Secuencia alternativa	4. El sistema no encuentra productos para esa selección por lo que no se puede aplicar la ordenación.

Tabla 16: CU16 - Consulta de productos favoritos

Nombre	Descripción
Caso de uso	CU16 - Consulta de productos favoritos
Participantes	Cliente
Finalidad	El cliente puede consultar la lista sus productos favoritos
Precondición	El cliente ha iniciado sesión correctamente y tiene productos favoritos añadidos
Postcondición	El cliente ve su lista de productos favoritos

Requisito asociado	RF-PRD-05
Secuencia principal	1. El cliente accede a la página de “Favoritos” desde la barra de navegación. 2. El sistema muestra todos los productos favoritos que el usuario tiene guardados.
Secuencia alternativa	2. El cliente no tiene productos favoritos guardados y el sistema no muestra ninguno.

Tabla 17: CU17 - Añadir productos a favoritos

Nombre	Descripción
Caso de uso	CU17 - Añadir productos a favoritos
Participantes	Cliente
Finalidad	El cliente puede añadir productos a su lista de favoritos
Precondición	El cliente ha iniciado sesión correctamente
Postcondición	El producto se añade a la lista de favoritos
Requisito asociado	RF-PRD-06
Secuencia principal	1. El cliente accede a la página de “Catálogo” desde la barra de navegación. 2. El cliente selecciona la opción de “Ver todos los productos”. 3. El sistema muestra la lista de productos registrados. 4. El cliente selecciona el corazón de la esquina superior derecha del producto. 5. El sistema almacena el producto en su lista de favoritos.
Secuencia alternativa	3. El sistema muestra un mensaje indicando que no hay productos registrados..

Tabla 18: CU18 - Eliminar productos de favoritos

Nombre	Descripción
Caso de uso	CU18 - Eliminar productos de favoritos

Participantes	Cliente
Finalidad	El cliente puede eliminar productos de su lista de favoritos
Precondición	El cliente ha iniciado sesión correctamente y tiene productos añadidos como favoritos
Postcondición	El producto se elimina de la lista de favoritos
Requisito asociado	RF-PRD-07
Secuencia principal	1. El cliente accede a la página de “Favoritos” desde la barra de navegación. 2. El sistema muestra la lista de productos guardados como favoritos del usuario. 3. El cliente quita la selección del corazón de la esquina superior derecha del producto que desea quitar de la lista. 4. El producto queda eliminado de la lista de favoritos.
Secuencia alternativa	1. El cliente accede a la lista de productos desde el catálogo. 2. El cliente quita la selección del corazón de la esquina superior derecha del producto que desea quitar de la lista. 3. El producto queda eliminado de la lista de favoritos.

Tabla 19: CU19 - Registro de una nueva categoría

Nombre	**Descripción**
Caso de uso	CU19 - Registro de una nueva categoría
Participantes	Administrador
Finalidad	El administrador registra una nueva categoría de productos en el catálogo
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	La categoría queda registrada y aparece en el catálogo
Requisito asociado	RF-PRD-08

Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación. 2. El administrador selecciona la opción de “Nueva categoría”. 3. El administrador rellena los campos necesarios para la creación de la categoría. 4. El administrador selecciona la opción de “Crear categoría”. 5. El sistema guarda la categoría en el catálogo de productos.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. 5. El sistema no guarda los cambios. Retorno al paso 1 de la secuencia principal.

Tabla 20: CU20 - Modificación de una categoría

Nombre	**Descripción**
Caso de uso	CU20 - Modificación de una categoría
Participantes	Administrador
Finalidad	El administrador modifica una categoría de productos en el catálogo
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	La categoría queda modificada y aparece en el catálogo
Requisito asociado	RF-PRD-09
Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación. 2. El administrador selecciona el lápiz de la esquina superior derecha de la categoría que desea editar. 3. El administrador modifica los campos que desee de la categoría. 4. El administrador selecciona la opción de “Guardar cambios”. 5. El sistema modifica la categoría y se guarda.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”.

	5. El sistema no guarda los cambios. Retorno al paso 1 de la secuencia principal.

Tabla 21: CU21 - Eliminación de una categoría

Nombre	**Descripción**
Caso de uso	CU21 - Eliminación de una categoría
Participantes	Administrador
Finalidad	El administrador elimina una categoría de productos en el catálogo
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	La categoría queda eliminada del catálogo
Requisito asociado	RF-PRD-10
Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación. 2. El administrador selecciona la papelera de la esquina superior derecha de la categoría que desee eliminar. 3. El sistema muestra una ventana emergente que solicita la confirmación de eliminación. 4. El administrador selecciona la opción de “Eliminar” 5. El sistema elimina la categoría del sistema.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. 5. El sistema no elimina la categoría. Retorno al paso 1 de la secuencia principal.

Tabla 22: CU22 - Registro de un nuevo producto

Nombre	**Descripción**
Caso de uso	CU22 - Registro de un nuevo producto
Participantes	Administrador

Finalidad	El administrador registra un nuevo producto en el catálogo
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	El producto queda registrado y aparece en el catálogo
Requisito asociado	RF-PRD-11
Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación. 2. El administrador selecciona la opción de “Nuevo producto”. 3. El administrador rellena los campos necesarios para la creación del producto. 4. El administrador selecciona la opción de “Crear producto”. 5. El sistema guarda el producto en el catálogo.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. 5. El sistema no guarda los cambios. Retorno al paso 1 de la secuencia principal.

Tabla 23: CU23 - Edición de un producto

Nombre	**Descripción**
Caso de uso	CU23 - Edición de un producto existente
Participantes	Administrador
Finalidad	El administrador modifica la información de un producto ya existente
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	La información del producto queda actualizada en la base de datos
Requisito asociado	RF-PRD-12
Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación.

	2. El administrador selecciona la opción de “Ver todos los productos”. 3. El administrador selecciona la opción de “Editar” de la parte inferior del producto que desee modificar. 4. El administrador modifica los campos que desea y selecciona la opción de “Guardar cambios”. 5. El sistema guarda el producto modificado.
Secuencia alternativa	4. El administrador selecciona la opción de “Cancelar”. 5. El sistema no guarda los cambios. Retorno al paso 1 de la secuencia principal.

Tabla 24: CU24 - Eliminación de un producto

Nombre	Descripción
Caso de uso	CU24 - Eliminación de un producto
Participantes	Administrador
Finalidad	El administrador elimina un producto del catálogo
Precondición	El administrador ha iniciado sesión correctamente
Postcondición	El producto se elimina del catálogo
Requisito asociado	RF-PRD-13
Secuencia principal	1. El administrador accede a la página de “Catálogo” desde la barra de navegación. 2. El administrador selecciona la opción de “Ver todos los productos”. 3. El administrador selecciona la opción de “Eliminar” de la parte inferior del producto que desee eliminar. 4. El sistema muestra una ventana emergente con la confirmación de eliminación del producto. 5. El administrador selecciona la opción de “Eliminar”. 6. El sistema elimina el producto del sistema.
Secuencia alternativa	5. El administrador selecciona la opción de “Cancelar”. 6. El sistema no elimina el producto. Retorno al paso 1 de la secuencia principal.

Tabla 25: CU25 - Añadir un producto al carrito

Nombre	Descripción
Caso de uso	CU25 - Añadir un producto al carrito
Participantes	Cliente
Finalidad	El cliente puede añadir productos al carrito desde el catálogo
Precondición	El cliente ha iniciado sesión correctamente
Postcondición	El producto se añade al carrito
Requisito asociado	RF-PED-01
Secuencia principal	1. El cliente accede a la página de “Catálogo” desde la barra de navegación. 2. El cliente selecciona la opción de “Añadir” en la parte inferior de un producto. 3. El sistema añade el producto al carrito de la compra del cliente.
Secuencia alternativa	2. El producto no está disponible y no se puede añadir. Retorno al paso 1 de la secuencia principal.

Tabla 26: CU26 - Modificación de un producto del carrito

Nombre	Descripción
Caso de uso	CU26 - Modificación de un producto del carrito
Participantes	Cliente
Finalidad	El cliente modifica la cantidad de un producto del carrito
Precondición	El cliente ha iniciado sesión correctamente y tiene productos en su carrito
Postcondición	El carrito se actualiza reflejando los cambios realizados
Requisito asociado	RF-PED-02
Secuencia principal	1. El cliente accede a la página de “Carrito” desde la barra

	de navegación superior. 2. El cliente modifica la cantidad de un producto. 3. El sistema valida y actualiza los cambios realizados.
Secuencia alternativa	2a. El producto no está disponible y no se puede modificar.
	1b. El cliente accede a la página de “Catálogo” desde la barra de navegación. Retorna al paso 2 de la secuencia principal.
	1c. El cliente accede a la página de “Catálogo” desde la barra de navegación. 2c. El cliente selecciona un producto del catálogo. Retorna al paso 2 de la secuencia principal.

Tabla 27: CU27 - Eliminación de un producto del carrito

Nombre	**Descripción**
Caso de uso	CU27 - Eliminación de un producto del carrito
Participantes	Cliente
Finalidad	El cliente elimina un producto del carrito
Precondición	El cliente ha iniciado sesión correctamente y tiene productos en su carrito
Postcondición	El producto se elimina del carrito
Requisito asociado	RF-PED-03
Secuencia principal	1. El cliente accede a la página de “Carrito” desde la barra de navegación superior. 2. El cliente selecciona la opción de “Eliminar” del producto que desea quitar del carrito. 3. El sistema elimina el producto del carrito.
Secuencia alternativa	1a. El cliente accede a la página de “Catálogo” desde la barra de navegación. Retorna al paso 2 de la secuencia principal.
	1b. El cliente accede a la página de “Catálogo” desde la barra de navegación.

	2b. El cliente selecciona un producto del catálogo. Retorna al paso 2 de la secuencia principal.

Tabla 28: CU28 - Añadir notas en un pedido

Nombre	Descripción
Caso de uso	CU28 - Añadir notas en un pedido
Participantes	Cliente
Finalidad	El cliente puede añadir una nota en un pedido
Precondición	El cliente ha iniciado sesión correctamente
Postcondición	Las anotaciones se guardan en el carrito correctamente
Requisito asociado	RF-PED-04
Secuencia principal	1. El cliente accede a la página de “Carrito” desde la barra de navegación superior. 2. El cliente escribe sus anotaciones en el apartado de “Notas”. 3. El sistema guarda las notas hasta la realización del pedido.
Secuencia alternativa	2. El usuario no realiza ninguna anotación en el pedido.

Tabla 29: CU29 - Realización de un pedido

Nombre	Descripción
Caso de uso	CU29 - Realización de un pedido
Participantes	Cliente
Finalidad	El cliente revisa su carrito y realiza el pedido
Precondición	El cliente ha iniciado sesión correctamente y tiene productos en su carrito.
Postcondición	El pedido queda registrado en la base de datos y se notifica al administrador.

Requisito asociado	RF-PED-05
Secuencia principal	1. El cliente accede a la página de “Carrito” desde la barra de navegación. 2. El cliente selecciona la opción de “Realizar pedido”. 3. El sistema comprueba que los productos añadidos están disponibles en el sistema, 4. El pedido queda registrado y se envía notificación de nuevo pedido al administrador.
Secuencia alternativa	2. El carrito está vacío y no se puede realizar ningún pedido.

Tabla 30: CU30 - Consulta de un pedido

Nombre	Descripción
Caso de uso	CU30 - Consulta de un pedido
Participantes	Cliente
Finalidad	El cliente consulta un pedido realizado anteriormente
Precondición	El cliente ha iniciado sesión correctamente y tiene algún pedido anterior realizado
Postcondición	Accede a la página de pedidos
Requisito asociado	RF-PED-06
Secuencia principal	1. El cliente accede a la página de “Pedidos” desde la barra de navegación. 2. El cliente selecciona un pedido anterior. 3. El sistema muestra los detalles del pedido.
Secuencia alternativa	No se contemplan escenarios alternativos.

Tabla 31: CU31 - Añadir al carrito un pedido anterior

Nombre	Descripción
Caso de uso	CU31 - Añadir al carrito un pedido anterior

Participantes	Cliente
Finalidad	El cliente accede a sus pedidos anteriores y puede añadir al carrito un pedido anteriormente realizado
Precondición	El cliente ha iniciado sesión correctamente y tiene pedidos anteriores registrados
Postcondición	El sistema carga los productos del pedido seleccionado en el carrito
Requisito asociado	RF-PED-07
Secuencia principal	1. El cliente accede a la página de “Pedidos” desde la barra de navegación. 2. El cliente consulta la lista de pedidos anteriores. 3. El cliente selecciona la opción de “Añadir al carrito” del pedido deseado. 4. El sistema navega hacia el carrito del cliente con los productos añadidos correctamente.
Secuencia alternativa	3a. El cliente selecciona el pedido anterior. Retorno al paso 3 de la secuencia principal.

Tabla 32: CU32 - Cancelación de un pedido ya realizado

Nombre	**Descripción**
Caso de uso	CU32 - Cancelación de un pedido ya realizado
Participantes	Cliente
Finalidad	El cliente cancela un pedido realizado anteriormente.
Precondición	El cliente ha iniciado sesión correctamente y tiene un algún pedido realizado.
Postcondición	El sistema cancela el pedido del cliente
Requisito asociado	RF-PED-08
Secuencia principal	1. El cliente accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista y estado de los pedidos.

	3. El cliente selecciona la opción de “Cancelar pedido” sobre un pedido “Pendiente”. 4. El sistema muestra una ventana emergente para confirmar la cancelación del pedido. 5. El cliente selecciona “Eliminar”. 6. El sistema elimina el pedido de la lista de pedidos.
Secuencia alternativa	3. El usuario no puede cancelar el pedido porque no está en estado “Pendiente”. Retorno al paso 2 de la secuencia principal.

Tabla 33: CU33 - Vaciado del carrito

Nombre	Descripción
Caso de uso	CU33 - Vaciado del carrito
Participantes	Cliente
Finalidad	El cliente puede vaciar el carrito de la compra
Precondición	El usuario ha iniciado sesión y tiene productos en el carrito
Postcondición	El sistema elimina los productos del carrito
Requisito asociado	RF-PED-09
Secuencia principal	1. El cliente accede a la página de “Carrito" desde la barra de navegación. 2. El cliente selecciona la opción de “Vaciar carrito”. 3. El sistema muestra una ventana emergente con la confirmación de vaciado. 4. El cliente selecciona la opción de “Aceptar”. 5. El sistema elimina todos los productos del carrito.
Secuencia alternativa	No se contempla ninguna secuencia alternativa.

Tabla 34: CU34 - Consulta de pedidos de clientes

Nombre	Descripción
Caso de uso	CU34 - Consulta de pedidos de clientes

Participantes	Administrador
Finalidad	El administrador puede ver los pedidos de los clientes
Precondición	El administrador ha iniciado sesión
Postcondición	El administrador consulta los pedidos de los clientes
Requisito asociado	RF-PED-10
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos de los clientes. 3. El administrador selecciona un pedido concreto. 4. El sistema muestra la información de ese pedido concreto.
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay pedidos registrados.

Tabla 35: CU35 - Filtrar lista de pedidos de clientes

Nombre	Descripción
Caso de uso	CU35 - Filtrar lista de pedidos de clientes
Participantes	Administrador
Finalidad	El administrador puede filtrar los pedidos de los clientes
Precondición	El administrador ha iniciado sesión
Postcondición	El administrador filtra los pedidos de los clientes
Requisito asociado	RF-PED-11
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos de los clientes. 3. El administrador abre el desplegable del apartado superior de “Filtrar por estado”. 4. El administrador selecciona el filtro deseado. 5. El sistema muestra la lista de pedidos filtrada por el estado seleccionado.

Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay pedidos registrados.

Tabla 36: CU36 - Ordenar lista de pedidos de clientes

Nombre	Descripción
Caso de uso	CU36 - Ordenar lista de pedidos de clientes
Participantes	Administrador
Finalidad	El administrador puede ordenar los pedidos de los clientes
Precondición	El administrador ha iniciado sesión
Postcondición	El administrador ordena los pedidos de los clientes
Requisito asociado	RF-PED-12
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos de los clientes. 3. El administrador abre el desplegable del apartado superior “Ordenar por”. 4. El administrador selecciona la ordenación deseada. 5. El sistema muestra la lista de pedidos ordenada según la selección.
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay pedidos registrados.

Tabla 37: CU37 - Cambio de estado de uno o varios pedidos

Nombre	Descripción
Caso de uso	CU37 - Cambio de estado de uno o varios pedidos
Participantes	Administrador
Finalidad	El administrador puede cambiar el estado de los pedidos de los clientes
Precondición	El administrador ha iniciado sesión y hay pedidos en el

	sistema
Postcondición	El administrador cambia el estado de los pedidos deseados
Requisito asociado	RF-PED-13
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos de los clientes. 3. El administrador selecciona uno o varios pedidos. 4. El administrador abre el desplegable de “Marcar como”. 5. El administrador selecciona el estado deseado. 6. El sistema cambia el estado a los pedidos elegidos.
Secuencia alternativa	3. El administrador selecciona un pedido concreto. 4. El administrador selecciona un nuevo estado desde el apartado de “Cambiar estado”. 5. El sistema cambia el estado del pedido seleccionado.

Tabla 38: CU38 - Descarga de uno o varios pedidos en PDF

Nombre	Descripción
Caso de uso	CU38 - Descarga de uno o varios pedidos en PDF
Participantes	Administrador
Finalidad	El administrador puede descargar uno o varios pedidos en formato PDF
Precondición	El administrador ha iniciado sesión correctamente y hay pedidos realizados
Postcondición	Se genera un archivo PDF con los datos de los pedidos seleccionados
Requisito asociado	RF-PED-14
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación 2. El sistema muestra la lista de pedidos de los clientes. 3. El administrador selecciona uno o varios pedidos. 4. El administrador selecciona la opción de “Descargar

	PDF” 5. El sistema genera un PDF con la lista de carga de los pedidos seleccionados.
Secuencia alternativa	No se contemplan escenarios alternativos.

Tabla 39: CU39 - Consulta del mapa de pedidos

Nombre	Descripción
Caso de uso	CU39 - Consulta del mapa de pedidos
Participantes	Administrador
Finalidad	El administrador puede consultar el mapa con las ubicaciones de los pedidos realizados por los clientes
Precondición	El administrador inicia sesión en la aplicación
Postcondición	El administrador consulta el mapa el mapa de pedidos
Requisito asociado	RF-RUT-01
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos realizados por los clientes. 3. El administrador selecciona la opción de “Ver mapa”. 4. El sistema muestra un mapa interactivo con las ubicaciones de los clientes que han realizado pedidos.
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay productos registrados.. Retorno al paso 3 de la secuencia principal.

Tabla 40: CU40 - Calcular ruta de reparto óptima

Nombre	Descripción
Caso de uso	CU40 - Calcular ruta de reparto óptima
Participantes	Administrador

Finalidad	El administrador puede calcular la ruta de reparto más óptima de los pedidos seleccionados
Precondición	El administrador inicia sesión en la aplicación
Postcondición	El administrador genera la ruta óptima de reparto
Requisito asociado	RF-RUT-02
Secuencia principal	1. El administrador accede a la página de “Pedidos” desde la barra de navegación. 2. El sistema muestra la lista de pedidos realizados. 3. El administrador selecciona el checkbox de los pedidos que quiere calcular la ruta de reparto. 4. El administrador selecciona la opción de “Calcular ruta”. 5. El sistema muestra el mapa con la ruta óptima en base a las ubicaciones de los clientes con pedidos realizados
Secuencia alternativa	2. El sistema muestra un mensaje indicando que no hay productos registrados.

Tabla 41: CU41 - Consultar ruta de reparto en Google Maps

Nombre	**Descripción**
Caso de uso	CU41 - Consultar ruta de reparto en Google Maps
Participantes	Administrador
Finalidad	El administrador puede consultar la ruta de reparto óptima en Google Maps
Precondición	El administrador inicia sesión en la aplicación y ha generado una ruta óptima
Postcondición	El administrador consulta la ruta de reparto en Google Maps
Requisito asociado	RF-RUT-03
Secuencia principal	1. El administrador selecciona la opción de “Ver en Google Maps” en el apartado de “Mapa de clientes y ruta de reparto”.

	2. El sistema abre una nueva pestaña de Google Maps con las paradas de la ruta óptima calculada. 3. El administrador consulta la ruta de reparto correctamente.
Secuencia alternativa	No se contemplan escenarios alternativos.

4

Diseño

En este apartado se describe cómo está diseñado el sistema Click2Order. No solo se explican las pantallas y prototipos de la aplicación, sino también la estructura interna, como la base de datos, los servicios utilizados, cómo se organiza la aplicación y el modelo de datos con las colecciones empleadas. Además, se explican los servicios externos utilizados, como Cloudinary para almacenar imágenes y las API de Mapbox para mostrar mapas y calcular rutas de reparto.

4.1. Arquitectura de la aplicación

En este apartado se detalla la arquitectura del proyecto Click2Order y se argumentan las decisiones clave establecidas para su desarrollo, según se presenta en la Figura 23.

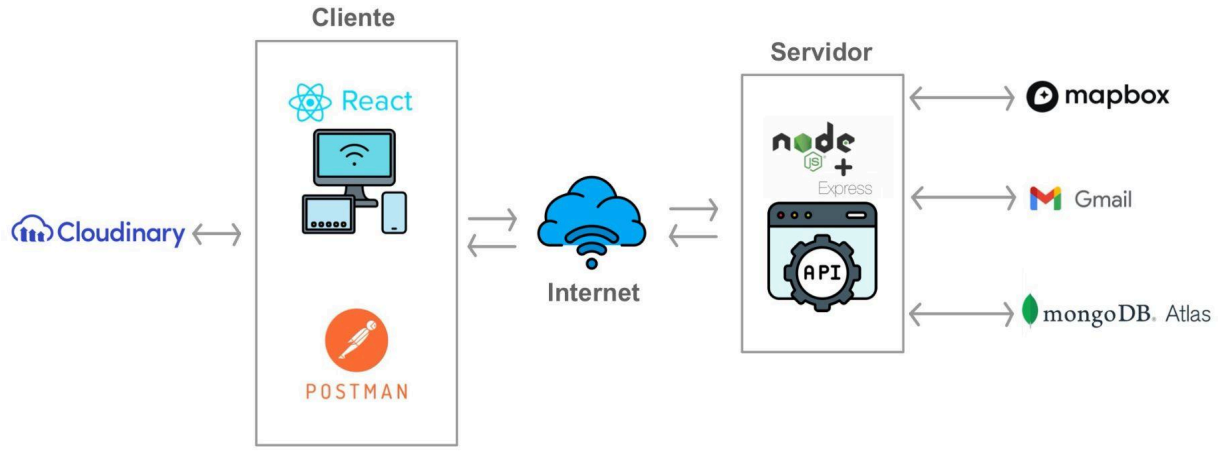

Figura 23. Esquema de la arquitectura de la aplicación

La arquitectura se basa en el modelo Cliente-Servidor, decisión justificada por la necesidad de centralizar la gestión de datos y facilitar la integración con servicios externos especializados. Esta separación permite mantener la lógica de negocio en el servidor mientras el cliente se enfoca únicamente en la presentación.

Para el funcionamiento de la aplicación se requiere un navegador web con conexión a Internet, un proyecto MongoDB Atlas para almacenamiento de datos, una cuenta de Cloudinary para gestión de imágenes y acceso a las APIs de Mapbox para funcionalidades de geolocalización.

El cliente es una aplicación web React que proporciona interfaces diferenciadas para administradores y clientes. Gestiona la interacción del usuario y se comunica con el servidor mediante peticiones HTTP. Además, maneja directamente la subida de imágenes a Cloudinary usando un preset sin firmar para agilizar el proceso.

El servidor implementa una API REST con Node.js y Express que procesa las solicitudes del cliente, gestiona la autenticación JWT y coordina las operaciones con MongoDB Atlas. Se encarga de las integraciones con Mapbox para geocodificación y optimización de rutas, y con Nodemailer para el envío automatizado de notificaciones por correo electrónico.

4.2. Prototipado y maquetas

Antes de iniciar con el desarrollo del frontend, se elaboraron unas maquetas que representaban cual era la idea inicial de cómo debía ser la interfaz de usuario.

Estas maquetas se han realizado a mano en la aplicación de Goodnotes del iPad y han servido como punto de partida para estructurar los apartados de la aplicación, facilitar una planificación visual y anticipar la navegación entre las vistas.

Cabe destacar que las pantallas que se van a mostrar no representan el resultado definitivo de la aplicación, si no una aproximación preliminar ya que a lo largo del desarrollo muchas de estas ideas han sido mejoradas o modificadas.

4.2.1. Maquetas de las vistas del cliente

En la Figura 24 se muestra la ilustración de inicio de sesión donde el cliente introduce su “CIF” y “Contraseña” para acceder a la aplicación. Una vez que las credenciales se han validado el usuario accede a la página inicial donde se mostrarán las categorías de productos existentes como se ve en la Figura 25.

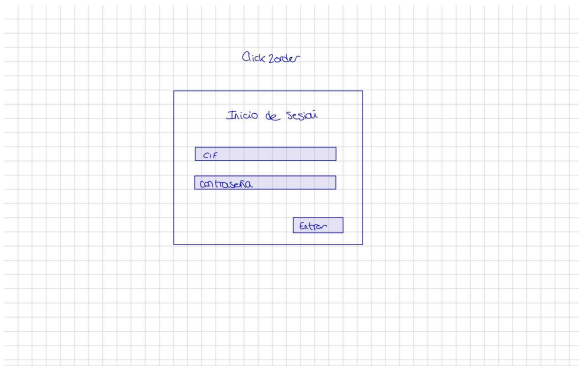

Figura 24. Prototipo inicio de sesión

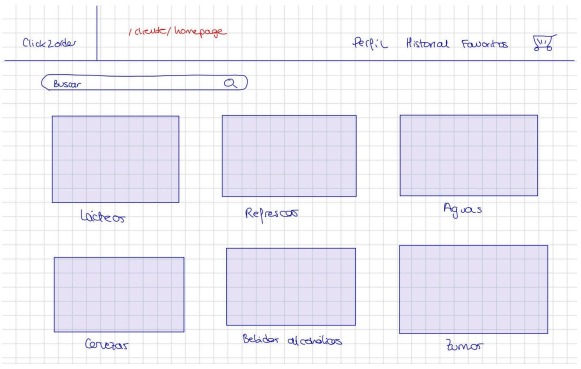

Figura 25. Prototipo página principal

Si el cliente accede a una de las categorías o a la lista de los productos se mostrará el catálogo como en la captura de la Figura 26 y si accede a un producto concreto se mostrará una página con la información de ese producto, como se puede observar en la Figura 27.

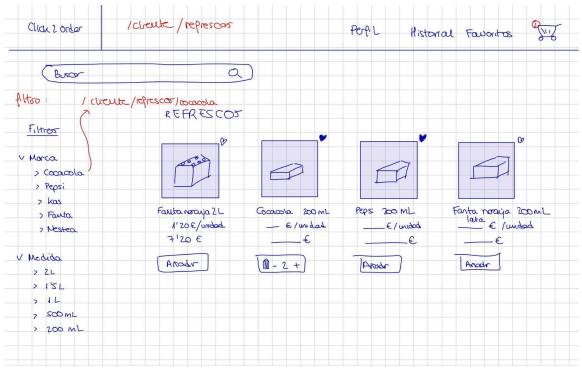

Figura 26. Prototipo catálogo productos

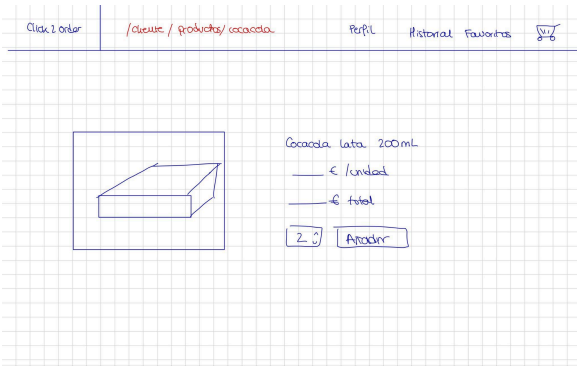

Figura 27. Prototipo productos

Los clientes también pueden realizar otras opciones como ver y modificar los datos de su perfil como se muestra en la Figura 28 y ver el historial de pedidos realizados como se recoge en la Figura 29.

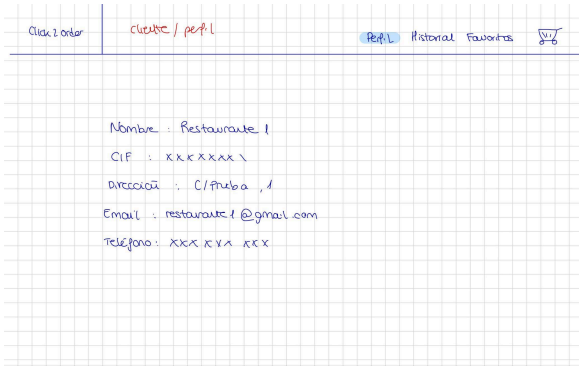

Figura 28. Prototipo perfil de cliente

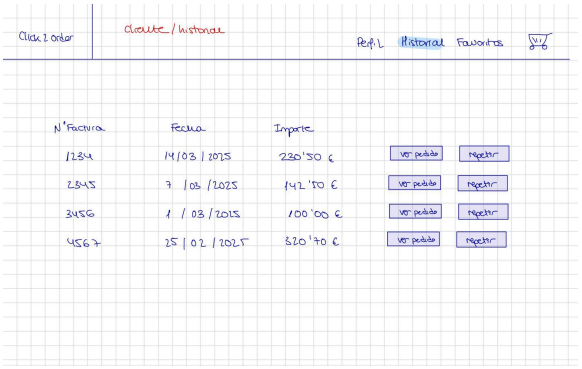

Figura 29. Prototipo historial pedidos

El cliente también podrá guardar los productos deseados en una lista de favoritos o añadir productos a un carrito para realizar un pedido como se ilustra en la Figura 30 y Figura 31.

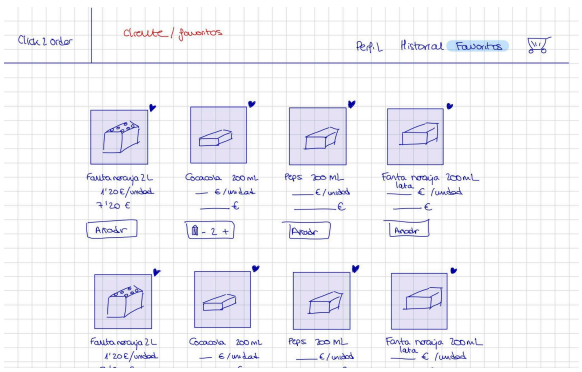

Figura 30. Prototipo favoritos

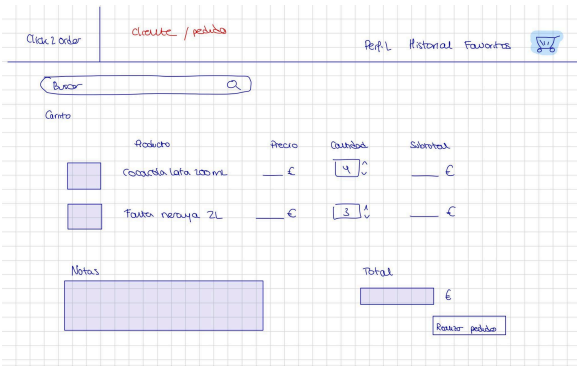

Figura 31. Prototipo carrito de compra

4.2.2. Maquetas de las vistas del administrador

El administrador por su parte, tendrá unas vistas similares a las de los clientes, pero podrá ver y gestionar a todos los comercios registrados en la plataforma como se muestran en las capturas Figura 32 y Figura 33.

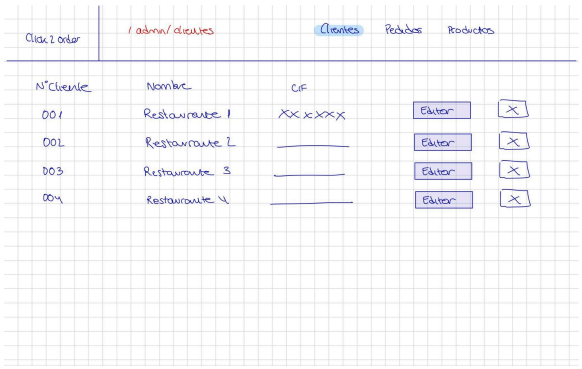

Figura 32. Prototipo vista de clientes

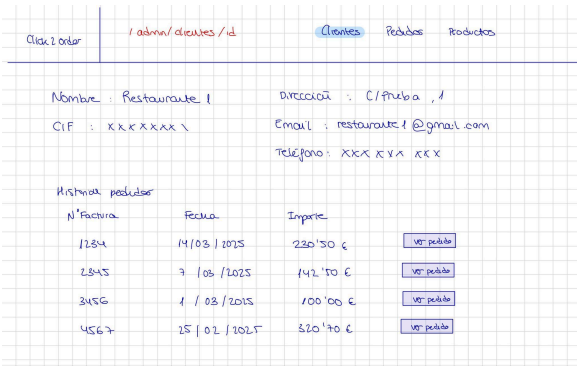

Figura 33. Prototipo perfil del cliente

El administrador también podrá ver la lista de pedidos realizados por los clientes y un mapa con las ubicaciones donde se ha realizado un pedido como se observa en la Figura 34 y podrá ver la información completa de cada uno de los pedidos como se muestra en la Figura 35.

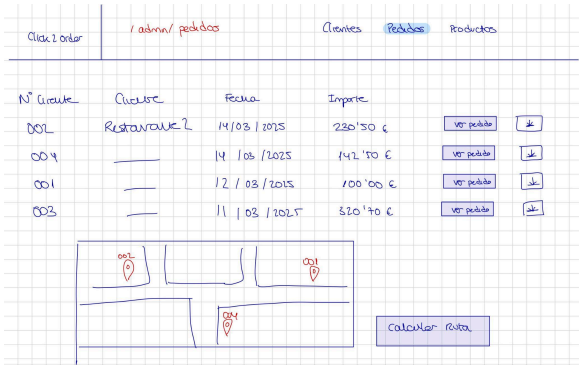

Figura 34. Prototipo pedidos pendientes

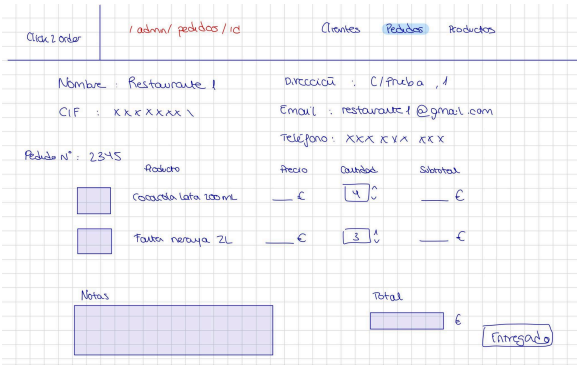

Figura 35. Prototipo pedido concreto

Al igual que los clientes, el administrador tendrá las mismas vistas de las categorías y del catálogo de productos, con la variable de que el administrador puede editar, crear y eliminar estas colecciones. Estas vistas se pueden observar en la Figura 36 y la Figura 37.

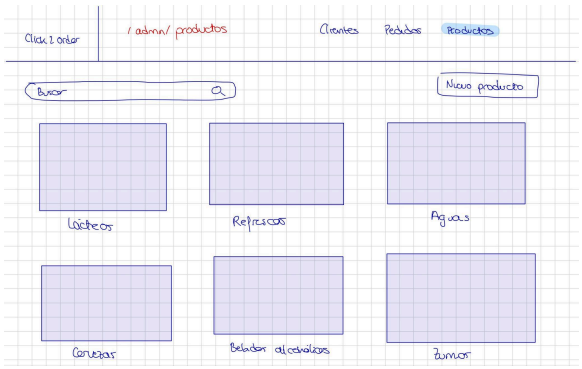

Figura 36. Prototipo categorías

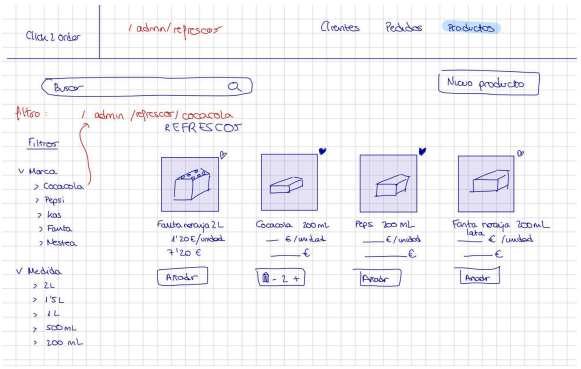

Figura 37. Prototipo catálogo

Por último, se ilustraron las maquetas de edición de un producto, como se observa en la Figura 38 y la creación de un producto nuevo como se observa en la Figura 39.

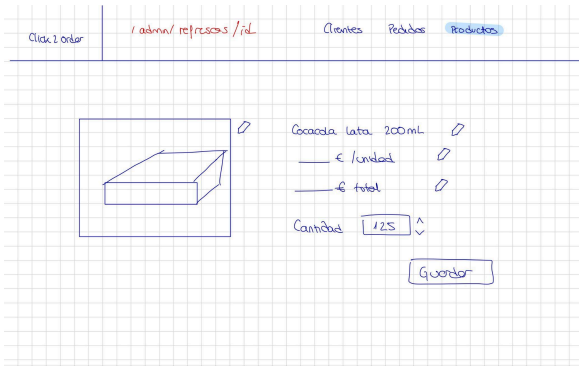

Figura 38. Prototipo edición producto

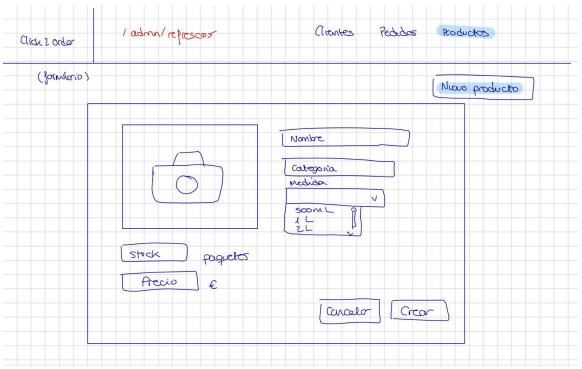

Figura 39. Prototipo producto nuevo

4.3. Colecciones de datos

La base de datos del sistema se ha implementado en MongoDB, un motor NoSQL orientado a documentos que permiten trabajar con esquemas más flexibles y adaptables. Las colecciones se crean automáticamente una vez se ha ejecutado la aplicación por primera vez gracias a la lógica de inicialización del backend.

A continuación, en la Figura 40 se muestra una ilustración con los principales modelos de la aplicación. Para cada uno se muestra un esquema visual de sus atributos.

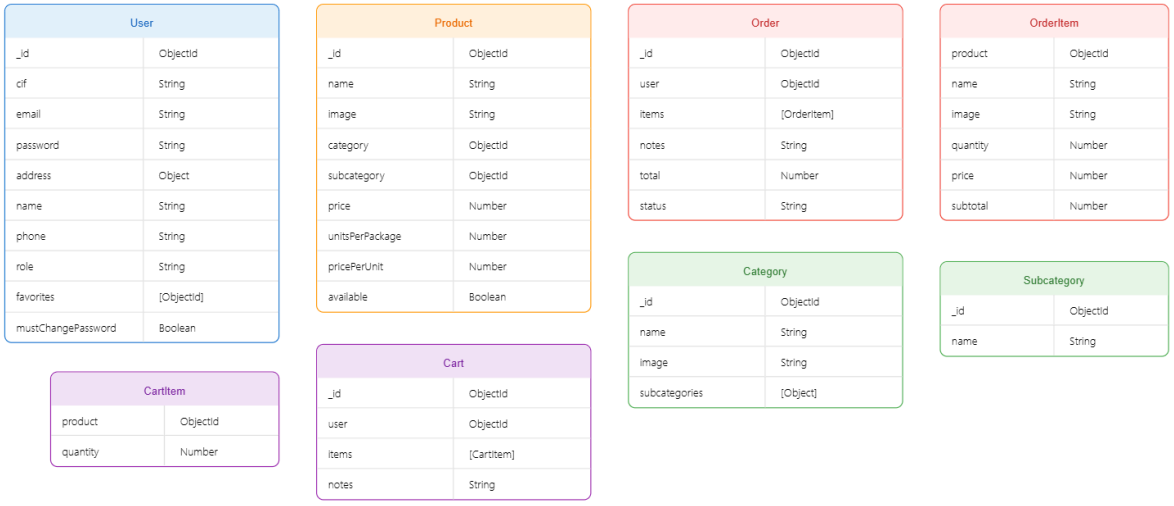

Figura 40. Esquema del modelo de datos

4.3.1. Colección User

El modelo del User almacena la información de los usuarios registrados en la plataforma, tanto los clientes como el administrador.

- **_id:** Identificador único del usuario (ObjectId).
- **cif:** Código de identificación fiscal del usuario.
- **email:** Correo electrónico único del usuario.
- **password:** Contraseña cifrada.
- **address:** Objeto que representa la dirección del usuario.
  - **text:** Dirección en formato texto.
  - **coordinates:** Objeto que representa las coordenadas de la dirección.
    - **latitude:** Latitud geográfica de la dirección.

    - **longitude:** Longitud geográfica de la dirección.
- **name:** Nombre completo del usuario.
- **phone:** Teléfono de contacto.
- **role:** Rol del usuario (administrador o cliente).
- **favorites:** Lista de referencias a los productos marcados favoritos (ObjectId).
- **mustChangePassword:** Booleano que indica si debe cambiar la contraseña. Esto sucede al dar de alta a un usuario, y al restablecer la contraseña.

4.3.2. Colección Product

El modelo Product almacena la información de cada producto disponible en el catálogo

- **_id**: Identificador único del producto (ObjectId).
- **name**: Nombre del producto.
- **image**: URL de la imagen asociada al producto.
- **category**: Referencia a la categoría principal (ObjectId).
- **subcategory**: Referencia a la subcategoría (ObjectId).
- **price**: Precio del producto.
- **unitsPerPackage**: Número de unidades por paquete.
- **pricePerUnit**: Precio por unidad.
- **available**: Booleano que indica si el producto está disponible o no.

4.3.3. Colección Cart

El modelo Cart representa el carrito de la compra de cada cliente.

- **_id**: Identificador único del carrito (ObjectId).
- **user**: Referencia al usuario propietario del carrito (ObjectId).
- **items**: Array de objetos, donde cada uno representa un producto añadido.
    - **product:** Referencia al producto (ObjectId).
    - **quantity:** Cantidad añadida de cada producto al carrito.
    - **subtotal:** Precio total de cada producto por unidad en el carrito (campo virtual).
- **notes**: Campo para notas o solicitudes especiales.

- **total:** Precio total del carrito (campo virtual).

4.3.4. Colección Order

El modelo Order almacena los pedidos realizados por los clientes.

- **_id**: Identificador único del pedido (ObjectId).
- **user**: Referencia al usuario que realiza el pedido (ObjectId).
- **items**: Array de objetos, cada uno representa un producto del pedido.
  - **product**: Referencia al producto (ObjectId).
  - **name**: Nombre del producto.
  - **image**: URL de la imagen del producto.
  - **quantity**: Cantidad solicitada.
  - **price**: Precio del producto.
  - **subtotal**: Precio subtotal en base a la cantidad de ese producto.
- **notes**: Notas añadidas al pedido.
- **total**: Importe total del pedido.
- **status:** Estado actual del pedido.

4.3.5. Colección Category

El modelo Category almacena las categorías disponibles en el catálogo de productos.

- **_id**: Identificador único de la categoría (ObjectId).
- **name**: Nombre de la categoría único.
- **image**: URL de la imagen asociada a la categoría.
- **subcategories**: Array de subcategorías, por defecto no contiene ninguna.
  - **name:** Nombre de la subcategoría (opcional).

5

Desarrollo e implementación

5.1. Desarrollo Backend

El backend de la aplicación está implementado como una API RESTful usando Node.js, Express y MongoDB. En esta parte se gestiona la lógica de negocio y el acceso a los datos para los usuarios, productos, categorías, carritos y pedidos.

5.1.1. Conexión a la base de datos e inicialización de modelos

Se utiliza MongoDB como gestor de base de datos y Mongoose como biblioteca intermediaria para facilitar la conexión y manipulación de datos. La lógica para la conexión con la base de datos se centraliza en `src/db.js`, donde se importa el paquete Mongoose y se utiliza el método `mongoose.conect()` para establecer la conexión usando URL definida en las variables de entorno. En caso de no definirse, se conecta a una base de datos local de MongoDB. Dicha conexión se puede observar en la Figura 41.

```
import mongoose from "mongoose";

const MONGODB_URI = process.env.MONGODB_URI || "mongodb://localhost/click2orderdb"

export const connectDB = async () => {
    try {
        console.log("Intentando conectar a la base de datos");
        await mongoose.connect(MONGODB_URI);
        console.log("Conexión a la base de datos establecida con éxito");
    } catch (error) {
        console.error(`Error al conectar a la base de datos: ${error.message}`);
    }
};
```

Figura 41. Conexión a la base de datos

En el archivo `src/index.js` antes de que se inicie el servidor Express, se importa y se ejecuta la función `connectDB()` como se muestra en la Figura 42, para asegurar que solo se arranque si la conexión a MongoDB es exitosa.

```
import 'dotenv/config';

import app from './app.js';
import { connectDB } from './db.js';

connectDB();
app.listen(4000)
console.log('Servidor en puerto', 4000)
```

Figura 42. Fichero index.js: Conexión antes de iniciar el servidor

Los modelos de datos se definen en la carpeta `src/models/`, donde cada archivo representa una de las colecciones definidas en la aplicación. Cada uno de estos modelos se exporta como un esquema de Mongoose usando `mongoose.Schema`. Cuando estos modelos se importan en los controladores o en otras partes de la aplicación, Mongoose se encarga de crear las colecciones automáticamente en la base de datos, con lo cual se inicializan al arrancar la aplicación.

5.1.2. Estructura de carpetas y archivos principales

La estructura de las carpetas en la parte backend se distribuye modularmente como se muestra en la Figura 43.

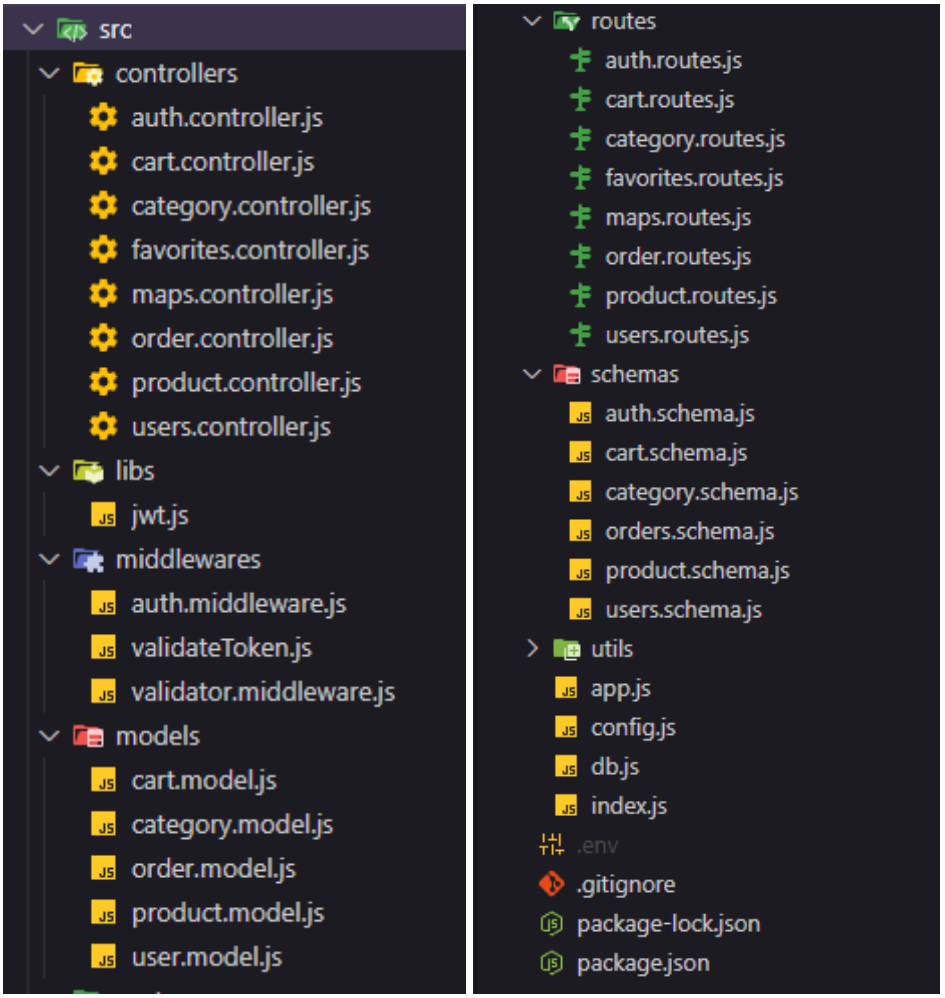

Figura 43. Estructura de las carpetas en el backend

Los principales archivos son:

- **`index.js`:** Se trata del archivo principal de la aplicación, su función es inicializar el entorno, conectar con la base de datos y arrancar el servidor.
- **`app.js`:** Este archivo inicializa la instancia de Express, configura los middlewares globales y monta las rutas iniciales de la API.
- **`db.js`:** centraliza la lógica para la conexión a la base de datos.
- **`.env`:** Se trata de un archivo de texto que almacena las variables de entorno como contraseñas, URLs o claves secretas. No se incluye en el repositorio (está en .gitignore) para proteger la seguridad.

Las principales carpetas son:

- **`controllers/:`** Esta carpeta contiene los controladores que manejan la lógica de negocio de cada recurso de la aplicación. Éstos reciben peticiones HTTP desde las rutas, procesan los datos y devuelven una respuesta al cliente. Por ejemplo en `product.controller.js` se encuentran varias funciones como `createProduct, updateProduct, o deleteProduct` que gestionan las operaciones relacionadas con productos.
- **`models/:`** Aquí se definen los modelos de datos. En cada archivo se define la estructura que tiene cada colección en la base de datos (campos, tipos de datos, relaciones). Por ejemplo en `user.model.js` se define el esquema del usuario y los campos que lo componen como nombre, cif, email, rol, etc.
- **`routes/:`** Esta carpeta contiene los archivos de las rutas donde se definen los endpoints de la API, que realizan las llamadas a los controladores correspondientes. En cada uno se agrupan las rutas de un recurso concreto (como productos, usuarios o pedidos). Por ejemplo en `product.routes.js` se definen rutas como `GET/products` o `POST/products`.
- **`middlewares/:`** En esta carpeta se definen las funciones intermediarias que se ejecutan antes de llegar a los controladores. Contiene por ejemplo a `auth.middleware.js` que verifica si un usuario está autenticado antes de permitir el acceso a algunas rutas. Esto valida los datos o maneja errores.
- **`schemas/:`** En esta carpeta se definen los esquemas de validación de datos de entrada que aseguran que los datos recibidos cumplan con un formato y unas reglas predefinidas. En este caso, en `auth.schema.js` se usa la librería de Zod para validar datos como el formato del email, o la contraseña.
- **`utils/:`** Esta carpeta contiene funcionalidades auxiliares que pueden ser usadas en distintas partes de la aplicación y que no dependen de un recurso específico, como por ejemplo `jwt.js` que se encarga de verificar tokens JWT para la autenticación de usuarios.

5.1.3. Endpoints disponibles en la API

El archivo principal del backend es `app.js`, donde se inicializa la aplicación, se configuran los principales middlewares y se agrupan todas las rutas de la API. Cualquier endpoint definido en los archivos de las rutas estará disponible bajo una URL que comienza por `/api`. Por ejemplo, la ruta para obtener los productos será `/api/products` en lugar de sólo `/products`.

Se le añade ese prefijo a todas las rutas para organizar y diferenciar claramente los endpoints de la API y mejorar la seguridad de la aplicación, se puede observar todas las rutas definidas en la Figura 44.

```
import express from 'express';
import morgan from 'morgan';
import cookieParser from 'cookie-parser';
import cors from 'cors';


import authRoutes from './routes/auth.routes.js';
import productRoutes from './routes/product.routes.js';
import cartRoutes from './routes/cart.routes.js';
import favoritesRoutes from './routes/favorites.routes.js';
import orderRoutes from './routes/order.routes.js';
import categoryRoutes from './routes/category.routes.js';
import userRoutes from './routes/users.routes.js';
import mapsRoutes from './routes/maps.routes.js';

const app = express();

app.use(cors({
    origin: 'http://localhost:5173',
    credentials: true
}));
app.use(morgan('dev'));
app.use(express.json());
app.use(cookieParser());

app.use('/api',authRoutes);
app.use('/api', userRoutes);
app.use('/api',productRoutes);
app.use('/api',cartRoutes);
app.use('/api', favoritesRoutes);
app.use('/api', orderRoutes);
app.use('/api', categoryRoutes);
app.use('/api', mapsRoutes);


export default app;
```

Figura 44. Fichero app.js

A continuación, se detallan los endpoints disponibles organizados en tablas por área funcional. Cada tabla incluye las siguientes columnas:

- **Método**: Define el tipo de operación que se va a realizar.
- **Ruta**: Indica la URL del endpoint, incluyendo los parámetros dinámicos, como por ejemplo ":id" para referirse a un caso concreto.
- **Descripción**: Explicación breve de la funcionalidad del endpoint.
- **Autenticación**: Indica si el endpoint requiere que el usuario esté autenticado, es decir, tenga un token válido.
- **Admin**: Especifica si el endpoint está restringido únicamente a el usuario con el rol de administrador.

Tabla 42. Endpoints de Autenticación

Método	Ruta	Descripción	Autenticación	Admin
POST	/register	Registro de un nuevo usuario en la plataforma	No	No
POST	/login	Permite a un usuario iniciar sesión	No	No
POST	/logout	Cierra la sesión de un usuario autenticado	Sí	No
GET	/verify	Verifica si un usuario tiene una sesión activa y válida	Sí	No
POST	/forgot-password	Envía email para recuperar una contraseña olvidada	No	No

Tabla 43. Endpoints de Usuarios

Método	Ruta	Descripción	Autenticación	Admin
PUT	/users/password	Permite a un usuario autenticado cambiar su contraseña	Sí	No
GET	/users/:id	Devuelve la información de un usuario en concreto	Sí	No
PUT	/users/:id	Actualiza los datos de un usuario en concreto	Sí	No

GET	/users/:userId/orders	Consultar pedidos de un usuario específico	Sí	Sí
GET	/users	Devuelve el listado de todos los usuarios registrados	Sí	Sí
POST	/users	Crea un nuevo usuario en la plataforma	Sí	Sí
DELETE	/users/:id	Elimina a un usuario de la plataforma	Sí	Sí
POST	/users/:id/reset-password	Permite a un administrador resetear la contraseña de un usuario	Sí	Sí

Tabla 44. Endpoints de Productos

Método	Ruta	Descripción	Autenticación	Admin
GET	/products	Devuelve la lista de todos los productos disponibles	Sí	No
GET	/products/:id	Devuelve la información de un producto en concreto	Sí	No
POST	/products	Crea un producto nuevo en la plataforma	Sí	Sí
PUT	/products/:id	Actualiza la información de un producto concreto	Sí	Sí
DELETE	/products/:id	Elimina un producto de la plataforma	Si	Sí

Tabla 45. Endpoints de Categorías

Método	Ruta	Descripción	Autenticación	Admin
GET	/categories	Devuelve la lista de todas las categorías de productos	Sí	No
GET	/categories/:id	Devuelve la información de una categoría en concreto	Sí	No

POST	/categories	Crea una nueva categoría	Sí	Sí
PUT	/categories/:id	Actualiza un a categoría en concreta	Sí	Sí
DELETE	/categories/:id	Elimina una categoría concreta	Sí	Sí

Tabla 46. Endpoints de Favoritos

Método	Ruta	Descripción	Autenticación	Admin
GET	/favorites	Devuelve la lista de los productos favoritos del usuario	Sí	No
POST	/favorites	Añade un producto a la lista de favoritos del usuario	Sí	No
DELETE	/favorites/:productId	Elimina un producto de la lista de favoritos del usuario	Sí	No

Tabla 47. Endpoints de Carrito

Método	Ruta	Descripción	Autenticación	Admin
GET	/cart	Devuelve el carrito de compra de un usuario autenticado	Sí	No
PUT	/cart/items/:id	Añade o actualiza la cantidad de un producto en el carrito	Sí	No
PUT	/cart/notes	Actualiza las notas del carrito	Sí	No
DELETE	/cart/items/:id	Elimina un producto del carrito	Sí	No
DELETE	/cart	Elimina completamente el carrito del usuario	Sí	No
POST	/cart/from-order/	Añade los productos de un	Sí	No

	:orderId	pedido anterior al carrito		

Tabla 48. Endpoints de Pedidos

Método	Ruta	Descripción	Autenticación	Admin
POST	/orders	Crea un pedido nuevo y vacía el carrito del usuario	Sí	No
GET	/orders	Devuelve la lista de pedidos del usuario o todos si es un admin	Sí	No
GET	/orders/:id	Devuelve los detalles de un pedido concreto	Sí	No
PUT	/orders/:id/status	Actualiza el estado de un pedido	Sí	No

Tabla 49. Endpoints de Mapas

Método	Ruta	Descripción	Autenticación	Admin
GET	/maps/reverse-geocoding	Convierte coordenadas a dirección legible	Sí	No
POST	/maps/optimize-route	Calcula ruta optimizada para múltiples puntos	Sí	No

5.1.4. Autenticación en las peticiones

La seguridad de la API de la aplicación se basa principalmente en el uso de JSON Web Tokens (JWT) para autenticar y autorizar a los usuarios.

Cuando un usuario envía sus credenciales para iniciar sesión (CIF y contraseña) al endpoint del login, el controlador de autenticación (`auth.controller.js`) lo recibe y éste busca el usuario en la base de datos y compara la contraseña recibida con la almacenada (usando una función hash).

Si las credenciales son correctas, el controlador del login realiza una llamada a `createAccessToken()` como se observa en la Figura 45, que genera un token

JWT del usuario encontrado con su id de usuario y rol asociado, usando una clave secreta definida en las variables de entorno (`.env`).

```
import jwt from 'jsonwebtoken';

export function createAccessToken(payload) {
    return new Promise((resolve, reject) => {
        jwt.sign(
            payload,
            process.env.TOKEN_SECRET,
            {
                expiresIn: "1d",
            },
            (err, token) => {
                if (err) reject(err);
                resolve(token);
            }
        );
    });
};
```

Figura 45. Fichero jwt.js

El token se envía al cliente (frontend) y se almacena en una cookie llamada token, véase la Figura 46. De esta manera en cada petición que se realice a una ruta protegida, el backend recupera el token de las cookies y comprueba si es válido.

Cuando un usuario realiza una petición a una ruta protegida por ejemplo, acceder a su carrito o realizar un pedido, el backend utiliza un middleware de validación de token para asegurar que la petición viene de un usuario autenticado.

```
export const authRequired = (req, res, next) => {
    const {token} = req.cookies;
    if (!token){
        return res.status(401).json({ meesage: "No existe token, autorización denegada. Debes iniciar sesión." });
    };


    jwt.verify(token, TOKEN_SECRET, (err, user) => {
        if (err) return res.status(403).json ({ message: "Token inválido. Debes iniciar sesión."});

        req.user = user;
        next();
    })
;}
```

Figura 46. Función authRequired

Si el token es válido, entonces la petición continua y llega al controlador correspondiente, donde puede acceder a la información del usuario autenticado a través de `req.user`.

Para las rutas de las acciones del administrador, se utiliza un middleware adicional como se observa en la Figura 47, que verifica que el rol del usuario sea administrador antes de permitir el acceso.

```
export const isAdmin = (req, res, next) => {
    if (req.user.role !== "admin") {
        return res
            .status(403)
            .json({ message: "Acceso denegado: Solo los administradores" });
    }
    next();
};
```

Figura 47. Función isAdmin

5.1.5. Validación datos de entrada

La validación de los datos es fundamental para garantizar la seguridad y la integridad de la API de la aplicación. En este proyecto para ello se utiliza Zod, una biblioteca que valida y parsea los datos. Esto permite definir la forma y las reglas que deben cumplir los datos recibidos en las peticiones, por ejemplo en el inicio de sesión, creación de productos, registro de usuarios, etc y valida que se cumplan los requisitos antes de procesar los datos. Zod permite crear una sintaxis clara, con validaciones personalizadas y mensajes de error detallados.

Estos esquemas se definen en la carpeta `src/schemas/`. Por ejemplo en `auth.schema.js` se valida que los datos del registro y login de usuario vengan en el formato correcto, se puede ver a continuación en la Figura 48.

```
export const loginSchema = z.object({
    cif: z.string({
        required_error: 'El CIF es requerido'
    }),
    password: z.string({
        required_error: 'La contraseña es requerida'
    }),
});
```

Figura 48. Esquema de validación del login

En el proyecto, el flujo de peticiones a una ruta se define como la Figura 49.

```
router.post("/users", authRequired, isAdmin, validateSchema(createUserSchema), createUser);
```

Figura 49. Petición post a los usuarios

- ➢ **`authRequired:`** Este middleware verifica que el usuario que está realizando la petición esté autenticado, es decir, que tenga un token válido. Si no lo está la petición se detiene y devuelve un error.
- ➢ **`isAdmin:`** Si el usuario está autenticado, se ejecuta el siguiente middleware que comprueba que el usuario es administrador, ya que esa acción requiere que solo la pueda realizar quién tenga ese rol específico. Si no es administrador, devuelve un error de acceso denegado al usuario.
- ➢ **`validateSchema(createUserSchema):`** Si el usuario tiene los permisos necesarios para crear a otro usuario, entonces los datos pasan por la validación del cuerpo de la petición para comprobar que los datos introducidos cumplen con los esquemas definidos con Zod.
- ➢ **`createUser:`** Si cumple todas la validaciones anteriores, finalmente se ejecuta la función del controlador correspondiente, que realiza la lógica de negocio y responde al cliente.

5.1.6. Envío de emails

El backend envía emails automáticos para distintas situaciones:

- ➢ **Registro de usuario:** Cuando el administrador crea un usuario nuevo en la aplicación, éste recibe un email de bienvenida a la plataforma y otro email con una contraseña temporal que le sugiere cambiarla al iniciar la aplicación.
- ➢ **Recuperación o reseteo de contraseña:** Si un usuario olvida su contraseña o un administrador la resetea, recibe un email con una nueva contraseña temporal y las instrucciones que debe seguir para restablecerla.
- ➢ **Recepción de nuevo pedido:** Cuando un cliente realiza un pedido, se le envía un email al administrador notificando que ha recibido un pedido nuevo con los datos del usuario y los productos solicitados.

Para enviar un email, se utiliza la biblioteca de Nodemailer, configurada para funcionar con una cuenta de Gmail. Esta configuración se realiza en el archivo `emailTemplates.js`.

Como se puede observar en la Figura 50, la configuración del servicio se gestiona mediante variables de entorno, almacenando de forma segura las credenciales de Gmail (email y contraseña de la aplicación) en el archivo `.env`. Este enfoque garantiza la privacidad de datos sensibles y facilita el cambio de configuración entre diferentes entornos sin necesidad de modificar el código fuente original. Este archivo, se excluye del control de versiones mediante `.gitignore` para evitar exponer las credenciales a repositorios como Github.

```
import nodemailer from 'nodemailer';

export const transporter = nodemailer.createTransport({
    service: "gmail",
    auth: {
        user: process.env.GMAIL_USER,
        pass: process.env.GMAIL_PASS,
    },
});
```

Figura 50. Configuración del servicio de correo

Los cuerpos de los emails se realizan mediante unas plantillas en HTML personalizadas con la información necesaria para cada caso.

Para enviar emails (véase Figura 51), se usan unas funciones que utilizan el transporte de Nodemailer y la plantilla descrita anteriormente para enviar el correo.

```
export const sendPasswordEmail = async (user, temporaryPassword, context = 'general') => {
    try {
        let subject = 'Tu nueva contraseña - Click2Order';
        switch (context) {
            case 'welcome':
                subject = 'Tu contraseña de acceso - Click2Order';
                break;
            case 'forgot':
                subject = 'Recuperación de contraseña - Click2Order';
                break;
            case 'admin_reset':
                subject = 'Tu contraseña ha sido restablecida - Click2Order';
                break;
        }
        const mailOptions = {
            from: `"Click2Order" <${process.env.GMAIL_USER}>`,
            to: user.email,
            subject: subject,
            html: getPasswordEmailTemplate(user, temporaryPassword, context)
        };
        return await transporter.sendMail(mailOptions);
    } catch (error) {
        throw error;
    }
};
```

Figura 51. Función para enviar email de contraseña

Por otro lado, estas funciones de envío de emails se llaman desde los controladores. Por ejemplo, en el momento de creación de un usuario, se realizan dos llamadas desde `user.controller.js` como se ve en la Figura 52, tras guardar un usuario en la base de datos, uno con el email de bienvenida y otro con la contraseña temporal inicial.

```
try {
    await sendWelcomeEmail(savedUser);
} catch (emailError) {
    console.error('Error enviando email de bienvenida:', emailError);
}

try {
    await sendPasswordEmail(savedUser, temporaryPassword, 'welcome');
} catch (emailError) {
    console.error('Error enviando email de contraseña:', emailError);
}
```

Figura 52. Envío de email al crear un usuario

5.1.7. Integración con API de Mapbox

Para las funcionalidades de geolocalización y optimización de rutas en el proyecto se ha integrado la API de Mapbox. En concreto se han integrado dos servicios que ayudan en la conversión de coordenadas geográficas a direcciones legibles mediante geocodificación inversa y el cálculo de rutas optimizadas para la distribución de pedidos.

En el controlador `maps.controller.js` se centraliza toda la lógica relacionada con los servicios de Mapbox y los endpoints principales que consumen la API. La configuración del token de acceso se gestiona mediante una variable de entorno para mantener la seguridad del sistema, como se observa en la Figura 53. El proceso de obtención de este token se explica en el Anexo B.

```
1  const MAPBOX_TOKEN = process.env.MAPBOX_ACCESS_TOKEN;
```

Figura 53. Token de acceso Mapbox

5.1.7.1. API de Geocodificación inversa

La geocodificación inversa o también llamado Reverse Geocoding es un proceso que permite convertir unas coordenadas geográficas numéricas (latitud y longitud) en direcciones legibles para los usuarios.

Esta funcionalidad es fundamental cuando los usuarios seleccionan su ubicación directamente en un mapa interactivo, ya que las coordenadas numéricas como por ejemplo “36.7165657,-4.480035”, no proporcionan información comprensible sobre la ubicación real.

La geocodificación inversa se usa en tres casos específicos en la aplicación:

- ➢ Registro de nuevos clientes donde el administrador selecciona su ubicación.
- ➢ Elección de ubicación de clientes durante la edición de perfil.

Implementación técnica de la API

Se ha empleado la API de Geocodificación V6 de Mapbox [24]. El endpoint implementado es `GET/api/maps/reverse-geocoding` que realiza una llamada a la función `reverseGeocode` donde se realiza toda la lógica.

En este método, primero se validan los datos de entrada como se puede comprobar en la Figura 54.

```
const { lat, lng } = req.query;

if (!lat || !lng) {
    return res.status(400).json({
        message: "Latitud y longitud son requeridas"
    });
}

// Validar coordenadas
const latitude = parseFloat(lat);
const longitude = parseFloat(lng);

if (isNaN(latitude) || isNaN(longitude)) {
    return res.status(400).json({
        message: "Coordenadas inválidas"
    });
}

if (latitude < -90 || latitude > 90 || longitude < -180 || longitude > 180) {
    return res.status(400).json({
        message: "Coordenadas fuera de rango"
    });
}
```

Figura 54. Validación de datos de entrada

Posteriormente se construye la petición a la API de Mapbox, como se muestra en la Figura 55, cuyos parámetros enviados son:

- `longitude/latitude`: Coordenadas geográficas específicas a convertir.
- `types:'address'`: Restringe la búsqueda a unas direcciones específicas.
- `limit:1`: Solicita solo el resultado más preciso.
- `language:'es'`: Configura las respuestas en español.
- `permanent:false`: Indica que los resultados se usan temporalmente, sin almacenamiento permanente.

```
const url = `https://api.mapbox.com/search/geocode/v6/reverse`;
        const params = {
            longitude: longitude,
            latitude: latitude,
            access_token: MAPBOX_TOKEN,
            types: 'address',
            limit: 1,
            language: 'es',
            permanent: false
        };
```

Figura 55. Construcción de la petición a Mapbox

Mapbox devolvería una respuesta en formato JSON similar al ejemplo siguiente:

```
{ "properties": {
        "mapbox_id": "...",
        "feature_type": "address",
            "full_address": "Bulevar Louis Pasteur 36, 29010
Málaga, provincia de Málaga, España",
        "name": "Bulevar Louis Pasteur 36",
        "name_preferred": "Bulevar Louis Pasteur 36",
        "coordinates": {
            "longitude": -4.48004,
            "latitude": 36.717403,
            "accuracy": "point",
            "routable_points": [...]
  }}}
```

El controlador procesa la respuesta y extrae específiamente `feature.properties.full_address` como se observa en la Figura 56, para obtener la dirección completa formateada que incluye número, calle, código postal y ciudad.

```
const feature = response.data.features[0];
const address = feature.properties.full_address || feature.properties.name;
```

Figura 56. Procesamiento de la respuesta Mapbox Geocoding v6 API

5.1.7.2. API de Optimización de Rutas

Para calcular las rutas óptimas de reparto se utiliza la API Optimization V1 de Mapbox [25], que implementa una solución avanzada del clásico Problema del vendedor viajero (Traveling Salesman Problem -TSP), uno de los problemas más estudiados en la teoría de optimización combinatoria y ciencias de la computación y se define formalmente como:

"Dado un grafo completo G = (V,E) donde V es el conjunto de vértices (ubicaciones) y E es el conjunto de aristas con pesos (distancias o tiempos), encontrar un ciclo hamiltoniano de peso mínimo que visite cada vértice exactamente una vez y regrese al vértice de origen".

En el contexto de la distribución de bebidas para este proyecto:

- **Vértices (V):** Representan las ubicaciones de los clientes que han realizado pedidos, más el centro de distribución de origen.
- **Aristas (E):** Representan las conexiones entre las ubicaciones con unos pesos que simulan el tiempo del viaje real por las carreteras.
- **Objetivo:** Minimizar el tiempo total de la ruta de reparto visitando todos los clientes exactamente una vez y volviendo al almacén de origen.

La complejidad de este problema crece de manera factorial $O(n!)$, donde n es el número de ubicaciones, ya que para 5 ubicaciones encontramos 4! = 24 rutas posibles, para 6 ubicaciones 5! = 120 rutas posibles y para 10 ubicaciones por ejemplo, 9! = 362.880 ubicaciones posibles, con lo cual el cálculo de todas las rutas posibles para un conjunto grande de ubicaciones es impracticable de realizar de forma manual, por eso se hace uso de algoritmos heurísticos avanzados.

Implementación técnica de la API

La API Optimization V1 de Mapbox resuelve el problema TSP de manera práctica y eficiente. A diferencia de los métodos exactos que buscan una solución matemáticamente perfecta, utiliza técnicas que encuentran soluciones muy buenas en tiempo razonable.

Esta API soporta múltiples perfiles de navegación, adaptados a diferentes medios de transporte:

- `mapbox/walking`: Optimizado para desplazamientos a pie.
- `mapbox/cycling`: Para trayectos en bicicleta.
- `mapbox/driving-traffic`: Considera datos de tráfico en tiempo real.
- `mapbox/driving`: Perfil estándar para vehículos.

En este proyecto se ha optado por el perfil de navegación `mapbox/driving` ya que es un perfil específico para vehículos automóviles, que consideran limitaciones de velocidad, restricciones de giro y accesibilidad vial, lo cual se adapta muy bien para la simulación de rutas para un vehículo de reparto.

Antes de realizar la petición a la API, se implementa un proceso de validación y formateo de coordenadas para convertirlas al formato requerido por Mapbox.

Si las validaciones son correctas, se construye una petición con los parámetros que se pueden observar en la Figura 57.

```
const coordinatesString = coordinates.map(coord => `${coord[0]},${coord[1]}`).join(';');

        const url = `https://api.mapbox.com/optimized-trips/v1/mapbox/driving/${coordinatesString}`;
        const params = {
            access_token: MAPBOX_TOKEN,
            steps: true,
            geometries: 'geojson',
            overview: 'full',
            roundtrip: true,
            source: 'any',
            destination: 'any',
            language: 'es'
        };
```

Figura 57. Llamada a la API Optimización V1 de Mapbox

Cuando la API recibe los parámetros de entrada configurados anteriormente, ejecuta un proceso sofisticado que va más allá del TSP clásico.

Primero proyecta esos puntos exactos a las carreteras navegables más cercanas dentro de un radio determinado y se valida que cada ubicación sea accesible para vehículos, considerando las restricciones de tráfico, sentidos únicos o zonas restringidas para asegurar que la ruta sea realmente navegable.

A continuación, el sistema calcula una matriz completa de tiempos y distancias reales entre todos los pares de ubicaciones utilizando la red de carreteras y teniendo en cuenta datos de tráfico real, penalizaciones o incorporaciones difíciles para que los costes de la matriz sean lo más precisas posibles.

Con la matriz de costos completa, la API ejecuta algoritmos de optimización que combinan técnicas como el método del vecino más cercano, algoritmos genéticos para explorar múltiples posibilidades y métodos de optimización local.

Finalmente se evalúan las posibles soluciones considerando el "peso de navegación" que combina el tiempo, la distancia y las posibles dificultades y se proporciona una solución con una ruta que no solo es la más corta en términos de distancia sino la más eficiente considerando datos reales de navegación.

Mapbox devuelve esta solución mediante una respuesta JSON bastante completa, pero que resulta compleja para el consumo directo del frontend debido a su granularidad y formato específico de la API. Por tanto, en el controlador se extraen los datos más relevantes y se reorganiza en una estructura más limpia y accesible como se puede observar en la Figura 58, tomándose los valores numéricos principales como la duración, distancia, puntos ordenados, etc.

```
res.json({
    success: true,
    route: {
        duration: trip.duration, // Duración en segundos
        distance: trip.distance, // Distancia en metros
        waypoints: response.data.waypoints, // Puntos ordenados por Mapbox
        geometry: trip.geometry, // Línea de la ruta
        legs: trip.legs, // Segmentos de la ruta
        weight: trip.weight,
        weight_name: trip.weight_name
    },
    original_coordinates: coordinates,
    optimized_coordinates: response.data.waypoints.map(wp => [wp.location[0], wp.location[1]]),
    summary: {
        total_time: Math.round(trip.duration / 60), // minutos
        total_distance: Math.round(trip.distance / 1000 * 100) / 100, // km con 2 decimales
        waypoints_count: response.data.waypoints.length
    }
});
```

Figura 58. Procesamiento de la respuesta JSON de Mapbox

Evaluación de alternativas consideradas:

Se evaluaron múltiples opciones para implementar las funcionalidades de geocodificación y optimización de rutas

- **Implementación manual del algoritmo Dijkstra**: Se consideró desarrollar una implementación propia usando el algoritmo de Dijkstra, con una complejidad $O(V^2)$ donde V es el número de vértices. Aunque es más eficiente que evaluar todas las permutaciones posibles, su implementación requiere estructuras de datos especializadas como colas de prioridad y optimizaciones avanzadas. Además carecería de datos de tráfico real, restricciones viales y optimizaciones de tráfico reales de navegación. Por estas razones se optó por Mapbox, que ya ofrece estas funcionalidades implementadas y optimizadas.
- **OpenStreetMap**: Aunque proporciona datos cartográficos de calidad y es gratuito, su uso directo presenta una serie de limitaciones estéticas que generan una apariencia más básica comparada con otras alternativas. Por esta razón, se optó por utilizar tiles de Mapbox (que internamente usa datos de OpenStreetMap) a través de React-Leaflet, obteniendo mejor presentación visual con solo cambiar una línea de configuración.
- **OSRM (Open Source Routing Machine)**: OSRM fue considerado inicialmente por ser una solución open source para optimización de rutas. Sin embargo, su API pública está destinada únicamente a demostraciones y no es adecuada para aplicaciones con múltiples peticiones concurrentes. Para uso real requiere desplegar infraestructura propia.
- **GraphHopper**: También fue considerado al principio pero presenta una limitación en su plan gratuito de máximo 5 ubicaciones por petición de optimización. Esta restricción sería insuficiente en una empresa de distribución real que posiblemente necesite planificar rutas con más paradas.
- **Mapbox**: Ante estas limitaciones, Mapbox surge como la solución óptima ya que permite hasta 12 ubicaciones por petición (adecuado para rutas de distribución reales), permite 100.000 peticiones gratuitas mensualmente (suficiente para aplicaciones pequeñas-medianas), integra geocodificación y algoritmos heurísticos probados industrialmente, e incluye datos de tráfico real y restricciones viales actualizadas.

5.2. Desarrollo Frontend

5.2.1. Estado global con React Context

En la capa fronted de la aplicación se ha optado por el uso de React Context para centralizar el estado y el uso compartido de las operaciones sobre él. Se usan varios contextos como AuthContext con la lógica de autenticación, CartContext para la gestión del carrito y ProductsContext y CategoriesContext para los productos y las categorías. Cada contexto expone un provider que envuelve a la aplicación y un hook personalizado para consumir fácilmente su estado y sus métodos.

En el caso del AuthContext, se centraliza el usuario autenticado y los flujos de login, registro y cierre de sesión. Al inicio de la aplicación, un efecto comprueba si existe un token válido en las cookies y, en función de la respuesta de la API, se actualiza el estado de autenticación y los datos del usuario. De esta manera, los métodos del contexto pueden conocer si un usuario está autenticado, si está registrado o si cerró la sesión.

Este mismo enfoque se aplica a la gestión del carrito o la gestión de productos y categorías. Así cada contexto conoce el estado y las funciones que operan sobre él.

Esta arquitectura basada en contextos evita que se pasen propiedades a través de múltiples niveles de componentes, mantiene la coherencia del estado global y facilita la reutilización de lógica de negocio entre componentes. Además, al separar las responsabilidades en contextos específicos, se mejora la mantenibilidad del código y se facilita la depuración al tener cada dominio de la aplicación claramente delimitado.

A continuación se puede observar en la Figura 59 cómo se ha diseñado la estructura de las rutas en la aplicación

```jsx
<UsersProvider>
      <CategoriesProvider>
        <ProductsProvider>
          <FavoritesProvider>
            <CartProvider>
              <OrdersProvider>
                <BrowserRouter>
                  {user && <Navbar />}
                  <Routes>
                    <Route path="/login" element={<LoginPage />} />

                    <Route element={<PrivateRoute><Outlet /></PrivateRoute>}>
                      <Route path="/" element={<HomePage />} />
                      <Route path="/perfil" element={<UserFormPage />} />
                      <Route path="/productos" element={<ProductsPage />} />
                      <Route path="/productos/:id" element={<ProductDetailPage />} />
                      <Route path="/pedidos" element={<OrdersPage />} />
                      <Route path="/pedidos/:id" element={<OrderDetailPage />} />
                      <Route path="/favoritos" element={<FavoritesPage />} />
                      <Route path="/carrito" element={<CartPage />} />

                      {/* Admin-only */}
                      <Route element={<AdminRoute><Outlet /></AdminRoute>}>
                        <Route path="/admin/clientes" element={<UsersPage />} />
                        <Route path="/admin/clientes/nuevo" element={<UserFormPage mode="create" />} />
                        <Route path="/admin/clientes/:id/info" element={<UserDetailPage />} />
                        <Route path="/admin/clientes/:id" element={<UserFormPage mode="edit" />} />
                        <Route path="/admin/categorias/nueva" element={<CategoryFormPage mode="create" />} />
                        <Route path="/admin/categorias/:id" element={<CategoryFormPage mode="edit" />} />
                        <Route path="/admin/productos/nuevo" element={<ProductFormPage mode="create" />} />
                        <Route path="/admin/productos/:id" element={<ProductFormPage mode="edit" />} />
                        <Route path="/admin/pedidos" element={<AdminOrdersPage />} />
                        <Route path="/admin/pedidos/:id" element={<AdminOrderDetailPage />} />
                      </Route>
                    </Route>
                    <Route path="*" element={<Navigate to={user ? "/" : "/login"} />} />
                  </Routes>
                </BrowserRouter>
              </OrdersProvider>
            </CartProvider>
          </FavoritesProvider>
        </ProductsProvider>
      </CategoriesProvider>
    </UsersProvider>
```

Figura 59. Estructura de rutas del frontend de la aplicación

5.2.2. Barra de navegación para clientes

La pantalla principal tras la página de inicio de sesión para los clientes incluye una barra de navegación fija diseñada para ofrecer un acceso rápido y sencillo a las secciones principales de la aplicación. En la versión para ordenador, la barra de navegación muestra los accesos de *Catálogo, Perfil, Pedidos, Favoritos, Carrito y Cerrar sesión* junto con una barra de *Buscar productos* por palabras clave. Al pulsar en el logo de Click2Order, también navega a la página inicial con las categorías de los productos. Estas opciones se pueden observar en la Figura 60.

Figura 60. Barra de navegación del cliente

Para dispositivos más estrechos como un dispositivo móvil, la barra de navegación se adapta mostrando un menú desplegable con los mismos accesos a las secciones de la aplicación.

5.2.3. Barra de navegación para el administrador

La pantalla principal para el administrador también dispone de una barra de navegación similar a la de los clientes, manteniendo una coherencia visual y de uso en toda la aplicación.

La barra aparece en la parte superior y permite al administrador acceder a sus principales funcionalidades como *Catálogo, Perfil, Clientes, Pedidos y Cerrar sesión,* como se muestra en la Figura 61. Al igual que en la interfaz de cliente, se incluye el buscador de productos y el icono de inicio que redirige a la página principal.

La principal diferencia con la barra de clientes radica en las opciones disponibles, ya que el administrador tiene el poder de gestionar los clientes, los productos y categorías y visualizar todos los pedidos pendientes y la ruta óptima sugerida en función de la ubicación de los clientes. Gracias a este diseño, el administrador puede acceder rápidamente a todas las herramientas necesarias para la gestión eficiente de la empresa desde cualquier dispositivo.

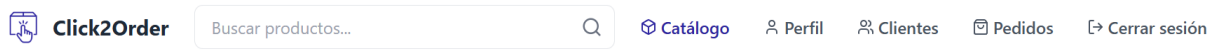

Figura 61. Barra de navegación del administrador

5.2.4. Subida de imágenes con Cloudinary

Para gestionar las imágenes de los productos y las categorías que sube el administrador, se ha integrado Cloudinary como servidor de almacenamiento. Esta subida se realiza desde la parte del cliente usando un *upload preset* llamado `click2order_preset` en modo *unsigned* como se observa en la Figura 62, para evitar la necesidad de firmar cada petición en el backend y así agilizar el desarrollo.

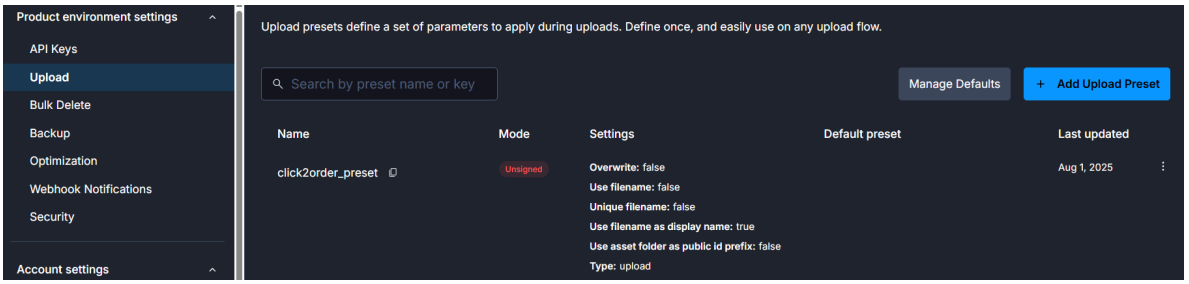

Figura 62. Configuración del preset en Cloudinary

El método de subida se encuentra en `ProductsContext` al igual que en `CategoriesContext` cambiando únicamente el nombre de la carpeta de destino. Se puede observar el método empleado para la subida de imágenes en la Figura 63.

```
const uploadToCloudinary = async (file) => {
    const formData = new FormData();
    formData.append('file', file);
    formData.append('upload_preset', 'click2order_preset');
    formData.append('folder', 'products');

    try {
      const response = await fetch(
        `https://api.cloudinary.com/v1_1/dnek7u3jy/image/upload`,
        {
          method: 'POST',
          body: formData
        }
      );

      if (!response.ok) {
        throw new Error('Error subiendo la imagen');
      }

      const data = await response.json();
      return data.secure_url;
    } catch (error) {
      console.error('Error uploading to Cloudinary:', error);
      throw error;
    }
  };
```

Figura 63. Método de subida de imágenes a Cloudinary

Cloudinary procesa cada subida según las transformaciones del preset y éste devuelve un JSON con el secure_url que es la URL de la imágen optimizada.

La llamada a la API de subida funciona enviando un POST a `https://api.cloudinary.com/v1_1/<cloud_name>/image/upload` donde se aplican los parámetros predefinidos en el preset.

5.2.5. Mapas con Mapbox

Para la integración de mapas interactivos en el frontend se utiliza la combinación de React-Leaflet como biblioteca de mapas para React junto con tiles de Mapbox para el renderizado visual proporcionando mapas de alta calidad y una interfaz interactiva fluida.

Selección de ubicación de usuarios

En los formularios de creación y edición de usuarios (tanto para clientes como administrador), se implementa un mapa interactivo que permite seleccionar la ubicación geográfica. Cuando el usuario hace clic en cualquier parte del mapa, se ejecuta un componente personalizado `MapClickHandler` como se observa en la Figura 64 que:

- ➢ Captura las coordenadas del clic (latitud y longitud).
- ➢ Actualiza automáticamente los campos del formulario con estas coordenadas.
- ➢ Realiza una petición al backend para obtener la dirección legible mediante geocodificación inversa.
- ➢ Muestra un marcador visual en la posición seleccionada.

```
function MapClickHandler({ onLocationSelect, onMapReady }) {
  const map = useMapEvents({
    click: (e) => {
      onLocationSelect(e.latlng);
    },
    ready: () => {
      onMapReady(map);
    }
  });
  return null;
}
```

Figura 64. Componente para manejar clicks en el mapa

Esta funcionalidad elimina la necesidad de introducir coordenadas manualmente y proporciona una experiencia de usuario intuitiva para la selección de ubicaciones, como se puede observar en la Figura 65.

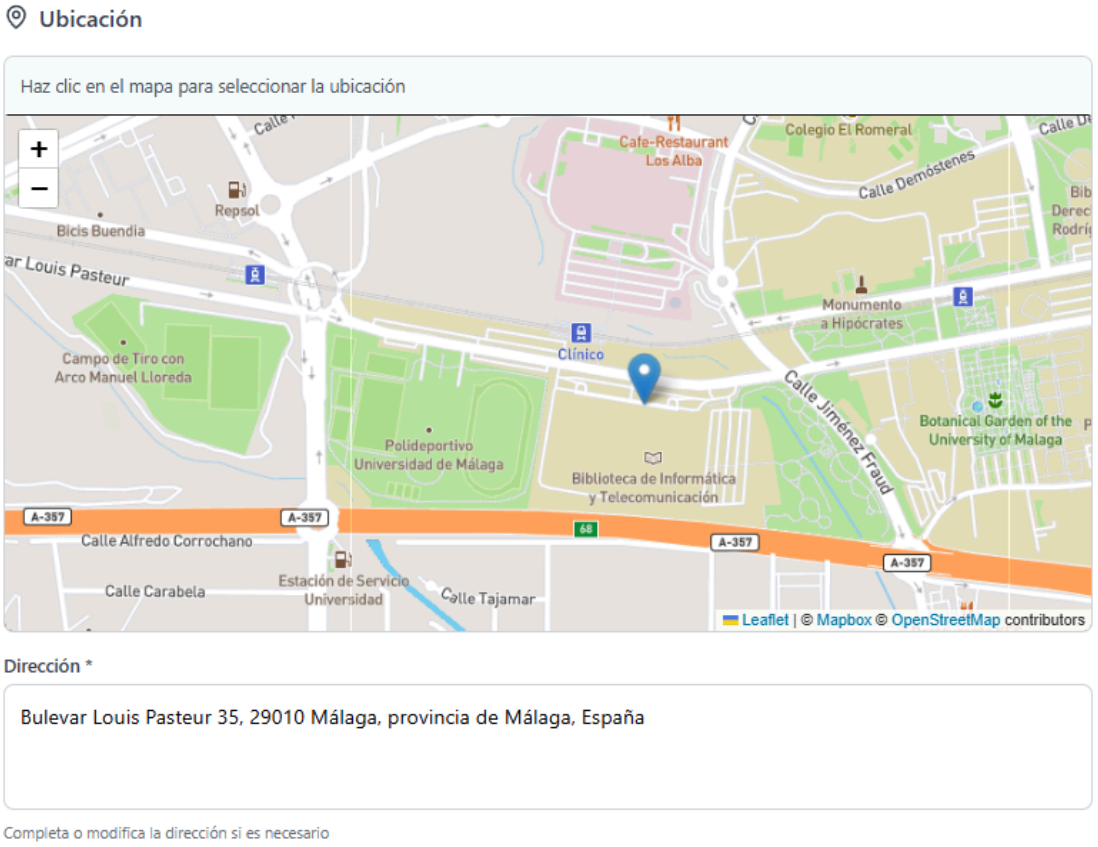

Figura 65. Selección de la ubicación de usuarios

Visualización de rutas optimizadas

En el panel de la administración de pedidos se implementa un mapa avanzado que muestra la optimización de rutas de reparto.

En dicho mapa se muestran unos marcadores personalizados como se puede ver en la Figura 66, que utilizan un código de colores diferenciado:

- Marcador rojo para la ubicación del administrador.
- Marcador azul para clientes con pedidos.
- Marcador verde para clientes con pedidos seleccionados para la ruta.
- Numeración secuencial en los marcadores que indica el orden de la ruta optimizada.

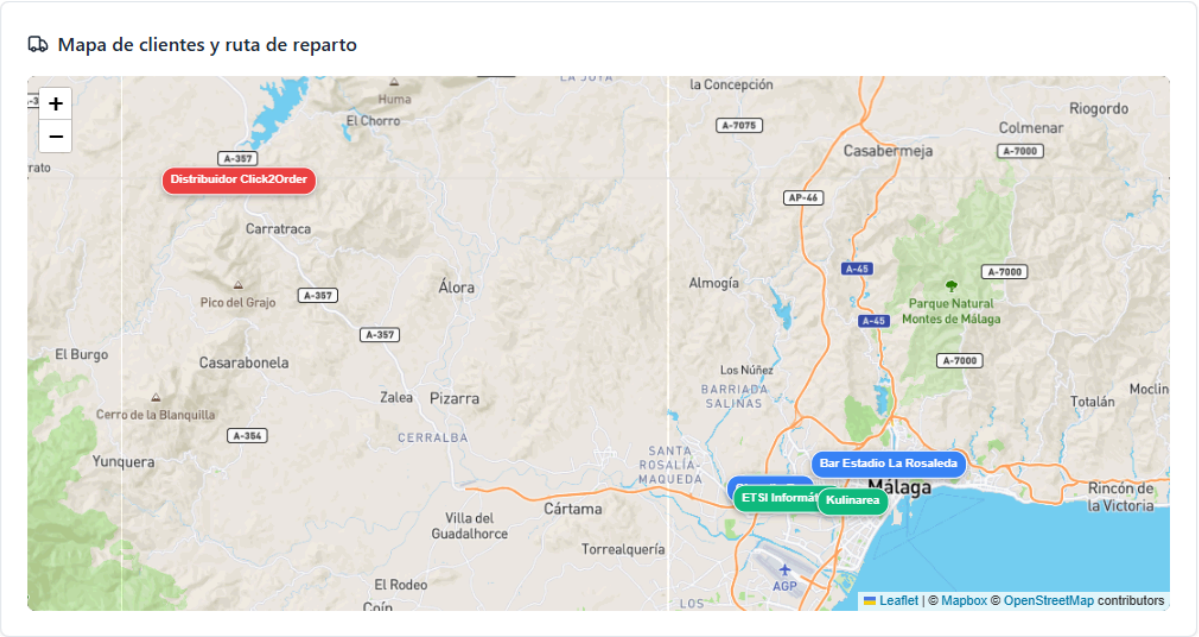

Figura 66. Visualización de marcadores personalizados

Cuando se calcula una ruta óptima de reparto, se dibuja en el mapa la trayectoria completa utilizando el componente `Polyline` de React-Leaflet, que conecta todas las coordenadas y en los marcadores se incluyen popups informativos que muestran información relevante sobre el pedido, como el nombre de la empresa, dirección completa, número de pedidos realizados y estado de los pedidos.

Los mapas consumen directamente los datos procesados por las APIs del backend, la de geocodificación inversa para convertir coordenadas en direcciones legibles durante la selección y la de optimización de rutas para obtener la secuencia óptima de paradas en el reparto.

El frontend procesa las respuestas del backend y las adapta al formato requerido por React-Leaflet, manejando específicamente la transformación de coordenadas (de formato longitud y latitud de Mapbox a longitud y latitud requerido por Leaflet) y la creación de elementos visuales dinámicos.

6

Pruebas

En este capítulo se exponen las pruebas desarrolladas para evaluar el correcto funcionamiento del sistema. La herramienta principal para las pruebas del backend ha sido Postman, que permite realizar llamadas controladas a la API REST sin depender de la interfaz de usuario. Por otro lado, el resto de pruebas para el frontend han sido realizadas mediante la interfaz del usuario

6.1. Metodologías de pruebas Postman

Las pruebas se han estructurado mediante peticiones HTTP, utilizando los métodos GET, POST, PUT y DELETE según la naturaleza de cada funcionalidad. Estas pruebas verifican que la API devuelve los códigos de estado HTTP apropiados según cada situación. Los principales códigos utilizados son:

- 200 (OK): Indica que la petición se ha procesado correctamente y se devuelven los datos solicitados.
- 201 (Created): Confirma que un nuevo recurso se ha creado exitosamente (registro de usuarios, creación de productos).
- 400 (Bad Request): Señala que la petición contiene datos inválidos o campos obligatorios faltantes.
- 401 (Unauthorized): Indica que la petición requiere autenticación o las credenciales proporcionadas son incorrectas.
- 403 (Forbidden): Confirma que el usuario está autenticado pero no tiene permisos para realizar la operación solicitada.

- ➢ 404 (Not Found): Indica que el recurso solicitado no existe en el sistema.
- ➢ 500 (Internal Server Error): Señala un error interno del servidor durante el procesamiento de la petición.

Durante el desarrollo se ha creado una colección de pruebas en Postman de los endpoints de la API REST de Click2Order. Como se puede observar en la Figura 67, la colección incluye pruebas organizadas por módulos funcionales: autenticación (auth), productos (products), carrito (cart), categorías (category), favoritos (favorites), usuarios (users) y pedidos (orders).

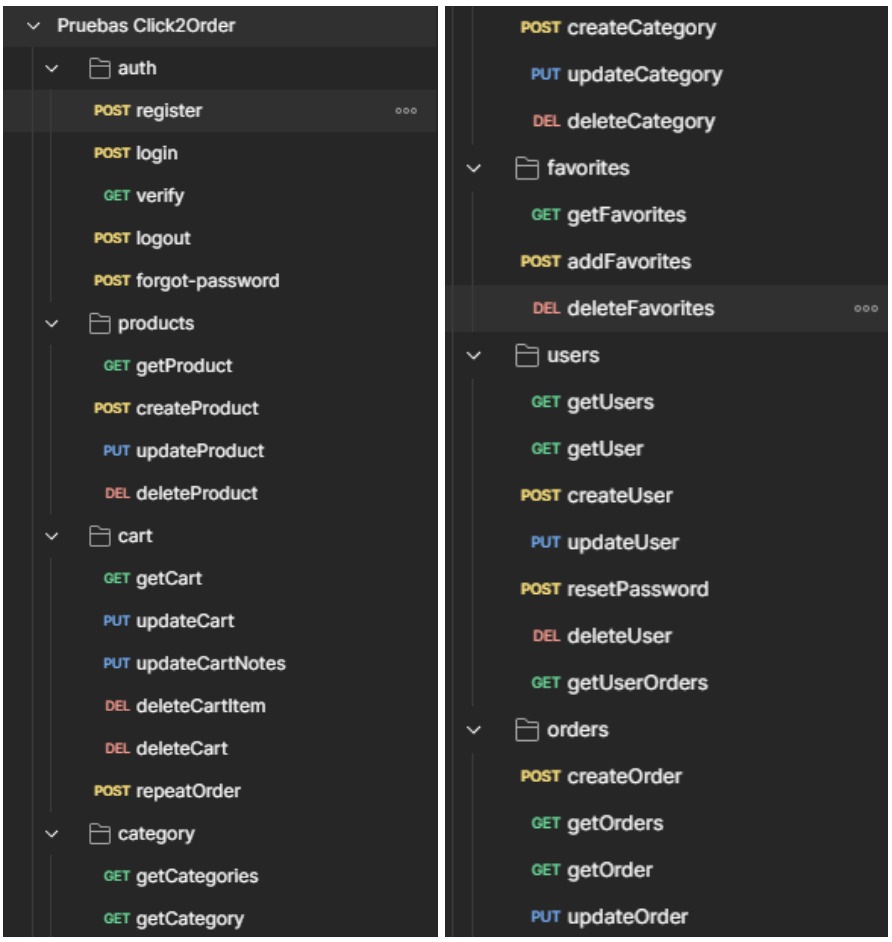

Figura 67. Lista de pruebas en Postman

La colección contiene diferentes pruebas que validan las funcionalidades del sistema, incluyendo casos válidos e inválidos para garantizar el correcto manejo de los errores. A continuación se muestran algunas de las pruebas realizadas como ejemplos del proceso de validación.

6.1.1. Pruebas de autenticación

➢ Registro de usuario

En la Figura 68, se muestra una prueba de registro con datos inválidos, al enviar una petición POST a `/api/register` con un email en formato incorrecto, el sistema valida los datos de entrada y devuelve un código 400 (Bad Request) en una respuesta con detalle “Formato de email inválido”, confirmando que las validaciones de Zod funcionan correctamente y previenen el registro de usuarios con datos incorrectos.

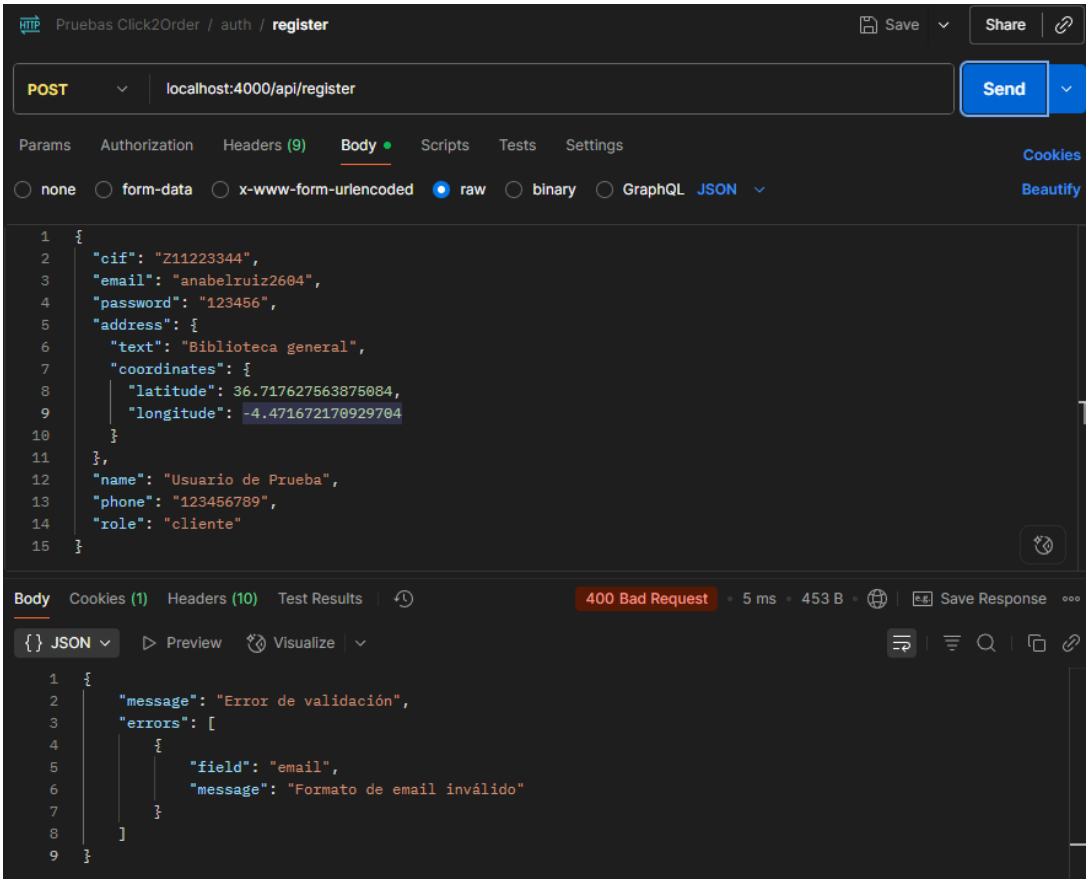

Figura 68. Petición POST /api/register inválida

En la Figura 69 en cambio, se muestra el flujo correcto de registro de usuario y donde el sistema ha devuelto un código 200 (OK), que indican que la petición se ha procesado exitosamente.

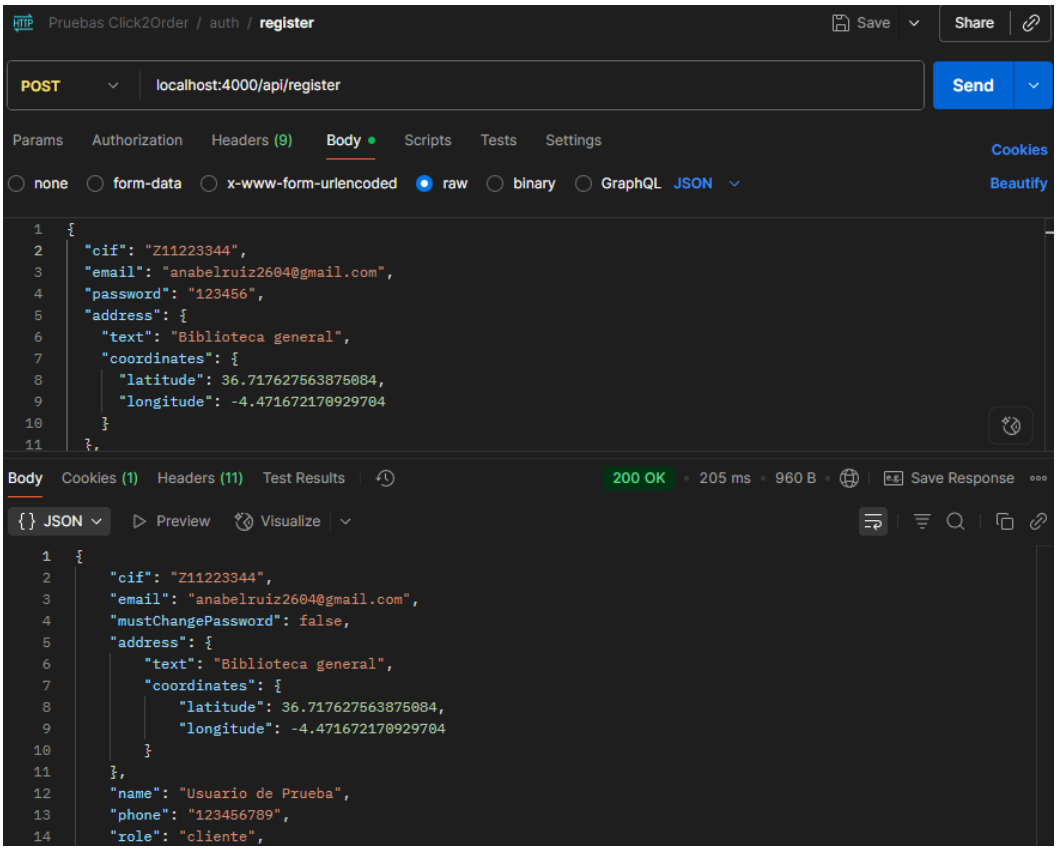

Figura 69. Petición POST /api/register válida

➢ Login de usuario

Para el login del usuario se han realizado pruebas introduciendo las credenciales inválidas, como se muestra en la Figura 70 y con las credenciales correctas como se muestra en la Figura 71.

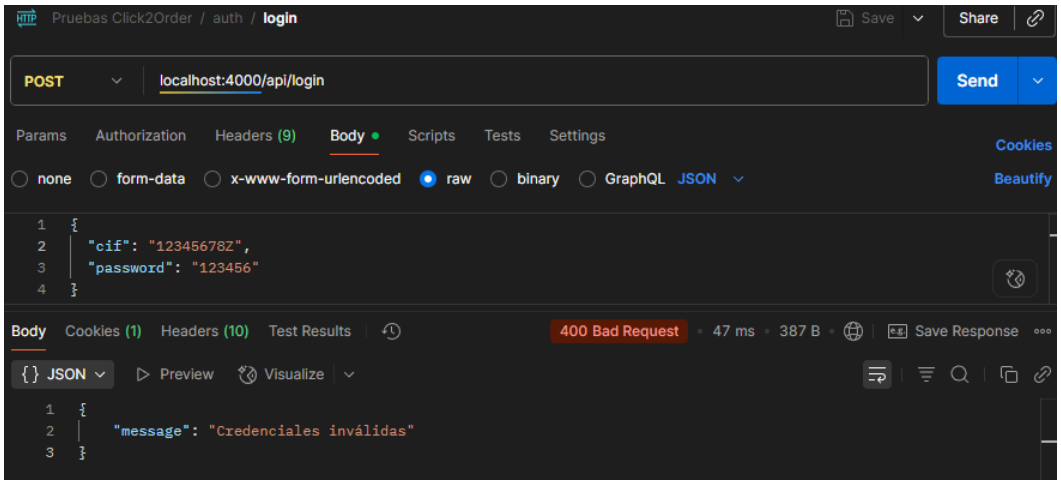

Figura 70. Petición POST /api/login inválida

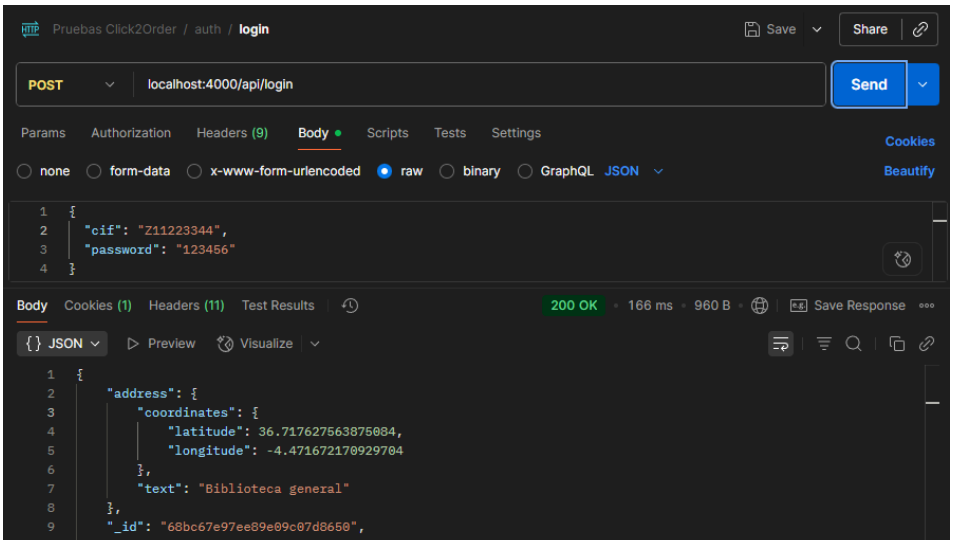

Figura 71. Petición POST /api/login válida

6.1.2. Pruebas de productos

➢ Creación de productos

Como se observa en la Figura 72, una prueba de creación de producto enviando una petición POST a `/api/products` con el campo “subcategory” vacío (“”), el sistema devuelve código 400 (Bad Request) con “Error de validación” especificando que “La subcategoría no puede estar vacía”, confirmando que las validaciones de Zod funcionan correctamente para los productos.

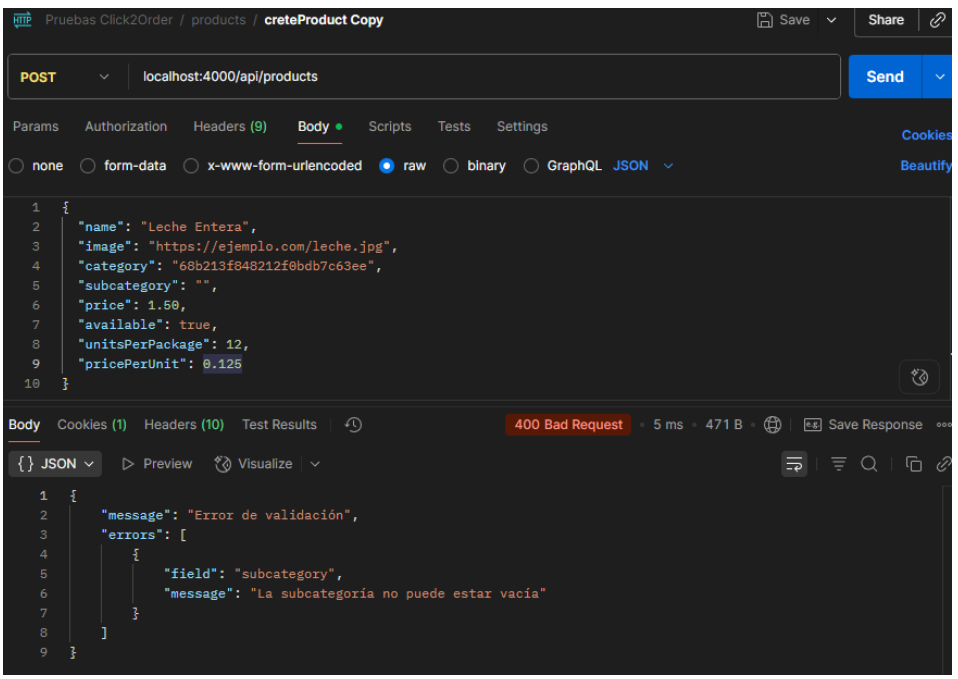

Figura 72. Petición POST /api/products inválida

En cambio en la Figura 73, se muestra la creación exitosa de un producto.

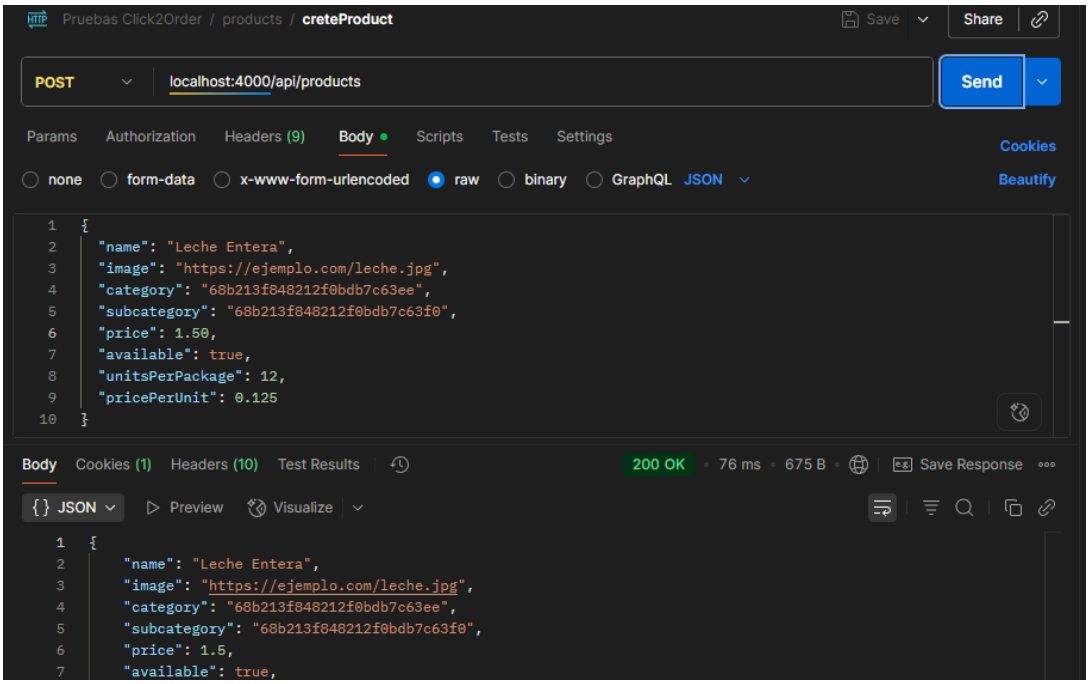

Figura 73. Petición POST /api/products válida

➢ Actualizar un producto

Como se puede observar en la Figura 74, se actualiza correctamente un producto.

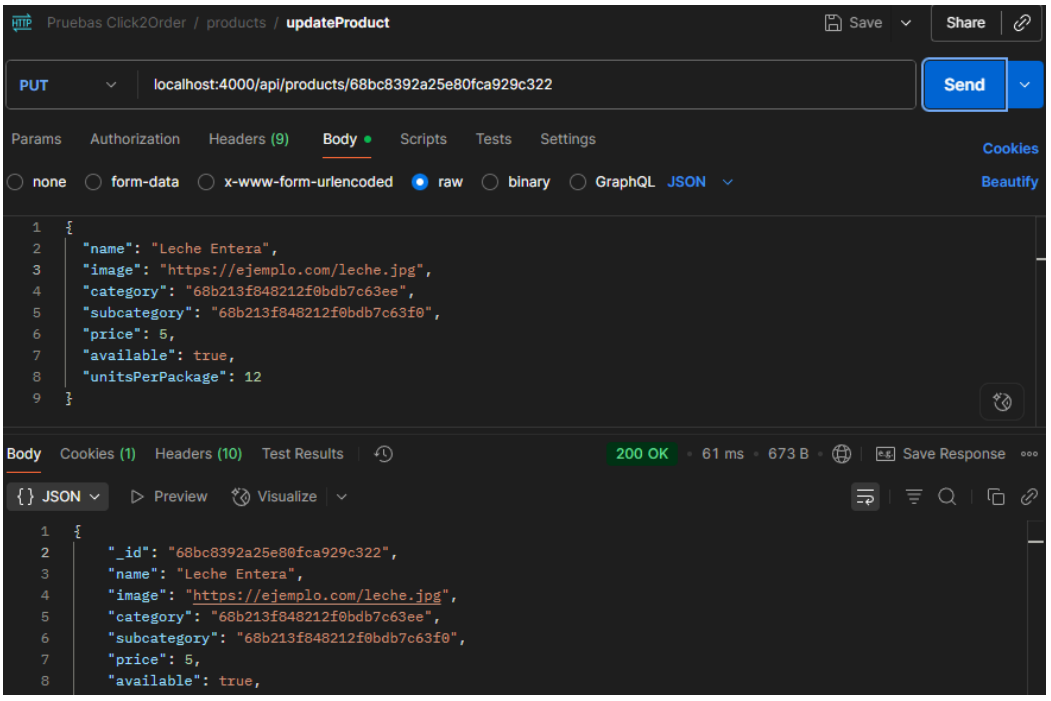

Figura 74. Petición PUT /api/product

6.1.3. Pruebas de carrito

➢ Actualizar un producto del carrito

La Figura 75 muestra una prueba PUT para modificar un producto del carrito con cantidad inválida (0). El sistema devuelve código 400 con “Error de validación” indicando que “La cantidad debe ser al menos 1”, validando correctamente las reglas de negocio del carrito. En cambio, en la Figura 76 muestra la actualización del producto realizada correctamente.

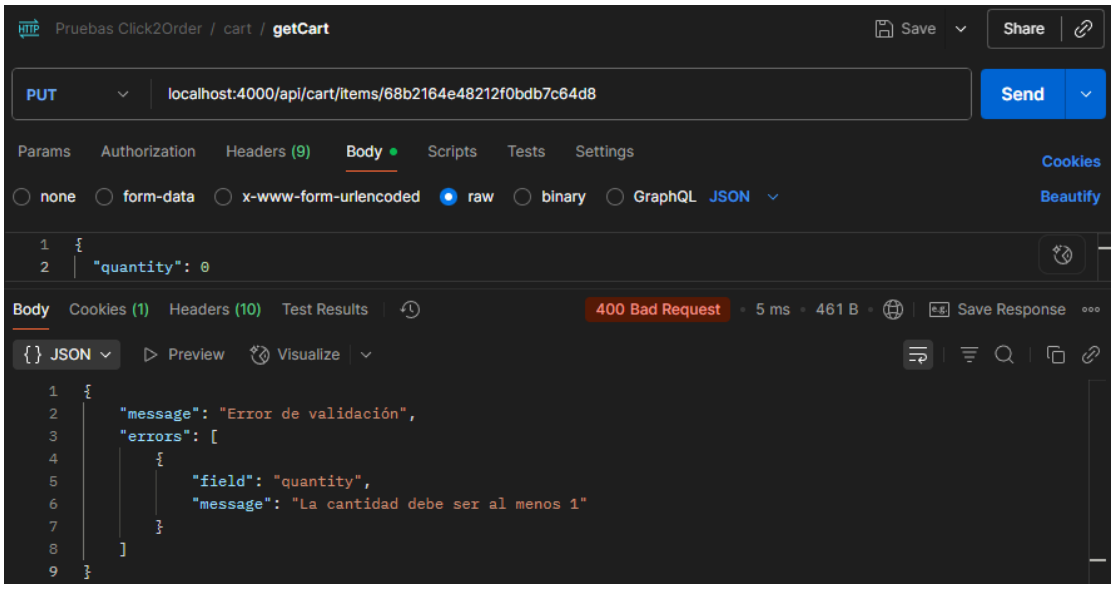

Figura 75. Petición PUT /api/cart/items/:id inválida

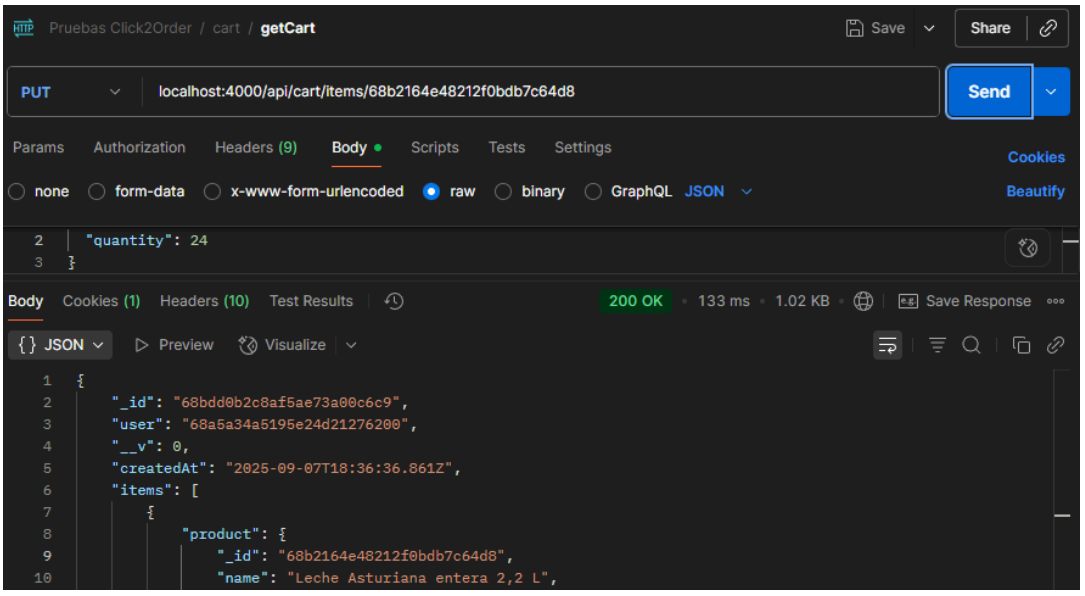

Figura 76. Petición PUT /api/cart/items/:id válida

➢ **Llenar carrito con pedido anterior**

Para rellenar el carrito con los productos de un pedido anterior se ha realizado una prueba POST usando el ID del pedido que se quiera cargar, como se puede observar en la Figura 77, el sistema devuelve código 200 confirmando que los productos del pedido anterior se han añadido exitosamente al carrito, mostrando los items con sus detalles.

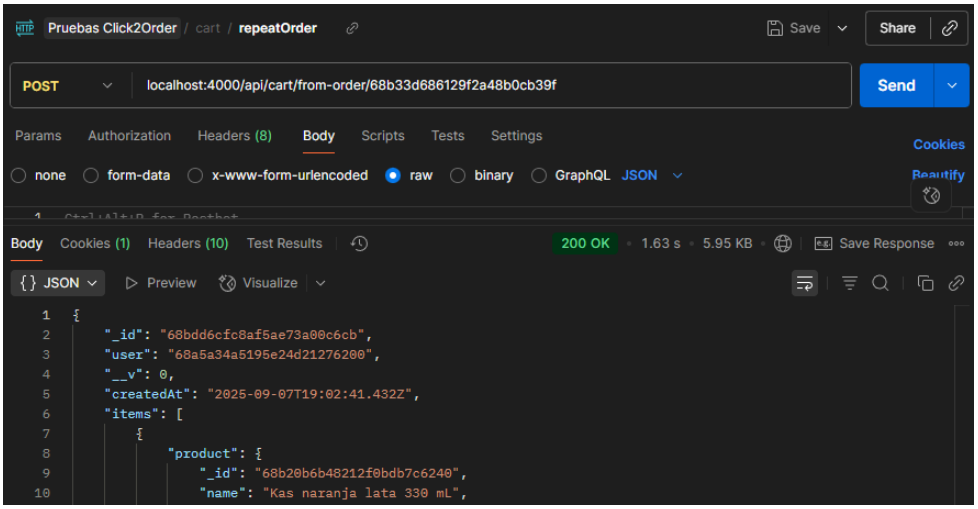

Figura 77. Petición POST api/cart/from-order/:orderId

6.1.4. Pruebas de pedidos

➢ **Consultar pedido concreto**

La Figura 78 muestra una prueba GET para consultar un pedido específico mediante su ID. El sistema devuelve código 200 con la información completa del pedido, validando que la consulta individual de pedidos funciona correctamente.

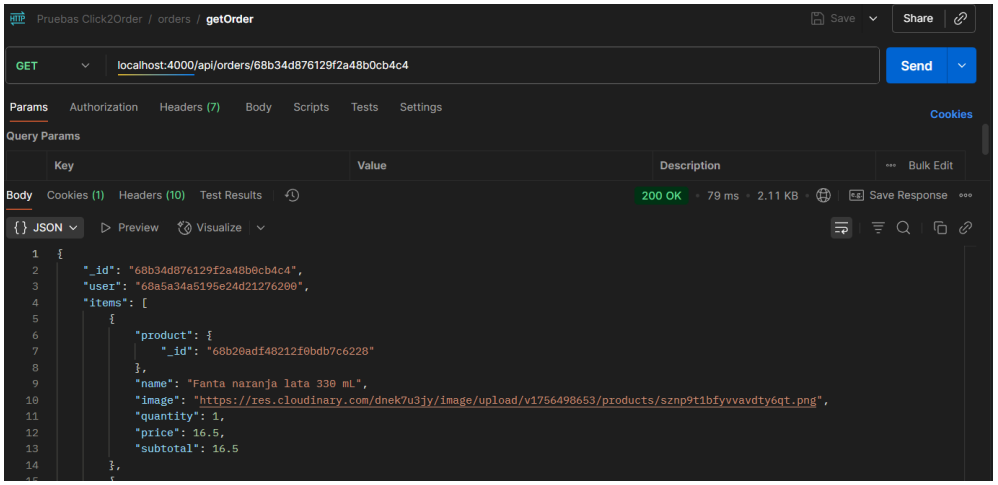

Figura 78. Petición GET api/orders/:Id

➢ Cambiar el estado de pedido

La Figura 79 muestra una prueba PUT fallida para cambiar el estado de un pedido enviando un valor inválido (123). El sistema devuelve código 400 con “Error de validación” indicando “Estado inválido”, confirmando que solo acepta estados predefinidos del sistema.

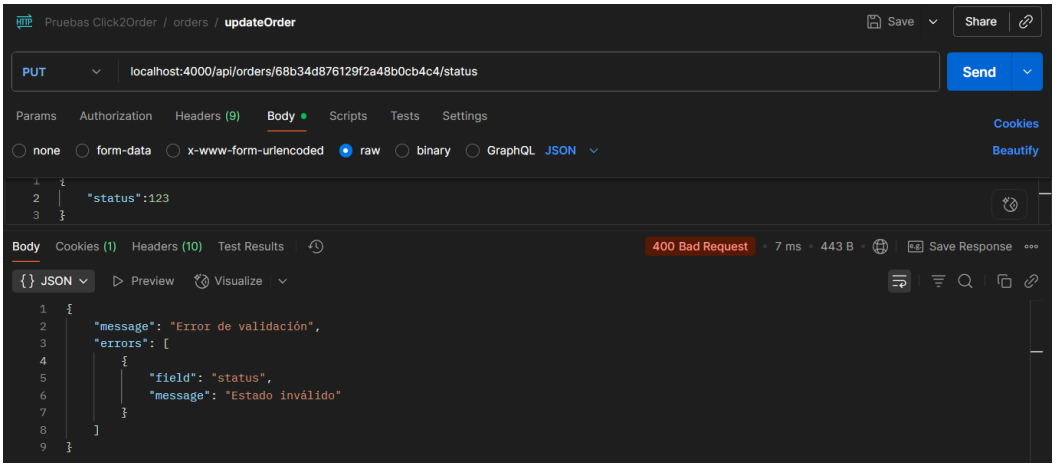

Figura 79. Petición PUT /api/orders/:id/status inválida

En contraste, la Figura 80 demuestra la actualización exitosa del estado enviando “pendiente” como valor válido. El sistema devuelve código 200 junto con el pedido actualizado, mostrando el nuevo estado aplicado correctamente.

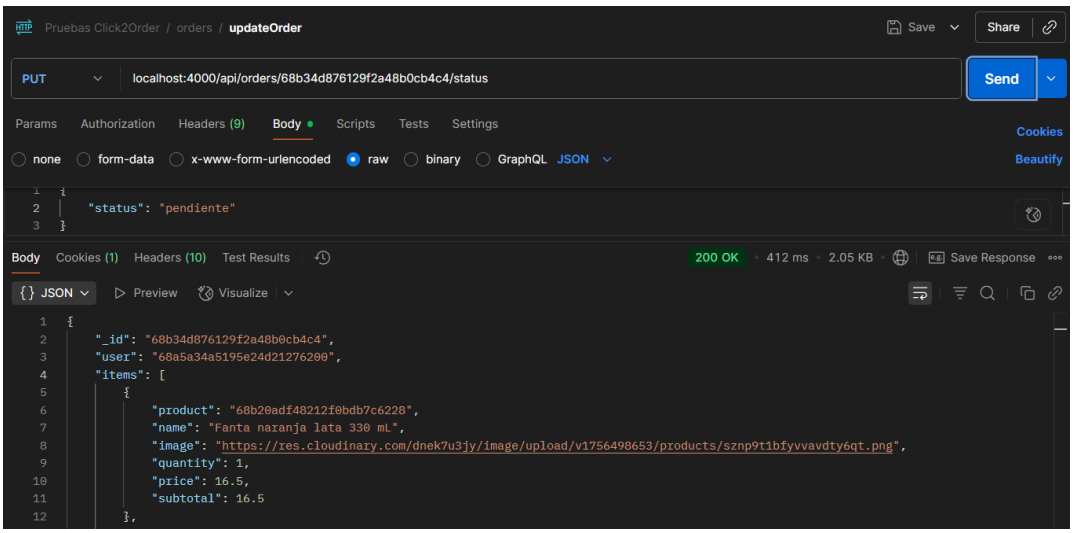

Figura 80. Petición PUT /api/orders/:id/status válida

7

Conclusiones y líneas futuras

Este capítulo final recoge las reflexiones y aprendizajes obtenidos durante el desarrollo del proyecto, así como las posibles ampliaciones y mejoras que podrían implementarse en versiones futuras.

7.1. Conclusiones

El desarrollo de Click2Order ha alcanzado satisfactoriamente la mayoría de los objetivos planteados inicialmente, logrando transformar un proceso de gestión de pedidos manual y desestructurado en una solución digital eficiente y escalable.

7.1.1. Desafíos y curva de aprendizaje

La implementación de esta plataforma web ha supuesto un reto técnico considerable, siendo el primer proyecto de estas características desarrollado de forma completamente autónoma. Al no contar con experiencia previa en proyectos de esta envergadura, fue necesario un proceso intensivo de investigación y autoaprendizaje de tecnologías que no habían sido utilizadas anteriormente en profundidad.

El dominio de React Context para la gestión de estado global, la configuración de una API REST robusta con Node.js y Express, y la integración de servicios externos como Mapbox y Cloudinary, requirió consultar documentación técnica extensa, tutoriales especializados y resolver problemas específicos que no estaban

cubiertos en la formación académica tradicional. La implementación del sistema de autenticación con JWT, la configuración de MongoDB Atlas, y la integración de algoritmos de optimización de rutas.

7.1.2. Resultados obtenidos

Click2Order ha logrado implementar exitosamente las funcionalidades establecidas: catálogo estructurado con filtros y búsquedas, gestión de carrito, historial de pedidos, repetición de pedidos anteriores, gestión de productos por el administrador, notificaciones por correo electrónico, descarga de pedidos en PDF, administración de clientes y cálculo básico de rutas optimizadas.

Sin embargo, algunas funcionalidades propuestas inicialmente no se desarrollaron completamente. El control de stock visible para el administrador se omitió intencionalmente, ya que Click2Order funciona como digitalización del proceso manual existente donde los clientes pueden solicitar cualquier cantidad. La exportación a formatos específicos de facturación no se implementó debido a la diversidad de sistemas propietarios que requeriría un desarrollo personalizado para cada plataforma. El cálculo dinámico de rutas mediante la prioridad de cada pedido se simplificó al cálculo básico debido a la complejidad algorítmica de considerar múltiples variables simultáneamente. La gestión de usuarios se limitó a eliminación directa en lugar de habilitación/deshabilitación, más apropiada para el contexto de empresa familiar.

7.1.3. Validación y efectividad de la solución

La experiencia directa con la problemática de la empresa familiar ha permitido validar de primera mano que Click2Order resuelve efectivamente los problemas más críticos del sector. Los errores de interpretación en pedidos telefónicos como “la leche que suele pedir” o “10 aguas sin especificar marca” se eliminan completamente mediante el catálogo estructurado. La pérdida de mensajes de WhatsApp y la desorganización de pedidos se solucionan con la centralización digital. La funcionalidad de repetir pedidos anteriores agiliza significativamente el proceso para establecimientos con patrones regulares de consumo.

El cálculo automático de rutas optimizadas transforma una tarea manual que podía requerir horas de planificación en un proceso automático de segundos, mejorando sustancialmente la eficiencia logística. La notificación automática por correo electrónico garantiza que ningún pedido pase desapercibido, problema frecuente con el sistema manual anterior.

La solución desarrollada cumple con el propósito principal de estructurar y agilizar la gestión de pedidos, demostrando ser una alternativa viable que mantiene la flexibilidad operativa mientras elimina las principales fuentes de error del proceso tradicional.

7.2. Líneas futuras

Aunque Click2Order cumple con las funcionalidades esenciales, existen oportunidades de mejora para futuras versiones.

La plataforma podría expandirse hacia un sistema multiusuario que soporte múltiples administradores, supervisores y repartidores, permitiendo gestionar operaciones de mayor escala. Un sistema de control de inventario con alertas automáticas cuando los productos alcancen niveles mínimos facilitaría la planificación de compras del administrador.

La integración con software de facturación existente eliminaría la duplicación de datos entre plataformas, mientras que el procesamiento automático de audios e imágenes permitiría convertir mensajes de voz o listas manuscritas en pedidos estructurados, manteniendo la comodidad de los métodos tradicionales.

Se podrían incorporar algoritmos de optimización más avanzados que consideren horarios de entrega preferidos, capacidad de vehículos y planificación para varios días. Un módulo de análisis con estadísticas de ventas, patrones de consumo y eficiencia de rutas proporcionaría información valiosa para el administrador.

Estas mejoras transformarían Click2Order en una plataforma integral de gestión comercial, manteniendo los principios de simplicidad y eficiencia de la propuesta actual.

Referencias

[1] Atlassian, «Kanban - A brief introduction», Atlassian. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.atlassian.com/agile/kanban

[2] «¿Qué es JavaScript? - Aprende desarrollo web | MDN», MDN Web Docs. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn_web_development/Core/Scripting/What_is_JavaScript

[3] «React». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://es.react.dev/

[4] «Vite», vitejs. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://vite.dev

[5] «Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.» Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://tailwindcss.com/

[6] «Web Accessibility Color Contrast Checker - Conform to WCAG». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://accessibleweb.com/color-contrast-checker/

[7] «Context – React». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://legacy.reactjs.org/docs/context.html

[8] «Node.js — About Node.js®». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://nodejs.org/en/about

[9] «Express - Node.js web application framework». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://expressjs.com/

[10] «Intro», Zod. Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://zod.dev/

[11] «Nodemailer | Nodemailer». Accedido: 26 de agosto de 2025. [En línea]. Disponible en: https://nodemailer.com/

[12] «Introduction to MongoDB - Database Manual - MongoDB Docs». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.mongodb.com/docs/manual/introduction/

[13] «What is MongoDB Atlas? - Atlas - MongoDB Docs». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.mongodb.com/docs/atlas/

[14] «MongoDB Compass», MongoDB. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.mongodb.com/products/tools/compass

[15] «Visual Studio Code - Code Editing. Redefined». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://code.visualstudio.com/

[16] «Postman: The World's Leading API Platform | Sign Up for Free». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.postman.com/

[17] «Image and Video Upload, Storage, Optimization and CDN», Cloudinary. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://cloudinary.com/

[18] «Mapbox | Maps, Navigation, Search, and Data». Accedido: 31 de agosto de 2025. [En línea]. Disponible en: https://www.mapbox.com

[19] «GitHub.com Documentación de ayuda», GitHub Docs. Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://docs-internal.github.com/es

[20] «Captura, organiza y aborda tus tareas pendientes en cualquier lugar | Trello». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://trello.com/es

[21] «Google Meet». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://meet.google.com/landing

[22] «Documentos de Google». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://docs.google.com/document/u/0/

[23] «Goodnotes | Notes Reimagined | Note-Taking App». Accedido: 24 de agosto de 2025. [En línea]. Disponible en: https://www.goodnotes.com

[24] «Geocoding API | API Docs», Mapbox. Accedido: 6 de septiembre de 2025. [En línea]. Disponible en: https://docs.mapbox.com//api/search/geocoding

[25] «Optimization API v1 | API Docs», Mapbox. Accedido: 6 de septiembre de 2025. [En línea]. Disponible en: https://docs.mapbox.com//api/navigation/optimization-v1

Anexo A

Manual de usuario

Este manual de usuario proporciona una guía completa para utilizar Click2Order, explicando todas las funcionalidades disponibles tanto para el rol de cliente como el rol de administrador.

Al ejecutar la aplicación, el usuario ve la página de acceso a la plataforma como se puede observar en la Figura 81, donde rellena sus datos (CIF y contraseña) y selecciona la opción de “Entrar”.

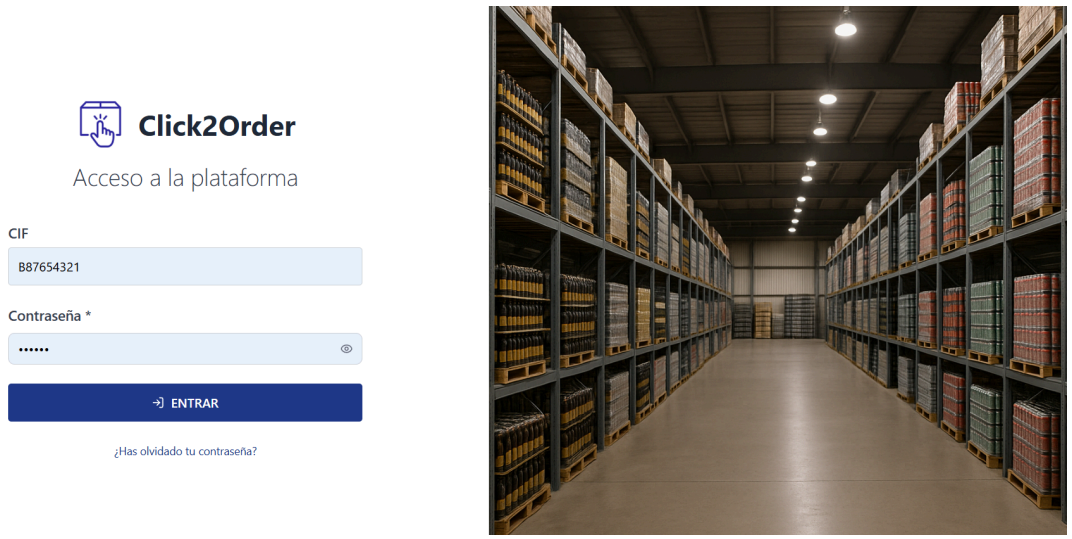

Figura 81. Pantalla de acceso a la plataforma

En el caso de que el usuario olvide su contraseña, selecciona la opción de “¿Has olvidado tu contraseña?”, donde se muestra un modal para rellenar con su CIF y Email, como se puede observar en la Figura 82, y donde posteriormente selecciona la opción de “Enviar” para recibir una contraseña temporal por email.

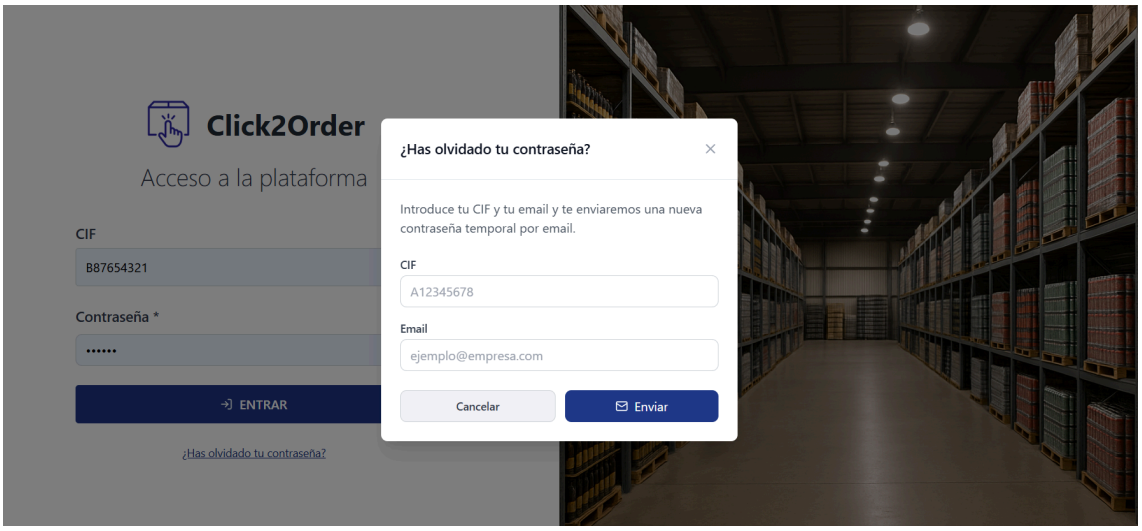

Figura 82. Pantalla de olvido de contraseña

Al acceder a la plataforma, en la página de “Perfil” desde la barra de navegación se mostrará el perfil del usuario autenticado, donde podrá consultar y modificar sus datos personales como se muestra en la Figura 83.

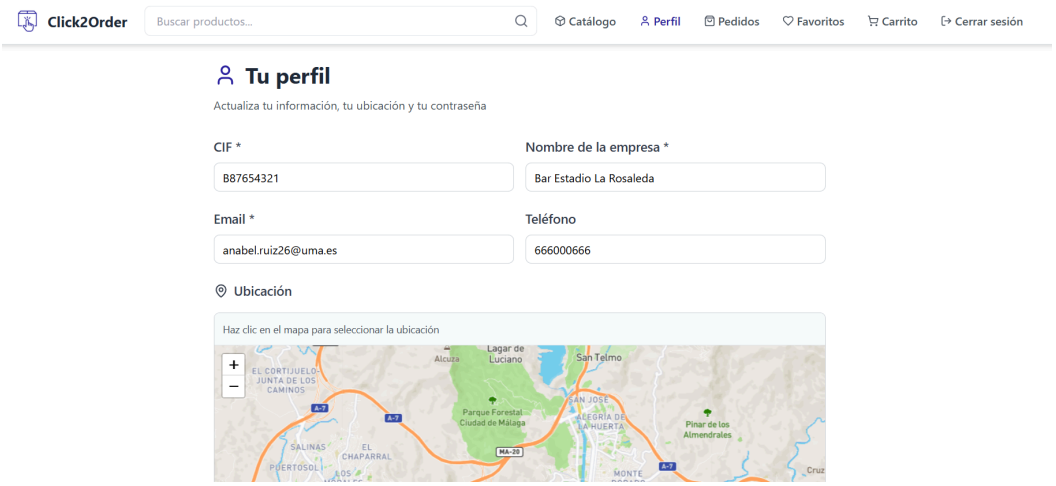

Figura 83. Pantalla de perfil

El usuario puede cambiar la ubicación en la que se encuentra buscando y clicando en el mapa la ubicación exacta, la cual se muestra en el apartado de “Dirección” automáticamente. Una vez se hayan modificado los datos, selecciona la opción de “Guardar cambios” o “Cancelar”, como se observa en la Figura 84.

Para cambiar la contraseña, el usuario selecciona la opción de “Mostrar” en el apartado de “Cambiar contraseña”, donde se muestra un formulario para introducir la antigua y la nueva contraseña. Una vez se hayan cumplimentado los cambios se selecciona la opción de “Cambiar contraseña”.

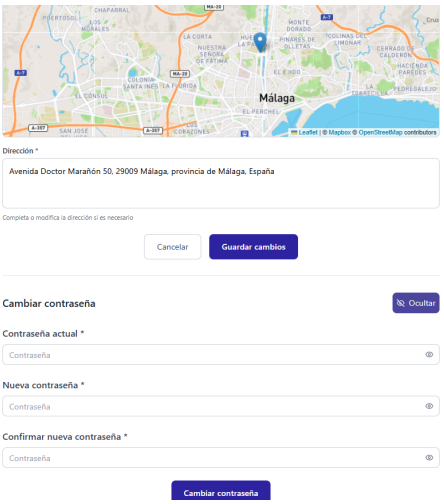

Figura 84. Pantalla de perfil para confirmar cambios

En función del rol del usuario, se mostrarán unas pantallas u otras, las cuales se van a explicar a continuación.

A.1. Guía de uso para el rol de cliente

Una vez que el cliente accede a la plataforma, se encuentra directamente en el apartado de “Catálogo” como se puede ver en la Figura 85, donde se pueden ver las distintas categorías de productos existentes y acceder a la demás funcionalidades del sistema. También podrá seleccionar la opción de “Cerrar sesión” para volver a la pantalla de acceso a la plataforma anterior.

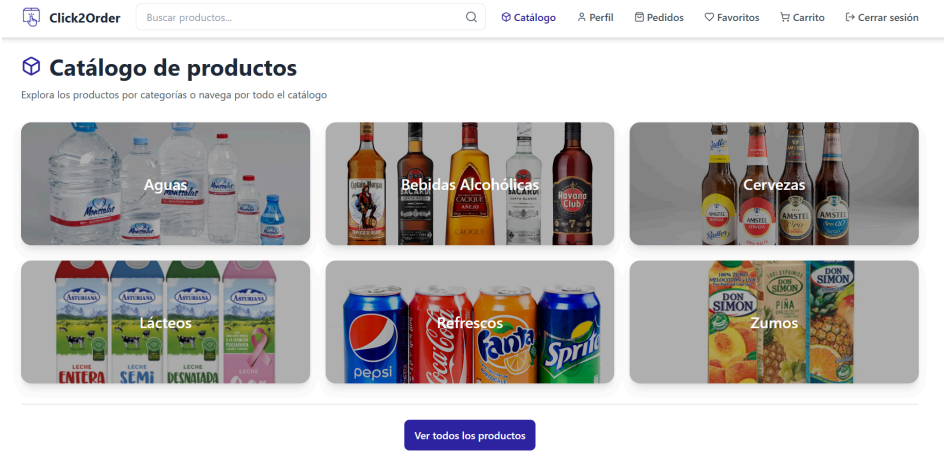

Figura 85. Pantalla de catálogo de productos

Si se selecciona la opción de “Ver todos los productos”, se accede a una de las categorías del catálogo o se realiza una búsqueda de productos desde la barra de navegación, aparece la lista de productos específica, los cuales se pueden

ordenar y filtrar, como se puede observar en la barra lateral izquierda de la Figura 86. Además, en esa pantalla se puede agregar un producto a favoritos o añadir y cambiar las cantidades de un producto en el carrito.

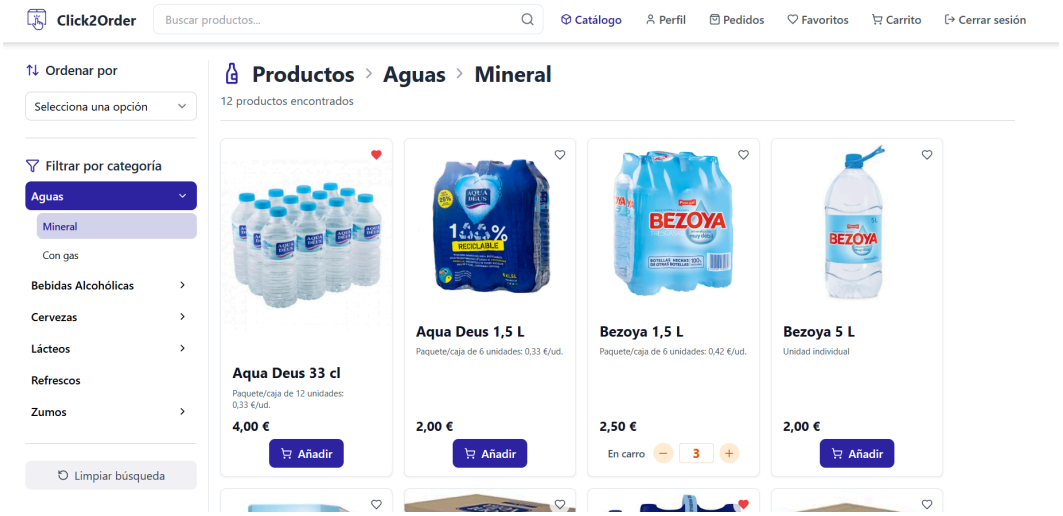

Figura 86. Pantalla de lista de productos

El cliente puede ver los detalles de un producto concreto haciendo clic sobre él y se mostrará una pantalla como se ve en la Figura 87. En esa pantalla puede observar la información concreta de ese producto, puede añadirlo en la lista de favoritos seleccionando la opción de “Marcar como favorito” o también puede añadirlo al carrito seleccionado la opción de “Añadir al carrito”, además de poder cambiar la cantidad que desea añadir.

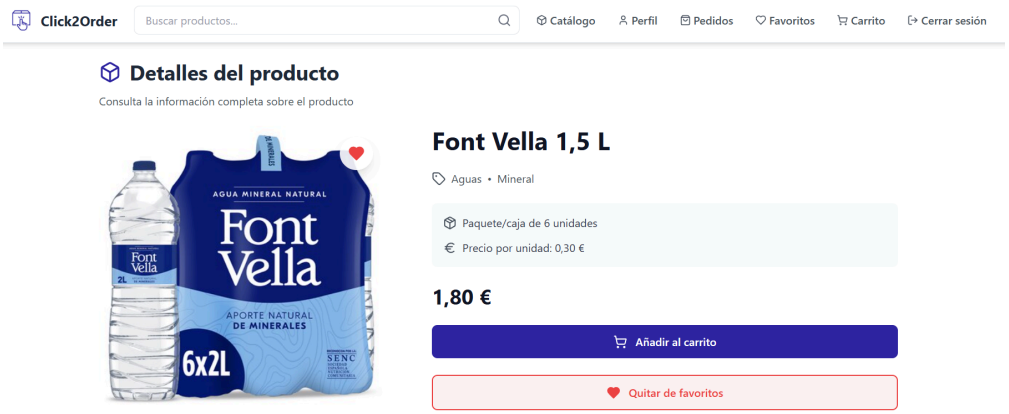

Figura 87. Pantalla de detalles del producto

En la página de “Pedidos” el cliente puede observar una lista con todos sus pedidos realizados, véase la Figura 88, donde los pedidos se pueden ordenar o filtrar según se desee. Además el cliente puede añadir un pedido de la lista en el

carrito para repetirlo, seleccionando la opción de “Añadir al carrito” o cancelar un pedido que aún esté en estado “Pendiente” seleccionando la opción de “Cancelar pedido”.

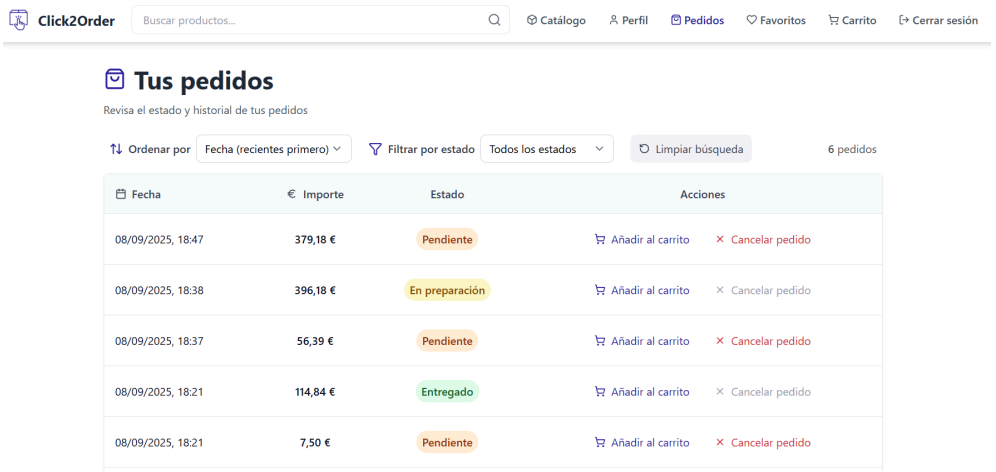

Figura 88. Pantalla de pedidos

Seleccionando un pedido concreto, se accede a la página de “Detalles del pedido” como se observa en la Figura 89, donde se puede consultar toda la información relevante además de añadir los productos de nuevo en el carrito seleccionando la opción de “Añadir al carrito” y cancelar el pedido si lo permite.

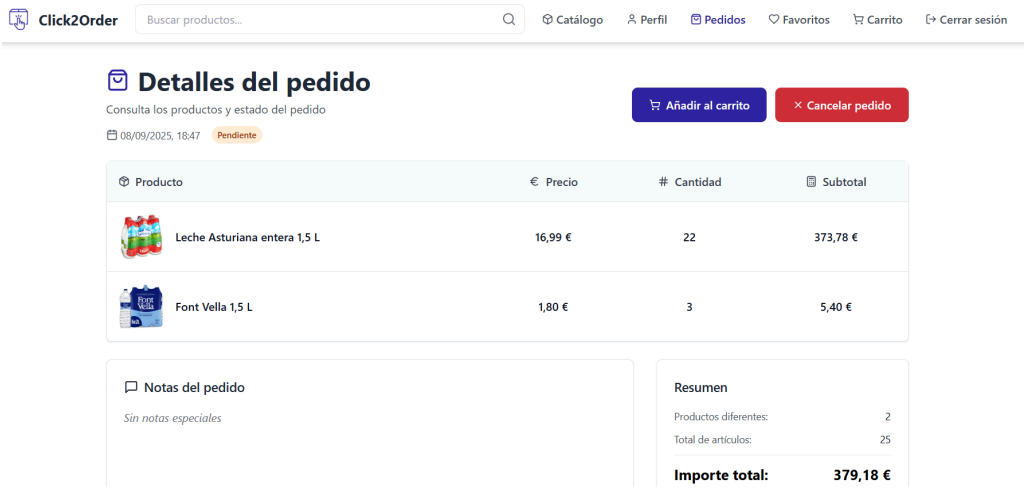

Figura 89. Pantalla de detalles del pedido

En la página de “Favoritos”, el cliente puede ver todos los productos que se haya guardado en la lista de favoritos para así poder visualizar rápidamente sus productos más recurrentes, como se observa en la Figura 90.

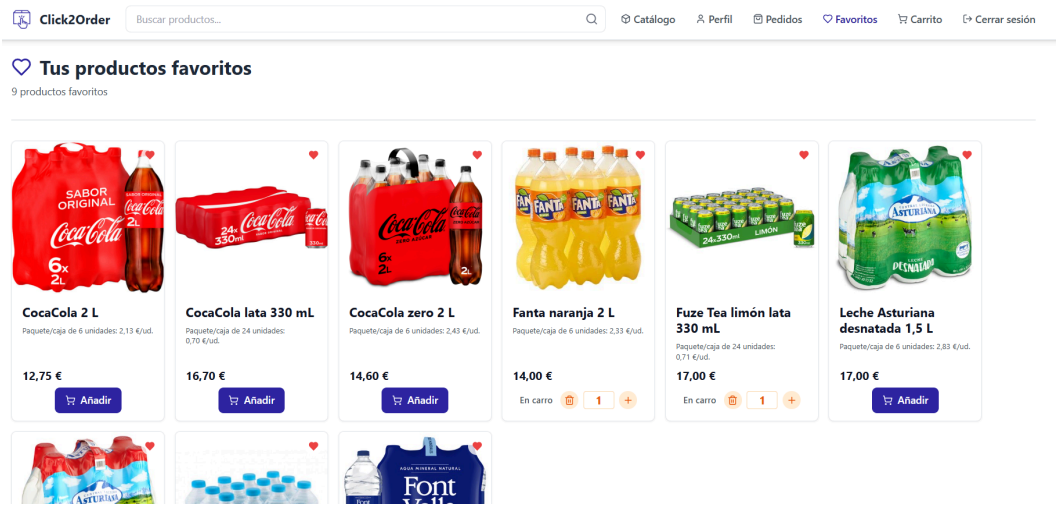

Figura 90. Pantalla de productos favoritos

Al acceder a la página de “Carrito”, se puede revisar y gestionar todos los productos que se hayan añadido, véase Figura 91, donde se podrá modificar las cantidades de cada producto o añadir alguna anotación al pedido.

También podrá realizar el pedido seleccionando la opción de “Realizar pedido” o volver a la lista de productos seleccionando la opción de “Seguir comprando”. Además se pueden eliminar todos los productos del carrito seleccionando la opción de “Vaciar carrito”.

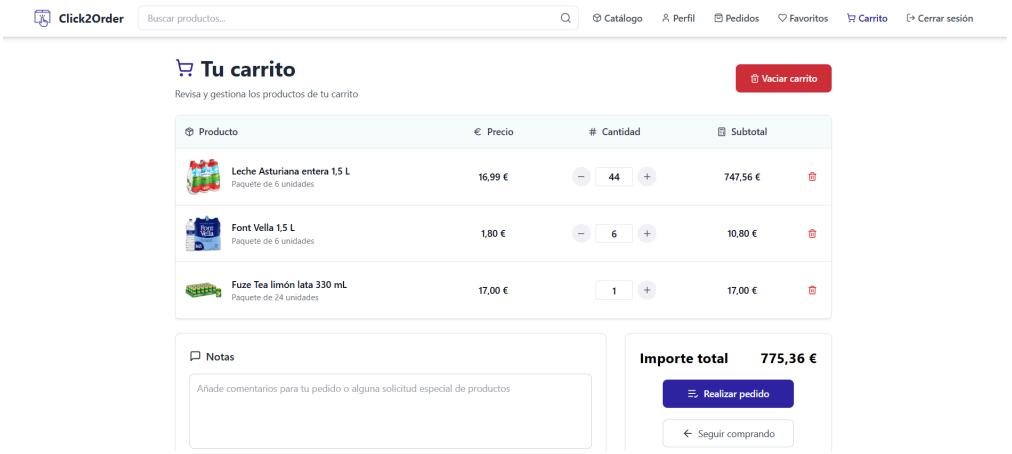

Figura 91. Pantalla de carrito

A.2. Guía de uso para el rol de administrador

Una vez el usuario con rol de administrador accede a la plataforma, se muestra la página inicial del Catálogo de productos con las categorías existentes igual que el resto de usuarios pero con la posibilidad de editar, eliminar y crear productos y categorías, como se muestra en la Figura 92.

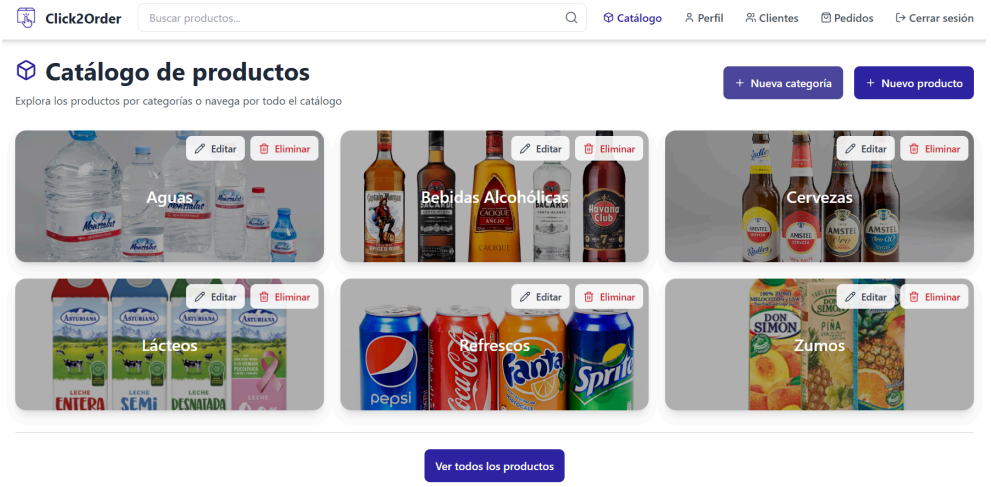

Figura 92. Pantalla de catálogo de productos del administrador

Tanto si edita, como si crea una nueva categoría, se muestra un formulario donde podrá editar la imagen y los campos requeridos de una categoría, y una vez se cumplimenten los campos necesarios se selecciona la opción de “Guardar cambios” o “Cancelar”, véase Figura 93.

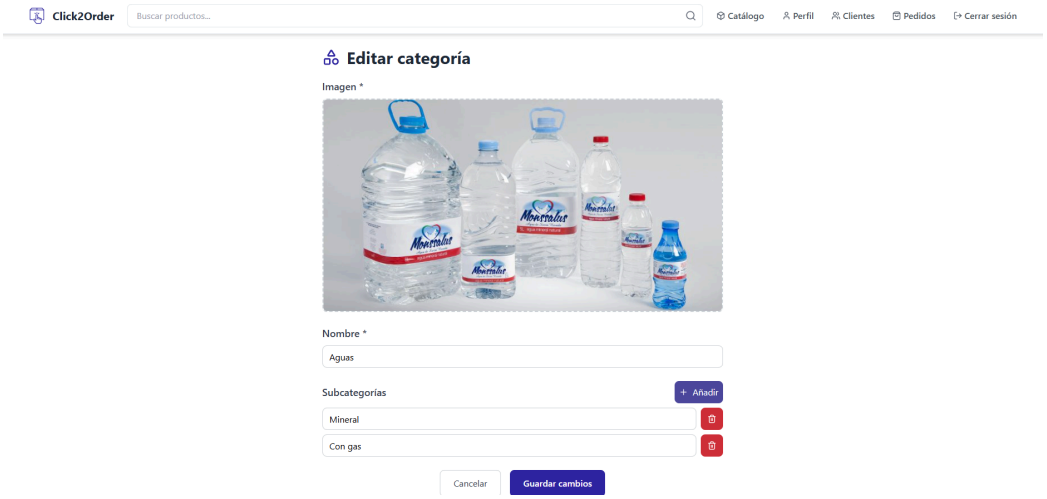

Figura 93. Pantalla de edición de categoría

Si se selecciona la opción de “Ver todos los productos”, se accede a una de las categorías del catálogo o se realiza una búsqueda de productos desde la barra de navegación, aparece la lista de productos específica, los cuales se pueden ordenar y filtrar, como se muestra en la Figura 94.

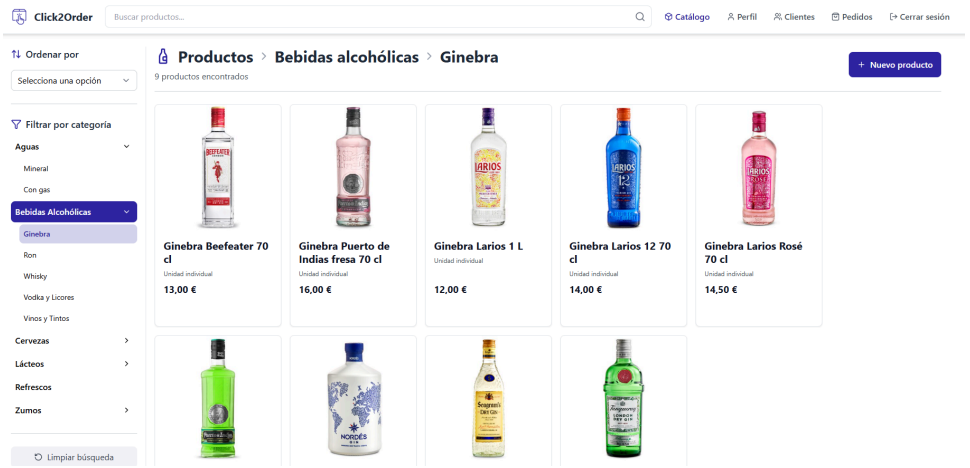

Figura 94. Pantalla de lista de productos del administrador

Si se selecciona la opción de “Nuevo producto”, se muestra un formulario para la creación de uno nuevo, donde se rellenan los campos requeridos y una vez estén completos se selecciona la opción de “Crear producto”, como se ve en la Figura 95.

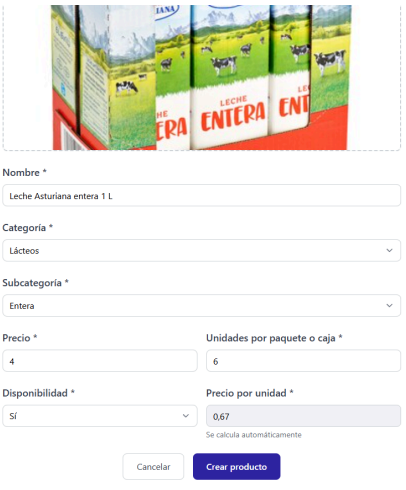

Figura 95. Pantalla de creación de un producto

Si en la lista de productos se selecciona un producto concreto se habilitan las opciones de “Editar producto” y “Eliminar producto” como se observa en la Figura 96.

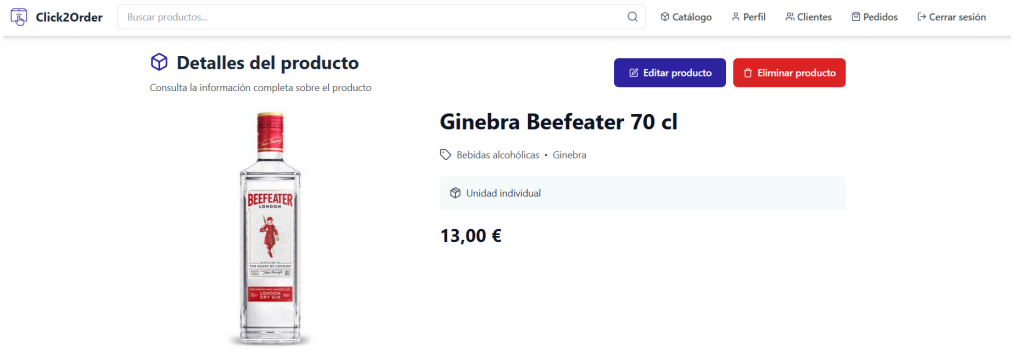

Figura 96. Pantalla de detalles de un producto del administrador

En la página de “Clientes”, el administrador puede ver y gestionar todos los clientes registrados en el sistema, como se puede observar en la Figura 97. Aquí también puede editar la información de un cliente concreto seleccionado la acción de “Editar” en la fila de cada uno, donde se le abrirá el formulario de edición del cliente o la acción de “Eliminar”, donde se le mostrará un mensaje de confirmación para borrar al cliente del sistema.

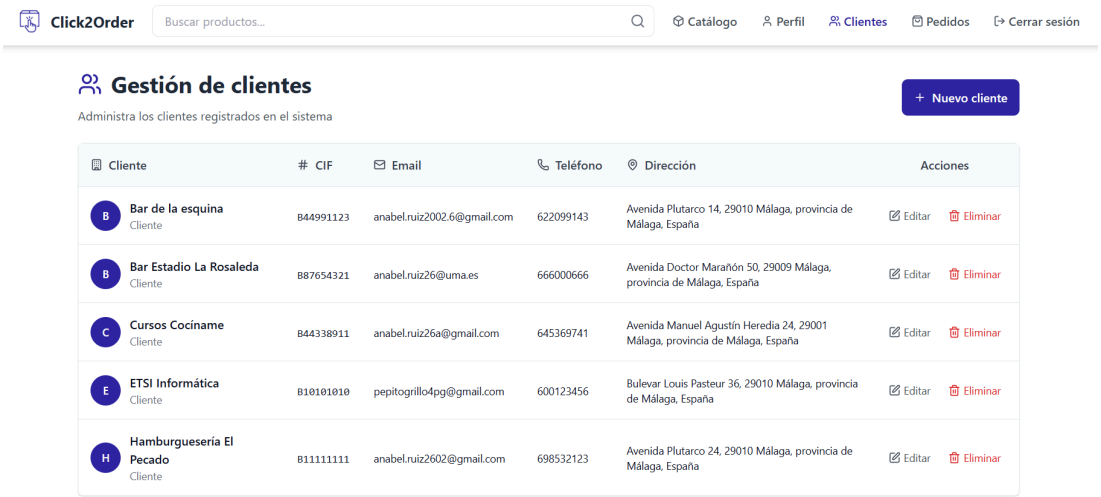

Figura 97. Pantalla de gestión de clientes

Si se selecciona la opción de “Nuevo cliente” se muestra un formulario donde se debe completar todos los campos para crear un nuevo cliente en el sistema como se muestra en la Figura 98.

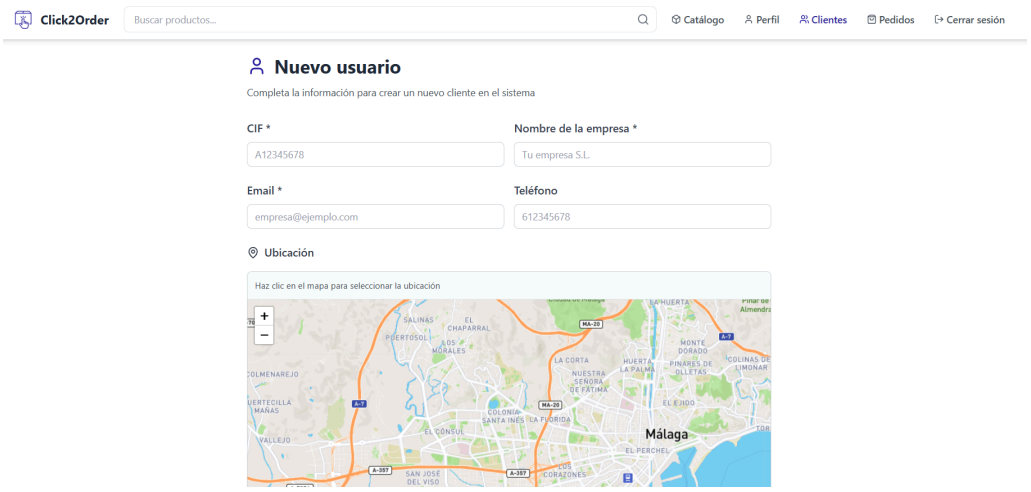

Figura 98. Pantalla de creación de nuevo usuario (I)

Una vez que los datos estén cumplimentados correctamente se selecciona la opción de “Crear usuario” para darlo de alta en el sistema y enviar una contraseña automática temporal al email del usuario, como se muestra en la Figura 99.

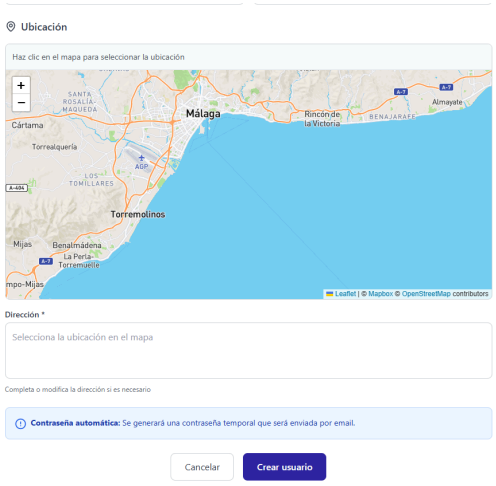

Figura 99. Pantalla de creación de nuevo usuario (II)

Si se selecciona cualquiera de los clientes de la lista, se muestra una página con la información detallada del cliente y el historial de los pedidos realizados de dicho cliente, como se observa en la Figura 100.

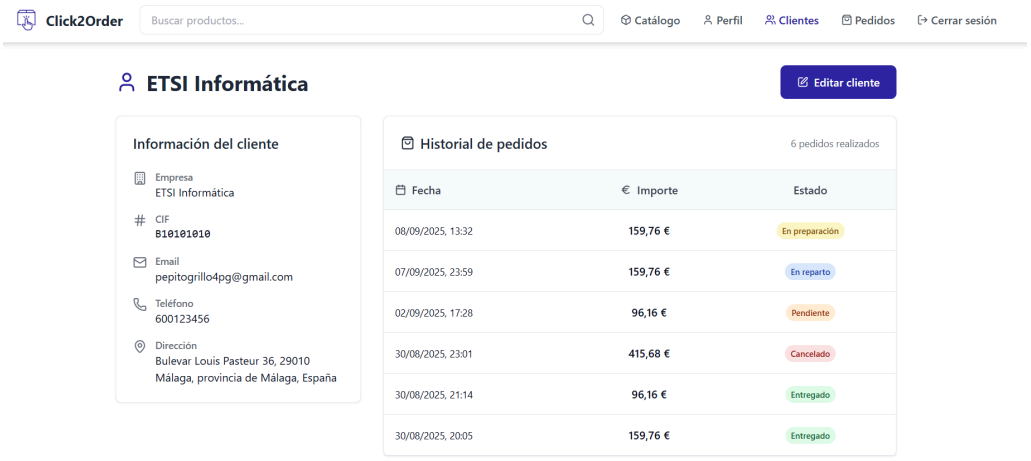

Figura 100. Pantalla de información de cliente

Si se selecciona la opción de “Editar cliente” se muestra el formulario anterior de edición del cliente y si se selecciona en un pedido concreto se muestra una página con los detalles del pedido, véase Figura 101. En esta página se puede acceder a cada producto concreto, se puede cambiar el estado del pedido a uno nuevo o se puede descargar el pedido en PDF para ver la lista de carga.

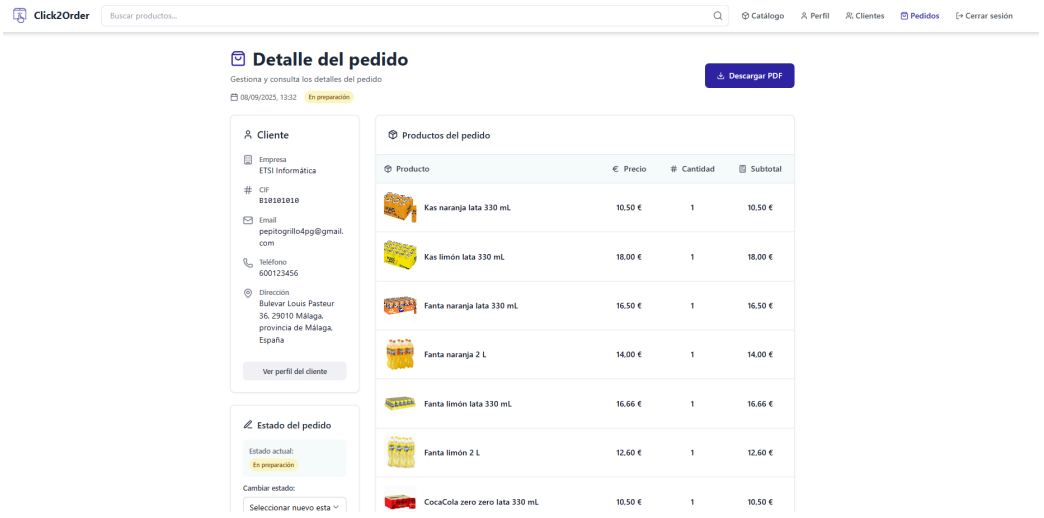

Figura 101. Pantalla de detalle del pedido del administrador

Si se selecciona la opción de “Descargar PDF” se descarga la lista de los productos que se deben cargar para ese cliente para facilitar al administrador la

carga de la mercancía en el vehículo de reparto. Dicho PDF se puede observar en la Figura 102.

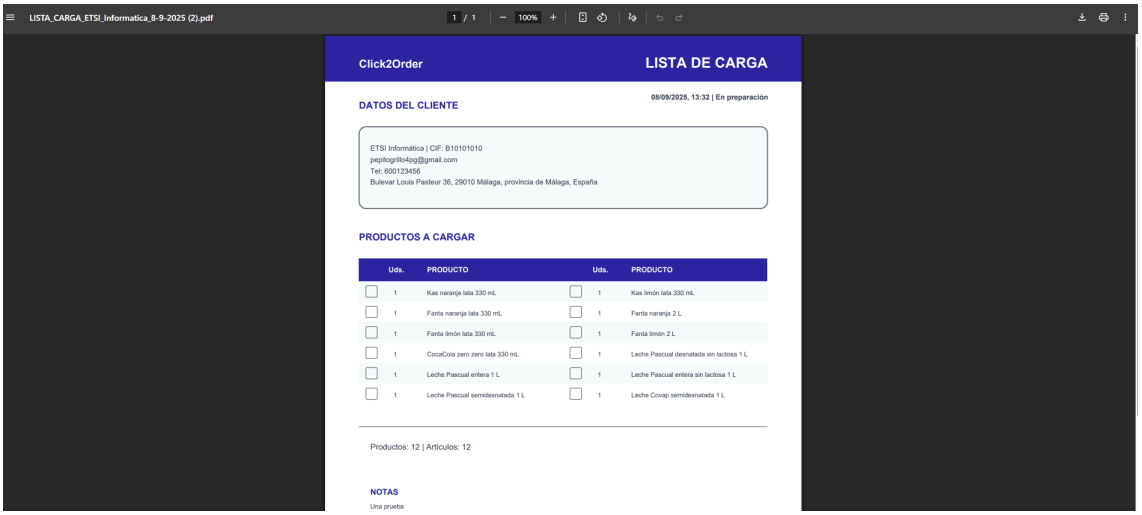

Figura 102. PDF de la lista de carga

Al acceder a la página de “Pedidos”, se mostrará la lista de todos los pedidos de todos los clientes, ordenados por la fecha más reciente por defecto como se muestra en Figura 103.

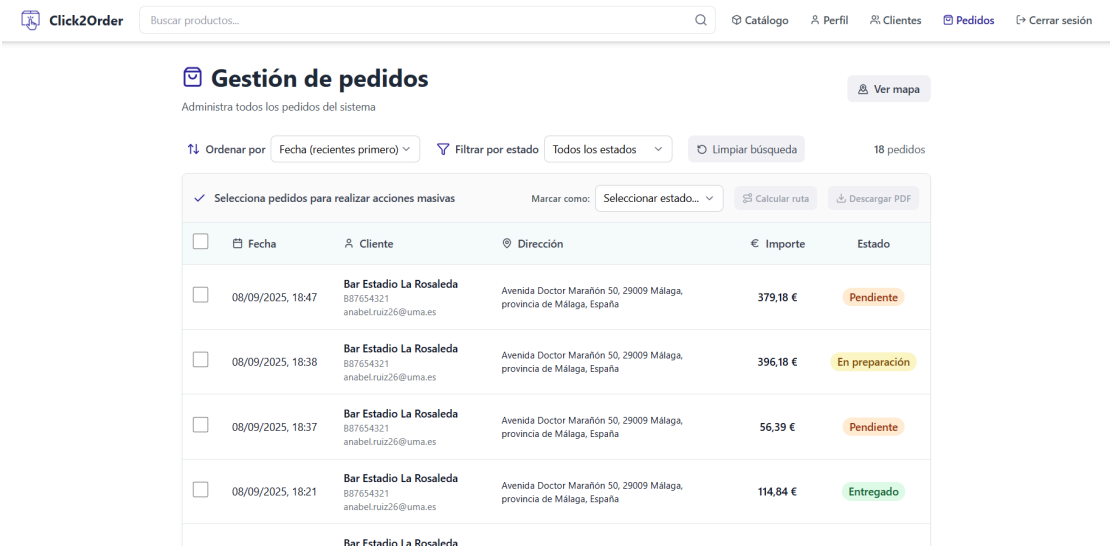

Figura 103. Pantalla de gestión de pedidos (I)

El administrador también puede seleccionar la opción de “Ver mapa” para ver todos los clientes con pedidos realizados en azul y el punto de distribución del administrador en rojo. Además, al seleccionar un cliente en el mapa se puede observar la información del establecimiento y los pedidos de forma compacta como se muestra en la Figura 104.

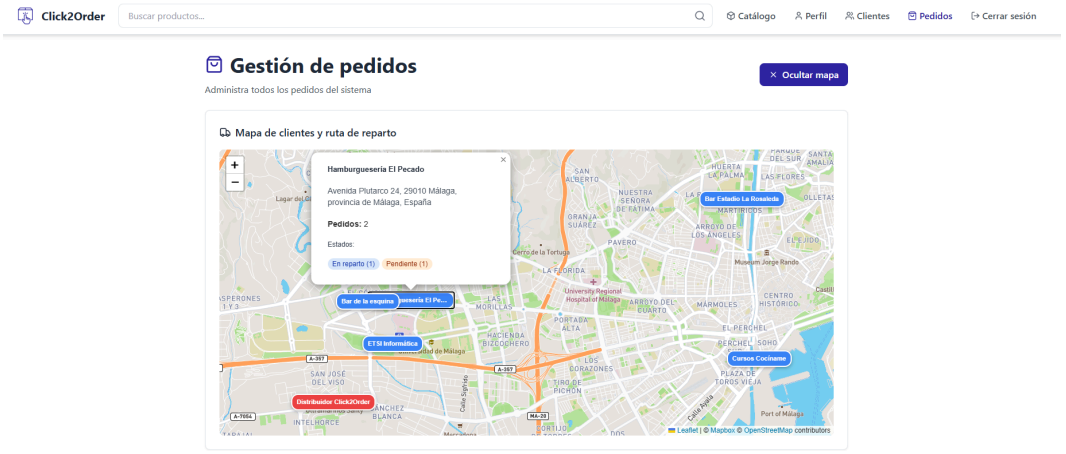

Figura 104. Pantalla de gestión de pedidos (II)

Por otra parte, el administrador también puede ver los detalles de cada pedido, ordenar la lista y filtrar por un estado concreto. Además puede seleccionar varios pedidos para realizar varias acciones masivamente, como cambiarles el estado como se puede observar en la Figura 105 o descargar los pedidos en PDF, los cuales se concatenan en un único PDF en el orden inverso a la ruta óptima calculada, para que se carguen primero los que se van a descargar después.

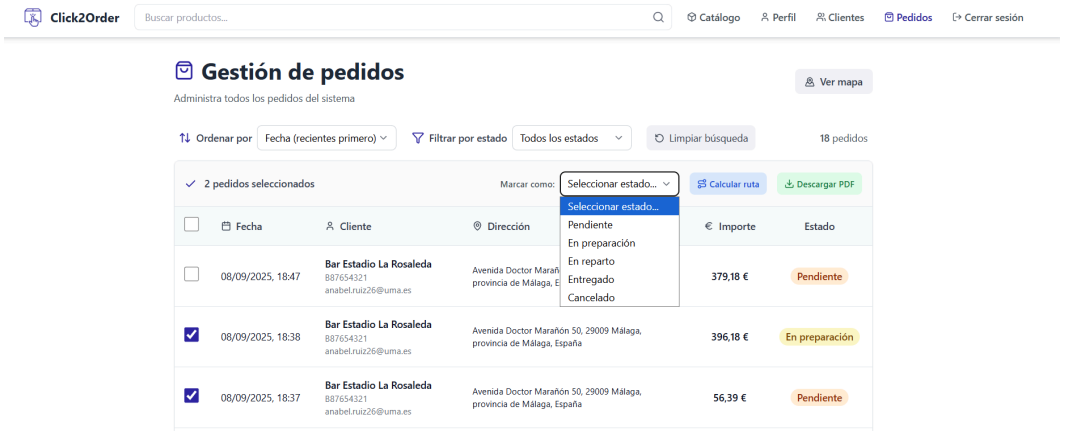

Figura 105. Pantalla de gestión de pedidos (III)

Al seleccionar múltiples pedidos, se calcula la ruta de reparto más eficiente mediante la opción “Calcular ruta”. El sistema genera un mapa interactivo que

muestra la trayectoria optimizada, donde las ubicaciones de los clientes incluidos en la ruta aparecen marcadas en verde y numeradas según el orden de visita recomendado, como se observa en la Figura 106. Bajo el mapa, se muestra un resumen con la distancia total del recorrido, el número de paradas programadas y la secuencia ordenada que debe seguir el repartidor.

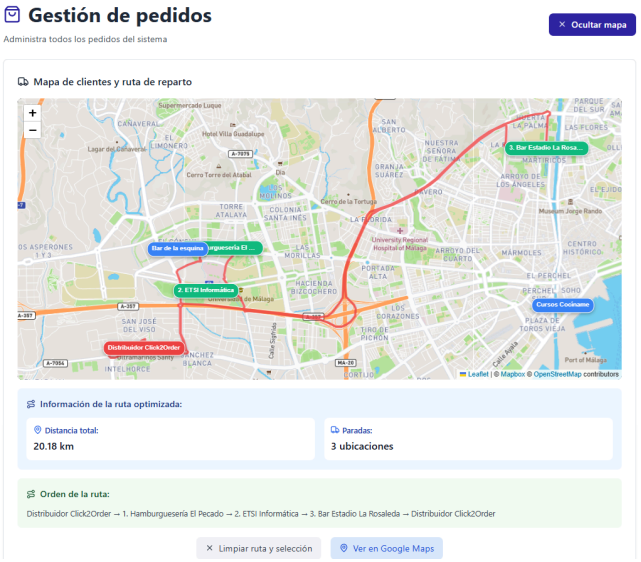

Figura 106. Pantalla de gestión de pedidos (IV)

Al seleccionar la opción de “Ver en Google Maps”, el sistema redirige a la pantalla de Google Maps con la ruta óptima calculada anteriormente y las paradas que debe hacer el repartidor, como se observa en la Figura 107.

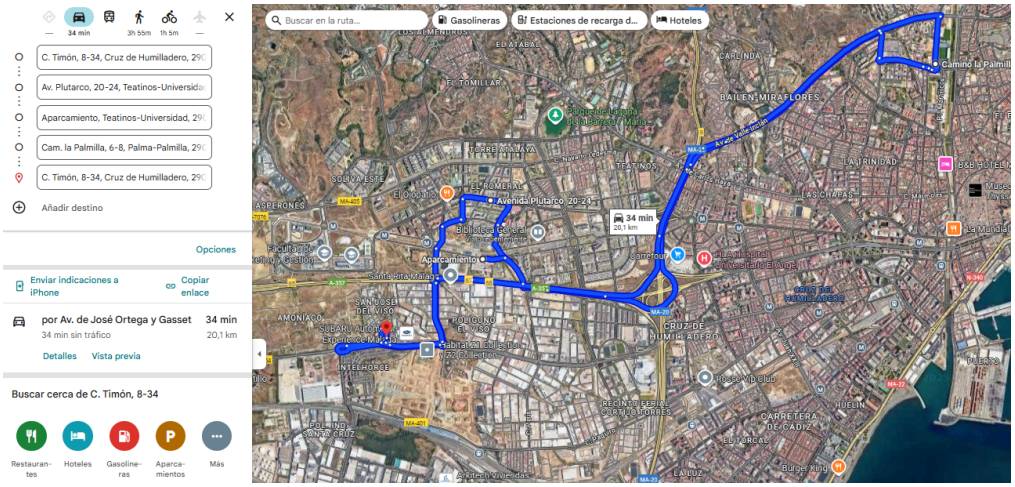

Figura 107. Pantalla de gestión de pedidos (V)

Anexo B

Manual de instalación y despliegue

B.1. Requisitos previos

Para ejecutar el proyecto en un dispositivo de manera local, primero es necesario tener instalado Node.js y MongoDB en el ordenador. Estos son los únicos requisitos técnicos imprescindibles antes de proceder con la configuración del proyecto.

B.2. Configuración del proyecto

Para configurar el proyecto, se debe abrir un entorno de desarrollo como por ejemplo Visual Studio Code y cargar la carpeta del proyecto Click2Order. Una vez abierto el proyecto en el IDE, es necesario configurar dos terminales: una en la raíz del proyecto para el backend y otra en el directorio `/client` para el frontend. En ambas terminales hay que ejecutar `npm install` para instalar los módulos necesarios, como se ve en la Figura 108.

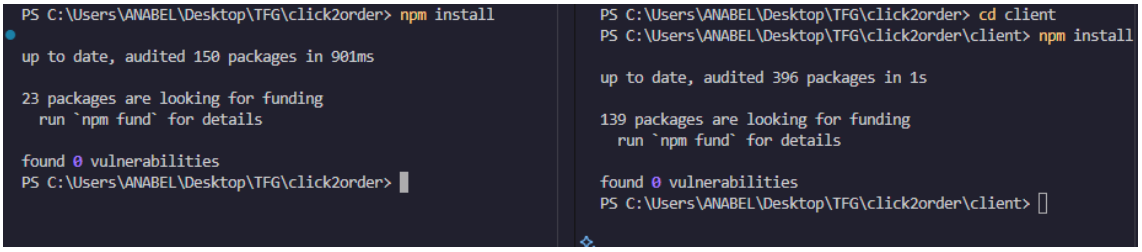

Figura 108. Terminales de VS Code configuradas

Antes de ejecutar ambas partes del proyecto, es necesario configurar las variables de entorno del backend. Estas variables incluyen la URI de la base de datos MongoDB en caso de querer alojarla en la nube, credenciales para la cuenta de Gmail para el envío de emails, y el token de Mapbox para las peticiones a su API. Para ello, se crea un fichero `.env` en la raíz del proyecto y se asignan valores a estas variables, como se muestra en la Figura 109.

```
.env
1  MONGODB_URI= tu_uri_mongodb
2  GMAIL_USER= tu_email@gmail.com
3  GMAIL_PASS= tu_contraseña_aplicacion
4  MAPBOX_ACCESS_TOKEN= tu_token_mapbox
```

Figura 109. Configuración fichero .env

B.3. Obtención de credenciales para .env

Para configurar el envío de emails a través de Nodemailer, se debe acceder a la cuenta de Gmail que se utilizará para enviar los emails y navegar a “Gestionar tu cuenta de Google” en la sección “Seguridad”. Es imprescindible activar la verificación en 2 pasos y se debe acceder a “Contraseñas de aplicación” para generar una nueva contraseña de 16 caracteres como se muestra en la Figura 110, que debe copiarse y utilizarse como valor para la variable GMAIL_PASS.

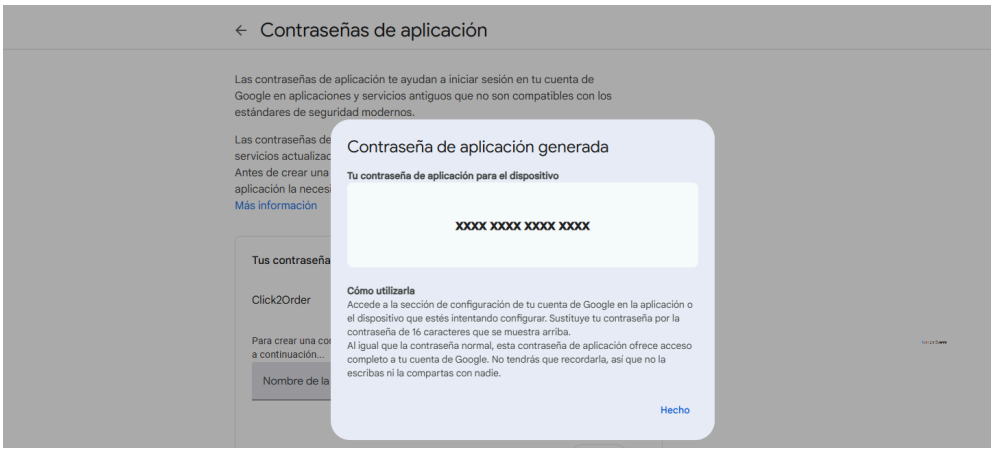

Figura 110. Generación de contraseña de aplicación

Para obtener el token de Mapbox, se debe crear una cuenta en Mapbox y acceder al panel de control o dashboard. En la sección “Access tokens” se puede copiar el token público predeterminado, o alternativamente crear un nuevo token personalizado para la aplicación, como se ve en la Figura 111.

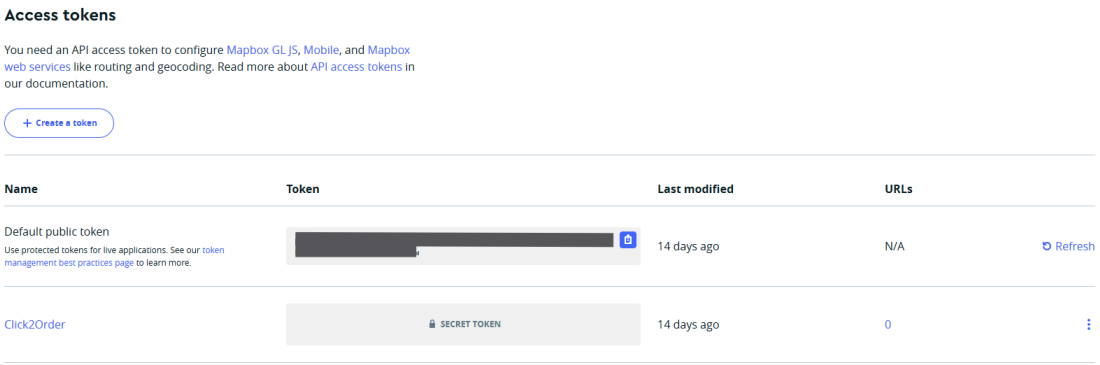

Figura 111. Generación de token de acceso de Mapbox

Para obtener la MongoDB URI, es necesario crear una cuenta en MongoDB Atlas y una vez registrado, crear un nuevo cluster gratuito como se muestra en la Figura 112, seleccionando la opción “Free” que proporciona 512 MB de almacenamiento compartido, suficiente para el desarrollo y pruebas.

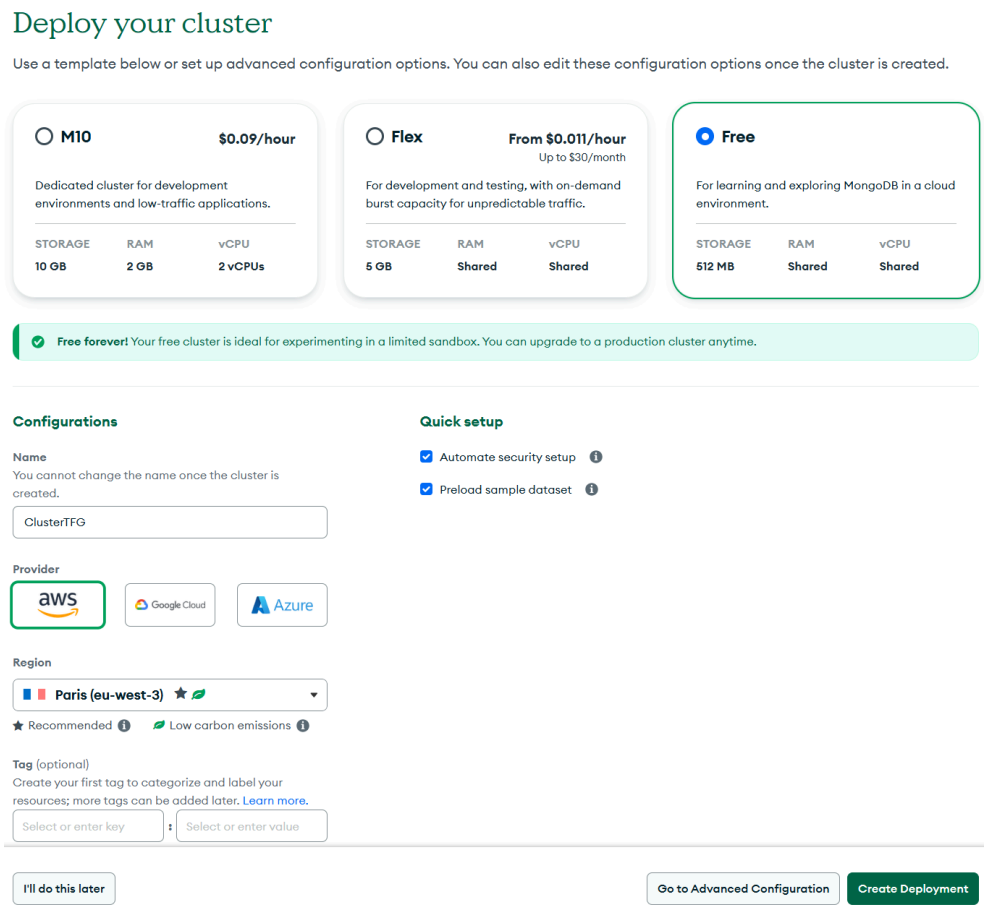

Figura 112. Creación de Cluster en MongoDB Atlas

Después de crear el cluster, hay que configurar un usuario de base de datos como se observa en la Figura 113. Seleccionar “Create Database User” e introducir nombre de usuario y contraseña, es importante guardar estas credenciales ya que se necesitarán para la URI de conexión.

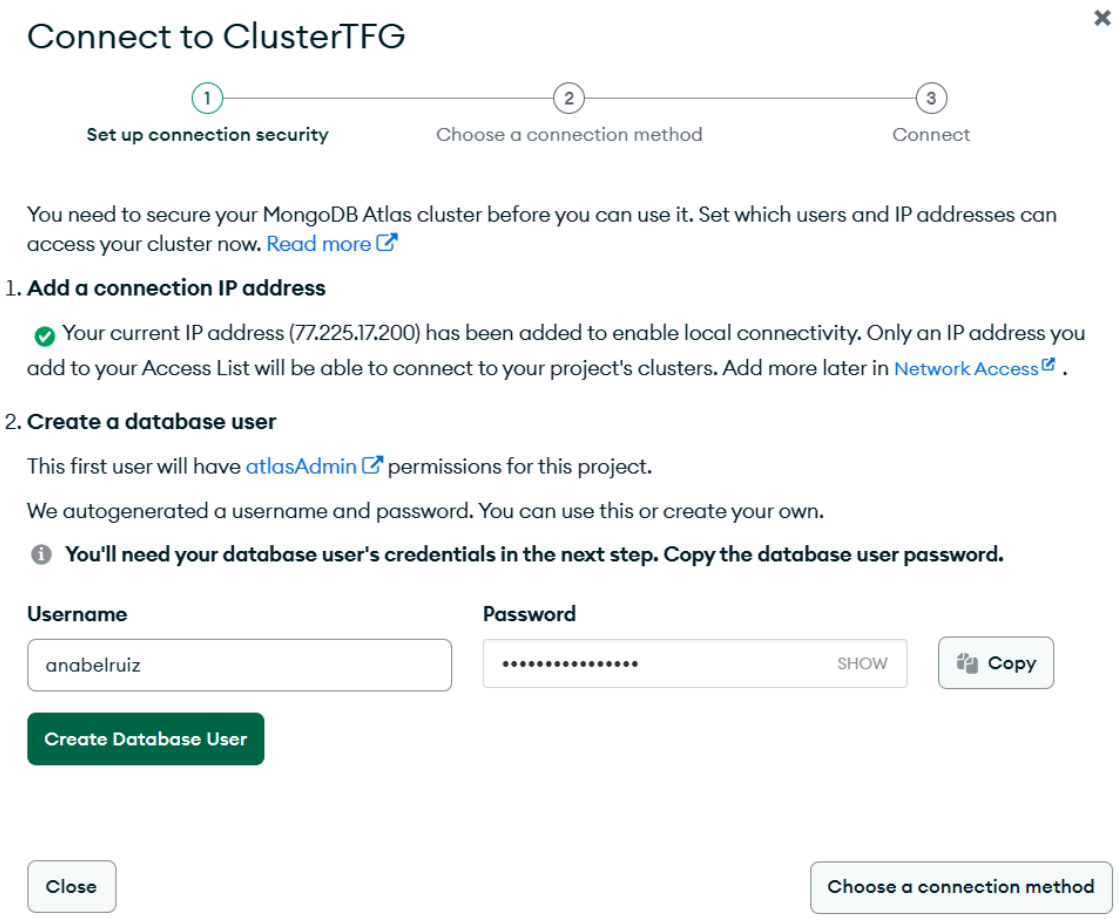

Figura 113. Creación del usuario de la base de datos

Una vez configurado el usuario, es necesario obtener la cadena de conexión desde “Connect” del cluster como se muestra en la Figura 114, Para ello hay que seleccionar “MongoDB Driver” para aplicaciones Node.js y copiar la cadena resultante (`mongodb+srv://usuario:contraseña@cluster.mongodb.net/nombre_base_datos`) para usarla como valor de MONGODB_URI en el archivo de configuración.

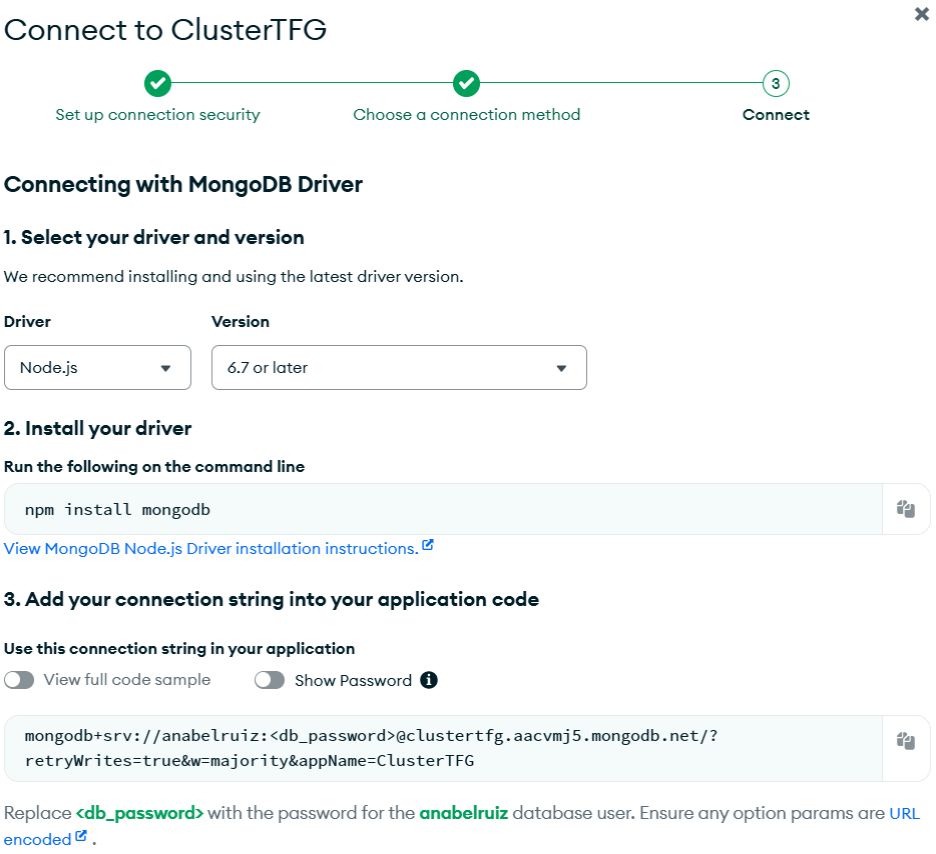

Figura 114. Obtención de la cadena de conexión MongoDB

Como alternativa para la gestión visual de la base de datos, se puede utilizar MongoDB Compass conectándose al cluster creado, tal como se observa en la Figura 115. Esta herramienta gráfica permite visualizar y administrar las bases de datos y colecciones de forma intuitiva, facilitando el monitoreo del contenido durante el desarrollo de la aplicación.

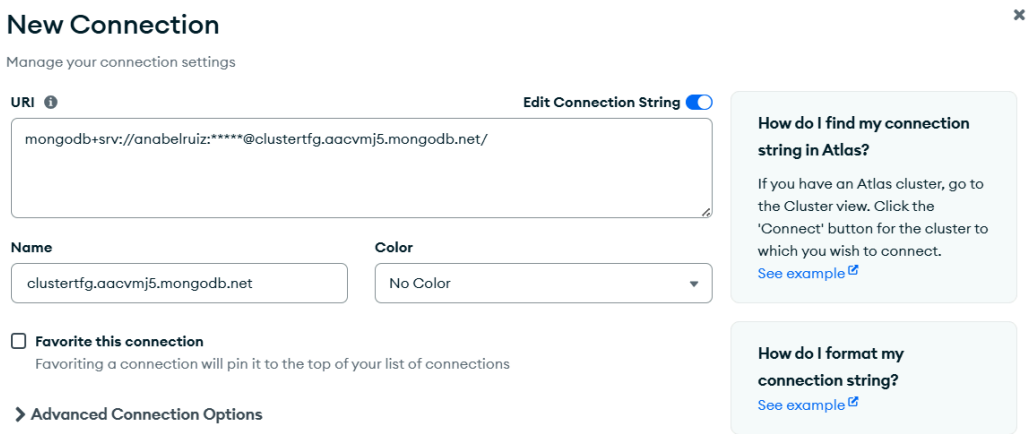

Figura 115. Conexión desde MongoDB Compass al cluster

B.4. Ejecutar la aplicación

Si se utiliza MongoDB de forma local, se debe iniciar el servicio ejecutando el comando `mongod` en una terminal del sistema. Una vez configuradas todas las variables de entorno y asegurándonos de que MongoDB está ejecutándose, se procede a lanzar tanto el backend como el frontend.

En las dos terminales previas, se debe ejecutar `npm run dev` en cada una de ellas. La terminal situada en la raíz del proyecto ejecutará el backend del servidor, mientras que la ubicada en el directorio `client` iniciará el frontend de la aplicación como se observa en la Figura 116. Una vez ambos estén ejecutándose correctamente, se puede acceder a la aplicación abriendo un navegador web y navegando a la dirección `http://localhost:5173`.

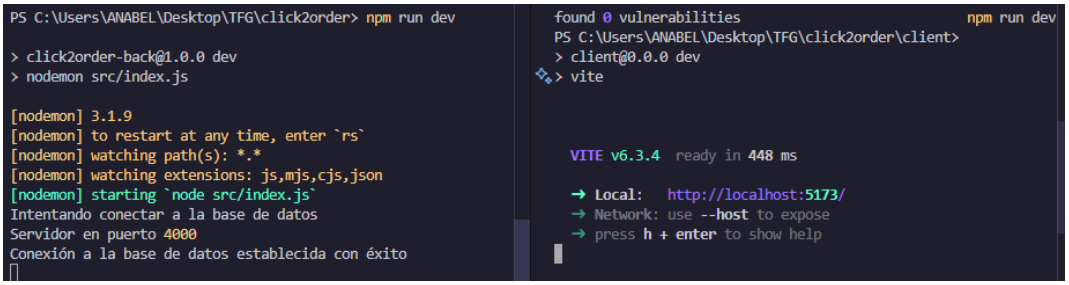

Figura 116. Terminales VS Code ejecutando

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