Universidad de Valladolid

Escuela de Ingeniería Informática de Valladolid

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención Ingeniería del Software

Mercado Rural: una web para digitalizar el comercio en las zonas rurales

Autor:

Dña. Sara Crespo Morán

Tutor:

Dña. Yania Crespo González-Carvajal

Resumen

Mercado Rural es una plataforma web diseñada para modernizar el comercio local y de proximidad en las zonas rurales de España. Su objetivo es facilitar a los pequeños comerciantes la digitalización de sus negocios, permitiéndoles ofrecer productos como alimentos, artículos de farmacia y combustibles de manera accesible para los compradores, sin pretender competir con las grandes plataformas de comercio electrónico.

El desarrollo del proyecto se ha llevado a cabo utilizando tecnologías como Angular para el frontend, Spring Boot con Java para el backend y MySQL como base de datos relacional. En cuanto a la metodología, se ha seguido un enfoque en cascada para las fases iniciales de análisis y diseño, complementado con un modelo de proceso ágil en la fase de implementación y pruebas, garantizando flexibilidad y eficiencia en el desarrollo.

Abstract

Mercado Rural is a web platform designed to modernize local and proximity commerce in rural areas of Spain. Its objective is to facilitate small merchants the digitalization of their businesses, allowing them to offer products such as food, pharmacy items and fuels in an accessible way for buyers, without pretending to compete with large e-commerce platforms.

The development of the project has been carried out using technologies such as Angular for the frontend, Spring Boot with Java for the backend and MySQL as a relational database. Regarding the methodology, a waterfall approach has been followed for the initial analysis and design phases, complemented with an agile process model in the implementation and testing phase, ensuring flexibility and efficiency in the development.

Índice general

Lista de figuras

Lista de tablas

Capítulo 1
Introducción

1.1. Contexto y motivación

En las zonas rurales de España, la modernización tecnológica ha avanzado de forma desigual en comparación con las ciudades. Estas comunidades, caracterizadas por una baja densidad de población y un alto porcentaje de personas mayores, enfrentan barreras que dificultan su acceso a las plataformas digitales y al comercio electrónico. Mientras que en entornos urbanos proliferan servicios como Just Eat, Glovo o Uber Eats, en los pueblos pequeños estas soluciones no son viables debido a la falta de infraestructura, el escaso volumen de usuarios y la baja rentabilidad para las grandes empresas.

Como resultado, los habitantes de estas zonas dependen de pequeños comercios locales y del reparto periódico de mercancías desde pueblos cercanos, un modelo tradicional que, aunque funcional, no siempre es práctico. En muchos casos, los productos llegan a través de camiones o furgonetas que reparten solo en determinados días y horarios, por lo que si alguien no puede estar presente en ese momento, corre el riesgo de quedarse sin acceso a ciertos bienes. Esta situación afecta especialmente a personas mayores o con problemas de movilidad, quienes enfrentan además el desafío de adaptación a la tecnología.

Este proyecto surge con la intención de modernizar el comercio local sin perder de vista las particularidades de estas comunidades. La plataforma que se propone permitirá a los comercios digitalizarse, ofreciendo a los habitantes la posibilidad de realizar pedidos desde sus hogares, con una interfaz accesible tanto para jóvenes como para personas mayores. Para estos últimos, se han diseñado opciones más tradicionales, como la posibilidad de contactar fácilmente con los comercios por teléfono.

Además de facilitar la vida de los residentes, esta solución busca apoyar a los pequeños comerciantes, ampliando su alcance sin necesidad de grandes inversiones. En definitiva, el objetivo es construir un puente entre tradición y tecnología, mejorando la calidad de vida en estas comunidades y contribuyendo a frenar la despoblación.

1.2. Alternativas existentes

En el contexto de la modernización de las áreas rurales, es fundamental analizar las plataformas de entrega y comercio electrónico que ya operan en el mercado. Aunque existen soluciones como Just Eat, Glovo y Uber Eats, su enfoque ha sido mayoritariamente urbano, lo que limita su efectividad en comunidades con baja densidad de población. En este apartado se exploran las características,

funcionalidades y limitaciones de estas aplicaciones, así como su público objetivo y las zonas en las que operan, para establecer una comparación clara con mi propuesta de plataforma.

1.2.1. Just Eat, Glovo y UberEats

Estas plataformas están diseñadas para operar en entornos urbanos, donde la alta densidad de población y la infraestructura favorecen la entrega rápida de pedidos.

- **Just Eat** conecta usuarios con restaurantes locales, centrándose exclusivamente en comida a domicilio.
- **Glovo** amplía su oferta con productos de supermercados, farmacias y otros comercios.
- **Uber Eats** se diferencia por aprovechar la red de conductores de Uber, lo que le permite ofrecer entregas más rápidas y mayor disponibilidad de repartidores.

A pesar de su éxito en ciudades, estas aplicaciones no operan en zonas rurales debido a la baja rentabilidad. Esta propuesta busca cubrir esa necesidad, facilitando el acceso digital a comercios locales en áreas menos conectadas.

1.2.2. Supermercados con tienda online

Algunas grandes cadenas de supermercados, como **Mercadona, Carrefour o Día**, han implementado plataformas online que permiten a los clientes realizar pedidos y recibirlos a domicilio. Estos servicios, aunque pueden facilitar la compra de productos sin necesidad de desplazarse, presentan limitaciones en el contexto rural. En muchos casos, el reparto solo está disponible en determinadas áreas y requiere un gasto mínimo para que el envío sea gratuito o rentable. Además, la oferta se centra en productos de grandes distribuidores, dejando fuera a los pequeños comercios y productores locales.

1.2.3. "Vamos ya!"

"Vamos Ya!" es un servicio de alquiler de coches con conductor similar a Uber, pero diseñado específicamente para cubrir rutas interurbanas en zonas con menor densidad de población. Este tipo de modelo demuestra que es posible adaptar soluciones tecnológicas a las particularidades del entorno rural, ofreciendo alternativas de movilidad más accesibles para quienes no disponen de transporte propio (Valladolid Movilidad Sostenible S.L., 2017).

Si bien no está enfocado en la compra de productos, su éxito en la adaptación de un servicio urbano a una necesidad rural refuerza la idea de que los entornos rurales pueden beneficiarse de plataformas digitales diseñadas específicamente para ellos.

1.3. Modelo de negocio

El modelo de negocio está diseñado para garantizar la sostenibilidad del sistema mientras ofrece a los comercios rurales una solución justa y accesible para digitalizar sus ventas. Este apartado se divide en dos secciones: la decisión actual, que describe las fuentes de financiación iniciales y el modelo implementado, y las posibilidades futuras, que plantean mejoras y ajustes que podrían introducirse a largo plazo.

1.3.1. Decisión actual

La principal fuente de ingresos iniciales será el apoyo económico de ayuntamientos y diputaciones provinciales, cubriendo costes operativos y promoviendo la digitalización del comercio local. Dado que muchos municipios pequeños tienen recursos limitados y los comercios y compradores suelen estar en localidades distintas dentro de una misma provincia, el respaldo de las diputaciones es clave para ampliar el alcance del servicio.

Complementariamente, se implementará un modelo basado en un porcentaje sobre cada compra que los clientes realicen a través de la plataforma. Para calcularlo, se parte de la estimación de los costes operativos anuales de la plataforma que se estiman en 8.808,85€ (ver apartado 7.3.3. Coste real) y del volumen de ventas esperado. En un escenario inicial con 15 comercios registrados, generando un promedio de 1.500€ al mes por comercio (270.000€ al año), el porcentaje necesario para cubrir los costes operativos sería del 3,26%. Para garantizar la sostenibilidad y un margen de beneficio del 20%, este porcentaje incrementa al 4%.

Este enfoque tiene las siguientes características:

- **Proporcionalidad.** El porcentaje aplicado será el mismo para todos los comercios y se calculará en función del importe total de cada compra realizada.
- **Distribución en pedidos conjuntos.** Si un cliente hace un pedido que incluye productos de varios comercios, el porcentaje se aplicará al subtotal correspondiente a cada comercio. Por ejemplo, si un cliente realiza un pedido de 20€, de los cuales 15€ corresponden al comercio A y 5€ al comercio B, aplicando un porcentaje del 4%, la plataforma retendría:
  - 0,60€ del comercio A (4% de 15€)
  - 0,20€ del comercio B (4% de 5€)
- **Simplicidad inicial.** Este modelo es fácil de implementar y alinea los ingresos de la plataforma con el volumen de ventas de los comercios, siendo justo para negocios con ventas bajas.

Esta combinación de financiación municipal y provincial, junto con el porcentaje sobre ventas, permite establecer una base sólida para el lanzamiento de la plataforma, asegurando ingresos recurrentes mientras se fomenta la digitalización del comercio local.

1.3.2. Posibilidades futuras

De cara al crecimiento de la plataforma y a su sostenibilidad a largo plazo, he identificado varias opciones para evolucionar el modelo de negocio:

- **Combinación de cuota fija y porcentaje sobre las ventas.** Se podría establecer una cuota fija anual o mensual para cubrir costes básicos de operación, combinada con el porcentaje sobre las ventas. Este enfoque permite garantizar ingresos recurrentes mientras se mantiene un modelo proporcional.
- **Planes de suscripción personalizados.** Diseñar planes de suscripción (básico, avanzado, premium) que ofrezcan características adicionales, como:
  - Mayor visibilidad de los productos (por ejemplo, que aparezcan en los primeros resultados de búsqueda).
  - Acceso a estadísticas avanzadas sobre ventas y clientes.
  - Servicios de promoción o publicidad dentro de la plataforma.

Estas alternativas permiten mayor flexibilidad en la monetización, adaptándose a las necesidades de los comercios y al crecimiento de la plataforma. Aunque actualmente no se implementarán, son opciones viables para futuras iteraciones del modelo de negocio.

1.4. Objetivos

1.4.1. Objetivos del proyecto

En primer lugar, se plantean los objetivos del proyecto, que están orientados a las funcionalidades y características que la plataforma web proporcionará para resolver las necesidades previamente planteadas.

- Desarrollar una plataforma web accesible para usuarios de diferentes edades y niveles de conocimientos tecnológicos.
- Modernizar las relaciones entre compradores y vendedores de las zonas rurales, permitiendo a los pequeños comerciantes ofrecer sus productos y servicios de manera digital a los residentes de su entorno.
- Incorporar funcionalidades de entrega flexible, como lockers y reparto a domicilio, adaptadas a las necesidades de los usuarios.
- Desarrollar una interfaz intuitiva y sencilla que permita tanto a los jóvenes como a las personas mayores realizar pedidos de manera eficiente, incluyendo alternativas para acceder a los servicios sin necesidad de usar la web (por ejemplo, llamadas telefónicas a los negocios locales).

- Apoyar a los pequeños comerciantes locales en la digitalización de sus negocios mediante herramientas simples que les permitan gestionar productos, recibir pedidos y coordinar entregas sin requerir grandes conocimientos técnicos.
- Garantizar la accesibilidad en áreas con baja conectividad a internet, optimizando la plataforma para funcionar en condiciones de conexión limitada.

1.4.2. Objetivos personales

- Adquirir experiencia práctica en el desarrollo full-stack, trabajando en el front-end con Angular y en el back-end con Spring Boot.
- Aprender a trabajar con diferentes modelos de proceso, como el desarrollo por fases (en cascada) y procesos ágiles, utilizando el marco de trabajo Scrum.
- Consolidar mis conocimientos sobre las actividades de un proyecto software (análisis, diseño, implementación y pruebas).
- Realizar una propuesta con un objetivo social, contribuyendo a la digitalización del comercio rural y mejorando el acceso a productos locales para diferentes grupos de población, incluidos mayores y personas con menor familiaridad tecnológica.

1.5. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1: Introducción. En este capítulo se describe el origen de la idea del proyecto, qué motivación hay para desarrollarla, algunas alternativas ya existentes y qué objetivos se pretenden conseguir con el proyecto.

Capítulo 2: Requisitos y planificación. En este capítulo se presenta el modelo de proceso que se seguirá, los requisitos funcionales y no funcionales del proyecto, los actores del sistema y los stakeholders. También se incluye un análisis de riesgos que podrían surgir durante el desarrollo del proyecto, así como un análisis de costes.

Capítulo 3: Análisis. En este capítulo se presentan los casos de uso del sistema, acompañados del diagrama de casos de uso y una matriz de correspondencia entre los casos de uso y los requisitos funcionales para garantizar que todos los requisitos están cubiertos. Además, se incluye el modelo de dominio del sistema, el diagrama de estados que representa los diferentes estados de un pedido, y los bocetos del diseño de la interfaz de usuario tanto para la parte del comprador como para los comercios.

Capítulo 4: Tecnologías utilizadas. En este capítulo se detallan las herramientas y tecnologías empleadas en el desarrollo del proyecto. Primero, se describen las herramientas utilizadas para la

comunicación, después las específicas para el análisis, diseño y documentación y, finalmente, las herramientas empleadas para la gestión y control del proyecto.

Capítulo 5: Diseño. En este capítulo se describe el diseño del sistema, incluyendo la arquitectura y los patrones de diseño aplicados. También se explican las decisiones de diseño, como la versión de Angular elegida, el logo, la tipografía, la paleta de colores y los bocetos de la interfaz de usuario. Finalmente, se presenta el diseño de la base de datos, ilustrado con un diagrama que muestra la estructura y relaciones entre las entidades.

Capítulo 6: Implementación y pruebas. En este capítulo se describe la estructura y organización del código fuente del proyecto en su totalidad, así como la recopilación y descripción de las pruebas realizadas, incluyendo todos los casos de prueba que se llevaron a cabo para garantizar el correcto funcionamiento de la aplicación.

Capítulo 7: Seguimiento del proyecto. En este capítulo se presentan los tiempos reales empleados en cada una de las fases del proyecto. Para aquellas fases en las que se aplicó un proceso ágil basado en SCRUM, se proporciona un seguimiento detallado de las tareas realizadas en cada sprint, incluyendo el tiempo invertido en cada una de ellas. Además, se llevan a cabo actividades de revisión y retrospectiva para analizar qué ha ido bien y qué ha ido mal, con el fin de aprender de ello para futuros sprints. Se analizan las diferencias entre lo planificado y lo ejecutado, identificando posibles mejoras que podrían implementarse para optimizar el desarrollo y el proceso de trabajo. Se hace un resumen de la ejecución general del proyecto y de las lecciones aprendidas durante su desarrollo.

Capítulo 8: Conclusiones. En este capítulo de conclusiones, se realiza un análisis de los objetivos planteados al inicio del proyecto, contrastándolos con los resultados obtenidos a lo largo de su desarrollo. Se evalúa si los objetivos fueron alcanzados y se reflexiona sobre el proceso en general, destacando los logros conseguidos y la satisfacción personal al ver cómo la idea inicial se ha transformado en una solución funcional. Además, se presentan las líneas de trabajo futuras, identificando áreas en las que el proyecto podría continuar evolucionando y mejorando, así como posibles nuevas funcionalidades y mejoras que podrían implementarse para seguir satisfaciendo las necesidades de los usuarios y ampliando el alcance del sistema.

Anexo A Manuales. Incluye manuales para la instalación y despliegue del sistema, así como un manual de usuario.

Anexo B Resumen de enlaces adicionales. Incluye enlaces de interés sobre el proyecto, como el repositorio del código.

Capítulo 2
Requisitos y Planificación

En este capítulo se definen los stakeholders, los actores del sistema, los requisitos funcionales y no funcionales que regirán el comportamiento del sistema, se hace un análisis de riesgos, se presenta la planificación del proyecto y, por último, se hace un análisis de costes.

2.1. Stakeholders

Los stakeholders o partes interesadas en un proyecto son aquellas personas, grupos o entidades que de alguna manera tienen un interés o pueden verse afectados por el desarrollo y los resultados del proyecto. A continuación, se describen los principales stakeholders de la plataforma, sus intereses y el papel que desempeñan en el contexto del desarrollo.

- **Comerciantes locales.** Son parte clave del sistema, ya que utilizarán la plataforma para digitalizar sus negocios, ofrecer productos y gestionar pedidos.
- **Compradores.** Quienes usarán la plataforma para adquirir los productos. Esto incluye tanto personas mayores que podrían beneficiarse de las facilidades de compra, como los jóvenes que buscan más modernización.
- **Desarrolladores o equipo técnico.** Aquellos que podrían participar en el desarrollo, mantenimiento y soporte técnico de la plataforma a largo plazo.
- **Administraciones locales, provinciales y regionales.** Las autoridades pueden tener un interés en apoyar iniciativas que ayuden al desarrollo de las zonas rurales, como un medio de modernización y mejora de la calidad de vida.

2.2. Actores del sistema

En el desarrollo de cualquier sistema software es fundamental identificar y definir los actores que interactuarán con la plataforma. Los actores representan a las entidades externas que se comunican con el sistema, ya sean personas, otros sistemas o dispositivos.

En este caso, los actores principales del sistema son los vendedores y los compradores.

- **Vendedores**. Son los comerciantes locales que registran su negocio en la plataforma y gestionan los productos y servicios que ofrecen a los clientes.

- **Compradores**. Son los residentes de las zonas rurales que utilizan la plataforma para adquirir los productos, ya sea para recogerlos en los puntos de entrega o recibirlos en su domicilio.

2.3. Requisitos funcionales

Los requisitos funcionales definen las acciones que debe ser capaz de realizar el sistema para cumplir con las expectativas de los usuarios. Estos requisitos describen las funcionalidades y características que los actores del sistema, ya sean vendedores o compradores, necesitan para interactuar de manera efectiva con la plataforma.

Los requisitos aquí listados corresponden al producto mínimo viable (MVP) para la primera release del proyecto, lo que significa que no se busca cubrir todos los posibles requisitos, sino aquellos que permiten cumplir con la idea principal y básica de la plataforma. A medida que el sistema evolucione, se podrán añadir más funcionalidades en futuras versiones.

Vendedores:

RF01. El sistema deberá permitir a los vendedores registrar su comercio proporcionando la información especificada en el requisito de información RI01.

RF02. El sistema deberá permitir a los vendedores iniciar sesión utilizando el correo electrónico de la empresa y la contraseña.

RF03. El sistema deberá permitir a los vendedores visualizar los pedidos realizados por los compradores a través de la plataforma. Los pedidos deberán estar organizados por localidades, y dentro de cada localidad, podrán ver el pedido realizado por cada persona, visualizando concretamente: nombre del cliente, dirección, teléfono, los productos pedidos (nombre y cantidad) y el tipo de entrega solicitado por el cliente (a domicilio o a locker).

RF04. El sistema deberá permitir a los vendedores actualizar el estado de un pedido (ver estados en el RI04), permitiendo cambiarlo de "solicitado" a "preparado" y, posteriormente, de "preparado" a "entregado".

RF05. El sistema deberá permitir a los vendedores registrar pedidos manualmente que reciban telefónicamente, indicando: nombre del cliente, su dirección de domicilio, teléfono del cliente, productos pedidos y cantidad, y el tipo de entrega (domicilio o locker).

RF06. El sistema deberá permitir a los vendedores añadir productos proporcionando la información especificada en RI03.

RF07. El sistema deberá permitir a los vendedores eliminar productos de su catálogo de venta.

RF08. El sistema deberá permitir a los vendedores modificar productos, pudiendo cambiar nombre, precio y descripción.

RF09. El sistema deberá permitir a los vendedores modificar la información registrada de su comercio, pero únicamente el nombre del comerciante, el teléfono y la contraseña.

RF10. El sistema deberá permitir a los vendedores visualizar el historial de pedidos, mostrando: nombre del cliente, productos pedidos (nombre y cantidad), y el costo total de la compra.

RF11. El sistema deberá permitir a los vendedores definir las localidades a las que pueden realizar entregas.

RF12. El sistema deberá permitir a los vendedores publicar avisos en un tablón de anuncios, eligiendo las localidades donde aparecerán y proporcionando la información especificada en RI07.

RF13. El sistema deberá permitir a los vendedores cerrar sesión.

Compradores:

RF14. El sistema deberá permitir a los compradores registrarse proporcionando la información especificada en el requisito de información RI02.

RF15. El sistema deberá permitir a los compradores iniciar sesión utilizando su correo electrónico y contraseña.

RF16. El sistema deberá permitir a los compradores cerrar sesión.

RF17. El sistema deberá permitir a los compradores modificar su información personal registrada, especificada en RI02.

RF18. El sistema deberá permitir a los compradores ver los productos de los comercios, pero únicamente de aquellos que venden en la localidad del comprador.

RF19. El sistema deberá permitir a los compradores agregar productos a su carrito de compra.

RF20. El sistema deberá permitir a los compradores realizar pedidos, los cuales constarán de la información especificada en el requisito RI05. El pedido en el momento de ser realizado tendrá el estado "solicitado".

RF21. El sistema deberá permitir a los compradores ver su historial de pedidos, mostrando: fecha del pedido, productos pedidos (nombre, cantidad y precio) y precio total de la compra.

RF22. El sistema deberá permitir a los compradores realizar el pago de sus pedidos mediante PayPal.

Respecto al RF22, se elige esta opción ya que PayPal, a través de su pasarela de pago, permite a los usuarios pagar con tarjetas de crédito registradas previamente o utilizando el saldo disponible en su cuenta de PayPal, lo que facilita la experiencia de pago y proporciona mayor flexibilidad para los usuarios. Se asume además que la cuenta de PayPal de cada comercio está asociada al correo electrónico indicado en el registro de la plataforma, de modo que los pagos se procesarán correctamente a la cuenta correspondiente.

2.4. Requisitos de información

Este tipo de requisitos definen los datos específicos que deben ser recolectados y gestionados por el sistema para garantizar su correcto funcionamiento y cumplir con las necesidades de los usuarios. Están estrechamente vinculados a los requisitos funcionales, ya que cada funcionalidad que implique el registro, modificación o consulta de datos debe especificar qué información es necesaria. En todo momento, el sistema deberá cumplir con lo especificado en el requisito RNF01.

A continuación, se detallan los requisitos de información:

RI01. El sistema deberá recoger la siguiente información de los vendedores durante el registro: nombre del comercio, dirección, correo electrónico, teléfono, CIF, categoría del negocio (panadería, carnicería, frutas y verduras, venta de combustibles), nombre y apellidos del responsable del negocio en la plataforma, y la contraseña con la que posteriormente se podrá identificar en la plataforma.

RI02. El sistema deberá recoger la siguiente información de los compradores durante el registro: nombre y apellidos, número de teléfono, dirección de domicilio, correo electrónico y contraseña.

RI03. El sistema deberá recoger la siguiente información de un producto registrado por un vendedor: nombre, precio, precio por unidad (precio/kg o precio/litro según corresponda) y los ingredientes que conforman dicho producto.

RI04. El sistema deberá registrar el estado de un pedido, el cual puede ser: "solicitado", "preparado" o "entregado".

RI05. El sistema deberá recoger la siguiente información de un pedido de un comprador: fecha del pedido, productos solicitados (nombre, cantidad y precio), estado del pedido y el precio total de la compra.

RI06. El sistema deberá recoger información sobre la entrega: fecha y hora.

RI07. El sistema deberá recoger la siguiente información de un aviso publicado por un vendedor: localidades en las que se quiere publicar y el texto del aviso.

2.5. Reglas de negocio

Las reglas de negocio definen políticas, restricciones y comportamientos específicos que deben seguirse en el funcionamiento del sistema. Estas reglas no son propiamente requisitos funcionales, pero influyen en la lógica del sistema y establecen condiciones sobre cómo deben cumplirse ciertos procesos o decisiones dentro del mismo.

Estas son:

RN01. Durante la realización de un pedido, si el comprador elige pagar en efectivo, solo puede seleccionar la opción de entrega a domicilio, mientras que si elige pagar con PayPal, puede optar por la entrega a domicilio o a un locker.

RN02. Un pedido no puede marcarse como "entregado" sin antes haber sido marcado como "preparado".

RN03. La cantidad máxima que se puede pedir de un producto es de 50 unidades.

RN04. Si la cantidad solicitada por un comprador no es viable para el vendedor, este tendrá acceso al número de teléfono del cliente para contactar y acordar una nueva cantidad. Asimismo, podrá editar el pedido directamente desde su área de vendedor para reflejar la nueva cantidad acordada.

RN05. Los pedidos deberán realizarse al menos con un día de antelación respecto al día de entrega.

RN06. Cuando el vendedor marque el pedido como entregado se considerará previamente pagado.

2.6. Requisitos no funcionales

Los requisitos no funcionales son aquellos criterios que definen cómo debe comportarse un sistema, en lugar de qué funciones debe realizar. Estos requisitos abarcan aspectos como el rendimiento, la seguridad, la usabilidad y la escalabilidad, y son fundamentales para garantizar que el sistema no solo cumpla con sus funciones, sino que también satisfaga las expectativas de los usuarios en términos de calidad y experiencia.

Estos serían:

RNF01. El sistema deberá garantizar que los datos personales de los usuarios sean tratados de manera segura y conforme a las normativas vigentes de protección de datos, como el RGPD (Reglamento General de Protección de Datos). Además, se asegurará de que solo se recojan y almacenen los datos estrictamente necesarios para el funcionamiento de la plataforma, evitando solicitar o almacenar información que no sea esencial para el propósito del servicio.

RNF02. El sistema deberá ofrecer tiempos de respuesta de menos de 10 segundos para garantizar una experiencia de usuario fluida, incluso en momentos de alta carga.

RNF03. El sistema deberá ser capaz de gestionar un número razonable de usuarios conectados simultáneamente, estimado en 1000 usuarios activos en un momento dado, sin afectar significativamente el rendimiento.

RNF04. Las transacciones en la plataforma deberán ser seguras, empleando mecanismos de cifrado para proteger la información sensible, en particular los datos de pago.

RNF05. El sistema deberá ser escalable, permitiendo una expansión gradual para soportar un mayor número de usuarios y operaciones conforme crezca la plataforma.

2.7. Análisis de riesgos

El plan de riesgos es una herramienta fundamental para asegurar el éxito de cualquier proyecto, ya que permite identificar, analizar y planificar cómo abordar los posibles problemas que puedan surgir durante su desarrollo. Los riesgos son eventos o situaciones inciertas que, de materializarse, pueden afectar negativamente el avance, la calidad o el resultado final del proyecto. Evaluar estos riesgos de manera anticipada es clave para minimizar su impacto o incluso evitar que ocurran.

Para cada riesgo identificado, se considerarán cuatro elementos esenciales que permiten un análisis más profundo:

- **Probabilidad.** Se refiere a la posibilidad de que un riesgo ocurra. Determinar la probabilidad de cada riesgo ayuda a priorizarlos, ya que aquellos con mayor probabilidad de materializarse deben ser gestionados con mayor urgencia y detalle.
- **Impacto.** Hace referencia a la magnitud del daño que causaría el riesgo si llegara a ocurrir. Al igual que la probabilidad, es importante conocer el impacto para evaluar la gravedad del riesgo y preparar un plan adecuado. Un riesgo con un alto impacto, aunque tenga baja probabilidad, también debe recibir especial atención.
- **Plan de mitigación.** Son las acciones preventivas que se implementan para reducir la probabilidad de que el riesgo se materialice o disminuir su impacto en caso de que ocurra. La mitigación busca reducir el riesgo antes de que suceda, anticipándose al problema.
- **Plan de contingencia.** Este plan establece qué se hará si el riesgo finalmente se produce, es decir, las acciones correctivas para gestionar el problema una vez ha ocurrido. A diferencia de la mitigación, que busca evitar el riesgo, la contingencia se activa para reducir el daño cuando ya se ha materializado.

Fuentes: (PMBC, 2025) y (SmartSheet, 2023)

Para evaluar adecuadamente la prioridad de cada riesgo, se utilizará una matriz de riesgos que para cada combinación de probabilidad e impacto indicará las acciones que se deben llevar a cabo. En la Tabla 2.1 se muestra esta matriz de riesgos.

Impacto / Probabilidad	BAJO	MEDIO	ALTO
BAJA	No es necesario hacer nada	No es necesario hacer nada	Mitigar
MEDIA	No es necesario hacer nada	Mitigar	Planificar la contingencia
ALTA	Mitigar	Planificar la contingencia	Planificar la contingencia

Tabla 2.1. Matriz de riesgos

A continuación, se presentan los riesgos desde la Tabla 2.2 hasta la Tabla 2.9.

ID del riesgo	R01
Título	Trabajo individual
Categoría	Personal
Descripción	Realizar el TFG de manera individual supone un reto, ya que la falta de colaboración y retroalimentación constante de otros puede generar inseguridad y dificultar la mejora continua del trabajo. Aunque se cuenta con el apoyo del tutor, gestionar todas las áreas del proyecto puede resultar abrumador en algunos momentos.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Pedir ayuda al tutor/a ▪ Establecer desde el inicio un plan de trabajo y una programación realista para prevenir la acumulación de tareas. Además, es crucial priorizar y gestionar el tiempo de manera eficaz para asegurar un progreso continuo y ordenado en el desarrollo del proyecto.
Plan de contingencia	▪ Si bien este riesgo es inevitable, es posible mitigar sus consecuencias (estrés o falta de tiempo) mediante la implementación de medidas como técnicas de relajación y una adecuada organización.

Tabla 2.2. R01: Trabajo individual

ID del riesgo	R02
Título	Falta de tiempo
Categoría	Personal
Descripción	La falta de tiempo es un riesgo significativo, especialmente al tener que compaginar el desarrollo del TFG con las prácticas en empresa y una asignatura. Estas responsabilidades adicionales pueden dificultar el

	cumplimiento de los plazos establecidos y la entrega del proyecto en la fecha prevista.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	▪ Elaborar un cronograma detallado que facilite el seguimiento del progreso a lo largo del tiempo y permita identificar problemas con anticipación. ▪ Mantener una comunicación regular con el tutor, ya que su experiencia en trabajos de fin de grado les permite identificar situaciones que pueden requerir más tiempo del que el estudiante podría anticipar.
Plan de contingencia	▪ Consultar con el tutor sobre la posibilidad de extender los plazos de entrega si fuera necesario. ▪ Priorizar las tareas esenciales y ajustar el cronograma para concentrarse en los elementos más críticos.

Tabla 2.3. R02: Falta de tiempo

ID del riesgo	R03
Título	Desafíos técnicos
Categoría	Tecnología
Descripción	Este proyecto requiere el uso de tecnologías como Angular, de la cual se tiene poco conocimiento, y la implementación de pagos con PayPal, que nunca se ha utilizado previamente. Estos desafíos técnicos pueden retrasar el avance del proyecto y generar dificultades adicionales.
Probabilidad	Media
Impacto	Alta
Plan de mitigación	▪ Si es posible, adquirir habilidades y conocimientos técnicos adicionales sobre Angular y la integración de pagos con PayPal a través de tutoriales o documentación oficial.
Plan de contingencia	▪ Revisar y ajustar el cronograma del proyecto para considerar posibles retrasos debido a la curva de aprendizaje y a la resolución de dificultades técnicas.

Tabla 2.4. R03: Desafios técnicos

ID del riesgo	R04
Título	Dificultades de redacción
Categoría	Personal
Descripción	La capacidad de redactar de manera clara y coherente es esencial en el TFG. Si se tienen dificultades para expresar ideas de forma estructurada o

	profesional, esto puede afectar negativamente la presentación final del trabajo.
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Realizar revisiones periódicas del trabajo escrito con el tutor, familiares o amigos que puedan ofrecer correcciones y sugerencias de redacción.
Plan de contingencia	▪ Solicitar revisiones periódicas al tutor, familiares o amigos.

Tabla 2.5. R04: Dificultades de redacción

ID del riesgo	R05
Título	Problemas de salud o personales
Categoría	Personal
Descripción	Problemas de salud imprevistos o situaciones personales difíciles pueden impactar gravemente en la dedicación y el progreso del TFG, comprometiendo la capacidad de cumplir con los plazos.
Probabilidad	Medio
Impacto	Medio
Plan de mitigación	▪ Mantener un estilo de vida saludable, que incluya una alimentación equilibrada, actividad física regular y descansos adecuados, para minimizar el riesgo de problemas de salud.
Plan de contingencia	▪ Comunicar al tutor la situación y solicitar una extensión de los plazos de entrega.

Tabla 2.6. R05: Problemas de salud o personales

ID del riesgo	R06
Título	Cambios de enfoque o de tema
Categoría	Personal
Descripción	Es común que durante el desarrollo del TFG surjan nuevas ideas o enfoques que requieran modificar parte del trabajo ya realizado. Esto puede generar retrasos considerables y la necesidad de rehacer secciones importantes.
Probabilidad	Medio
Impacto	Alto
Plan de mitigación	▪ Registrar cualquier nueva idea, evaluando cuidadosamente su relevancia antes de realizar cambios importantes en el enfoque del trabajo.
Plan de contingencia	▪ Si se decide cambiar el enfoque del proyecto, elaborar un nuevo plan y obtener la aprobación del profesor, teniendo en cuenta los plazos disponibles.

Tabla 2.7. R06: Cambios de enfoque o tema

ID del riesgo	R07
Título	Pérdida de información
Categoría	Técnicos
Descripción	Existe la posibilidad de que, debido a errores en el sistema de archivos o fallos en los procesos de guardado, se pierda información crítica, como las versiones más recientes del proyecto o documentos clave del TFG. Esto podría suceder por la corrupción de archivos, fallos en el disco duro, o por un mal funcionamiento del sistema operativo, causando la pérdida de datos no respaldados.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Implementar un sistema de copias de seguridad automáticas y regulares que guarde versiones del trabajo en diferentes ubicaciones (nube y dispositivos físicos). ▪ Utilizar software de control de versiones, como Git, para asegurar que los cambios en el código y los documentos importantes sean respaldados continuamente.
Plan de contingencia	▪ Recuperar la última versión guardada o un archivo de respaldo almacenado en un servicio en la nube o un dispositivo externo. ▪ Utilizar herramientas de recuperación de datos para intentar restaurar los archivos dañados o corruptos.

Tabla 2.8. R07: Pérdida de información

ID del riesgo	R08
Título	Fallos de hardware
Categoría	Técnicos
Descripción	El riesgo de que el ordenador o los dispositivos utilizados para desarrollar el proyecto sufran daños o fallos es relevante, como la avería del disco duro, fallos en la memoria RAM o problemas con la fuente de alimentación. Estos fallos podrían interrumpir el desarrollo del TFG y ocasionar la pérdida de avances recientes si no se cuenta con respaldos.
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	▪ Mantener el equipo en condiciones óptimas mediante revisiones periódicas y la utilización de software de diagnóstico para prever posibles fallos. ▪ Contar con dispositivos de respaldo o acceso a otro equipo para continuar el desarrollo en caso de problemas técnicos.

Plan de contingencia	▪ Si el equipo principal falla, utilizar un equipo de respaldo o uno temporal mientras se repara el principal. ▪ Enviar el equipo a reparación técnica especializada y, mientras tanto, continuar trabajando desde servicios en la nube o en un equipo alternativo.

Tabla 2.9. R08: Fallos de hardware

2.8. Modelo de proceso

La planificación y selección del modelo de desarrollo son aspectos cruciales para garantizar el éxito de cualquier proyecto. Dado que cada fase del proyecto tiene características diferentes, he optado por utilizar un modelo de desarrollo mixto que combine lo mejor de dos enfoques ampliamente utilizados en ingeniería de software: el modelo en cascada y el marco de trabajo ágil SCRUM.

Para las fases iniciales del proyecto, que incluyen la **especificación de requisitos**, la **fase de análisis y fase de diseño**, es fundamental establecer una base sólida y clara que no requiera constantes revisiones. Por ello, el **modelo en cascada**, que se desarrolla de manera secuencial, es la opción más adecuada.

Sin embargo, en las **fases de implementación, pruebas y mantenimiento**, donde se codifica el proyecto, es habitual que surjan ajustes, correcciones y cambios de último momento. Aquí es donde entra en juego la flexibilidad del marco ágil **SCRUM**, permitiendo dividir el trabajo en períodos cortos y gestionar de forma eficiente cualquier imprevisto que surja.

Antes de detallar cómo se aplicarán ambos enfoques al proyecto, es importante hacer una breve introducción sobre cada uno de ellos.

2.8.1. Modelo en cascada

El modelo en cascada es uno de los enfoques más tradicionales en el desarrollo de software que se caracteriza por ser un modelo secuencial en el que el proyecto avanza a través de una serie de fases bien definidas. Estas fases deben completarse antes de poder pasar a la siguiente.

Fases del modelo en cascada:

- **Análisis de requisitos.** En esta fase se recogen y documentan todas las necesidades del cliente o del sistema. Se realiza un estudio detallado de los requisitos que el sistema deberá cumplir, estableciendo una base clara de lo que se espera desarrollar.
- **Diseño del sistema.** Una vez definidos los requisitos, se pasa a diseñar la arquitectura del sistema. En esta fase se determina cómo se estructurará el software, definiendo los componentes, interfaces, bases de datos y otros aspectos técnicos clave.

- **Implementación.** Después de la fase de diseño, se procede a la codificación del sistema según las especificaciones ya establecidas.
- **Pruebas.** En esta fase se verifica que el sistema desarrollado cumple con los requisitos definidos en la fase inicial. Se realizan pruebas de integración, funcionalidad y rendimiento para asegurarse de que el software está libre de errores y funciona correctamente.
- **Despliegue y mantenimiento.** Una vez que el sistema ha sido probado y aprobado, se despliega en el entorno de producción. A partir de este punto, el software entra en la fase de mantenimiento, donde se resuelven posibles errores o se aplican mejoras según sea necesario.

Este modelo tiene varias ventajas que lo hacen ideal para las primeras fases de mi proyecto:

- ✓ Claridad en los requisitos: al exigir que todos los requisitos estén bien definidos desde el principio, se garantiza una visión clara y precisa de lo que se necesita, reduciendo el riesgo de confusión o cambios de última hora.
- ✓ Documentación exhaustiva: cada fase se documenta cuidadosamente, lo que facilita el seguimiento del progreso y asegura que el proyecto se desarrolle conforme a lo planificado.
- ✓ Fases bien estructuradas: el enfoque secuencial del modelo en cascada es perfecto para las etapas iniciales de mi proyecto, como la definición y especificación de requisitos y el diseño, ya que estas etapas necesitan un enfoque más estructurado y riguroso. El hecho de no volver atrás una vez que se completa una fase asegura que no se produzcan cambios drásticos o confusiones que puedan desestabilizar el proyecto.

Las fuentes consultadas han sido: (Ganttpro, 2024) y (IONOS, 2019).

2.8.2. SCRUM

Scrum es un marco de trabajo ágil para el desarrollo de software que facilita el desarrollo de proyectos de manera incremental y flexible. A diferencia de otros enfoques más tradicionales, como el modelo en cascada, Scrum se caracteriza por permitir adaptaciones constantes y la entrega continua a lo largo del ciclo de vida del proyecto. Está diseñado para gestionar proyectos complejos y en evolución, donde los requisitos pueden cambiar a medida que se avanza en el desarrollo. Está basado en tres pilares:

- **Transparencia.** El trabajo debe ser visible tanto para los que lo realizan como para los que lo reciben, debido a que muchas decisiones se basan en la percepción de este.
- **Inspección.** El progreso del trabajo y el cumplimiento de los objetivos debe ser revisado frecuentemente para evitar problemas o sesgos indeseados en el producto.

- **Adaptación.** Si el trabajo se desvía de lo deseado se debe ajustar tan pronto como sea posible para evitar mayor desviación.

Hay tres **roles** fundamentales que son claves para el éxito del proyecto:

- **Product Owner (PO).** Es el responsable de maximizar el valor del producto que se está desarrollando. El PO se encarga de gestionar el Product Backlog (lista de tareas pendientes) y de priorizar los elementos que se deben desarrollar en cada sprint. El PO también interactúa con los interesados para asegurarse de que el producto final cumpla con sus expectativas y necesidades.
- **Scrum Master (SM).** El Scrum Master actúa como facilitador y guía para el equipo de desarrollo. Se asegura de que se sigan los principios y prácticas de Scrum, eliminando obstáculos que puedan surgir durante el desarrollo. El Scrum Master no es un jefe, sino un líder servicial que fomenta la autoorganización del equipo y facilita la comunicación entre todos los roles.
- **Equipo de desarrollo.** Es un grupo de profesionales multidisciplinarios que trabajan juntos para entregar incrementos del producto al final de cada sprint. El equipo de desarrollo es autoorganizado y colabora para cumplir con los objetivos del sprint, tomando decisiones sobre cómo abordar cada tarea.

Scrum se organiza en una serie de **eventos** repetitivos que estructuran el trabajo del equipo:

- **Sprint.** Es el corazón de Scrum, un ciclo de desarrollo corto y constante que suele durar entre 1 y 4 semanas. Al final de cada sprint, se entrega un incremento del producto que debe ser funcional y presentable. Los sprints se realizan uno tras otro sin interrupciones, asegurando que siempre haya un objetivo a corto plazo que el equipo debe cumplir.
- **Sprint Planning.** Es la reunión que se lleva a cabo al inicio de cada sprint, donde el equipo selecciona del Product Backlog los elementos que se desarrollarán durante el sprint. El equipo discute qué se puede completar durante el sprint y cómo se llevará a cabo ese trabajo.
- **Daily Scrum.** Es una reunión diaria, de aproximadamente 15 minutos, en la que el equipo revisa su progreso y ajusta el plan de trabajo según sea necesario. Cada miembro responde a tres preguntas clave: qué hizo ayer, qué planea hacer hoy y si hay algún obstáculo que impida su progreso.
- **Sprint Review.** Se realiza al final de cada sprint, donde el equipo muestra el incremento del producto al Product Owner y a otros interesados. El propósito es obtener retroalimentación y asegurarse de que el producto desarrollado cumpla con las expectativas.

- **Sprint Retrospective.** Después de la revisión, el equipo reflexiona sobre el proceso y cómo puede mejorarlo. Es una oportunidad para identificar áreas de mejora tanto en el flujo de trabajo como en la colaboración del equipo.

Scrum utiliza varios **artefactos** para organizar y hacer visible el trabajo del equipo. Estos artefactos ayudan a gestionar el progreso y asegurar que el equipo se mantenga enfocado en los objetivos clave:

- **Product Backlog.** Es una lista priorizada de todas las funcionalidades, tareas y mejoras que el producto necesita. El Product Owner es el encargado de mantener y priorizar el Product Backlog, asegurándose de que siempre esté actualizado.
- **Sprint Backlog.** Una lista más específica que contiene los elementos seleccionados del Product Backlog que el equipo de desarrollo se compromete a completar durante el sprint. Incluye las tareas necesarias para cumplir con el objetivo del sprint.
- **Incremento.** Es el resultado del trabajo realizado durante el sprint. El incremento debe ser funcional y estar en un estado presentable, listo para ser entregado al usuario o cliente final. Cada incremento se suma al anterior, construyendo así el producto final de manera continua.

Las fuentes consultadas han sido: (Atlassian, 2013), (Proyectos Ágiles, 2015) y (We are marketing, 2024).

2.9. Plan de proyecto

Ahora que se ha analizado teóricamente el modelo en cascada y el marco de trabajo Scrum, en este apartado vamos a ver cómo aplicar estos enfoques al proyecto. Se realizará una planificación inicial donde se estimarán los tiempos necesarios para completar las diferentes tareas. En primer lugar, se abordará la planificación de la primera mitad del proyecto, guiada por el modelo en cascada, y posteriormente, se detallará la segunda parte, que será gestionada bajo el marco de trabajo Scrum.

2.9.1. Adaptación del modelo de cascada al proyecto y calendarización

A diferencia del enfoque tradicional, que establece la fase de análisis como la primera de forma directa, en mi planificación he decidido introducir una fase inicial con la contextualización, motivación y objetivos del proyecto, así como la identificación y definición de los requisitos funcionales y no funcionales del sistema, los stakeholders, un análisis de riesgos y un análisis de costes. Esta fase inicial es esencial para sentar las bases del desarrollo y garantizar que todas las decisiones posteriores estén alineadas con las expectativas y necesidades del proyecto.

Además, esta adaptación incluye una calendarización detallada, en la que se establecen los tiempos estimados para cada fase del modelo en cascada, asegurando una visión clara del progreso y los plazos del proyecto.

De la Tabla 2.10 a la Tabla 2.12 se muestran las fases y sus tareas.

Tarea	Fecha inicio	Fecha fin	Tiempo empleado
Contexto	14/09/2024	22/09/2024	2h
Motivación	16/09/2024	22/09/2024	2h
Análisis de alternativas	16/09/2024	22/09/2024	3h
Objetivos	14/09/2024	22/09/2024	1h 30m
Identificación de stakeholders	20/09/2024	20/09/2024	30m
Definición de requisitos funcionales	16/09/2024	22/09/2024	1h
Definición de requisitos no funcionales	16/09/2024	22/09/2024	30m
Análisis de riesgos	20/09/2024	22/09/2024	3h
Planificación	25/09/2024	29/09/2024	2h 30m
Análisis de costes	26/09/2024	29/09/2024	2h 30m
Resumen de la fase	14/09/2024	29/09/2024	18h 30m

Tabla 2.10. Estimación de tiempos de la fase inicial

Tarea	Fecha inicio	Fecha fin	Tiempo empleado
Definición de los casos de uso	30/09/2024	6/10/2024	5h
Modelo de casos de uso	6/10/2024	6/10/2024	30m
Modelo del dominio inicial	6/10/2024	6/10/2024	2h
Modelado de objetos como diagramas de estados	6/10/2024	6/10/2024	30m
Resumen de la fase	30/09/2024	6/10/2024	8h

Tabla 2.11. Estimación de tiempos de la fase de análisis

Tarea	Fecha inicio	Fecha fin	Duración (días)
Especificación de las tecnologías utilizadas	7/10/2024	9/10/2024	3h
Diseño arquitectónico	11/10/2024	13/10/2024	5h
Patrones de diseño	11/10/2024	13/10/2024	5h
Diseño de la base de datos	12/10/2024	13/10/2024	5h
Diseño de la interfaz de usuario	18/10/2024	27/10/2024	20h
Resumen de la fase	7/10/2024	27/10/2024	38h

Tabla 2.12. Estimación de tiempos de la fase de diseño

2.9.2. Adaptación de Scrum al proyecto y calendarización

En esta sección, se detalla cómo se adaptará el marco de trabajo Scrum para las fases de implementación, pruebas y mantenimiento.

La duración de cada sprint se ha fijado en dos semanas, durante las cuales se contemplará una carga de trabajo máxima de 40 horas. Cada sprint comenzará con un Sprint Planning, donde se definirán las tareas a abordar y se establecerá el objetivo del sprint. Además, se llevará a cabo una reunión cada martes con la tutora para revisar el progreso y ajustar el rumbo del proyecto según sea necesario y esto, a su vez, permitirá definir el inicio y fin de cada sprint. Al final de cada sprint, se realizará una Sprint Review para presentar el trabajo completado y recibir feedback, seguido de una Sprint Retrospective para reflexionar sobre el proceso y buscar oportunidades de mejora para futuros sprints.

Después de completar las fases de requisitos, análisis y diseño, quedan aproximadamente 277 horas de trabajo. Dado que se ha estimado un total de 350 horas dedicadas al TFG, esto resulta en un total de 7 sprints. Debido a los posibles riesgos en la planificación del proyecto, se ha añadido un sprint adicional como refuerzo. Este sprint servirá para completar tareas que no se hayan finalizado a tiempo o para trabajar en los detalles finales de la documentación que se debe entregar.

En la Tabla 2.13 se puede observar la calendarización de los diferentes sprints.

Sprint	Comienzo	Finalización	Observaciones
Sprint 1	29/10/2024	12/11/2024	
Sprint Planning	29/10/2024		
Scrum semanal	5/11/2024		
Sprint Review Sprint Retrospective	12/11/2024		
Sprint 2	12/11/2024	26/11/2024	

Scrum semanal	19/11/2024		
Sprint Review Sprint Retrospective Sprint Planning	26/11/2024		
Sprint 3	26/11/2024	10/12/2024	
Sprint Planning	26/11/2024		
Scrum semanal	3/12/2024		
Sprint Review Sprint Retrospective	10/12/2024		
Sprint 4	10/12/2024	24/12/2024	
Sprint Planning	10/12/2024		
Scrum semanal	17/12/2024		
Sprint Review Sprint Retrospective	24/12/2024		
Sprint 5	24/12/2024	7/01/2025	
Sprint Planning	24/12/2024		
Scrum semanal	-		La reunión con la tutora se deberá posponer debido a los días festivos de estas fechas, así que queda pendiente de fijar.
Sprint Review Sprint Retrospective	7/01/2025		
Sprint 6	7/01/2025	21/01/2025	
Sprint Planning	7/01/2025		
Scrum semanal	14/01/2025		
Sprint Review Sprint Retrospective	21/01/2025		
Sprint 7	21/01/2025	4/02/2025	
Sprint Planning	21/01/2025		
Scrum semanal	28/01/2025		
Sprint Review Sprint Retrospective	4/02/2025		
Sprint refuerzo 1	4/02/2025	18/02/2025	
Scrum semanal	11/02/2025		
Sprint Review Sprint Retrospective	18/02/2025		
Solicitud de defensa 1	11/02/2025		Si todo sale bien y acabo en los 7 sprints previstos
Solicitud de defensa 2	25/02/2025		Si debo usar el sprint 1 de refuerzo

Tabla 2.13. Planificación inicial SCRUM

2.10. Estimación de costes

Para la estimación de costes se considerarán tanto los recursos humanos como los costes de hardware y software necesarios para la ejecución del proyecto. En primer lugar, se presentará un caso simulado, seguido por el análisis del caso real específico de este proyecto.

2.10.1. Coste simulado

El **salario de un desarrollador de software** en España está entre 22.000€ y 34.000€ anuales (Glassdoor, 2024). Para este caso, tomamos la cifra más baja ya que vamos a suponer que se cuenta con un desarrollador software junior. Además, al salario base debemos agregar un 28'30% correspondiente a la aportación de la empresa a la Seguridad Social (Seguridad Social, 2024). Esto significa que el coste total para la empresa asciende a 28.226€ anuales. Para calcular el coste del desarrollador durante el tiempo que dura el TFG, se parte de una jornada laboral completa de 8 horas al día y un total de 252 días laborables anuales (descontando fines de semana y festivos) tal y como indica la página: (Working Days, 2024). Esto da un total de 2.016 horas de trabajo al año. Por lo tanto, el coste estimado final para un desarrollador software sería de 4.900€.

En cuanto al **hardware** empleado, se utilizará un portátil HP Pavilion 15 valorado en 850€. Teniendo en cuenta que la vida útil estimada de este equipo es de 4 años (48 meses), se ha calculado su coste amortizado mensual. Dividiendo el valor total del ordenador entre el número de meses de vida útil, obtenemos un coste de 17'71€ al mes. Dado que la duración estimada del proyecto es de aproximadamente 4 meses, el coste total del hardware durante este periodo asciende a 70'83€.

Con respecto al **software**, se empleará para la comunicación con la tutora la aplicación de Microsoft Teams que cuesta 3'70€/mes (Microsoft, 2024). Para la creación de los bocetos de las interfaces se utilizará la herramienta Moqups, que tiene un coste de 9€ al mes (Moqups, 2024). Dado que se requerirán 4 meses de uso, el coste total de esta herramienta asciende a 36€. Además, para la elaboración de diagramas y otros elementos visuales se utilizará Astah Professional, cuyo precio es de aproximadamente 11'15€ al mes (Astah, 2024). Para los 4 meses necesarios para el desarrollo del TFG, el coste total de Astah Professional será de 44'60€. Así que el coste total del software necesario para el proyecto es de aproximadamente 80'60€.

Otro de los gastos que se deben tener en cuenta para este proyecto son los relacionados con la **infraestructura de la oficina**. Para una oficina pequeña en Valladolid, dedicada a actividades de informática, los gastos mensuales típicos se pueden desglosar de la siguiente manera:

- **Alquiler del local.** Los precios pueden variar según la ubicación y tamaño de la oficina. Para este caso, se tomará un alquiler básico de unos 300€ al mes.
- **Electricidad.** El coste de electricidad, dependiendo del uso de equipos informáticos y aire acondicionado, puede oscilar entre 100€ y 200€ al mes. Se tomará una estimación media de 150€ mensuales.

- **Agua.** Este gasto suele ser menor en oficinas, alrededor de 20€ al mes.
- **Internet y teléfono.** Un plan de internet de alta velocidad y línea telefónica puede costar entre 50€ y 100€ al mes. Tomaremos el valor de 75€.
- **Otros gastos.** Estos incluyen limpieza, mantenimiento y otros servicios, estimados en unos 50€ al mes.

En total, los gastos mensuales de oficina se estiman en 595€ al mes. Dado que la duración del proyecto es de 4 meses, el coste total relacionado con la oficina asciende a 2.380€. Dicha información proviene de las fuentes (Idealista, 2024) y (Fotocasa, 2024).

El coste total del proyecto hasta este punto es de 7.431,43€. Sin embargo, para evitar que este coste se sobrepase debido a posibles extensiones del tiempo de desarrollo o al uso de herramientas no planificadas, se incrementará el presupuesto total en un 20%. Así, el coste final del proyecto se establece en **8.917,71€**.

Nota aclaratoria: Este cálculo de costes es una estimación interna para el seguimiento del proyecto y no constituye un presupuesto formal ni un coste que se pueda presentar a un cliente.

En la Tabla 2.14 se muestra el resumen con la estimación de los costes simulados.

Concepto	**Precio**	**Tiempo**	**Total**
Desarrollador software	28.226€/año	350 horas	4.900,00€
Portátil	850€	350 horas	70,83€
Licencia Microsoft Teams	3,70€/mes	350 horas	14,80€
Licencia Moqups	9€/mes	350 horas	36,00€
Licencia Astah Professional	11,15€/mes	350 horas	44,60€
Oficina	595€/mes	350 horas	2.380,00€
Total			7.446,23€
Total extendido (+20%)			**8.935,50€**

Tabla 2.14. Estimación coste simulado

2.10.2. Coste real

Dado que este Trabajo Fin de Grado tiene un fin académico, el equipo de desarrollo, que en este caso es únicamente la autora de este proyecto, no recibirá ningún tipo de salario. Por lo tanto, no se incluirán costes asociados a recursos humanos.

Respecto al hardware, se utilizará el portátil HP Pavilion 15, tal como se mencionó en el coste simulado. Por ende, el primer coste a considerar será de 70,83€ que cubrirá la duración aproximada del trabajo.

En cuanto al software, aunque previamente se estimaron unos costes asociados a Teams, Moqups y Astah Professional, realmente no generarán ningún gasto. La versión de Moqups que se empleará es la gratuita a pesar de sus limitaciones, y tanto Teams como Astah Professional será utilizada sin coste ya que cuento con una licencia de estudiante proporcionada por la Universidad de Valladolid y la Escuela de Ingeniería Informática.

Por último, el proyecto se desarrollará principalmente desde mi casa, lo que implica que los costes de electricidad, internet y otros servicios no se tendrán en cuenta, ya que están cubiertos por entidades externas al proyecto. Así que el coste real es de **70,83€**.

En la Tabla 2.15 se muestra el resumen con estos nuevos costes.

Concepto	**Precio**	**Tiempo**	**Total**
Desarrollador software	0€/año	350 horas	0€
Portátil	850€	350 horas	70,83€
Licencia Microsoft Teams	0€/mes	350 horas	0€
Licencia Moqups	0€/mes	350 horas	0€
Licencia Astah Professional	0€/mes	350 horas	0€
Total			**70,83€**

Tabla 2.15. Estimación coste real

Capítulo 3
Análisis

En este capítulo se lleva a cabo la identificación y definición de los casos de uso, se incluye una matriz de correspondencia entre los casos de uso y los requisitos funcionales y se presenta el modelo de dominio, tanto en una versión inicial como final, como resultado de la fase de modelado conceptual. Además, se analizan qué objetos del dominio se comportan como máquinas de estado y, en su caso, se modelan mediante diagramas de estados.

3.1. Casos de uso

Los casos de uso son descripciones detalladas de cómo los usuarios interactúan con un sistema para lograr un objetivo específico. Sirven como herramienta fundamental para capturar los requisitos funcionales del sistema (ver apartado 2.3. Requisitos funcionales) y como una manera de describir más concretamente las interacciones necesarias para cumplir con esos requisitos. De este modo, los casos de uso permiten visualizar las diversas interacciones que los actores (ver apartado 2.2. Actores del sistema) tendrán con la aplicación, facilitando el análisis de cómo se deben implementar dichas funcionalidades.

En esta sección se listan los casos de uso que corresponden a cada rol en el sistema, destacando que los actores principales son el vendedor y el comprador. Sin embargo, al definir los casos de uso y observar que algunas funcionalidades son muy similares para ambos roles (como el inicio de sesión), se ha introducido un actor generalización, llamado "usuario". Este actor representa tanto a compradores como a vendedores en aquellas funcionalidades compartidas, simplificando el análisis y evitando redundancias.

Vendedor:

- **CU01.** Registrar comercio
- **CU02.** Modificar información del comercio
- **CU03.** Ver pedidos
- **CU04.** Registrar nuevo pedido
- **CU05.** Actualizar estado de un pedido
- **CU06.** Ver productos
- **CU07.** Añadir producto
- **CU08.** Eliminar producto
- **CU09.** Modificar producto
- **CU10.** Añadir área de entrega

CU11. Eliminar área de entrega

CU12. Publicar nuevo aviso

Comprador:

CU13. Registrarse

CU14. Modificar información personal

CU15. Hacer nuevo pedido

CU16. Consultar productos

Usuario:

CU17. Iniciar sesión

CU18. Cerrar sesión

CU19. Ver historial de pedidos

Desde la Tabla 3.1 hasta la Tabla 3.19 se detallan concretamente cada uno de los casos de uso recién definidos. Cabe mencionar que, en esta versión que se va a presentar, el sistema no implementa un proceso de confirmación de correo electrónico tras el registro. Sin embargo, para líneas de trabajo futuras, se deberá introducir un mecanismo de verificación de cuenta, lo que implicará gestionar dos estados para el usuario: 'sin verificar' y 'verificado'. Esto garantizará mayor seguridad y evitará registros con correos no válidos.

CU01	Registrar comercio
Actor	Vendedor
Descripción	El actor vendedor registra su comercio en el sistema.
Precondición	-
Postcondición	El actor vendedor ha registrado su comercio en el sistema.
Secuencia normal	1. El actor vendedor solicita registrar su comercio en el sistema. 2. El sistema pide el nombre del comercio, la dirección (localidad, provincia, código postal y calle), el correo electrónico, el teléfono, el CIF, la categoría del comercio, el nombre y apellidos del propietario y la contraseña (2 veces). 3. El actor vendedor introduce todos los datos. 4. El sistema comprueba que todos los datos son correctos. 5. El sistema registra la empresa y el caso de uso finaliza.

Excepciones	4a. Si hay algún campo vacío, el sistema informa de ello y vuelve al paso 2. 4b. El correo electrónico tiene un formato inválido, es decir, no incluye el símbolo "@" y al menos una letra después de dicho símbolo, el sistema informa de ello y vuelve al paso 2. 4c. El correo electrónico ya está asociado a otro comercio registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4d. El teléfono ya está asociado a otro usuario registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4e. El CIF no es válido, es decir, no cumple con las reglas establecidas oficialmente para este identificador fiscal en España. Estas reglas incluyen que el CIF debe comenzar con una letra válida que identifica el tipo de entidad, seguida de siete dígitos y un carácter de control (que puede ser numérico o alfabético, dependiendo del tipo de entidad). Además, el carácter de control debe ser calculado correctamente según un algoritmo definido que valida los números del CIF. El sistema informa del error y vuelve al paso 2. 4f. El CIF ya está asociado a otro comercio registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4g. La contraseña no tiene un formato válido, es decir, no contiene como mínimo 8 caracteres, al menos una mayúscula, una minúscula y un número, por lo que el sistema informa de ello y vuelve al paso 2. 4h. Las contraseñas no coinciden, el sistema informa de ello y vuelve al paso 2.

Tabla 3.1. Descripción CU01 "Registrar comercio"

CU02	Modificar información del comercio
Actor	Vendedor
Descripción	El actor vendedor modifica la información registrada sobre su comercio e información personal, pero únicamente el teléfono, nombre del comerciante y la contraseña.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	La información referente a su comercio y/o a su información personal ha sido modificada.
Secuencia normal	1. El actor vendedor solicita modificar la información de su comercio. 2. El sistema muestra la información registrada sobre su comercio y la referente al comerciante. 3. El actor vendedor introduce un nuevo valor para el teléfono, un nuevo nombre para el vendedor y el nuevo par de contraseñas.

	4. El sistema comprueba la validez de cada dato introducido. 5. El sistema registra y actualiza con la nueva información y el caso de uso finaliza.
Excepciones	3a. El actor vendedor cancela y el caso de uso queda sin efecto. 4a. El teléfono introducido se corresponde con un teléfono ya registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4b. La contraseña no cumple el formato esperado, es decir, no contiene como mínimo 8 caracteres, al menos una mayúscula, una minúscula y un número. Por tanto, el sistema informa de ello y vuelve al paso 2. 4c. El nuevo par de contraseñas no coincide, el sistema informa de ello y vuelve al paso 2.

Tabla 3.2. Descripción CU02 " Modificar información del comercio "

CU03	Ver pedidos
Actor	Vendedor
Descripción	El actor vendedor visualiza los pedidos recibidos.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se muestra una lista con los pedidos recibidos.
Secuencia normal	1. El actor vendedor solicita ver los nuevos pedidos. 2. El sistema muestra los pedidos organizados por localidades y clientes.
Excepciones	-

Tabla 3.3. Descripción CU03 "Ver pedidos"

CU04	Registrar nuevo pedido
Actor	Vendedor
Descripción	Cuando el actor vendedor recibe una llamada de un cliente que desea realizar un pedido, el vendedor tendrá la opción de registrar manualmente dicho pedido, permitiendo así el registro de pedidos telefónicos en el sistema.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha registrado un nuevo pedido en el sistema con pago en efectivo y entrega a domicilio.
Secuencia normal	1. El actor vendedor solicita registrar un nuevo pedido manualmente. 2. El sistema pide el teléfono del cliente, su nombre y apellidos y su dirección de domicilio. 3. El actor vendedor introduce los datos solicitados. 4. El sistema muestra la lista de productos, y solicita los productos y las cantidades correspondientes.

	5. El actor vendedor selecciona los productos y sus cantidades correspondientes. 6. El sistema pide confirmación. 7. El actor vendedor confirma. 8. El sistema registra el nuevo pedido y el caso de uso finaliza.
Excepciones	3a, 5a, 7a. El actor vendedor cancela y el caso de uso queda sin efecto. 3b. El actor vendedor introduce únicamente el teléfono del cliente. El sistema comprueba si el teléfono introducido corresponde con un comprador registrado en el sistema y si es así continúa por el paso 4. En caso contrario, vuelve al paso 2.

Tabla 3.4. Descripción CU04 "Registrar nuevo pedido"

CU05	Actualizar estado de un pedido
Actor	Vendedor
Descripción	El actor vendedor actualiza el estado de un pedido. Si está en estado "solicitado" se puede pasar a estado "preparado" o "entregado", y si está en estado "preparado" sólo lo puede pasar a estado "entregado".
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	El sistema ha actualizado el nuevo estado del pedido.
Secuencia normal	1. El actor vendedor solicita actualizar el estado de un pedido. 2. El sistema muestra una lista con los pedidos solicitados y preparados. 3. El actor vendedor cambia el estado de uno de los pedidos. 4. El sistema registra el nuevo estado y el caso de uso finaliza.
Excepciones	-

Tabla 3.5. Descripción CU05 "Actualizar estado de un pedido"

CU06	Ver productos
Actor	Vendedor
Descripción	El sistema muestra los productos del comercio que gestiona el actor vendedor.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	El sistema muestra los productos del comercio.
Secuencia normal	1. El actor vendedor solicita ver todos los productos que gestiona. 2. El sistema muestra los productos gestionados por el comercio.
Excepciones	-

Tabla 3.6. Descripción CU06 "Ver productos"

CU07	Añadir producto
Actor	Vendedor
Descripción	El actor vendedor añade un nuevo producto para que esté a la venta.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha registrado un nuevo producto en el sistema asociado al comercio correspondiente.
Secuencia normal	1. El actor vendedor solicita añadir un nuevo producto. 2. El sistema pide el nombre del producto, el precio, el precio por unidad y la lista de ingredientes. 3. El actor vendedor introduce los datos. 4. El sistema comprueba la validez de los datos. 5. El sistema registra el nuevo producto y el caso de uso finaliza.
Excepciones	3a. El actor vendedor cancela y el caso de uso queda sin efecto. 4a. Hay campos vacíos, el sistema informa de ello y vuelve al paso 2. 4b. El nombre del producto supera los 50 caracteres, el sistema informa de ello y se vuelve al paso 2. 4c. El precio introducido es menor o igual que 0, el sistema informa del error y se vuelve al paso 2. 4d. El precio por unidad es menor o igual que 0, el sistema informa de ello y se vuelve al paso 2. 4e. La lista de ingrediente supera los 500 caracteres, el sistema informa de ello y se vuelve al paso 2.

Tabla 3.7. Descripción CU07 "Añadir producto"

CU08	Eliminar producto
Actor	Vendedor
Descripción	El actor vendedor elimina un producto registrado de su comercio.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha eliminado el producto del sistema.
Secuencia normal	1. El actor vendedor solicita eliminar un producto. 2. El sistema muestra una lista con todos los productos registrados asociados a su comercio. 3. El actor vendedor elimina uno de ellos. 4. El sistema pide confirmación. 5. El actor vendedor confirma la eliminación. 6. El sistema lo registra y el caso de uso finaliza.
Excepciones	3a, 5a. El actor vendedor cancela y el caso de uso queda sin efecto.

Tabla 3.8. Descripción CU08 "Eliminar producto"

CU09	Modificar producto
Actor	Vendedor
Descripción	Modificar la información de un producto registrado.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha modificado la información relativa a uno de los productos.
Secuencia normal	1. El actor vendedor solicita modificar la información de uno de sus productos. 2. El sistema muestra la información actualmente registrada para el producto seleccionado. 3. El actor vendedor introduce un nuevo nombre, precio, precio por unidad y la lista de ingredientes. 4. El sistema comprueba la validez de los datos introducidos. 5. El sistema actualiza la nueva información del producto y el caso de uso finaliza.
Excepciones	3a. El actor vendedor cancela y el caso de uso queda sin efecto. 4a. Hay campos vacíos, el sistema informa de ello y vuelve al paso 2. 4b. El nombre del producto supera los 50 caracteres, el sistema informa de ello y se vuelve al paso 2. 4c. El precio introducido es menor o igual que 0, el sistema informa del error y se vuelve al paso 2. 4d. El precio por unidad es menor o igual que 0, el sistema informa de ello y se vuelve al paso 2. 4e. La lista de ingrediente supera los 500 caracteres, el sistema informa de ello y se vuelve al paso 2.

Tabla 3.9. Descripción CU09 "Modificar producto"

CU10	Añadir área de entrega
Actor	Vendedor
Descripción	El actor vendedor tiene la opción de agregar localidades en las que desea vender sus productos.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha registrado una nueva localidad donde el actor vendedor comenzará a ofrecer y vender sus productos.
Secuencia normal	1. El actor vendedor solicita añadir una nueva área de entrega. 2. El sistema muestra las localidades que están cerca de su comercio y pide que se seleccione una o varias de ellas. 3. El actor vendedor selecciona una o varias localidades. 4. El sistema pide confirmación. 5. El actor vendedor confirma. 6. El sistema registra las localidades y el caso de uso finaliza.

Excepciones	3a, 5a. El actor vendedor cancela y el caso de uso queda sin efecto.

Tabla 3.10. Descripción CU10 "Añadir área de entrega"

CU11	Eliminar área de entrega
Actor	Vendedor
Descripción	El actor vendedor tiene la opción de eliminar localidades en las que vende sus productos.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se han eliminado una localidad en la que el vendedor hacía repartos.
Secuencia normal	1. El actor vendedor solicita eliminar un área de entrega. 2. El sistema muestra una lista con las localidades en las que actualmente vende y pide eliminar una. 3. El actor selecciona la localidad que se desea eliminar. 4. El sistema pide confirmación. 5. El actor vendedor confirma la eliminación de la localidad. 6. El sistema registra la eliminación y el caso de uso finaliza.
Excepciones	3a, 5a. El actor vendedor cancela y el caso de uso queda sin efecto.

Tabla 3.11. Descripción CU11 "Eliminar área de entrega"

CU12	Publicar nuevo aviso
Actor	Vendedor
Descripción	El actor vendedor tiene la opción de publicar avisos en las localidades que elija, proporcionando información sobre los días de reparto, posibles ausencias o cualquier otro detalle relevante relacionado con la entrega de sus productos.
Precondición	El actor vendedor debe haber iniciado sesión previamente.
Postcondición	Se ha registrado el aviso en el sistema.
Secuencia normal	1. El actor vendedor solicita publicar un nuevo aviso. 2. El sistema muestra las localidades en las que reparte. 3. El actor vendedor selecciona la localidad en la que se desea que se publique el aviso. 4. El sistema pide el título y el mensaje del aviso. 5. El actor vendedor introduce los datos. 6. El sistema comprueba la validez de los datos introducidos. 7. El sistema lo registra y el caso de uso finaliza.

Excepciones	3a, 5a. El actor vendedor cancela y el caso de uso queda sin efecto. 6a. Hay campos vacíos, el sistema informa de ello y vuelve al paso 4. 6b. El título del aviso supera los 100 caracteres, el sistema informa de ello y vuelve al paso 4. 6c. El mensaje del aviso supera los 500 caracteres, el sistema informa de ello y vuelve al paso 4.

Tabla 3.12. Descripción CU12 "Publicar nuevo aviso"

CU13	Registrarse
Actor	Comprador
Descripción	El actor comprador se registra en el sistema.
Precondición	-
Postcondición	El actor comprador queda registrado en el sistema.
Secuencia normal	1. El actor comprador solicita registrarse en el sistema. 2. El sistema pide el nombre, apellidos, teléfono, correo electrónico, dirección (localidad, provincia, código postal y calle) y la contraseña (dos veces). 3. El actor comprador introduce los datos. 4. El sistema comprueba que los datos son correctos. 5. El sistema registra al nuevo usuario y el caso de uso finaliza.
Excepciones	3a. El actor comprador cancela y el caso de uso queda sin efecto. 4a. Si hay algún campo vacío, el sistema informa de ello y vuelve al paso 2. 4b. El correo electrónico tiene un formato inválido, es decir, no incluye el símbolo "@" y al menos una letra después de dicho símbolo, el sistema informa de ello y vuelve al paso 2. 4c. El correo electrónico ya está asociado a otro usuario registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4d. El teléfono introducido corresponde con el de un usuario ya registrado en el sistema, el sistema informa de ello y vuelve al paso 1. 4e. La contraseña no tiene un formato válido, es decir, no contiene como mínimo 8 caracteres, al menos una mayúscula, una minúscula y un número, por lo que el sistema informa de ello y vuelve al paso 2. 4f. Las contraseñas no coinciden, el sistema informa de ello y vuelve al paso 2.

Tabla 3.13. Descripción CU13 "Registrarse"

CU14	Modificar información personal
Actor	Comprador
Descripción	El actor comprador tiene la posibilidad de modificar toda la información que proporcionó al momento de su registro.
Precondición	El actor comprador debe haber iniciado sesión previamente.
Postcondición	La información personal del actor comprador se ha actualizado.
Secuencia normal	1. El actor comprador solicita modificar su información personal. 2. El sistema muestra la información actual registrada (nombre, apellidos, teléfono, correo electrónico, dirección: localidad, provincia, código postal y calle, y la contraseña). 3. El actor comprador introduce nuevos valores para el nombre, apellidos, teléfono, correo electrónico y dirección. 4. El sistema comprueba la validez de la información introducida. 5. El sistema registra los cambios y el caso de uso finaliza.
Excepciones	3a. El actor comprador cancela y el caso de uso queda sin efecto. 4a. Si hay algún campo vacío, el sistema informa de ello y vuelve al paso 2. 4b. El correo electrónico tiene un formato inválido, es decir, no incluye el símbolo "@" y al menos una letra después de dicho símbolo, el sistema informa de ello y vuelve al paso 2. 4c. El correo electrónico ya está asociado a otro comercio registrado en el sistema, el sistema informa de ello y vuelve al paso 2. 4d. Si el teléfono introducido corresponde con el de otro usuario ya registrado en el sistema, el sistema informa de ello y vuelve al paso 3. 4e. La contraseña no tiene un formato válido, es decir, no contiene como mínimo 8 caracteres, al menos una mayúscula, una minúscula y un número, por lo que el sistema informa de ello y vuelve al paso 2. 4f. Las contraseñas no coinciden, el sistema informa de ello y vuelve al paso 2.

Tabla 3.14. Descripción CU14 "Modificar información personal"

CU15	Hacer nuevo pedido
Actor	Comprador
Descripción	El comprador inicia un nuevo pedido.
Precondición	-
Postcondición	Se ha creado un nuevo pedido
Secuencia normal	1. Se ejecuta el CU16 "Consultar productos".

	2. El actor comprador selecciona los productos que desea comprar, especifica la cantidad de cada uno y los añade a la cesta. 3. El sistema pide confirmación de la compra. 4. El actor comprador confirma que va a realizar la compra. 5. El sistema muestra la información personal del comprador (nombre, dirección, teléfono) y solicita la forma de pago: efectivo o PayPal. 6. El actor comprador selecciona como forma de pago la PayPal. 7. El sistema solicita el tipo de entrega: a domicilio o al locker. 8. El actor comprador elige la entrega a domicilio. 9. El sistema pide confirmación. 10. El actor comprador confirma. 11. El sistema redirige a la pasarela de pagos. 12. El sistema registra el nuevo pedido y el caso de uso finaliza.
Excepciones	4a, 6b, 8b, 10a. El actor comprador cancela y el caso de uso queda sin efecto. 6a. El actor comprador elige la opción de pago en efectivo y el sistema selecciona automáticamente el tipo de entrega "a domicilio". El caso de uso continúa por el paso 9 y después de ejecutarse el paso 10, se redirige al paso 12.

Tabla 3.15. Descripción CU15 "Hacer nuevo pedido"

CU16	Consultar productos
Actor	Comprador
Descripción	El sistema muestra los productos de los comercios al comprador.
Precondición	-
Postcondición	Se muestra una lista de todos los productos.
Secuencia normal	1. El actor comprador solicita ver todos los productos de los comercios. 2. El sistema muestra únicamente los productos de los comercios que venden en la dirección del comprador y el caso de uso finaliza.
Excepciones	-

Tabla 3.16. Descripción CU16 "Consultar productos"

CU17	Iniciar sesión
Actor	Usuario
Descripción	El usuario inicia sesión en el sistema.
Precondición	El usuario debe estar registrado en el sistema.
Postcondición	El usuario ha iniciado sesión.
Secuencia normal	1. El actor usuario solicita iniciar sesión en el sistema. 2. El sistema pide el correo electrónico y la contraseña. 3. El actor usuario introduce los datos.

	4. El sistema comprueba que los datos corresponden con un usuario registrado, según el tipo de usuario muestra las funcionalidades que puede realizar en el sistema y el caso de uso finaliza.
Excepciones	4a. Si hay algún campo vacío, el sistema informa de ello y vuelve al paso 2. 4b. El correo electrónico tiene un formato inválido, es decir, no incluye el símbolo "@" y al menos una letra después de dicho símbolo, el sistema informa de ello y vuelve al paso 2. 4c. El correo electrónico y la contraseña no coinciden con un usuario registrado, el sistema informa de ello y vuelve al paso 2.

Tabla 3.17. Descripción CU17 "Iniciar sesión"

CU18	Cerrar sesión
Actor	Usuario
Descripción	El usuario cierra sesión.
Precondición	El usuario debe haber iniciado sesión previamente.
Postcondición	El usuario ha cerrado sesión.
Secuencia normal	1. El actor usuario solicita cerrar sesión. 2. El sistema elimina la información temporal, cierra sesión y el caso de uso finaliza.
Excepciones	-

Tabla 3.18. Descripción CU18 "Cerrar sesión"

CU19	Ver historial de pedidos
Actor	Usuario
Descripción	El sistema muestra una lista con los pedidos.
Precondición	El usuario debe haber iniciado sesión previamente.
Postcondición	El usuario ve una lista con los pedidos.
Secuencia normal	1. El actor usuario solicita ver los pedidos. 2. El sistema muestra una lista con los pedidos realizados o recibidos en función del tipo de usuario que hace la solicitud.
Excepciones	-

Tabla 3.19. Descripción CU19 "Ver historial de pedidos"

3.1.1. Matriz de correspondencia entre CUs y RFs

Esta matriz tiene como objetivo establecer una relación clara entre los casos de uso y los requisitos funcionales del sistema. De esta manera, se garantiza que cada requisito funcional esté cubierto por al menos un caso de uso, asegurando que el sistema propuesto cumpla con todas las funcionalidades necesarias.

En la Tabla 3.20 se muestra la matriz.

	CU 01	CU 02	CU 03	CU 04	CU 05	CU 06	CU 07	CU 08	CU 09	CU 10	CU 11	CU 12	CU 13	CU 14	CU 15	CU 16	CU 17	CU 18	CU 19
RF01	x																		
RF02																	x		
RF03			x																
RF04					x														
RF05				x															
RF06							x												
RF07								x											
RF08									x										
RF09		x																	
RF10																			x
RF11										x	x								
RF12												x							
RF13																		x	
RF14													x						
RF15																	x		
RF16																		x	
RF17														x					
RF18																x			
RF19															x				
RF20															x				
RF21																			x
RF22															x				

Tabla 3.20. Matriz de correspondencia CUs-RFs

3.1.2. Modelo de casos de uso

En la Figura 3.1,se presenta un diagrama UML de casos de uso donde se representan a los actores y sus interacciones con el sistema. Uno de los actores es "Usuario", que generaliza a "Comprador" y "Vendedor", lo que permite representar de forma clara las interacciones comunes a ambos roles, mientras que cada especialización se encarga de sus propias interacciones específicas. También se ha

incluido un actor externo llamado "Pasarela de pagos" que se relaciona con el sistema únicamente cuando se realiza un pedido, concretamente con la acción de realizar un pago electrónico, donde el sistema delega la validación y autorización de las transacciones en esta entidad externa.

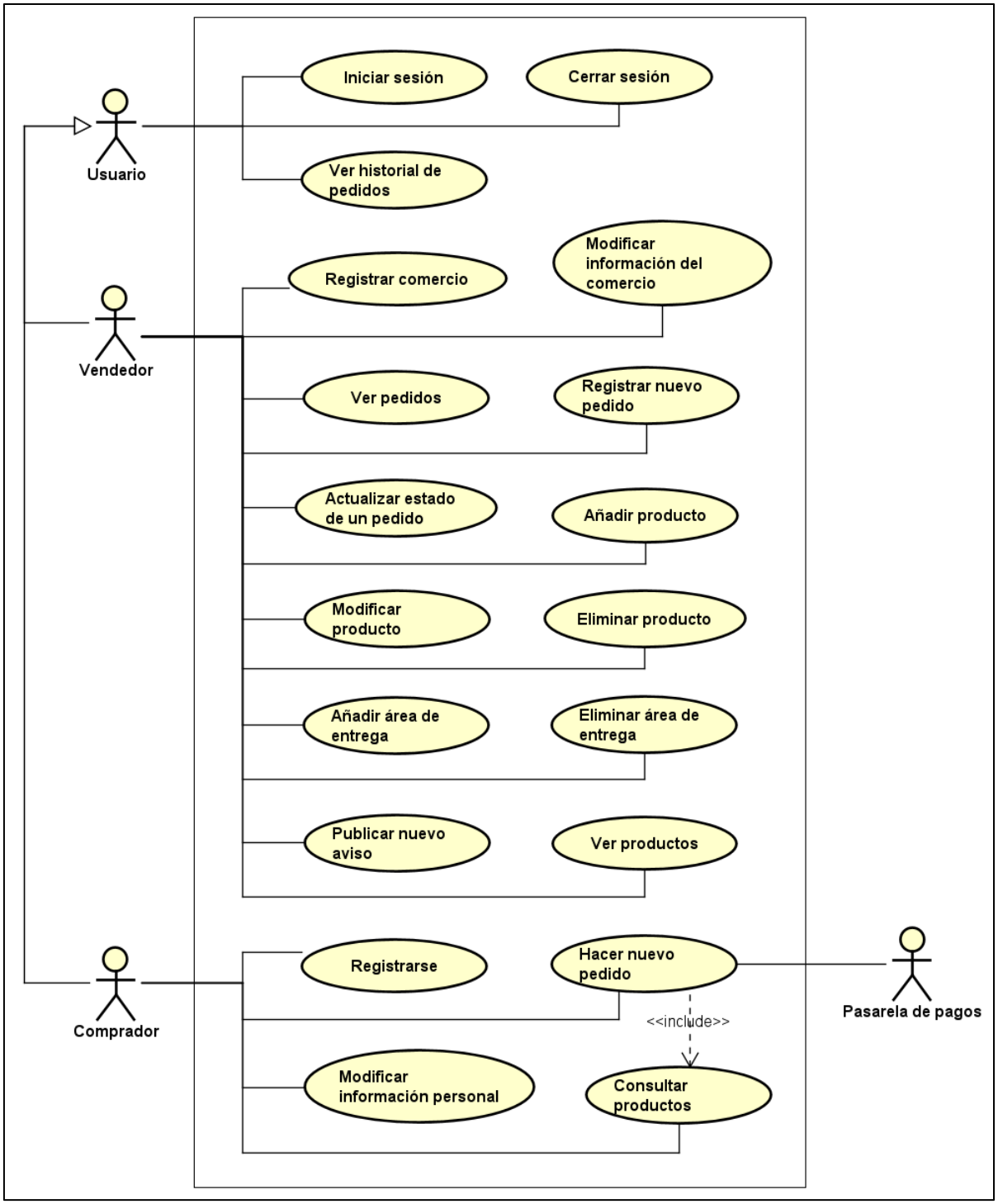

Figura 3.1. Diagrama de casos de uso

3.2. Modelo del dominio

El modelo de dominio es una representación conceptual del sistema que describe las principales entidades, atributos y relaciones. Tiene como objetivo proporcionar una visión clara de la estructura del sistema, identificando los elementos clave que interactúan dentro del mismo.

En la Figura 3.2, se presenta el resultado del modelado conceptual mediante un diagrama de clases UML.

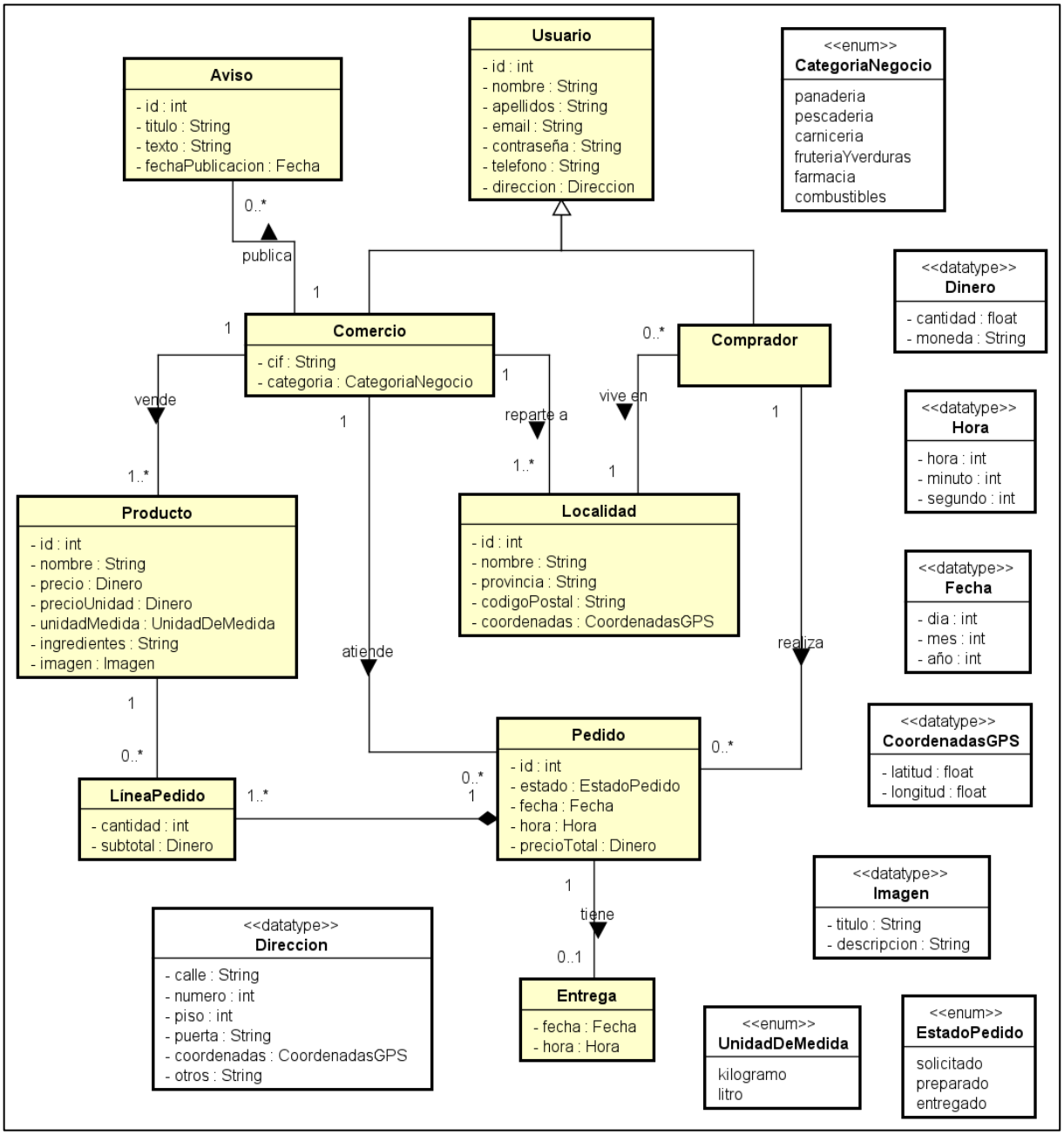

Figura 3.2. Modelo del dominio

3.3. Modelado de estados de un pedido

Este tipo de diagrama muestra los diferentes estados por los que puede pasar un objeto en el sistema a lo largo de su ciclo de vida, así como los eventos o acciones que provocan las transiciones entre estos estados. En este caso, se identifica el pedido como el único objeto que se comporta como una máquina de estados, por lo que su funcionamiento se especifica mediante un diagrama de estados UML. En la Figura 3.3 se presenta dicho modelado, donde un pedido puede pasar por las fases de "solicitado", "preparado" y "entregado". Este diagrama es útil para visualizar y entender las posibles transiciones y las acciones que las desencadenan, asegurando una correcta gestión del flujo de los pedidos en el sistema.

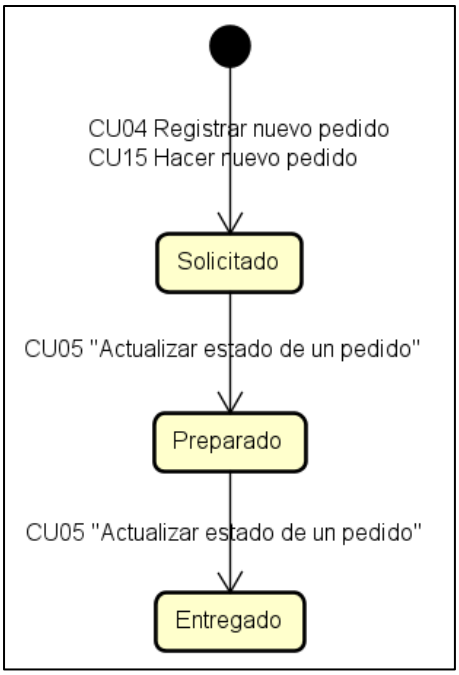

Figura 3.3. Diagrama de estados de un pedido

Capítulo 4
Tecnologías utilizadas

En este capítulo se presenta un desglose de las diferentes herramientas y tecnologías que se han utilizado a lo largo del desarrollo del proyecto. Se detallan las herramientas utilizadas para la comunicación con la tutora, las empleadas para el análisis, diseño y documentación del sistema, para la gestión y control del proyecto y se describen las tecnologías utilizadas para el desarrollo del software.

4.1. Herramientas para la comunicación

4.1.1. Microsoft Teams

Es una plataforma de comunicación y colaboración desarrollada por Microsoft. Está diseñada para facilitar el trabajo en equipo, permitiendo la mensajería instantánea, videollamadas, reuniones, y la integración con otras herramientas de Office 365 como Word, Excel o PowerPoint. Además, ofrece la opción de compartir archivos, colaborar en documentos en tiempo real y organizar tareas a través de canales dedicados para diferentes proyectos o equipos. Esta herramienta se ha convertido en una opción popular tanto en entornos empresariales como académicos por su versatilidad y facilidad de uso (Microsoft Support, 2020).

4.2. Herramientas para el análisis, diseño y documentación

4.2.1. Astah Professional

Es una herramienta de modelado visual orientada a la creación de diagramas UML (Unified Modeling Language), lo que la convierte en una opción ideal para el análisis (ver Capítulo 3) y diseño de software (ver Capítulo 5). Esta aplicación permite a los desarrolladores crear diagramas de clases, casos de uso, secuencia, actividad y otros, facilitando la planificación y documentación de sistemas complejos. Además, ofrece una interfaz intuitiva y soporte para la exportación de diagramas en diferentes formatos, lo que la convierte en una herramienta útil tanto para la fase de diseño como para la documentación técnica de proyectos de software (Astah, 2011).

4.2.2. Moqups

Moqups es una herramienta en línea que permite la creación de wireframes, maquetas y prototipos interactivos. Está diseñada para facilitar el diseño visual de interfaces de usuario, permitiendo a los usuarios esbozar rápidamente ideas y estructurar la navegación y funcionalidad de una aplicación o sitio web. Con su interfaz intuitiva, Moqups permite a diseñadores y desarrolladores colaborar en tiempo real, realizar ajustes visuales y obtener una representación clara de la estructura de la aplicación antes de la implementación. Es especialmente útil en la fase de diseño para validar y comunicar ideas a través de bocetos visuales (Virtual Project Manager, 2022).

4.2.3. Microsoft Word

Microsoft Word es un procesador de textos que permite crear, editar y formatear documentos con una gran variedad de opciones de personalización. Es parte de la suite de Office y es especialmente útil para la elaboración de trabajos escritos, informes y otros documentos textuales. Word ofrece una interfaz intuitiva y herramientas de colaboración que permiten a los usuarios compartir documentos y trabajar en ellos en tiempo real.

En el contexto de este proyecto, Word ha sido fundamental para la redacción y estructuración de mi TFG. Además, gracias a su integración con OneDrive (ver apartado 4.2.4. OneDrive), he podido compartir el progreso del documento con mi tutora de manera ágil y efectiva, permitiendo una colaboración y revisión continua del trabajo.

4.2.4. OneDrive

Es un servicio de almacenamiento en la nube ofrecido por Microsoft que permite a los usuarios guardar, sincronizar y acceder a archivos desde cualquier dispositivo conectado a internet. Además, facilita la colaboración en tiempo real, ya que varios usuarios pueden trabajar simultáneamente en documentos compartidos.

Para mi Trabajo Fin de Grado, OneDrive ha sido fundamental para almacenar el proyecto en la nube, lo que me ha permitido compartirlo fácilmente con mi tutora y que pudiera proporcionarme feedback sobre mi progreso. Además, esto contribuye a la implementación del plan de mitigación de los riesgos mencionados en la Tabla 2.8 y la Tabla 2.9.

4.3. Herramientas para la gestión y control del proyecto

4.3.1. Git

Git es un sistema de control de versiones distribuido que permite gestionar y hacer un seguimiento de los cambios en el código fuente de un proyecto. Ofrece funcionalidades clave como la gestión de ramas, la fusión de cambios y la posibilidad de revertir versiones anteriores del código. Aunque está diseñado para el trabajo en equipo, también es útil en proyectos individuales, ya que permite mantener un historial de cambios organizado, recuperar versiones anteriores y almacenar el código tanto en local como en un repositorio remoto.

En este proyecto se utiliza GitLab, una plataforma que proporciona una interfaz web para la gestión de repositorios Git y que facilita su uso mediante herramientas adicionales, como integración y entrega continua (CI/CD), gestión de proyectos y revisiones de código. Sin embargo, GitLab no es una tecnología en sí misma, sino una plataforma que implementa Git y añade funcionalidades extra para mejorar su uso.

El uso de Git, junto con un repositorio remoto en GitLab, permite mantener una copia segura del código en caso de fallos en el equipo local. Esto es especialmente útil para mitigar riesgos como el R07: Pérdida de información y R08: Fallos de hardware, ya que garantiza la disponibilidad del historial de cambios y la posibilidad de restaurar versiones anteriores si es necesario.

Fuentes: (Atlassian, 2020) y (Hostinger, 2025).

4.3.2. GitLab Issue Board

GitLab Issue Board es una herramienta de gestión de tareas integrada en GitLab que permite a los equipos organizar, priorizar y realizar seguimiento de sus tareas y problemas de manera visual. Aunque está basada en un sistema de tableros de tareas similar a Kanban, donde cada tarea (o "issue") se representa como una tarjeta que puede moverse entre distintas columnas, este tablero es también altamente adaptable a otros métodos, como Scrum. En el contexto de Scrum, las columnas pueden representar las etapas del flujo de trabajo dentro de un sprint, como "Por hacer", "En progreso", "Revisión" y "Terminado", facilitando así la gestión del backlog y el seguimiento del progreso durante cada ciclo de sprint.

Cada issue en el tablero puede contener descripciones detalladas, etiquetas, asignaciones, fechas límite y comentarios, lo que facilita la colaboración y el seguimiento de los avances en el equipo. Además, GitLab Issue Board permite personalizar las columnas y automatizar el movimiento de las tarjetas, por ejemplo, mediante etiquetas o cambios de estado, lo que ayuda a reflejar automáticamente el progreso de las tareas (GitLab, 2020).

4.4. Tecnologías para el desarrollo del proyecto

4.4.1. Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft que, aunque ligero, es muy potente. Soporta una amplia gama de lenguajes de programación y tecnologías, y ofrece características como depuración, control de versiones, y un sistema de extensiones que permite personalizar la experiencia de desarrollo. Es especialmente popular entre los desarrolladores web y de aplicaciones debido a su facilidad de uso y capacidades de integración (OpenWebinars, 2022).

4.4.2. Angular

Angular es un framework de desarrollo de aplicaciones web desarrollado por Google. Permite crear aplicaciones web de una sola página (SPA) de manera eficiente, utilizando TypeScript, un superconjunto de JavaScript que añade tipos estáticos. Angular proporciona una estructura robusta para desarrollar aplicaciones escalables, facilitando el manejo del DOM, la gestión del estado, y la creación de componentes reutilizables.

Para el desarrollo de aplicaciones en Angular, es fundamental contar con un entorno adecuado que permita gestionar tanto las dependencias del proyecto como las herramientas necesarias para su ejecución. Es aquí donde *Node.js* y *npm* juegan un papel crucial (Genbeta, 2014).

- **Node.js**. Proporciona el entorno de ejecución para JavaScript en el lado del servidor, lo que permite que Angular funcione de manera óptima. Al utilizar Node.js, los desarrolladores pueden aprovechar una serie de herramientas de desarrollo y servidores de pruebas que facilitan el proceso de creación y depuración de aplicaciones Angular.
- **npm (Node Package Manager).** Es el gestor de paquetes que se utiliza para instalar y gestionar las bibliotecas y dependencias que Angular necesita para su funcionamiento. A través de *npm*, los desarrolladores pueden acceder fácilmente a una amplia gama de paquetes de software y herramientas que enriquecen sus proyectos.

Fuentes: (Hiberus, 2021), (HubSpot, 2023) y (Angular University, 2024).

4.4.3. Spring Boot

Spring Boot es un framework que simplifica el desarrollo de aplicaciones Java al proporcionar una serie de configuraciones y convenciones predeterminadas que permiten iniciar rápidamente un proyecto. Se basa en el Spring Framework y es ideal para crear aplicaciones web y servicios RESTful.

Spring Boot facilita la configuración de aplicaciones y proporciona características como la autoconfiguración, lo que ahorra tiempo a los desarrolladores (IBM, 2021).

4.4.4. Apache Maven

Apache Maven es una herramienta de gestión de proyectos y comprensión de proyectos para proyectos Java. Se utiliza para gestionar las dependencias del proyecto y automatizar el proceso de construcción, incluyendo tareas como la compilación, pruebas y empaquetado de aplicaciones (OpenWebinars, 2019).

En el contexto de Spring Boot, Maven se utiliza para definir cómo se construye el proyecto y qué bibliotecas son necesarias.

4.4.5. Hibernate y JPA

La persistencia de datos es un aspecto crucial en el desarrollo de aplicaciones, ya que permite almacenar y recuperar información de manera eficiente. En el ecosistema de Java, la Java Persistence API (JPA) es una especificación que proporciona una forma estándar de gestionar la persistencia de datos en aplicaciones Java. JPA permite a los desarrolladores interactuar con bases de datos relacionales mediante el uso de objetos Java, simplificando el proceso de mapeo entre estos objetos y las tablas de la base de datos (Campus MVP, 2021).

Hibernate es una implementación de JPA que se ha convertido en una de las herramientas más populares para la gestión de la persistencia en aplicaciones Java. Proporciona una serie de funcionalidades avanzadas, como la gestión automática de conexiones, el manejo de transacciones y un sistema de caché, lo que permite a los desarrolladores centrarse en la lógica de negocio sin preocuparse por los detalles de la base de datos.

En conjunto, Hibernate y JPA permiten a los desarrolladores crear aplicaciones más robustas y eficientes al facilitar la interacción con bases de datos relacionales, permitiendo la creación, lectura, actualización y eliminación de datos de manera sencilla y estructurada.

Otras fuentes: (NTT Data, 2019) y (Arquitectura Java, 2021).

4.4.6. MySQL

MySQL es el sistema de gestión de bases de datos relacional utilizado en este proyecto. Como base de datos relacional, organiza los datos en tablas que se relacionan entre sí, permitiendo almacenar, consultar y manipular grandes volúmenes de datos estructurados.

Para facilitar la interacción entre la aplicación Java y la base de datos MySQL, se ha incluido el MySQL JDBC Driver como dependencia en el proyecto. Este driver, junto con las tecnologías JPA y Hibernate,

permite que Spring Boot acceda a la base de datos y realice operaciones de manera eficiente y sin necesidad de consultas SQL complejas.

Fuentes: (OpenWebinars, 2019) y (Arsys, 2024).

4.4.7. Herramientas de Inteligencia Artificial

Durante el desarrollo del proyecto, se ha utilizado la herramienta de inteligencia artificial **ChatGPT** para facilitar tanto la redacción de la documentación como la implementación del código. ChatGPT ha sido de gran apoyo en la generación y estructuración de contenido en la memoria, así como en la obtención de orientación sobre temas técnicos. Su principal utilidad ha sido ayudar a definir conceptos clave y ofrecer una base sobre la cual investigar de manera más eficiente en fuentes adicionales. Un ejemplo de esto fue su aporte en la integración de pagos con PayPal, proporcionando una visión general que facilitó la búsqueda de información más específica.

Capítulo 5
Diseño

Este capítulo aborda la estructura y los patrones utilizados para la construcción de la plataforma. Se inicia con la descripción de la arquitectura del sistema explicando los diferentes patrones arquitectónicos empleados tanto para el frontend como para el backend y, tras ello, se presentan los patrones de diseño aplicados en la solución de diseño de ambas partes. Finalmente, se expone el diseño de la interfaz de usuario, detallando las decisiones tomadas para ofrecer una experiencia eficiente e intuitiva.

5.1. Arquitectura del sistema

El sistema desarrollado es una aplicación web, lo que implica una arquitectura basada en la separación entre frontend y backend. Aunque en entornos de despliegue convencionales estos componentes se ejecutan en servidores distintos, durante el desarrollo y pruebas, toda la aplicación ha sido ejecutada en un entorno local (localhost), donde tanto el frontend como el backend residen en la misma máquina.

Para estructurar la plataforma, se ha seguido una arquitectura cliente-servidor, donde el frontend actúa como la interfaz de usuario y el backend gestiona la lógica de negocio y el acceso a los datos. Sin embargo, es importante aclarar que en aplicaciones web modernas, el "cliente" no es un dispositivo separado, sino que se refiere al código que se ejecuta en el navegador del usuario. Este cliente web interactúa con el servidor a través de peticiones HTTP, obteniendo datos y enviando acciones según la interacción del usuario.

A continuación, se describen los principales patrones arquitectónicos empleados en el desarrollo del frontend y el backend.

5.1.1. Patrón arquitectónico MVVM

En el frontend de la aplicación, se ha decidido seguir el patrón arquitectónico Model-View-ViewModel (MVVM). Este patrón es ampliamente utilizado en el desarrollo de interfaces de usuario, ya que permite dividir las responsabilidades entre la lógica de presentación, la lógica de negocio y los datos, promoviendo una estructura clara y modular. Angular, el framework que se utiliza en este proyecto para el frontend, implementa de manera implícita varias de las características clave de este patrón, lo que facilita su adopción.

MVVM es una evolución del patrón MVC (Model-View-Controller) y busca mejorar la organización y mantenibilidad del código, permitiendo la reutilización de componentes y facilitando el desarrollo y prueba de cada parte de la aplicación de manera independiente.

Este patrón divide el sistema en tres elementos principales:

- **Model (Modelo):** representa los datos y la lógica de negocio. Se encarga de gestionar el acceso a la base de datos, servicios o cualquier fuente de datos externa, encapsulando las reglas del negocio y asegurando la coherencia de los datos.
- **View (Vista):** es la interfaz de usuario (UI) con la que interactúa el usuario. Su responsabilidad principal es presentar los datos que provienen del ViewModel y responder a las acciones del usuario. No contiene lógica de negocio.
- **ViewModel (Modelo de Vista):** actúa como intermediario entre la View y el Model. Es responsable de manejar la lógica de presentación, formatear los datos para la vista y recibir los eventos de la vista, como interacciones del usuario, y pasarlos al modelo. Además, gestiona el estado de la vista y asegura que los datos se mantengan sincronizados entre el modelo y la vista.

MVVM y **MVC** comparten una filosofía común: dividir las responsabilidades entre los distintos elementos para hacer más manejable el desarrollo de aplicaciones complejas. Sin embargo, tienen diferencias clave:

- En el patrón MVC, el controlador gestiona la interacción entre el modelo y la vista, es decir, cuando el controlador percibe cambios en la vista, actualiza la vista y el modelo en consecuencia.
- En MVVM, en lugar de tener un controlador explícito, se utiliza el ViewModel, que gestiona el estado de la interfaz y expone datos formateados a la vista. Aquí, la vista se enlaza automáticamente con el ViewModel, y el flujo de datos entre ambos suele gestionarse mediante "data binding" (vinculación de datos), lo que minimiza el código necesario para sincronizar la interfaz con el modelo.

Como se mencionó, **Angular** promueve el uso del patrón MVVM de manera implícita a través de las siguientes características clave:

- **Componentes.** Los componentes en Angular pueden considerarse como el ViewModel en el patrón MVVM. Son los responsables de gestionar los datos, exponer propiedades y manejar eventos, todo ello vinculado directamente a la vista mediante el sistema de "two-way data binding" de Angular.
- **Data Binding.** Angular facilita el enlace de datos entre el componente (ViewModel) y la plantilla (template) de la vista (View), lo que es esencialmente el binding que caracteriza al

MVVM. Con el two-way data binding, los cambios en los datos del ViewModel se reflejan automáticamente en la vista, y viceversa.

- **Templates.** En Angular, las plantillas definen la View, y están vinculadas directamente al componente, proporcionando una clara representación de cómo debe verse la interfaz en función de los datos expuestos por el componente.

Fuentes utilizadas: (Microsoft, 2023), (KeepCoding, 2024), (Imagina Formación, 2024) y (OpenWebinars, 2018).

5.1.2. API REST

Se trata de un conjunto de principios arquitectónicos que guían la creación de servicios web, especialmente en la interacción entre el frontend y el backend. Se basa en el uso del protocolo HTTP para interactuar con los recursos del sistema a través de URLs bien definidas, siguiendo los principios del diseño REST (Representational State Transfer).

REST es un estilo arquitectónico que define un conjunto de restricciones para crear servicios web escalables y eficientes. Un servicio que cumple con estas restricciones se denomina RESTful. Estos servicios RESTful exponen sus recursos (como usuarios, pedidos, productos, etc.) mediante una serie de endpoints o URLs, y utilizan los métodos HTTP estándar para realizar acciones sobre ellos. Los más comunes son:

- **GET.** Recuperar información.
- **POST.** Crear un nuevo recurso.
- **PUT.** Actualizar un recurso existente.
- **DELETE.** Eliminar un recurso.

Los servicios RESTful son sin estado (stateless), lo que significa que el servidor no almacena información sobre las solicitudes anteriores del cliente. Toda la información necesaria para procesar una solicitud debe proporcionarse en cada interacción.

En conclusión, la aplicación de una API RESTful permite al backend exponer una serie de recursos a través de endpoints accesibles mediante HTTP, que el frontend puede consumir a través de peticiones HTTP.

Fuentes utilizadas: (RedHat, 2023) y (AWS, 2022).

5.2. Patrones de diseño

Los patrones de diseño son soluciones reutilizables y probadas para problemas comunes en el desarrollo de software. Su propósito es proporcionar una estructura clara y eficiente para resolver ciertos tipos de desafíos, facilitando la legibilidad, el mantenimiento y la escalabilidad del código.

A continuación, se presentan los utilizados en este proyecto.

5.2.1. Patrón Singleton

El objetivo principal de este patrón es garantizar que una clase tenga una única instancia en toda la aplicación y proporcionar un punto de acceso global a esa instancia. Esto es útil cuando solo se necesita un objeto que coordine acciones o gestione el estado compartido de una aplicación, como el acceso a una base de datos o la gestión de la configuración del sistema.

Ventajas:

- ✓ **Ahorro de recursos.** Evita la creación de múltiples instancias innecesarias.
- ✓ **Consistencia de estado.** Permite que toda la aplicación trabaje con el mismo objeto, lo que ayuda a mantener un estado compartido y coherente.
- ✓ **Fácil acceso.** Cualquier parte del código puede acceder fácilmente a la única instancia sin necesidad de pasarla manualmente.

En el contexto de Spring, el patrón Singleton se implementa de forma implícita. Por defecto, los beans de Spring tienen un ámbito Singleton, lo que significa que una única instancia del bean será creada y compartida en toda la aplicación, a menos que se especifique un comportamiento diferente. Cuando una clase es marcada con anotaciones como @Component, @Service o @Repository, Spring garantiza que solo una instancia de esa clase sea creada a menos que se especifique lo contrario. Por lo tanto, sin tener que escribir el patrón manualmente, ya se está aplicando en gran parte del sistema.

Fuentes utilizadas: (Refactoring Guru, 2020) y (Blancarte, 2018).

5.2.2. Patrón DAO y Repository

En el desarrollo de aplicaciones modernas, los patrones de diseño DAO (Data Access Object) y Repository se utilizan para separar la lógica de acceso a datos de la lógica de negocio. Ambos patrones comparten el objetivo de proporcionar una abstracción que facilite la interacción con los datos, pero cada uno lo aborda desde perspectivas ligeramente diferentes.

El **patrón DAO** se centra en proporcionar una capa de abstracción entre la lógica de negocio y la capa de acceso a datos. Este patrón define interfaces o clases que encapsulan la lógica específica de acceso a datos y proporcionan operaciones básicas de CRUD (crear, leer, actualizar y eliminar). Sus principales características son:

- **Abstracción del acceso a datos.** El DAO oculta los detalles técnicos sobre cómo se almacenan y recuperan los datos.
- **Interfaces consistentes.** Proporciona una interfaz coherente para que la lógica de negocio interactúe con los datos sin preocuparse por la implementación específica.
- **Centralización de la lógica.** La lógica de acceso a datos se centraliza, lo que facilita su reutilización en toda la aplicación.

El **patrón Repository** también abstrae el acceso a datos, pero se enfoca en proporcionar una capa orientada a objetos que gestione las entidades y los agregados del dominio. Trabaja con los datos como colecciones de objetos y proporciona una interfaz para recuperar y almacenar estos objetos. Sus principales características son:

- **Abstracción de la persistencia.** Oculta los detalles de almacenamiento y consulta, permitiendo que la lógica de negocio se enfoque en la manipulación de entidades del dominio.
- **Operaciones específicas.** Además de las operaciones CRUD, puede incluir métodos personalizados que se alineen con los requerimientos del dominio.
- **Enfoque orientado a objetos.** Trata las entidades como colecciones y permite un manejo más natural en sistemas basados en modelos de dominio.

En sistemas complejos es común **combinar ambos patrones** para aprovechar sus fortalezas. Por ejemplo:

- El DAO se utiliza para encapsular los detalles específicos de acceso a datos, como las consultas SQL o las operaciones directas sobre la base de datos.
- El Repository se utiliza como una capa de nivel superior que orquesta múltiples DAOs para proporcionar una interfaz orientada al dominio.

Su uso combinado garantiza una clara separación de responsabilidades, mejora el mantenimiento del código y permite una mayor adaptabilidad a cambios futuros.

Fuentes utilizadas: (OurAcademy, 2019), (Arquitectura Java, 2023), (Medium, 2016) y (Platzi, 2023).

5.2.3. Patrón DTO

El patrón DTO (Data Transfer Object) se utiliza para transferir datos de forma eficiente entre diferentes partes de una aplicación, en especial entre el backend y el frontend. Los DTO son objetos diseñados exclusivamente para transportar datos y no contienen lógica de negocio ni métodos innecesarios.

Para este proyecto, se utilizan los DTO para pasar información de las entidades gestionadas en el backend hacia el frontend. Por ejemplo, cuando el backend necesita devolver datos al frontend, las consultas a la base de datos que generan objetos de dominio completos (entidades) pueden contener información innecesaria o sensible para el frontend. En lugar de exponer directamente las entidades, los datos relevantes se encapsulan en un DTO, que es el objeto que finalmente se envía al frontend.

El uso de DTO también promueve una separación clara de responsabilidades. En el frontend, la información recibida se mapea a estructuras específicas, como interfaces, que permiten organizar y manejar los datos según las necesidades de la interfaz de usuario, manteniendo el diseño limpio y escalable.

Ventajas del patrón DTO

- ✓ **Separación de responsabilidades.** Aísla la lógica de negocio en el backend, dejando al frontend únicamente los datos necesarios.
- ✓ **Seguridad y control.** Evita exponer información sensible o innecesaria al frontend.
- ✓ **Reutilización y mantenimiento.** Facilita la evolución de la aplicación, ya que los cambios en las entidades del dominio no afectan directamente a las estructuras que utiliza el frontend.
- ✓ **Eficiencia.** Reduce la cantidad de datos enviados al frontend, optimizando la comunicación entre cliente y servidor.

Fuentes utilizadas: (Oscar Blancarte, 2018), (Oscar Blancarte, 2020) y (Medium (Fernando Andrade), 2024)

5.3. Decisiones de diseño

5.3.1. Versión de Angular

Para el desarrollo de este proyecto, se ha decidido utilizar Angular en su versión 18. Esta elección se basa en las mejoras significativas que ofrece en términos de rendimiento y funcionalidades. Sin embargo, se optará por seguir utilizando módulos en lugar de componentes *standalone*, lo que permitirá una mayor organización del código y facilitará la gestión de las dependencias dentro de la

aplicación. Asimismo, se aprovecharán las nuevas directivas introducidas en Angular 17, como @if y @for, que brindan una mayor flexibilidad y eficiencia en la manipulación del DOM.

5.3.2. Angular Material

Angular Material es una biblioteca de componentes de interfaz de usuario que implementa el diseño Material Design de Google (Angular Material, 2020). Esta biblioteca proporciona un conjunto de componentes de alta calidad y listos para usar, lo que permite crear aplicaciones web de manera más rápida y eficiente. Angular Material está diseñado específicamente para funcionar con Angular, lo que asegura una integración fluida y un rendimiento optimizado.

Entre las utilidades y ventajas que ofrece se incluyen las siguientes:

- **Componentes predefinidos.** Angular Material ofrece una amplia variedad de componentes, como botones, cuadros de diálogo, menús, formularios y tablas, que siguen las mejores prácticas de diseño. Esto reduce la necesidad de crear componentes desde cero, acelerando así el proceso de desarrollo.
- **Consistencia visual.** La biblioteca asegura una apariencia coherente en toda la aplicación, lo que mejora la experiencia del usuario y ayuda a mantener una estética profesional. Todos los componentes están diseñados según las directrices de Material Design, lo que garantiza que la aplicación sea intuitiva y fácil de usar.
- **Accesibilidad.** Angular Material se preocupa por la accesibilidad, incorporando funcionalidades que permiten que la aplicación sea usable por personas con diversas capacidades. Esto incluye el soporte para lectores de pantalla y la implementación de prácticas de accesibilidad recomendadas.
- **Diseño responsive.** Los componentes de Angular Material son responsive por defecto, lo que significa que se adaptan automáticamente a diferentes tamaños de pantalla y dispositivos. Esto es crucial para ofrecer una experiencia de usuario fluida en móviles y tablets.
- **Facilidad de uso.** La biblioteca incluye documentación extensa y ejemplos de uso, lo que facilita aprender a implementar los componentes y personalizarlos según sus necesidades.

Con la implementación de Angular Material en este proyecto, se busca no solo optimizar el desarrollo y diseño de la interfaz de usuario, sino también proporcionar a los usuarios una experiencia más atractiva y funcional.

Fuente utilizada: (Desarrollo Web, 2016).

5.3.3. Logo, tipografía y paleta de colores

El logo de la plataforma será sencillo y minimalista, diseñado en un solo color que variará en tonalidad según el contexto de la aplicación, adaptándose a fondos claros y oscuros para mantener la visibilidad y armonía visual. Como se puede ver en la Figura 5.1, el logo representa un campo y una casa, elementos que reflejan la esencia de la plataforma: cercanía, origen local y conexión con el entorno rural.

En cuanto a la tipografía, se usará Lato de Google Fonts, una fuente reconocida por su claridad y estilo accesible, lo que facilita la lectura en dispositivos de todos los tamaños y añade un toque moderno pero sencillo a la interfaz.

Para la paleta de colores se ha optado por tonos marrones evocando el entorno rural. En la Figura 5.2 se muestran concretamente los tonos que se utilizarán.

Figura 5.1. Logo de la plataforma

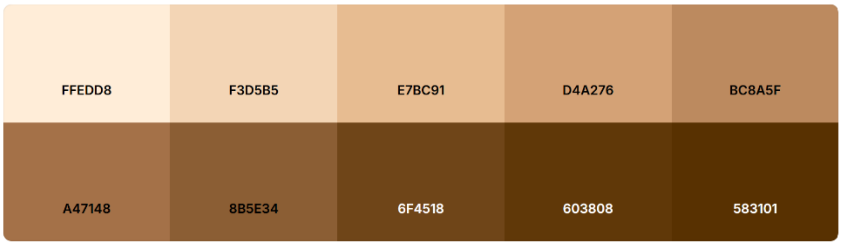

Figura 5.2. Paleta de colores

5.4. Diseño de la interfaz de usuario

En este apartado se presentan los bocetos del diseño de la interfaz de usuario, considerando tanto la perspectiva del comprador como la de los comercios.

El diseño sigue principios de usabilidad y accesibilidad, priorizando una experiencia moderna y sencilla. Para ello, se ha seguido la guía de Material Design, un estándar de Google que promueve interfaces intuitivas mediante principios de movimiento, profundidad y organización clara del contenido. Un ejemplo destacado de su aplicación en este proyecto es el uso de Tabs (Figura 5.10) o pestañas, que permiten estructurar la información en secciones separadas, mejorando la navegación

y evitando la sobrecarga de contenido en una sola vista. Este enfoque sigue el principio de "Progressive Disclosure", mostrando solo la información necesaria en cada momento para una experiencia más fluida.

Además, se ha tomado una decisión clave en la interfaz: la incorporación del concepto de carrito de la compra. Aunque en el análisis inicial (Figura 3.2) y en la estructura de la base de datos (Figura 5.42) no se modela como entidad propia, se ha representado visualmente en la interfaz de usuario porque es un concepto familiar para los usuarios en plataformas de comercio electrónico. Aunque internamente no se gestiona como un carrito tradicional, su presencia facilita la comprensión del proceso de compra. Se puede observar en el boceto de la Figura 5.16.

A continuación, se presentan los bocetos diseñados y realizados con la herramienta Moqups (previamente explicada en el apartado 4.2.2. Moqups) desde la Figura 5.3 hasta la Figura 5.41.

Fuentes utilizadas: (Material Design, 2016), (KeepCoding, 2024), (UXPin, 2023) y (SalySEO, 2024).

Figura 5.3. Página inicial "Mercado Rural"

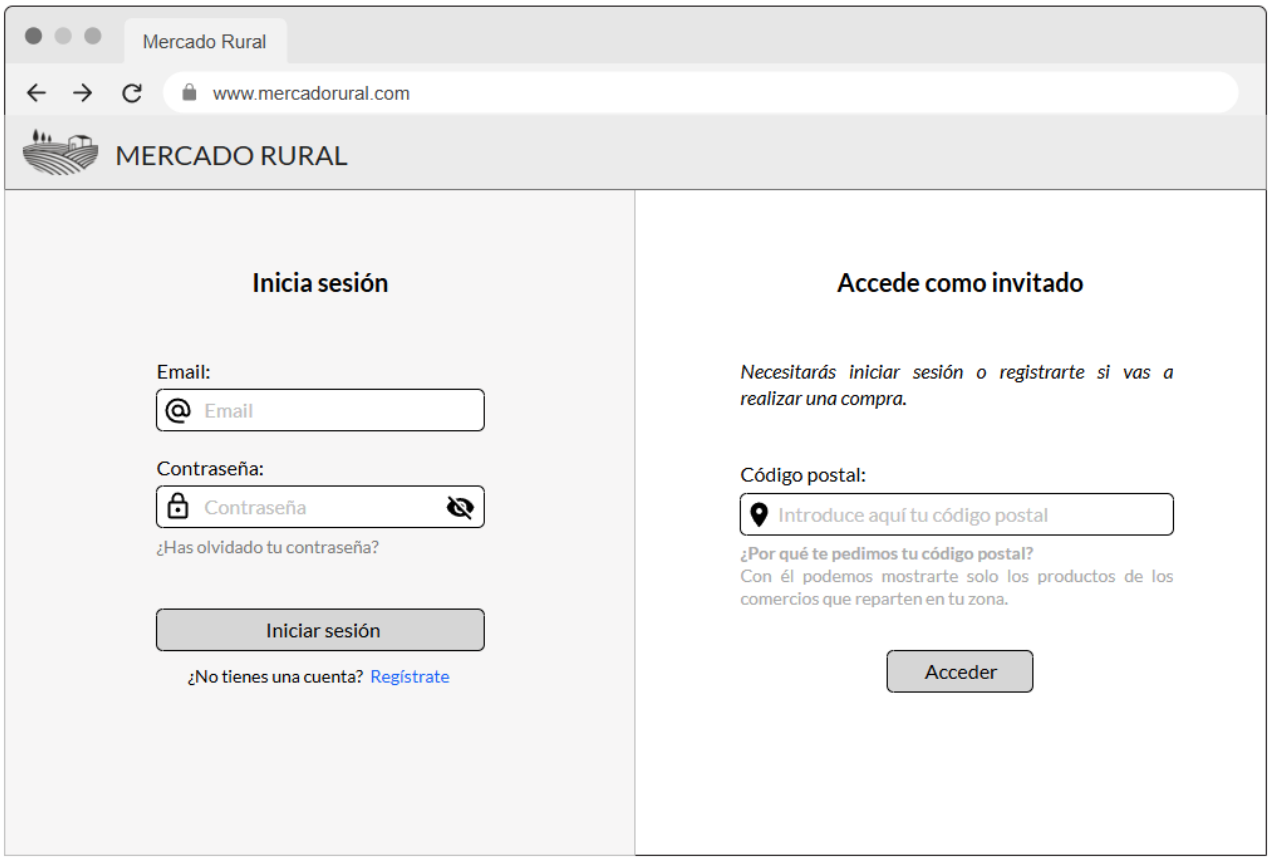

Figura 5.4. Inicio de sesión - compradores

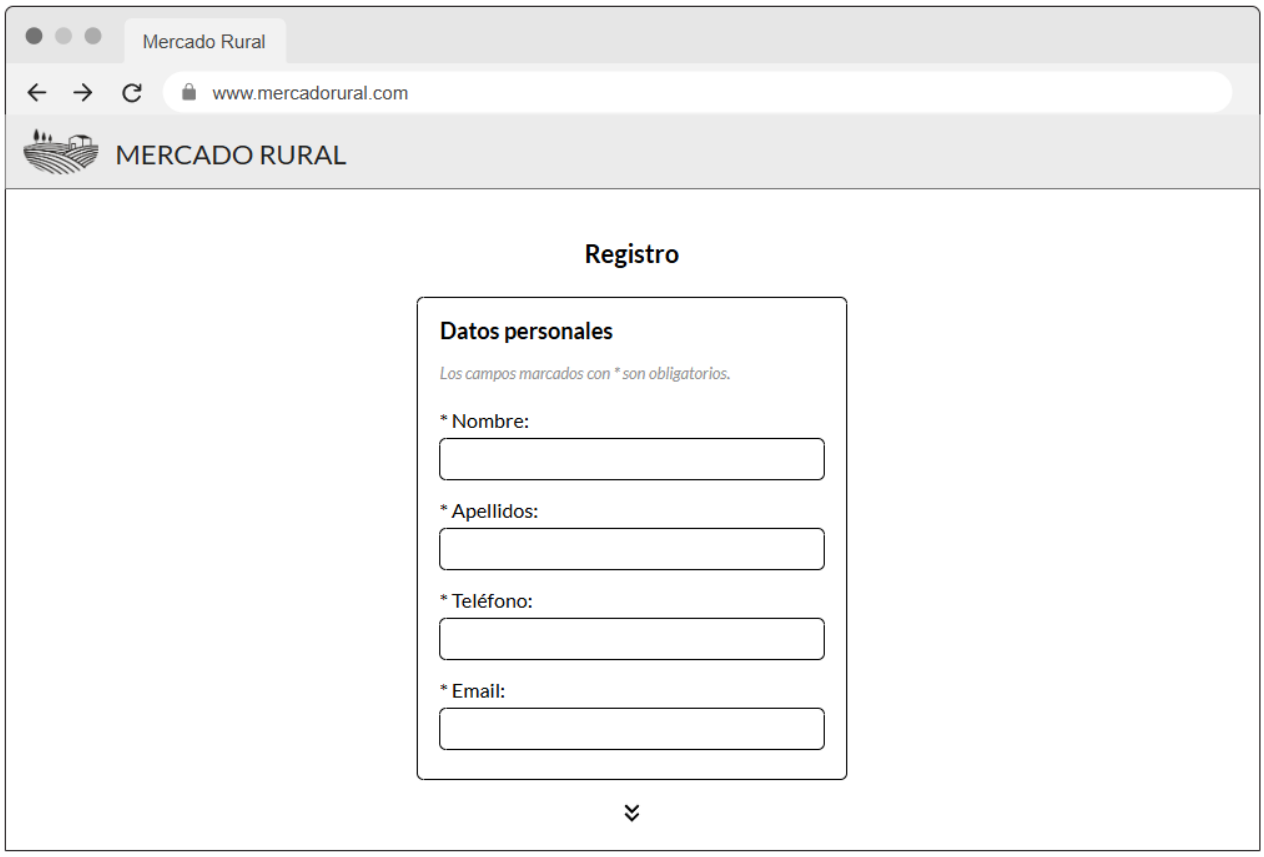

Figura 5.5. Registro compradores (parte 1)

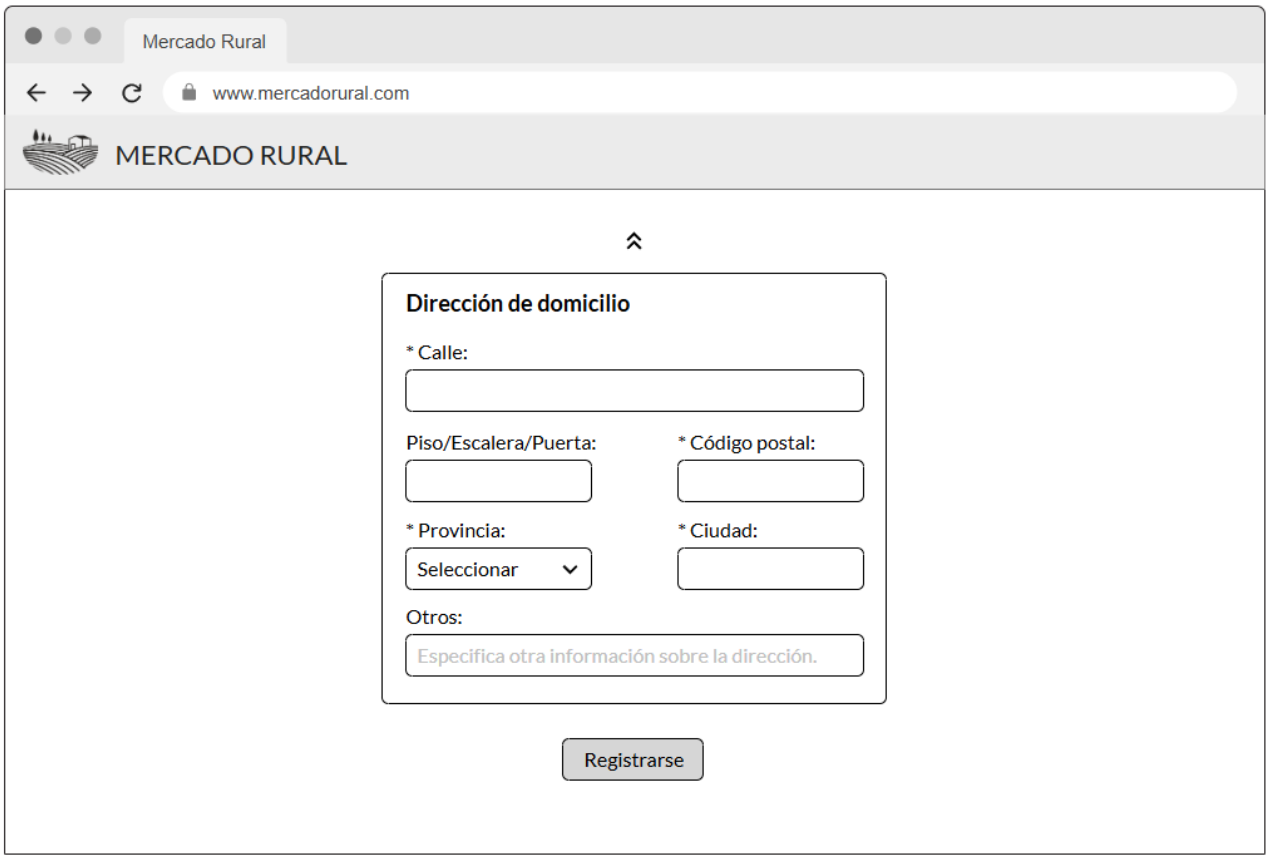

Figura 5.6. Registro compradores (parte 2)

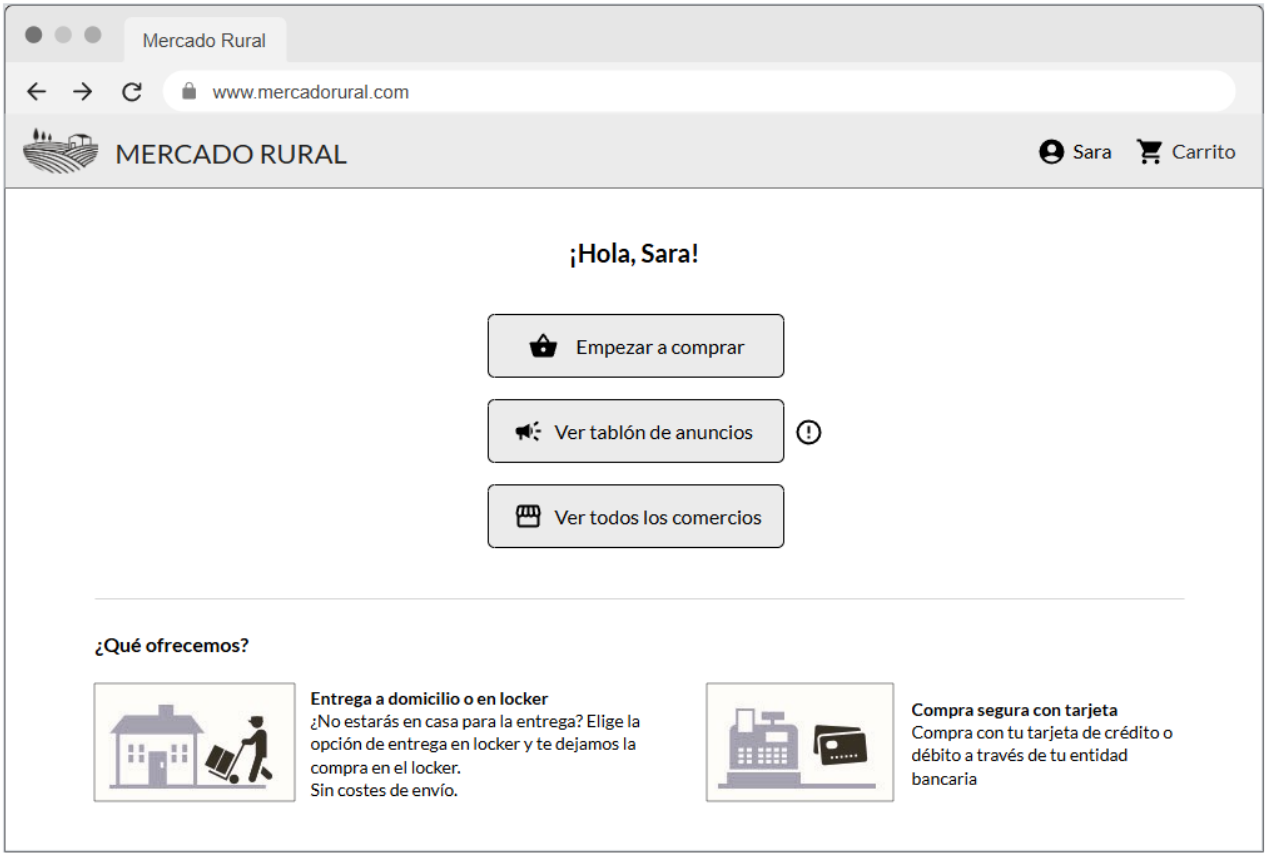

Figura 5.7. Página principal - compradores

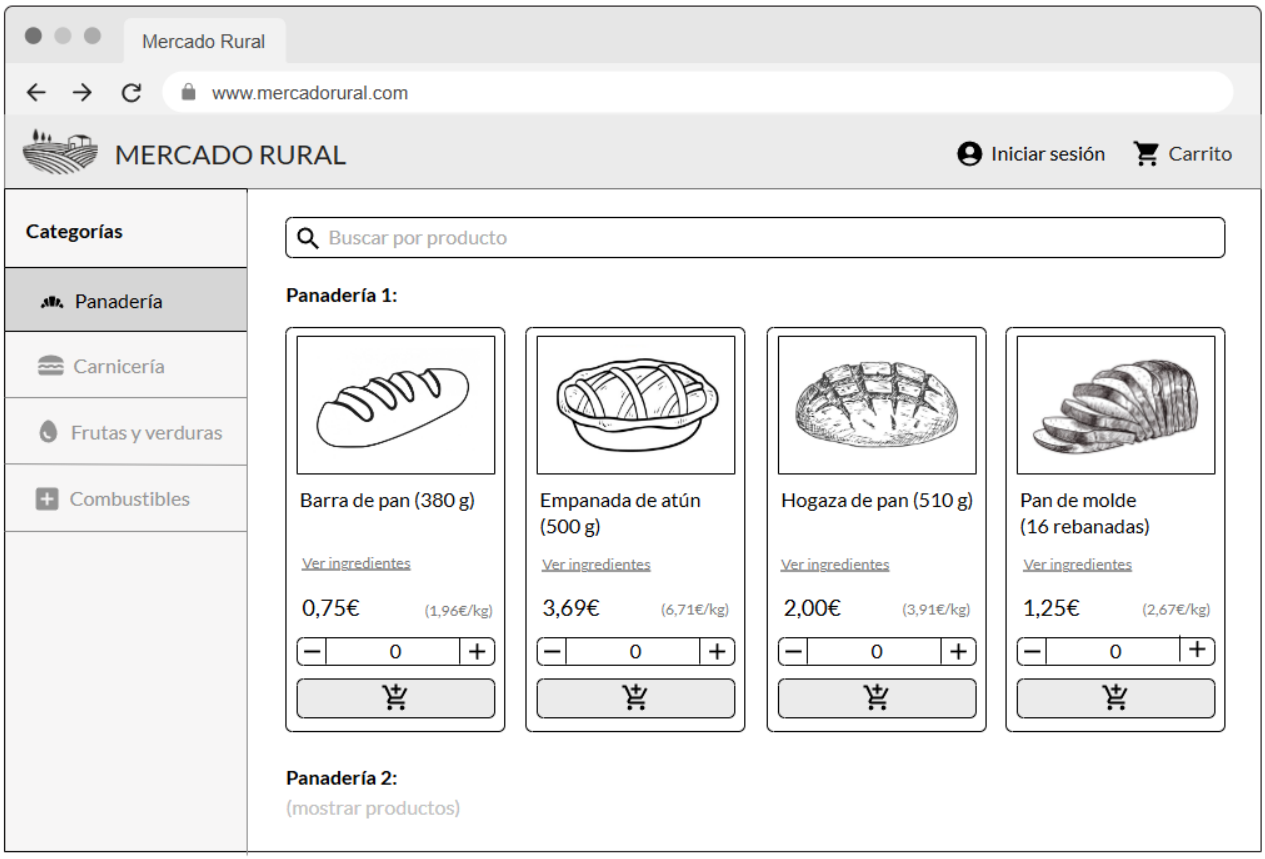

Figura 5.8. Comprar en sección panadería

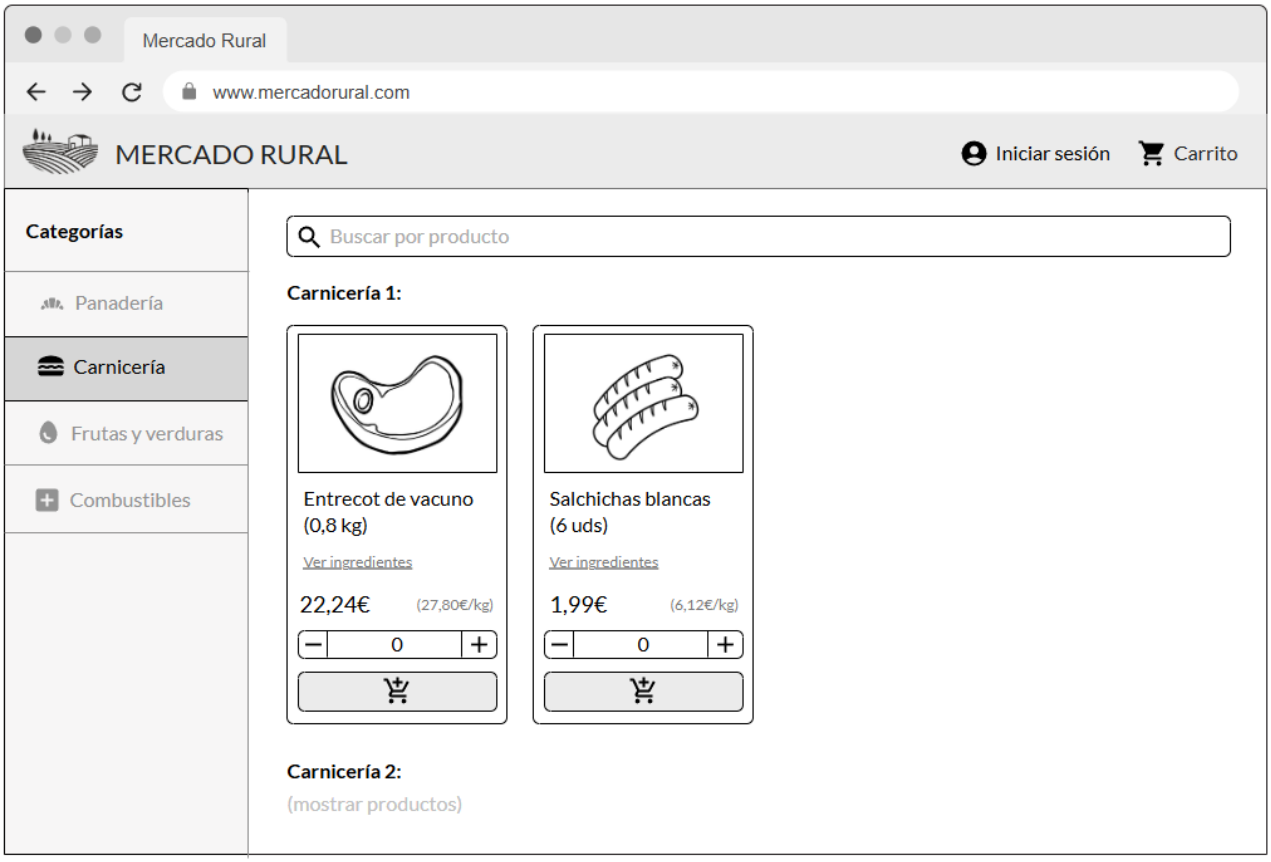

Figura 5.9. Comprar en sección carnicería

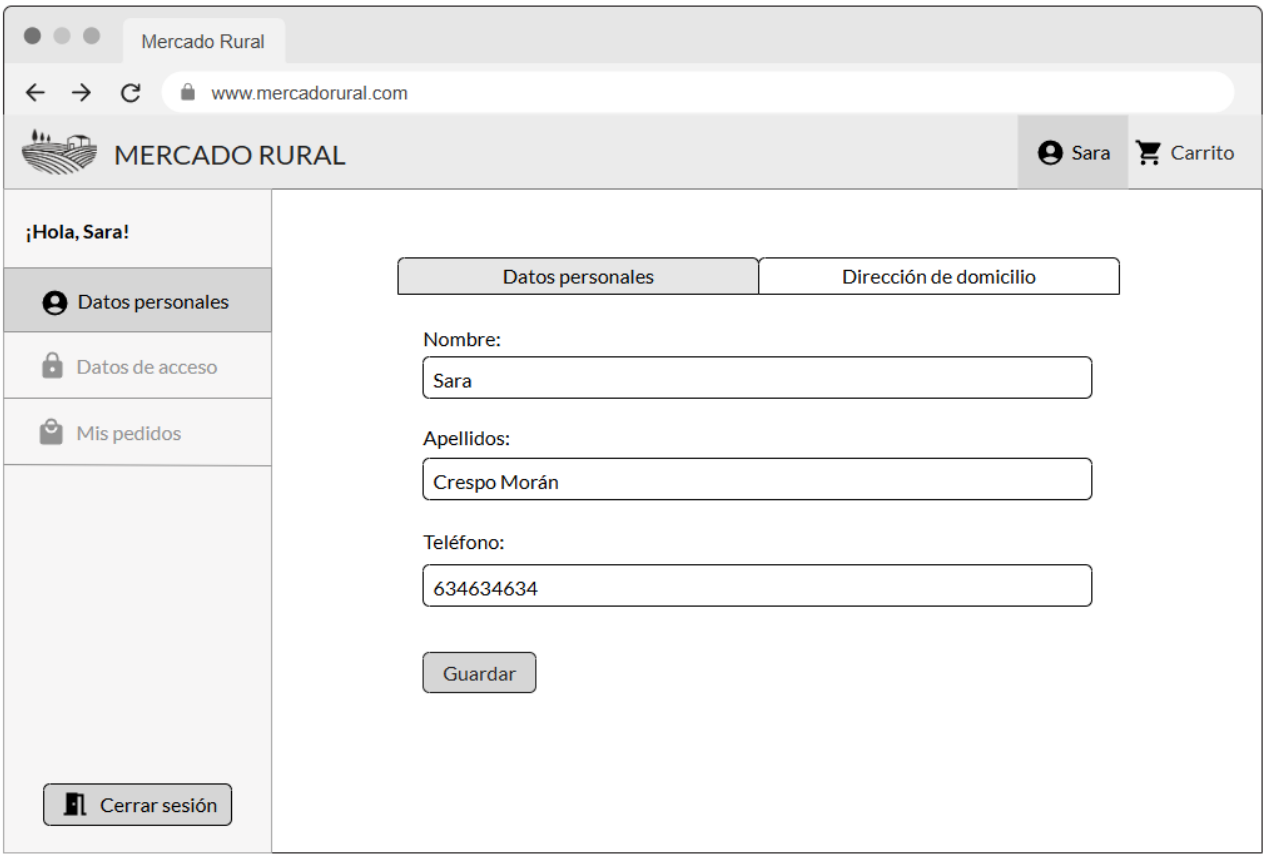

Figura 5.10. Datos personales - comprador

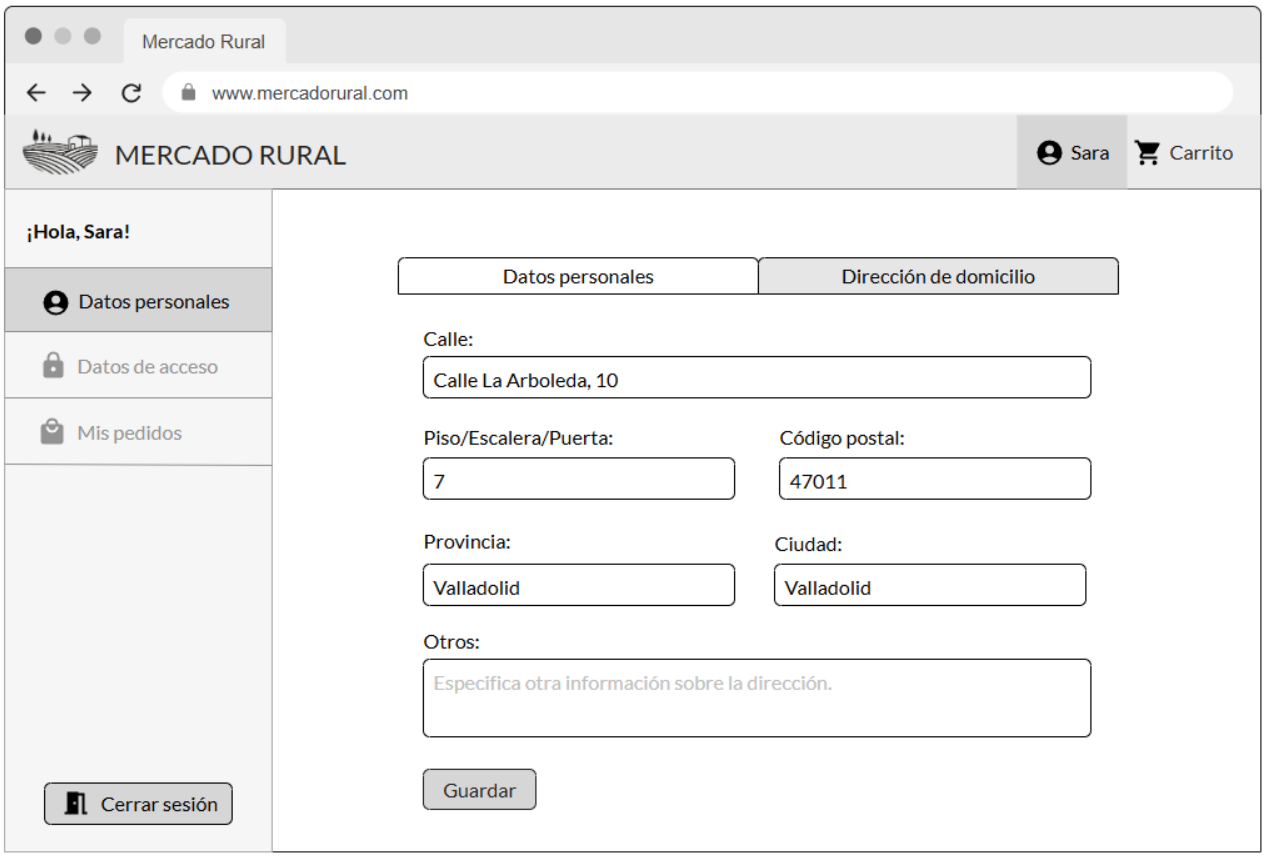

Figura 5.11. Datos de la dirección de domicilio - comprador

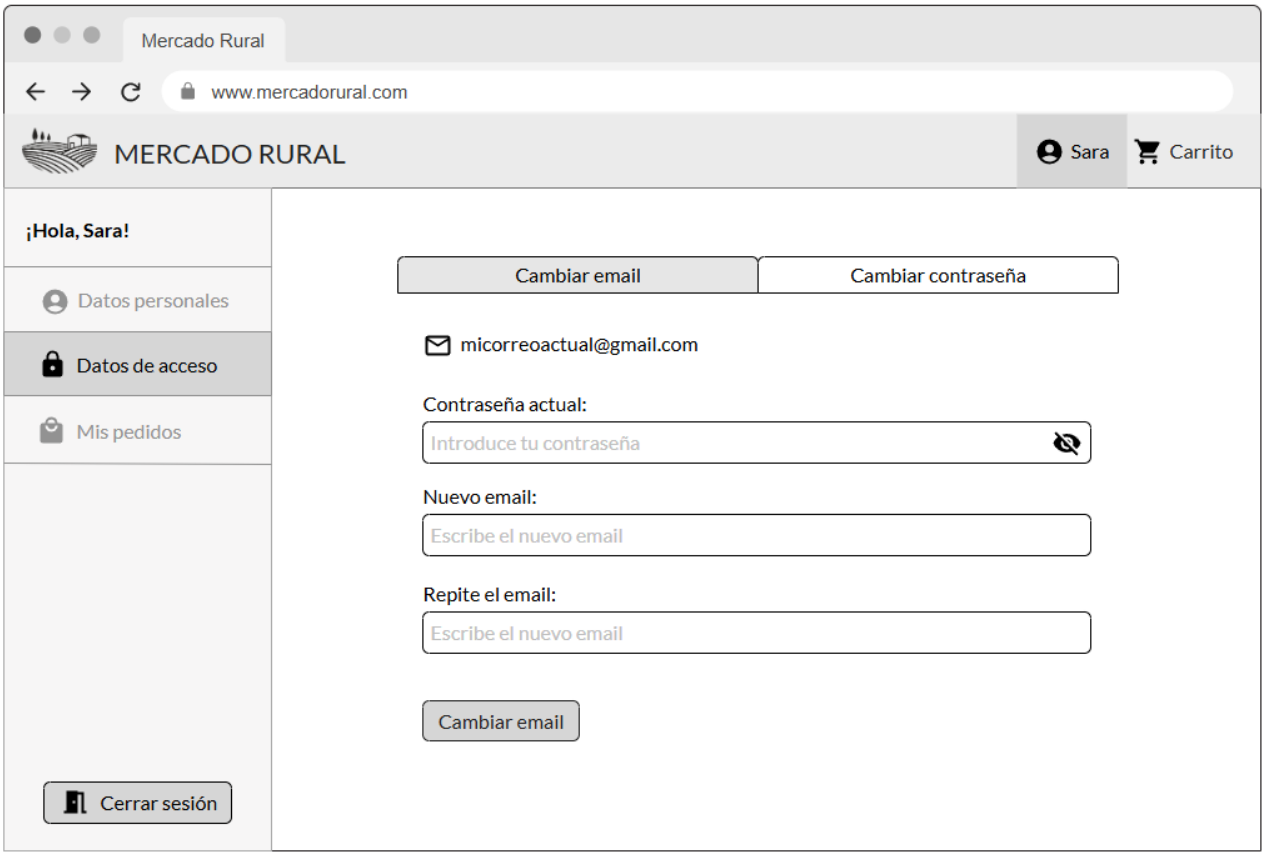

Figura 5.12. Cambiar email - comprador

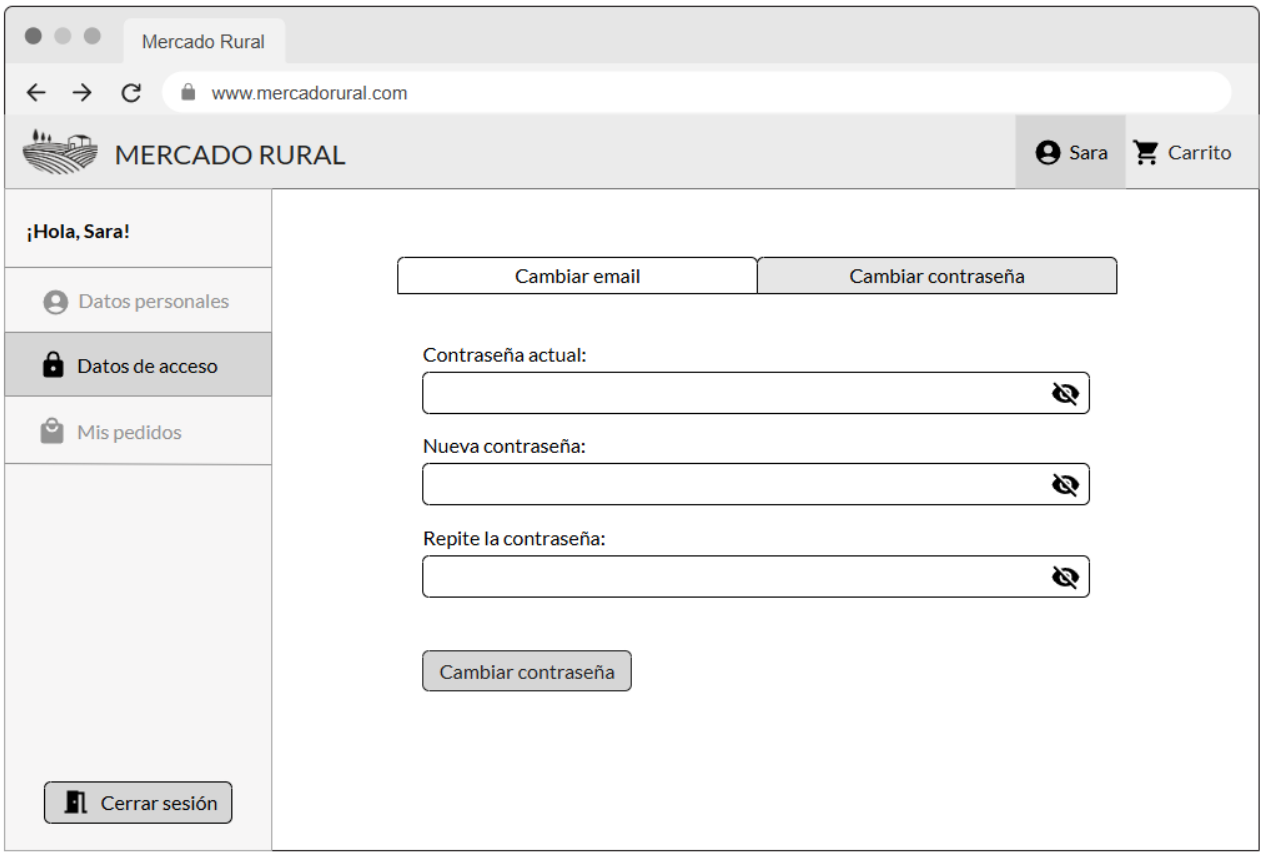

Figura 5.13. Cambiar contraseña - comprador

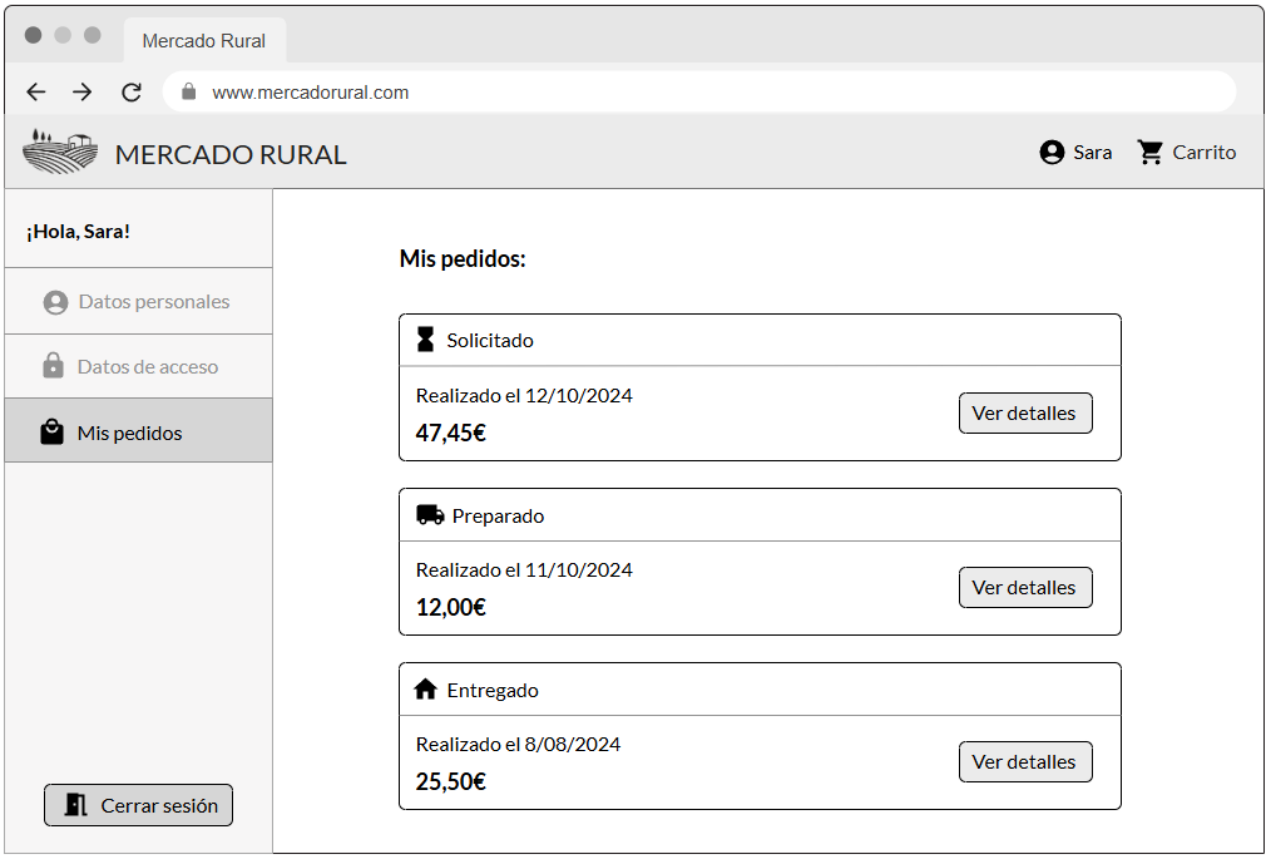

Figura 5.14. Historial de pedidos - comprador

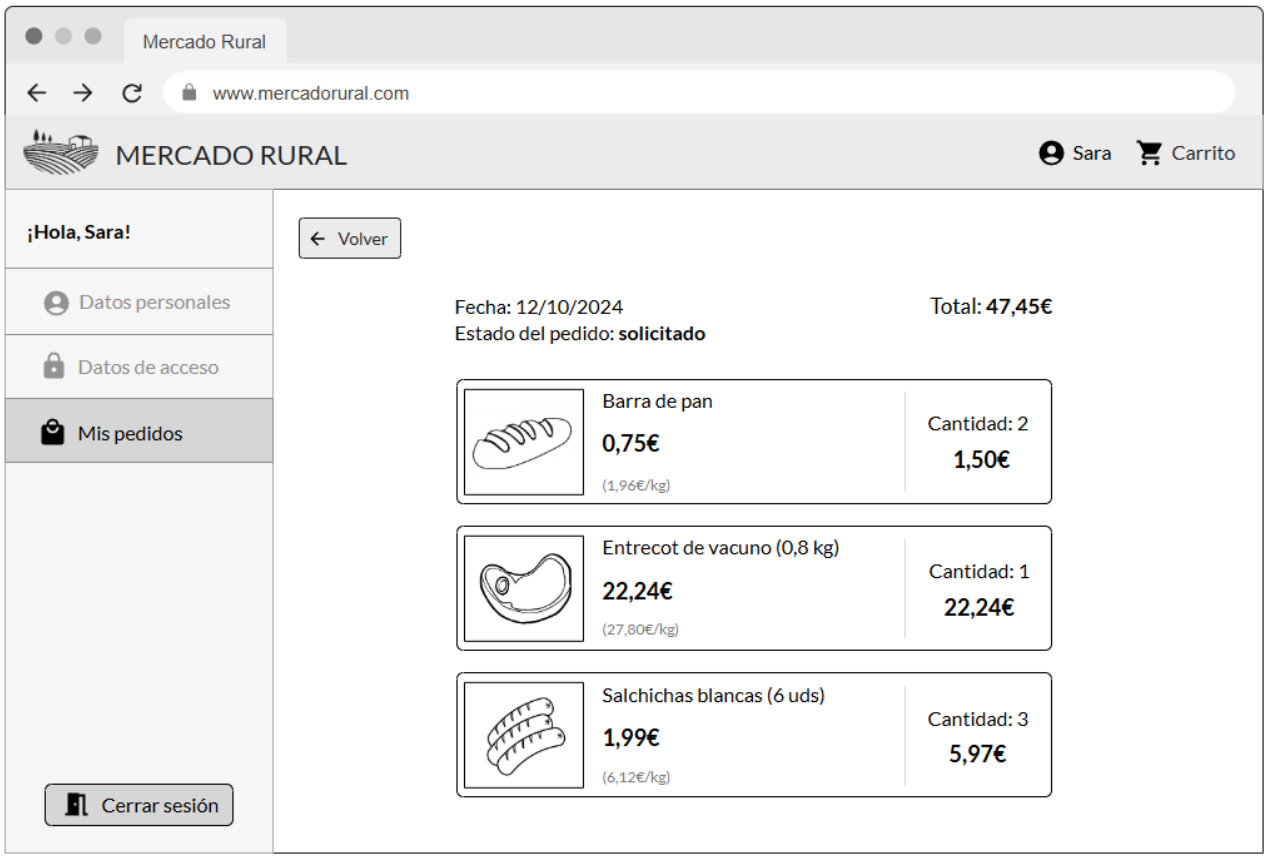

Figura 5.15. Detalles del pedido - comprador

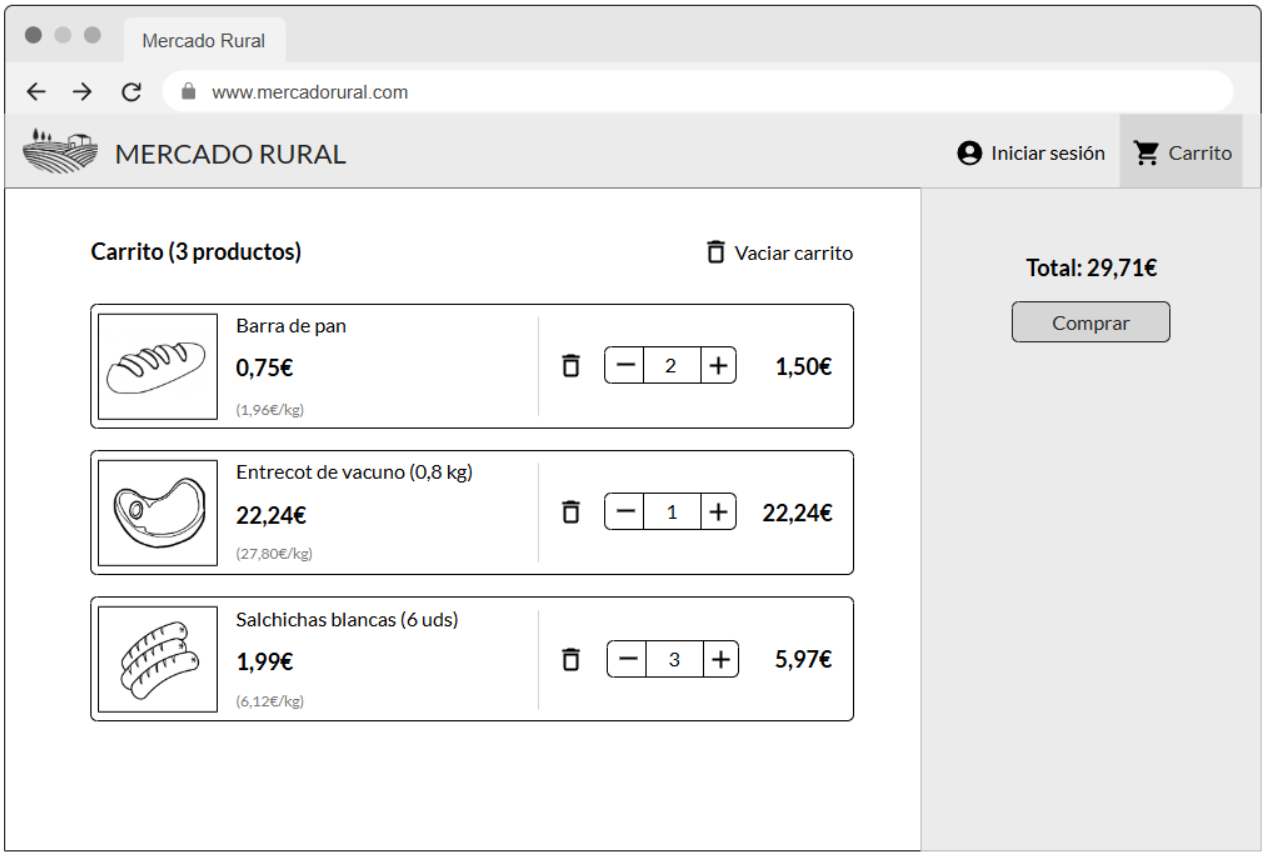

Figura 5.16. Carrito de compra

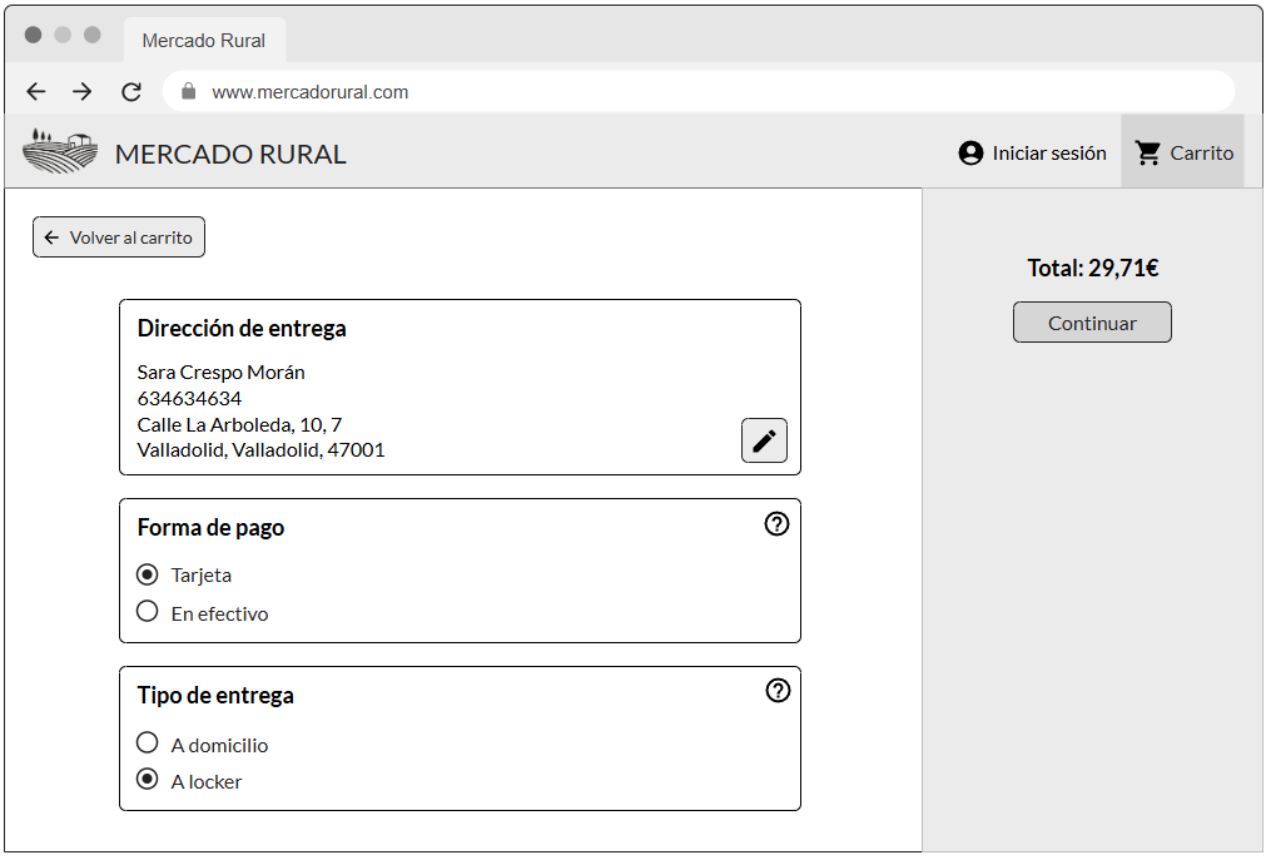

Figura 5.17. Selección de forma de pago y envío

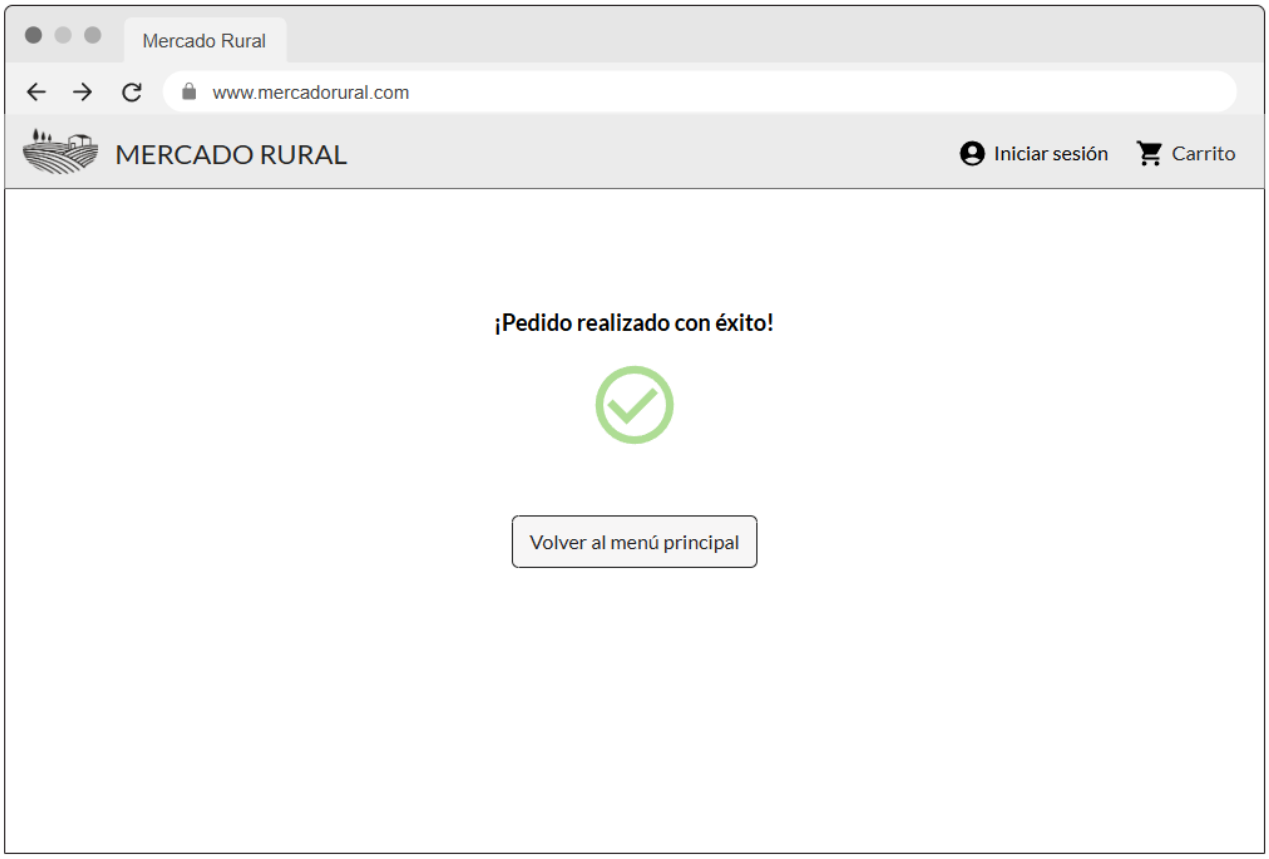

Figura 5.18. Pedido realizado con éxito

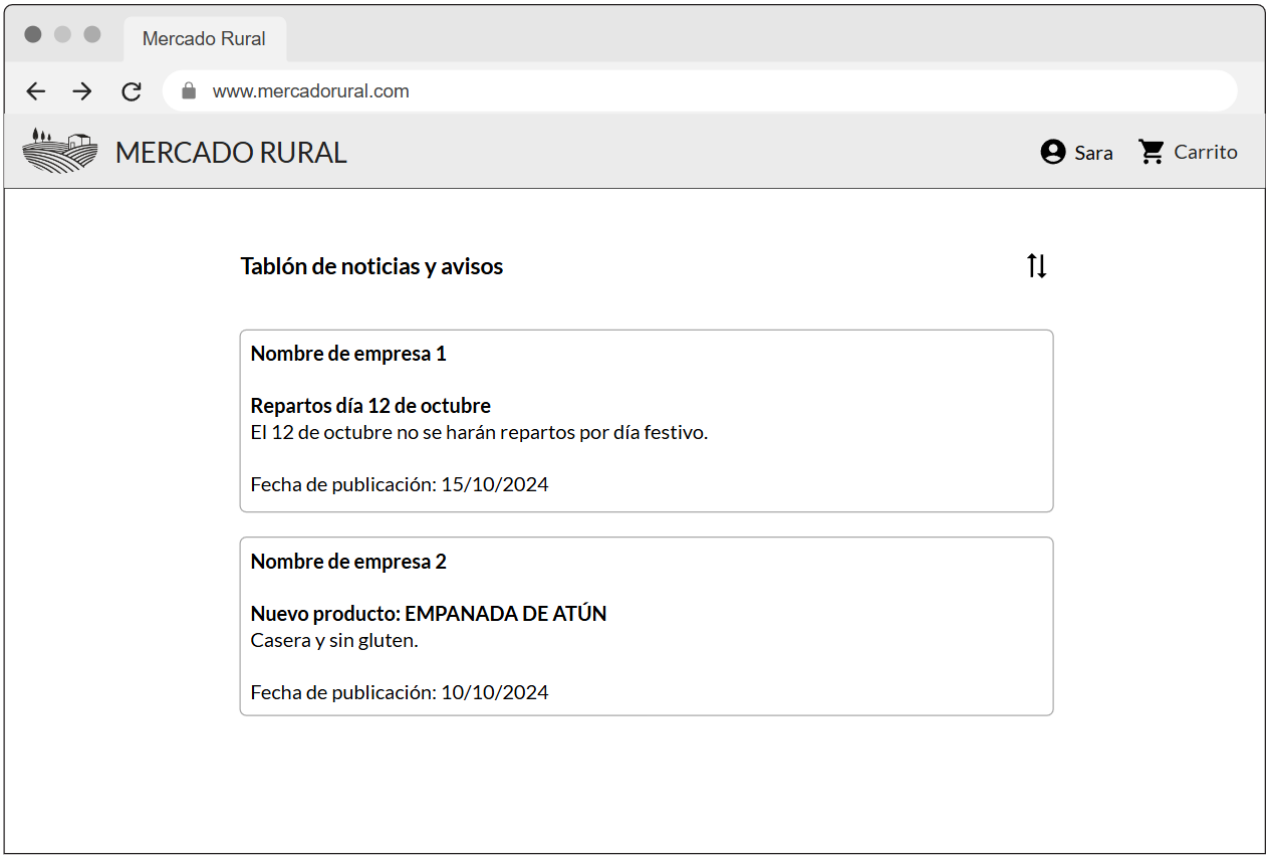

Figura 5.19. Tablón de avisos – comprador

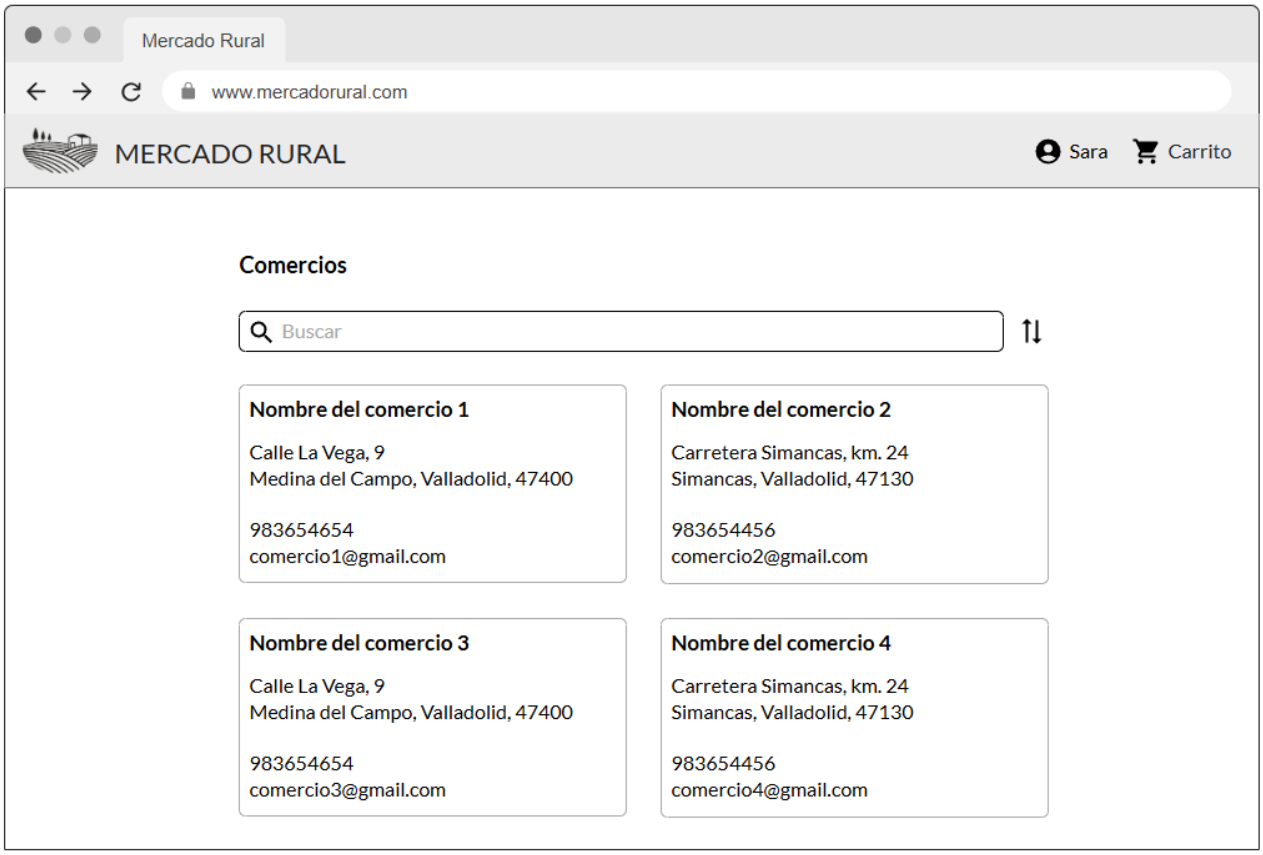

Figura 5.20. Lista de comercios - comprador

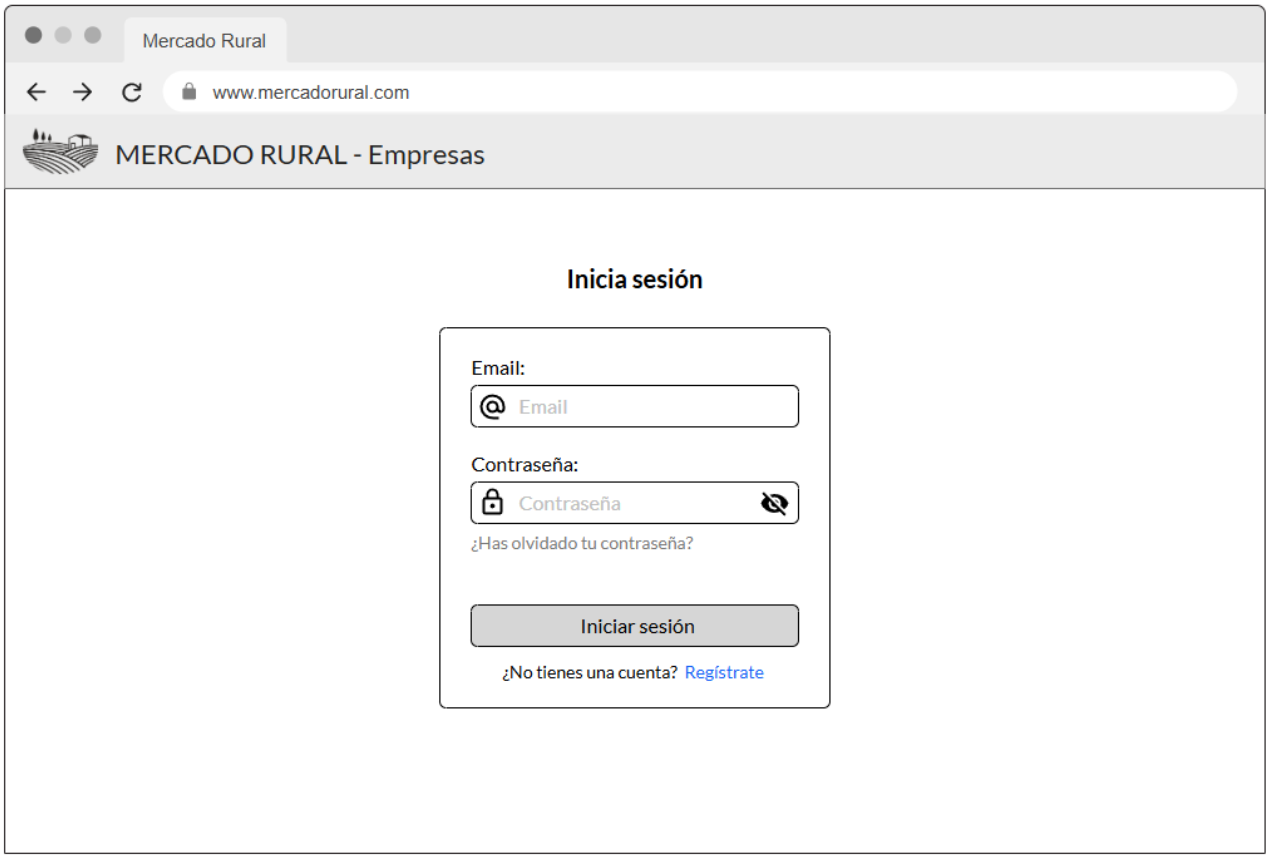

Figura 5.21. Inicio de sesión - empresas

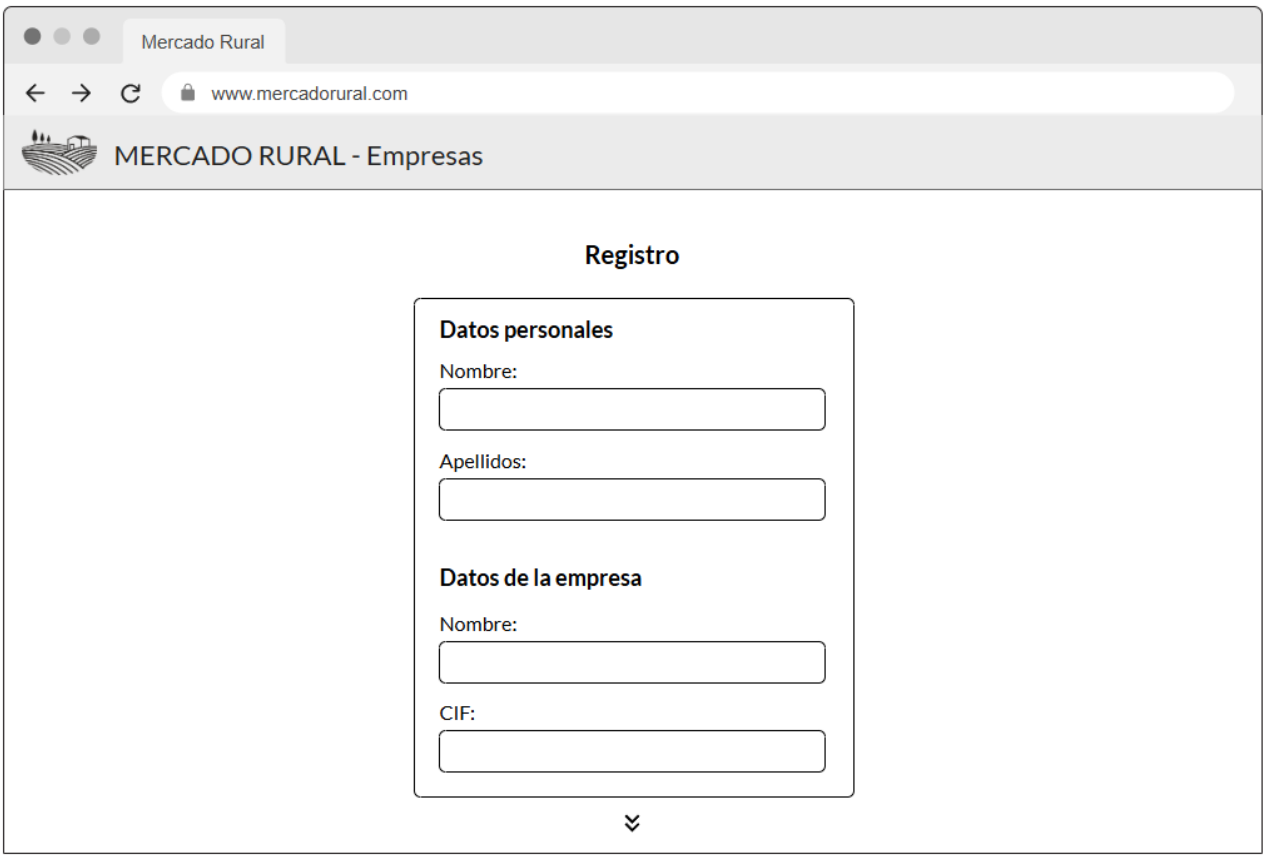

Figura 5.22. Registro empresas (parte 1)

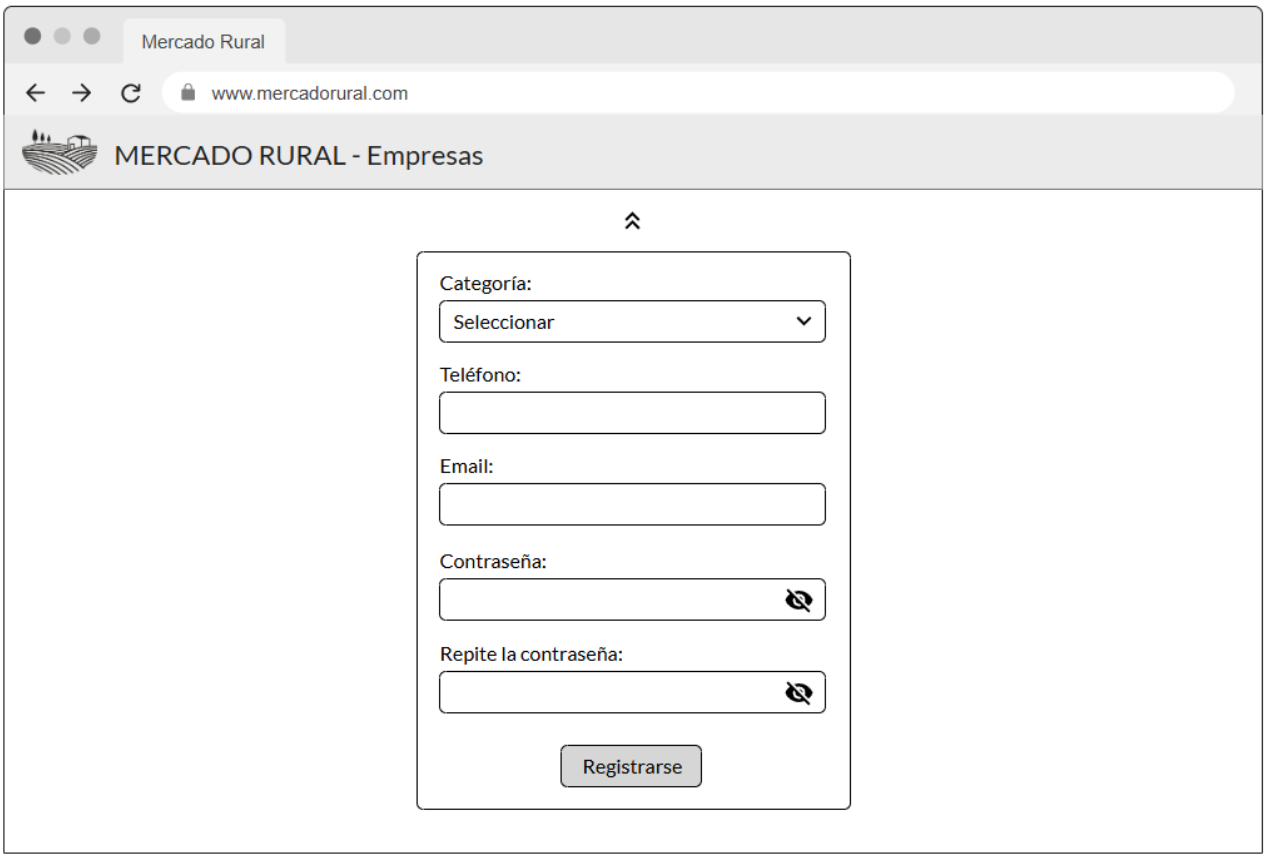

Figura 5.23. Registro empresas (parte 2)

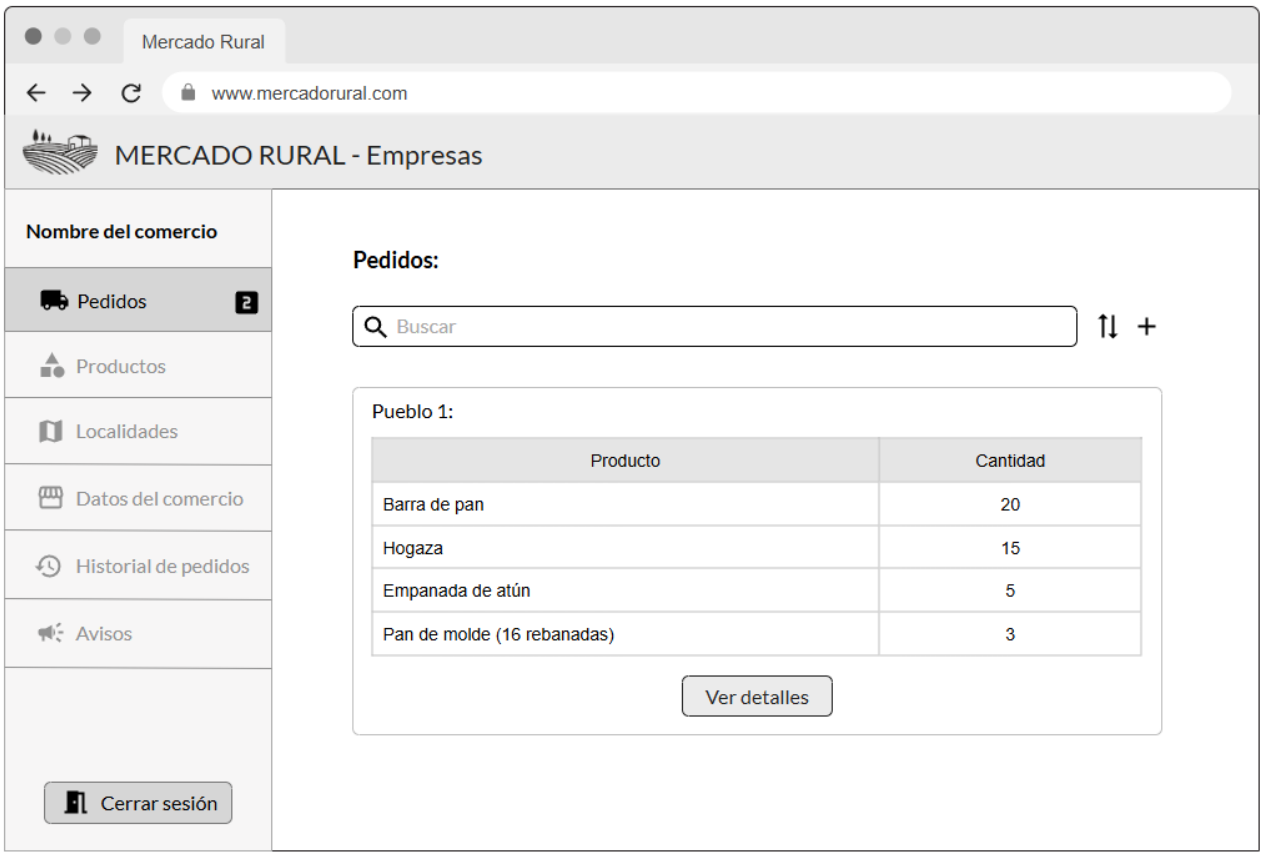

Figura 5.24. Lista de pedidos - empresa

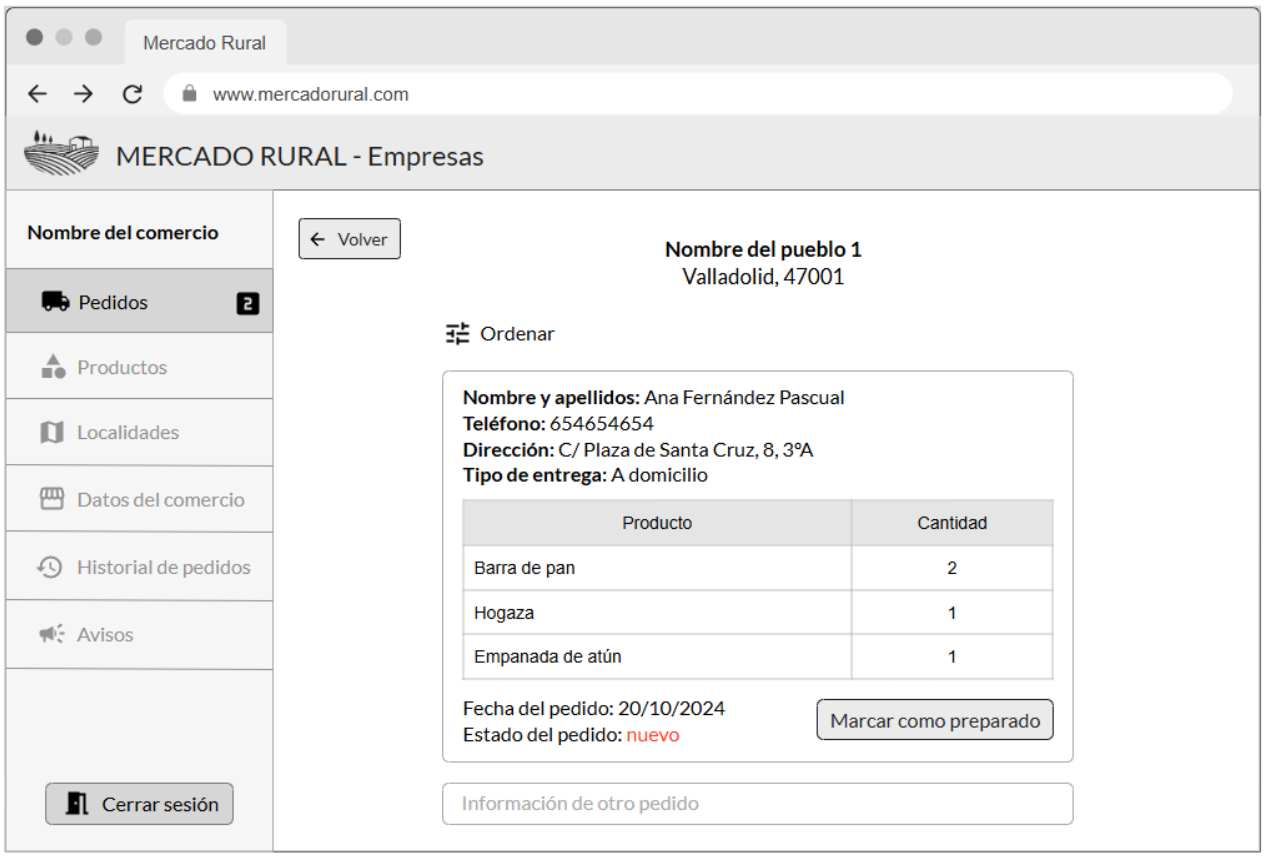

Figura 5.25. Ver detalles de un pedido - empresa

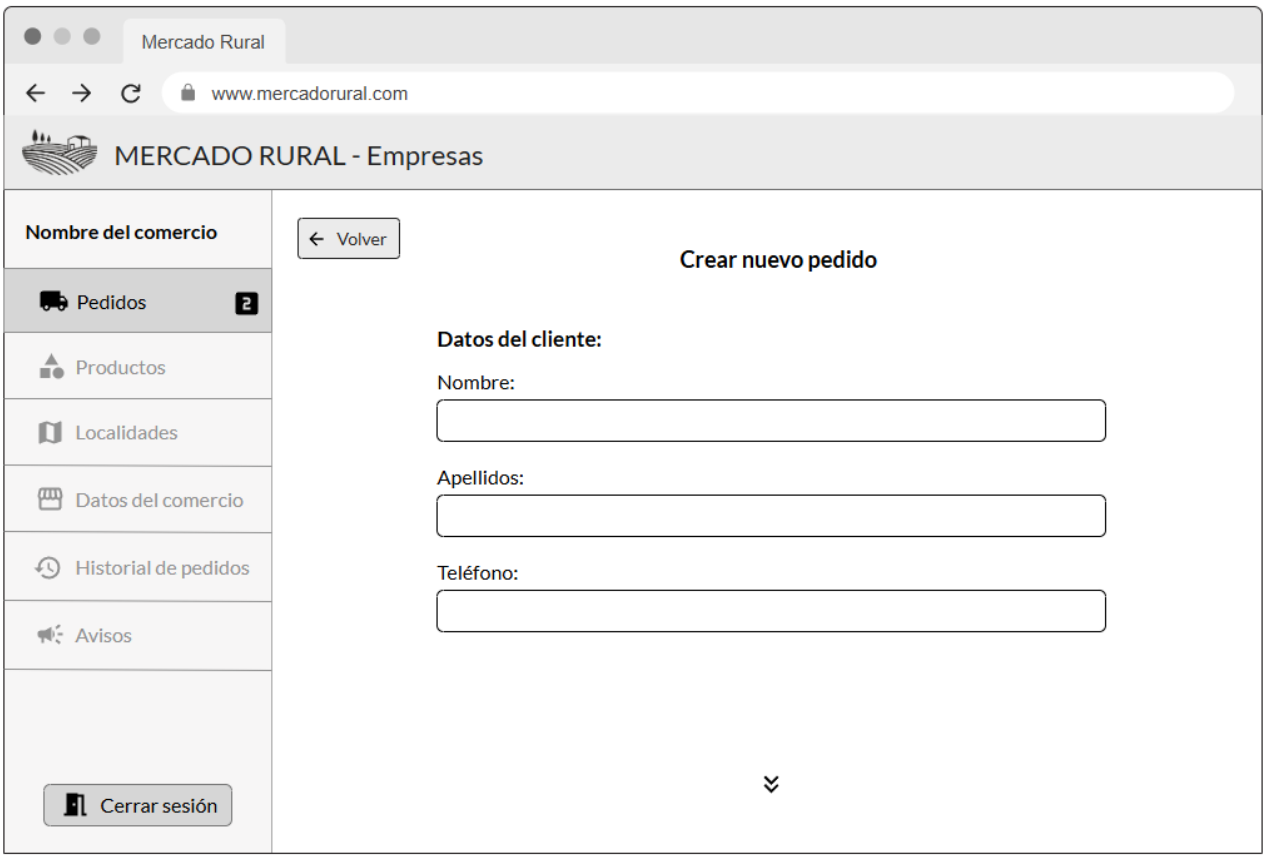

Figura 5.26. Crear nuevo pedido (parte 1) - empresa

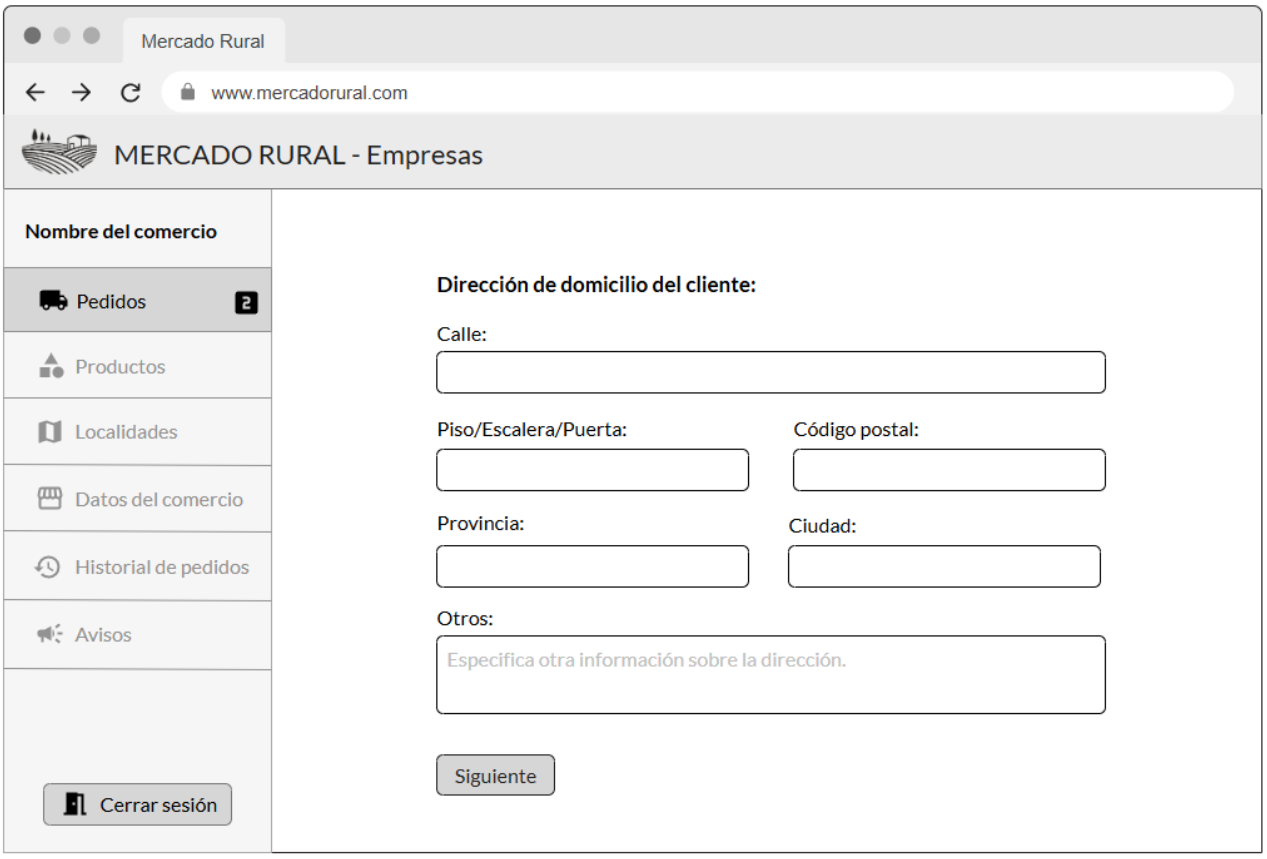

Figura 5.27. Crear nuevo pedido (parte 2) - empresa

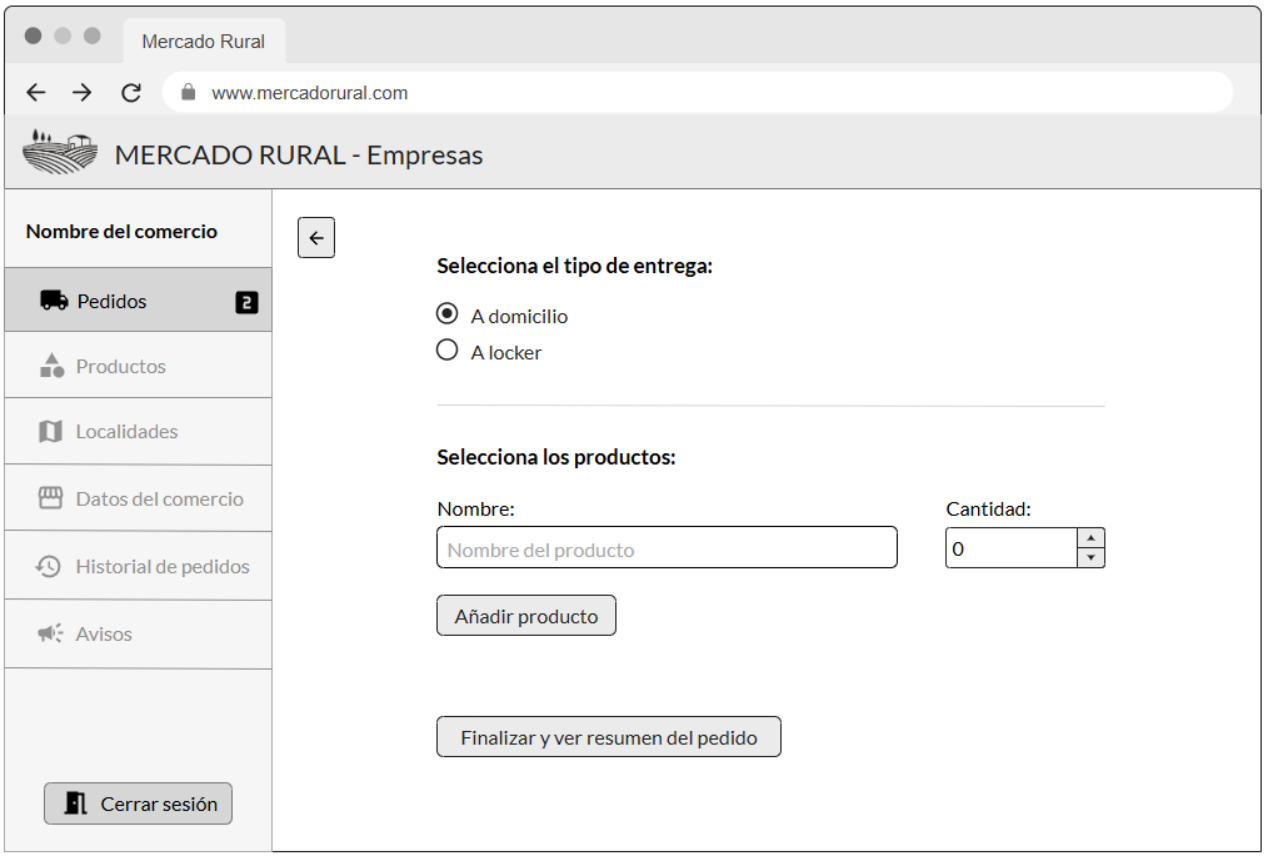

Figura 5.28. Crear nuevo pedido (parte 3) - empresa

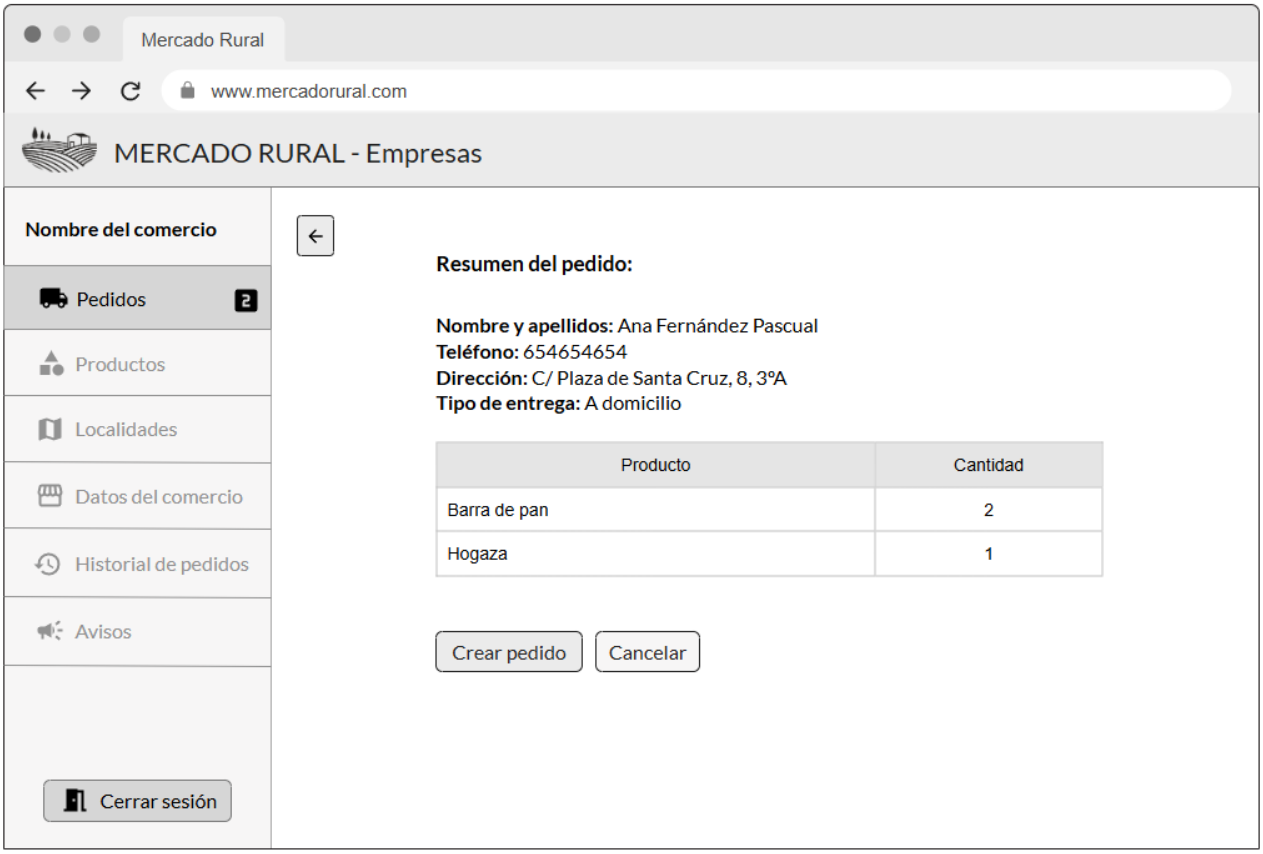

Figura 5.29. Crear nuevo pedido (parte 4) - empresa

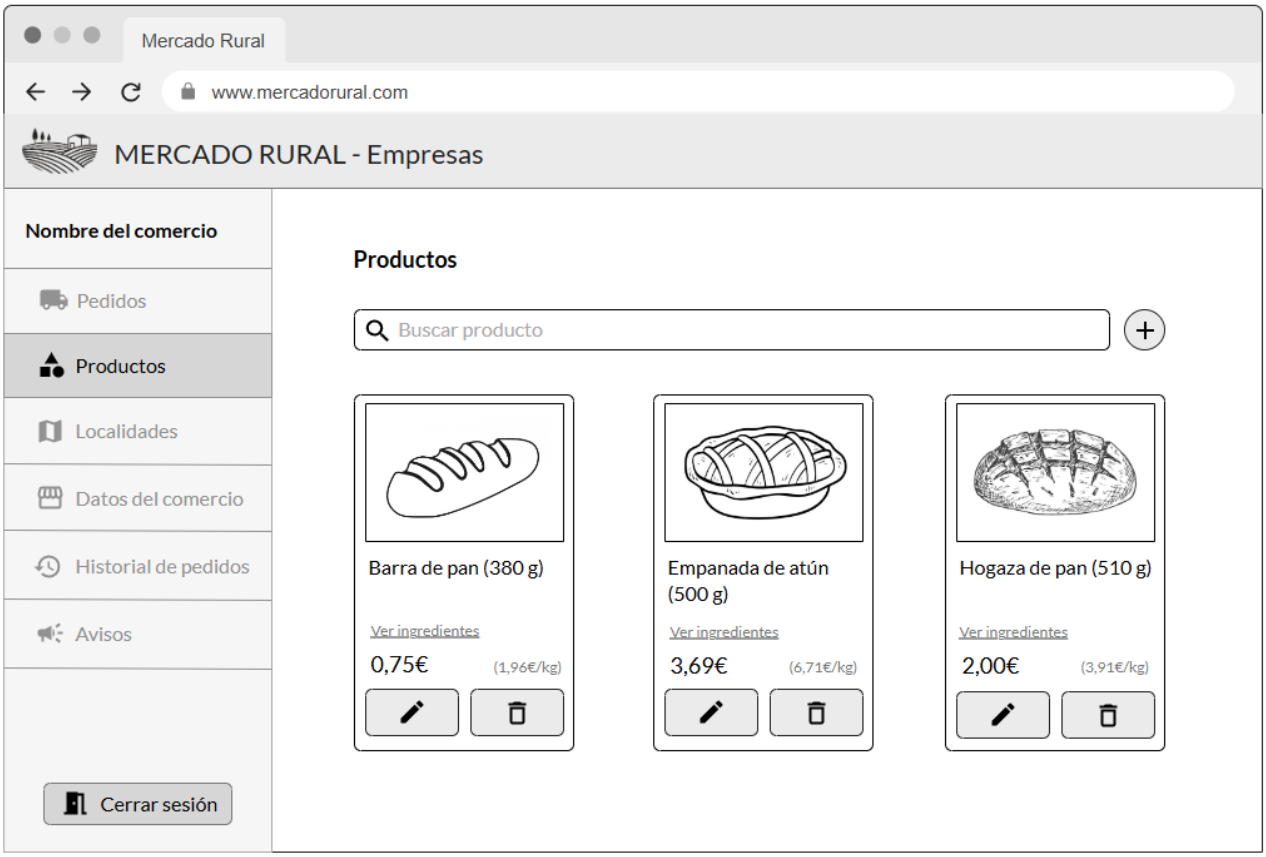

Figura 5.30. Lista de productos - empresa

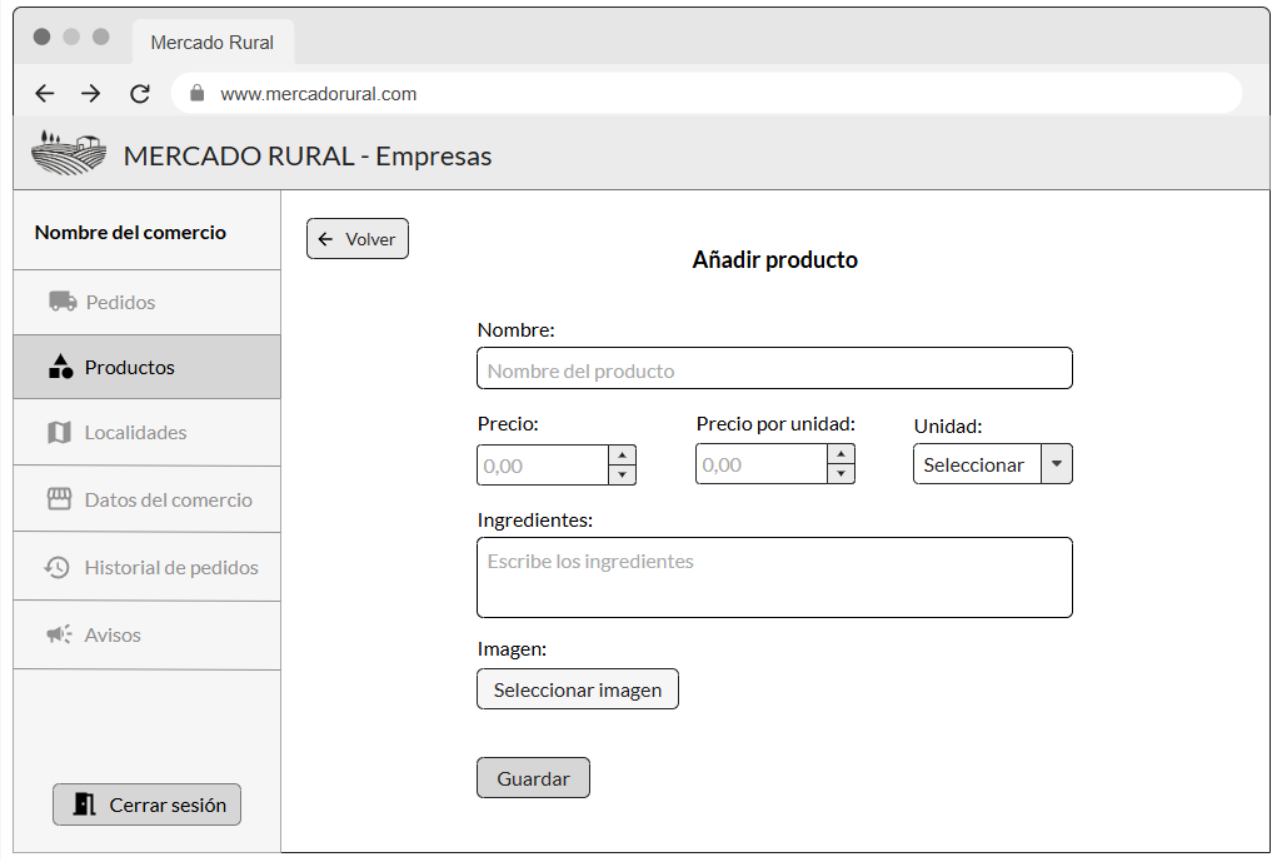

Figura 5.31. Añadir nuevo producto - empresa

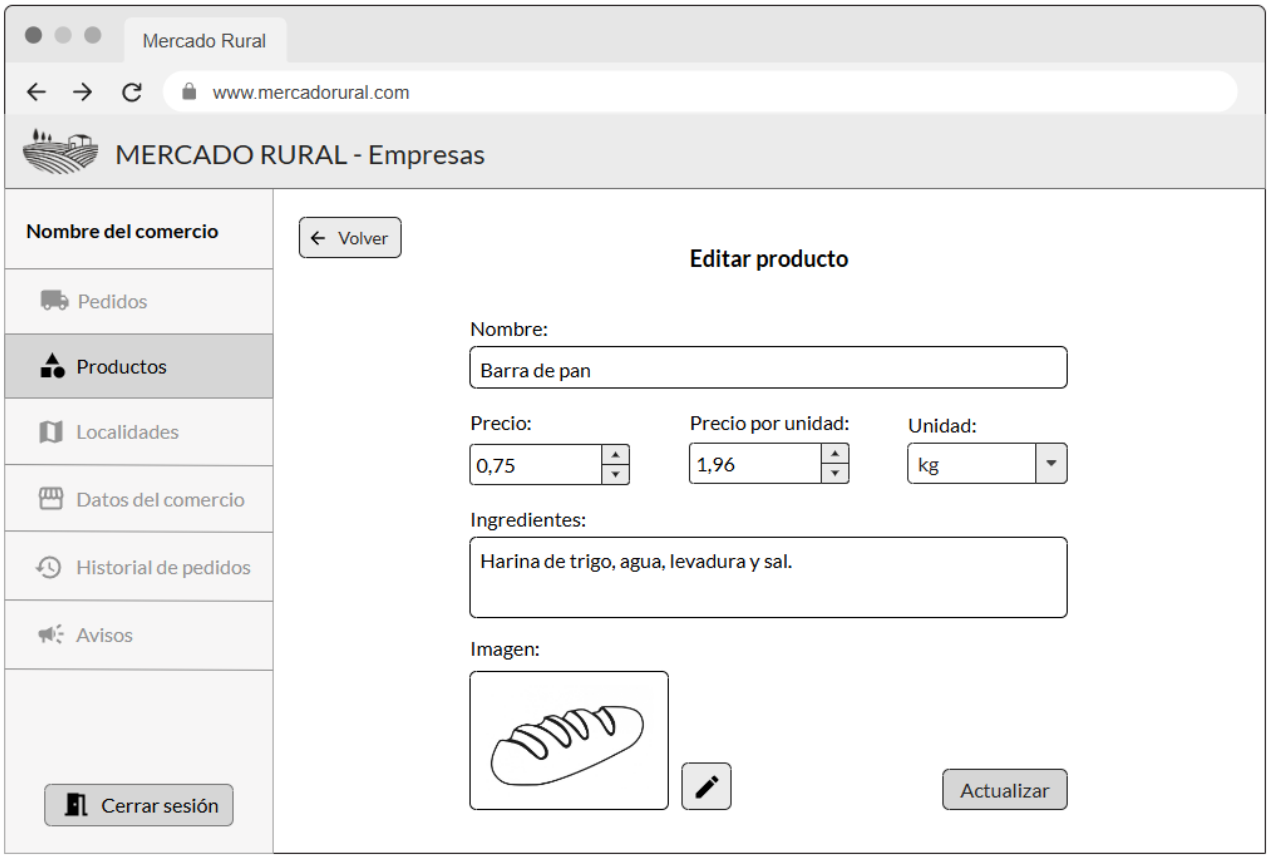

Figura 5.32. Editar producto - empresa

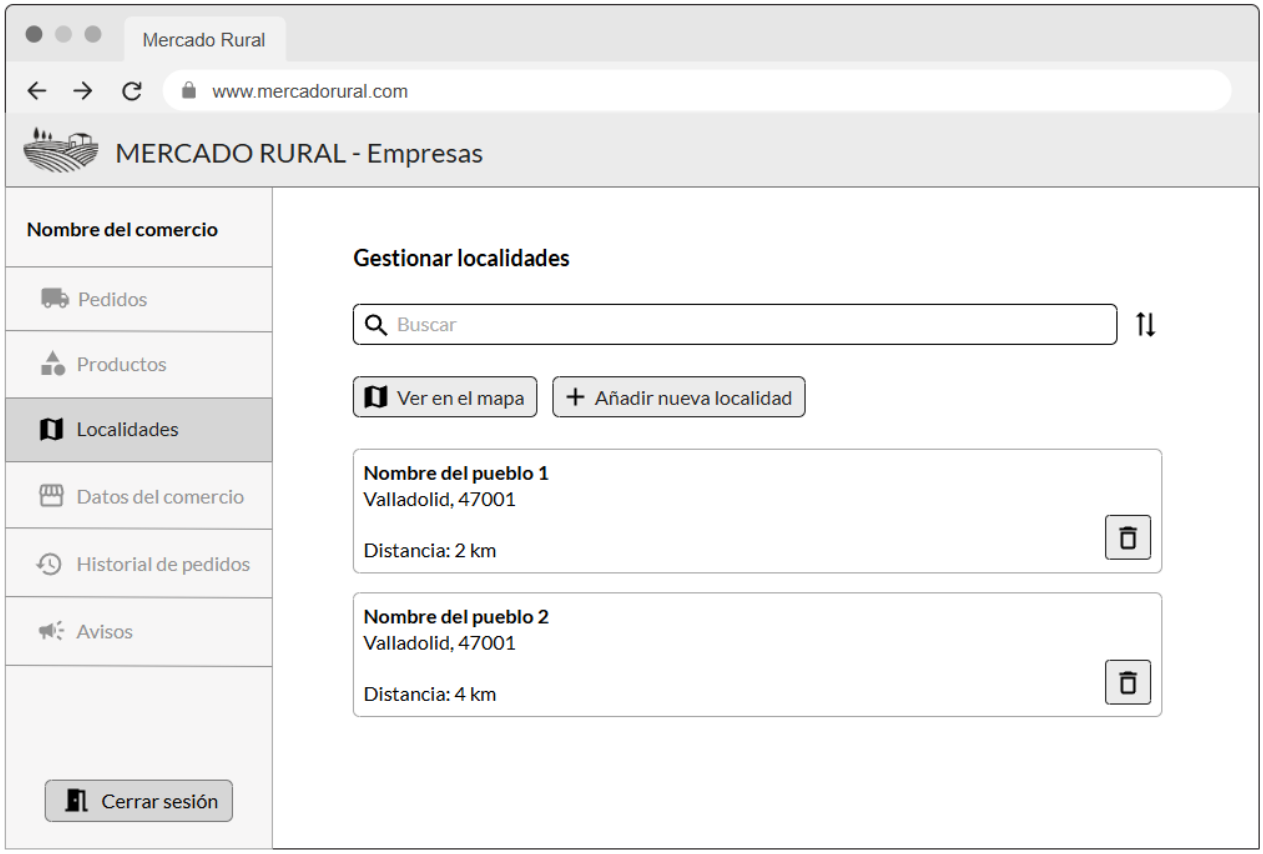

Figura 5.33. Gestionar localidades

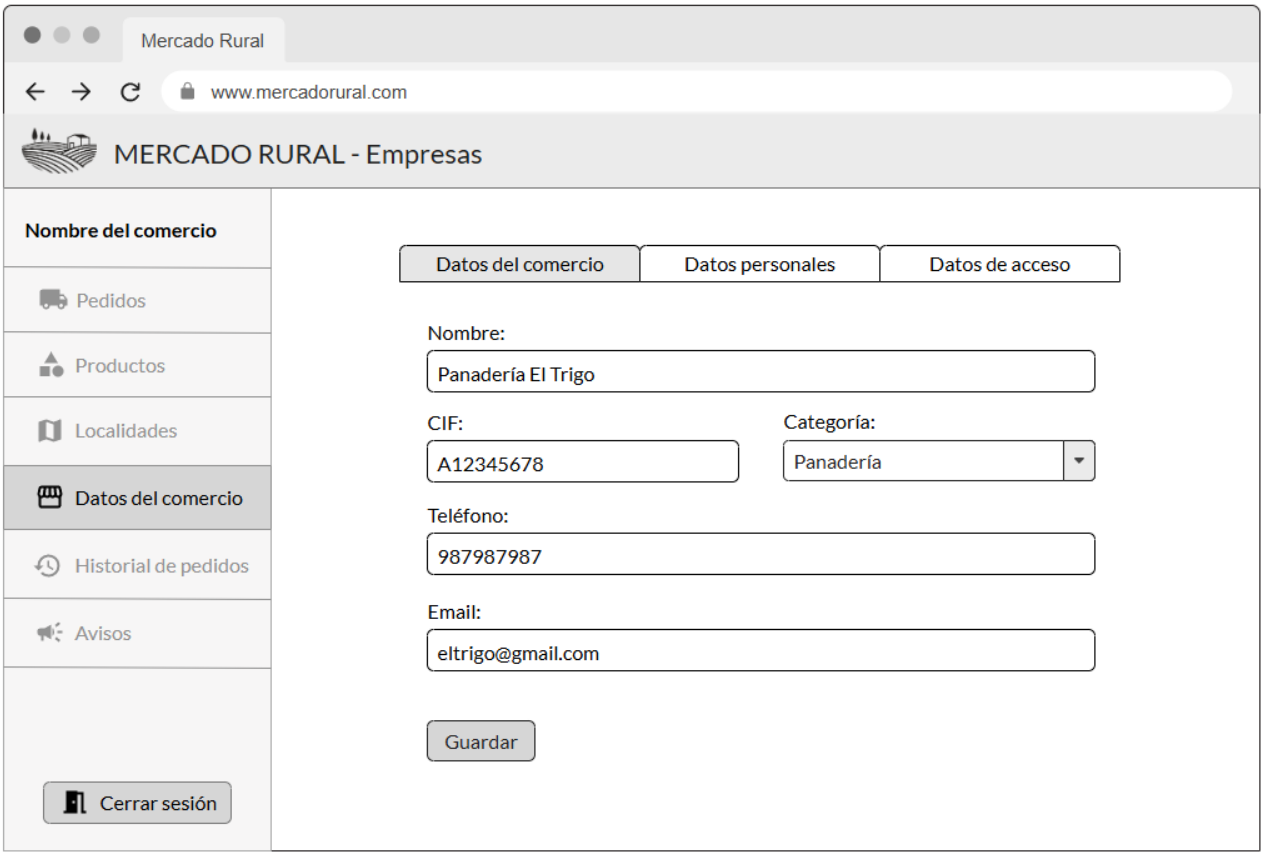

Figura 5.34. Datos del comercio

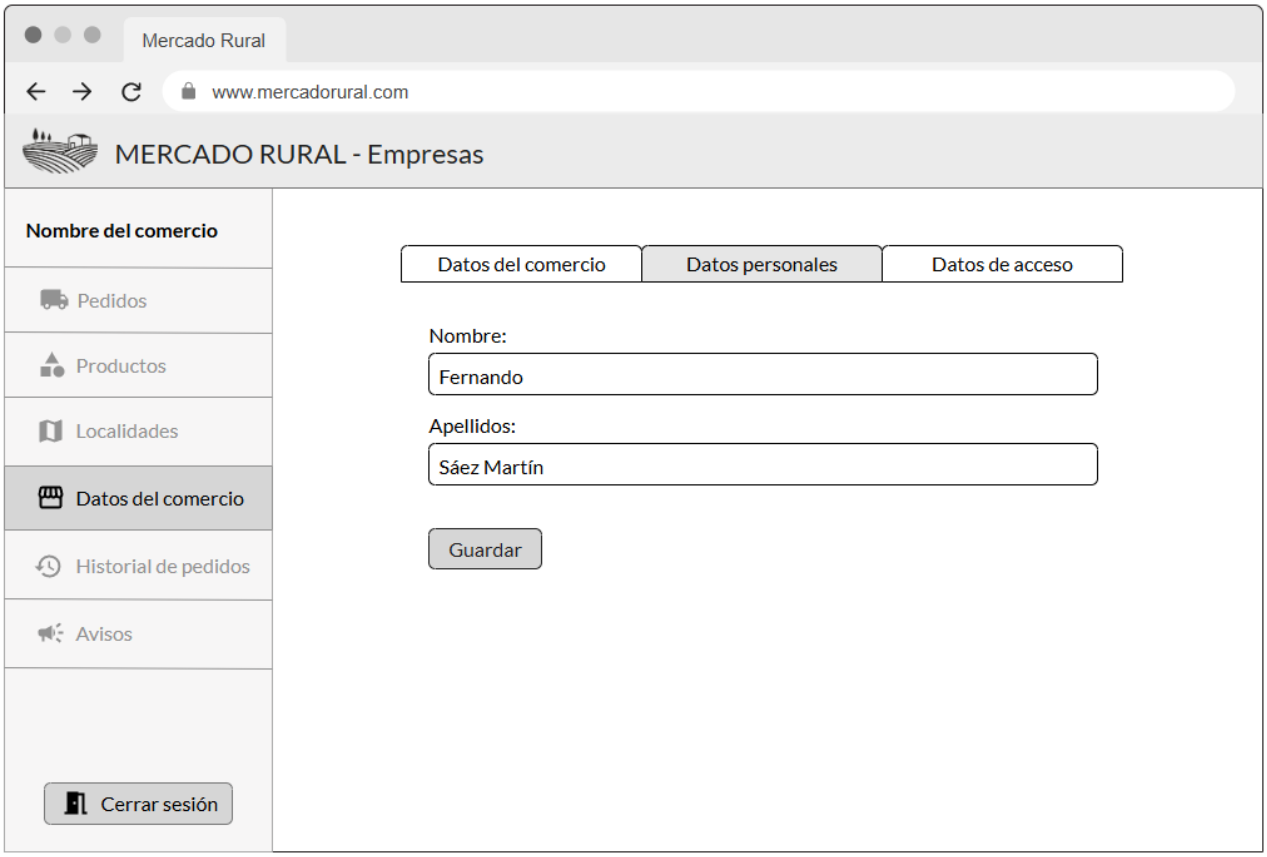

Figura 5.35. Datos personales - empresa

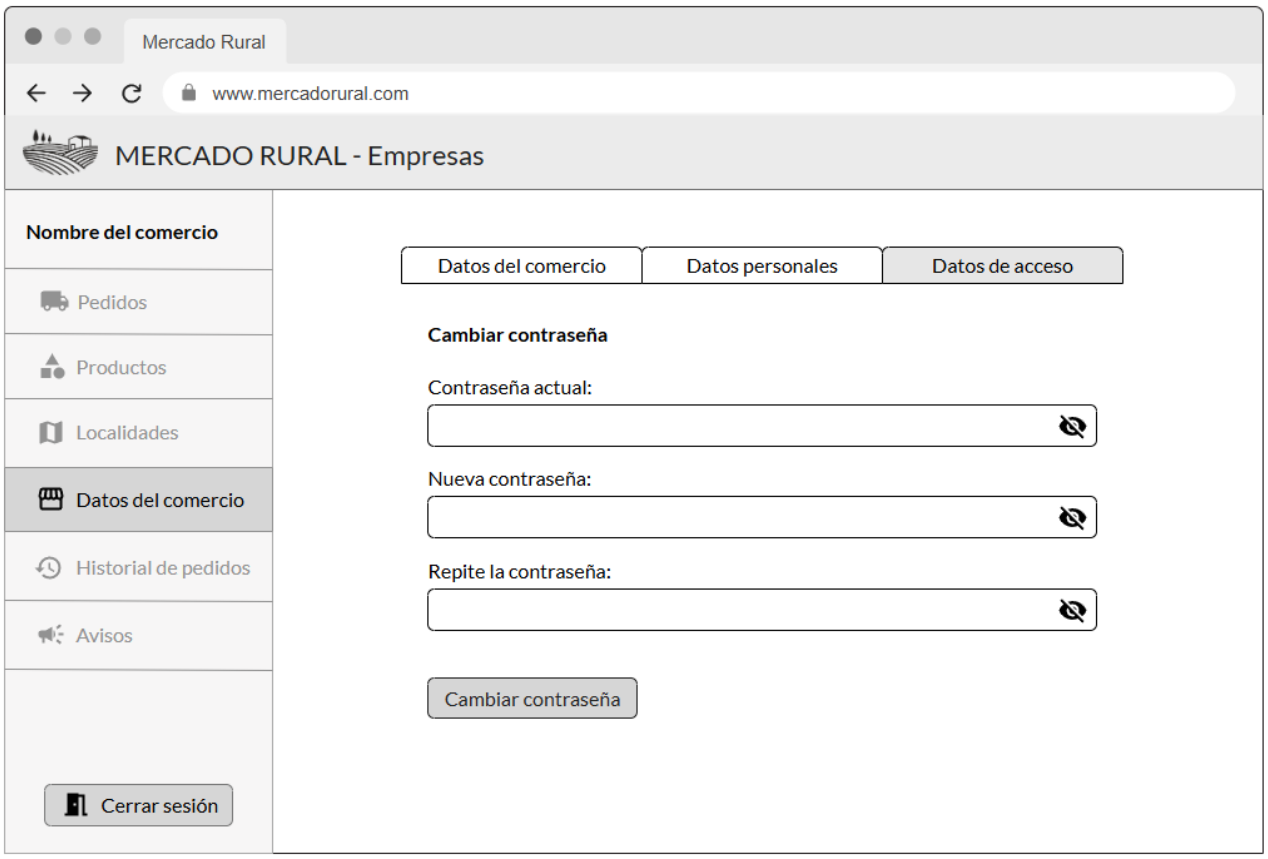

Figura 5.36. Cambiar contraseña - empresa

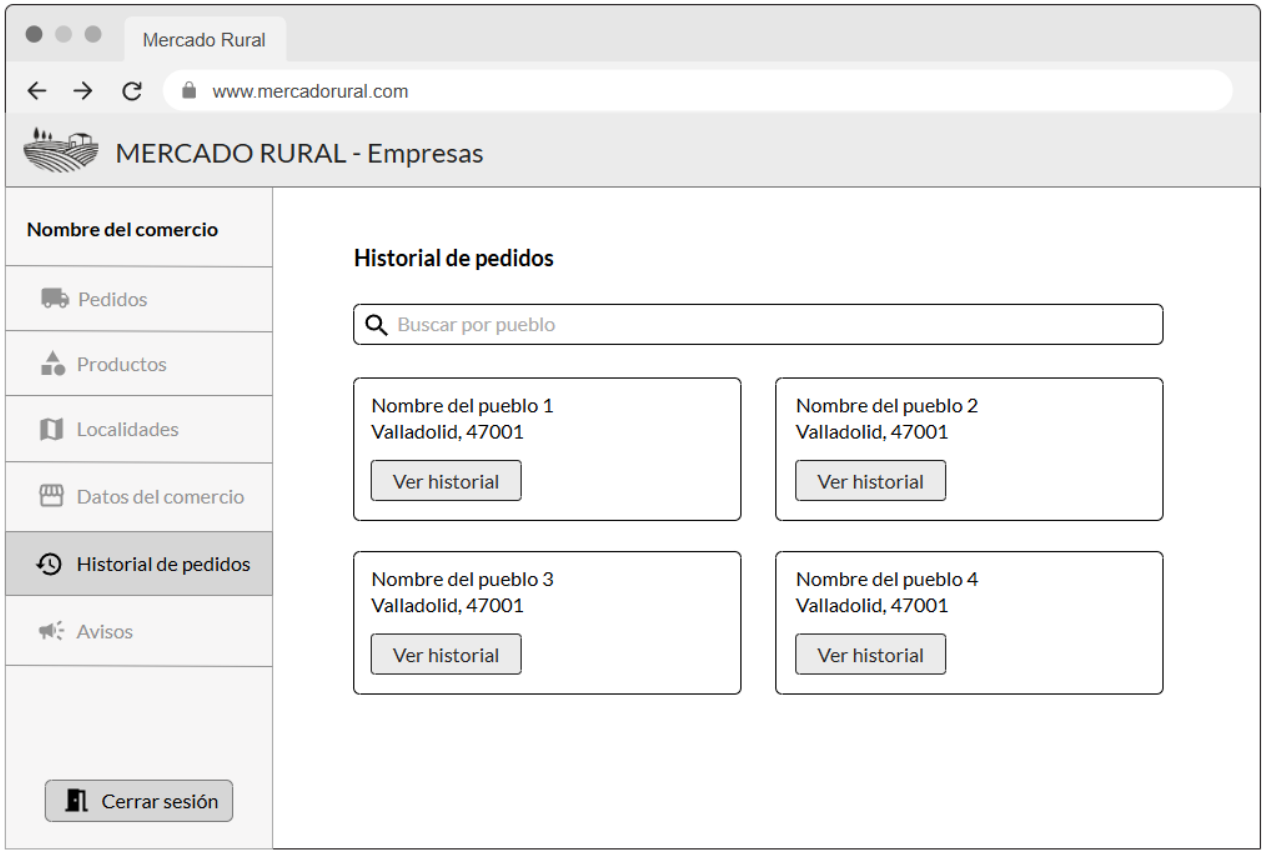

Figura 5.37. Historial de pedidos - empresa

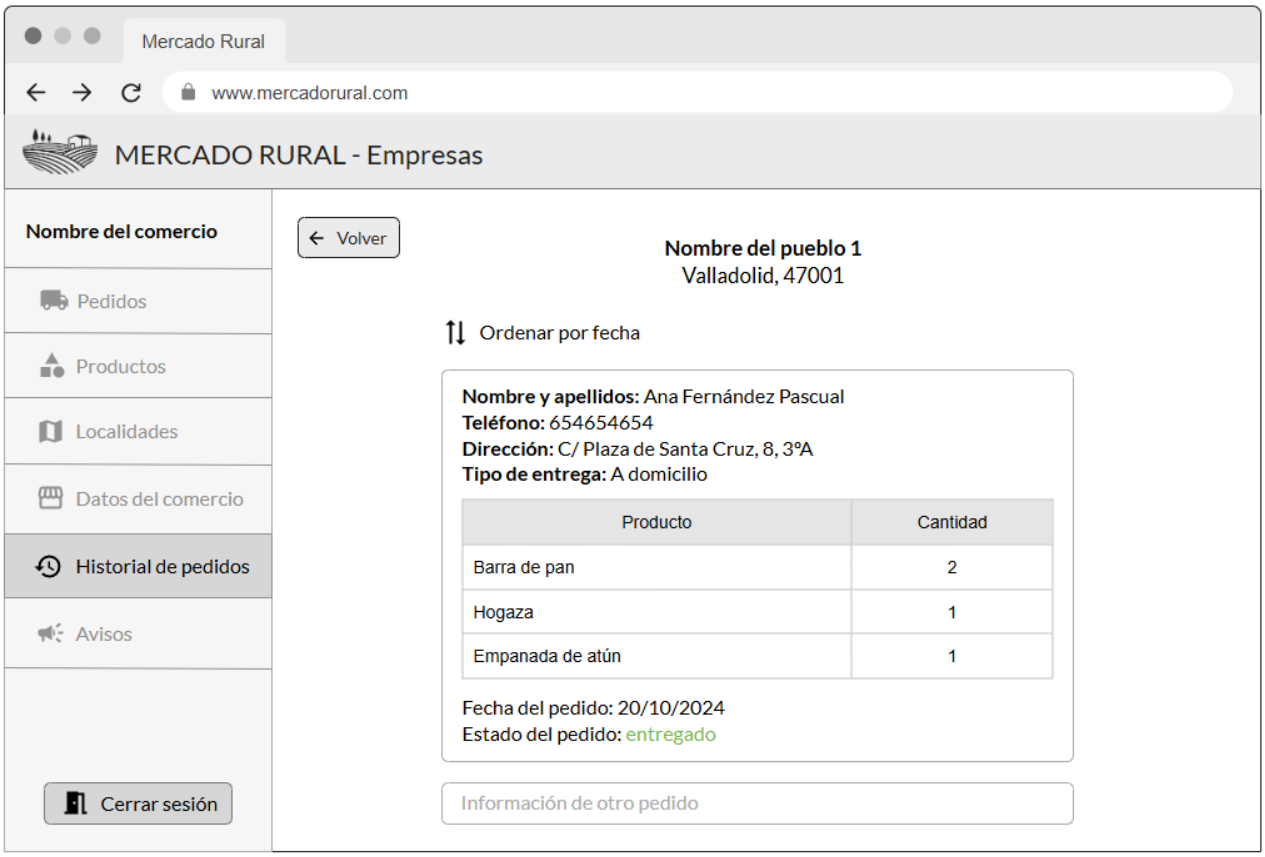

Figura 5.38. Historial de pedidos: detalles - empresa

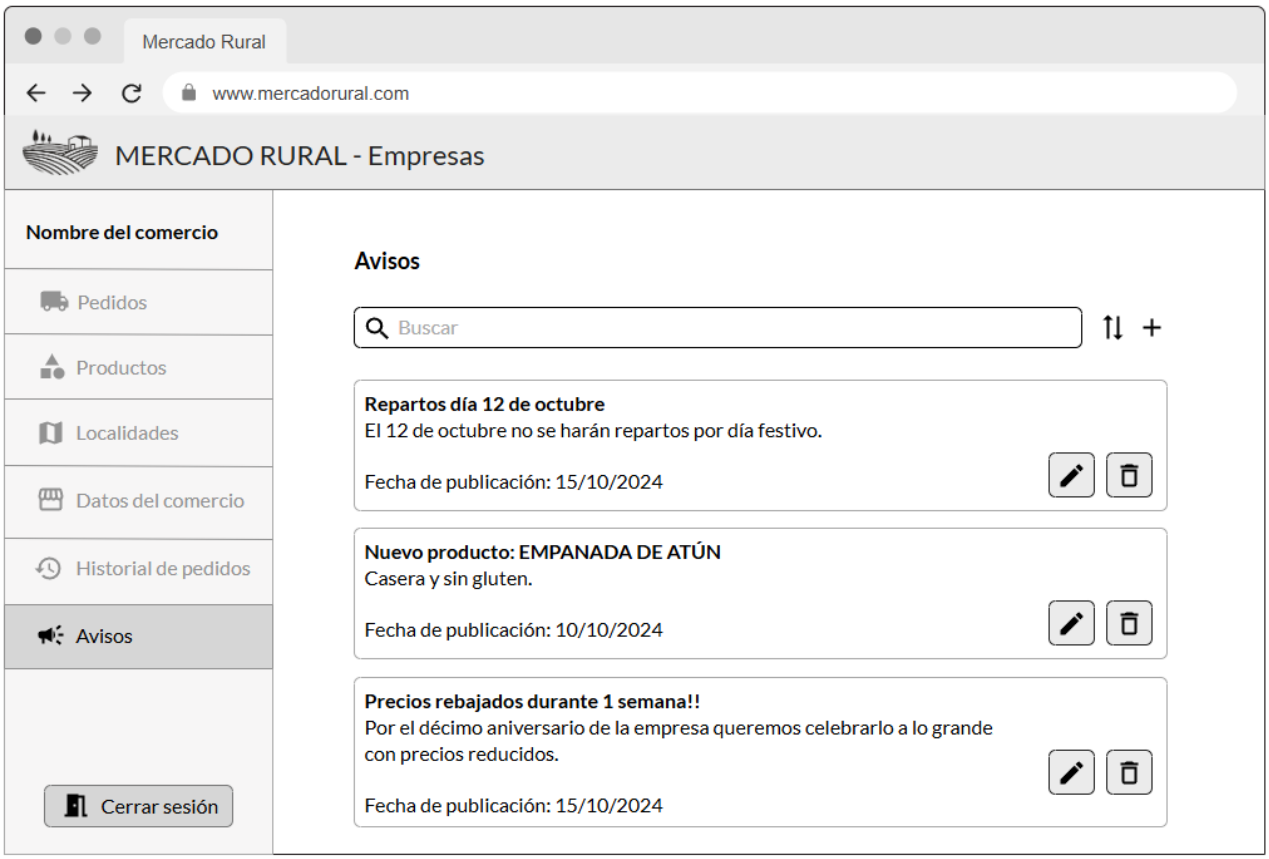

Figura 5.39. Avisos - empresa

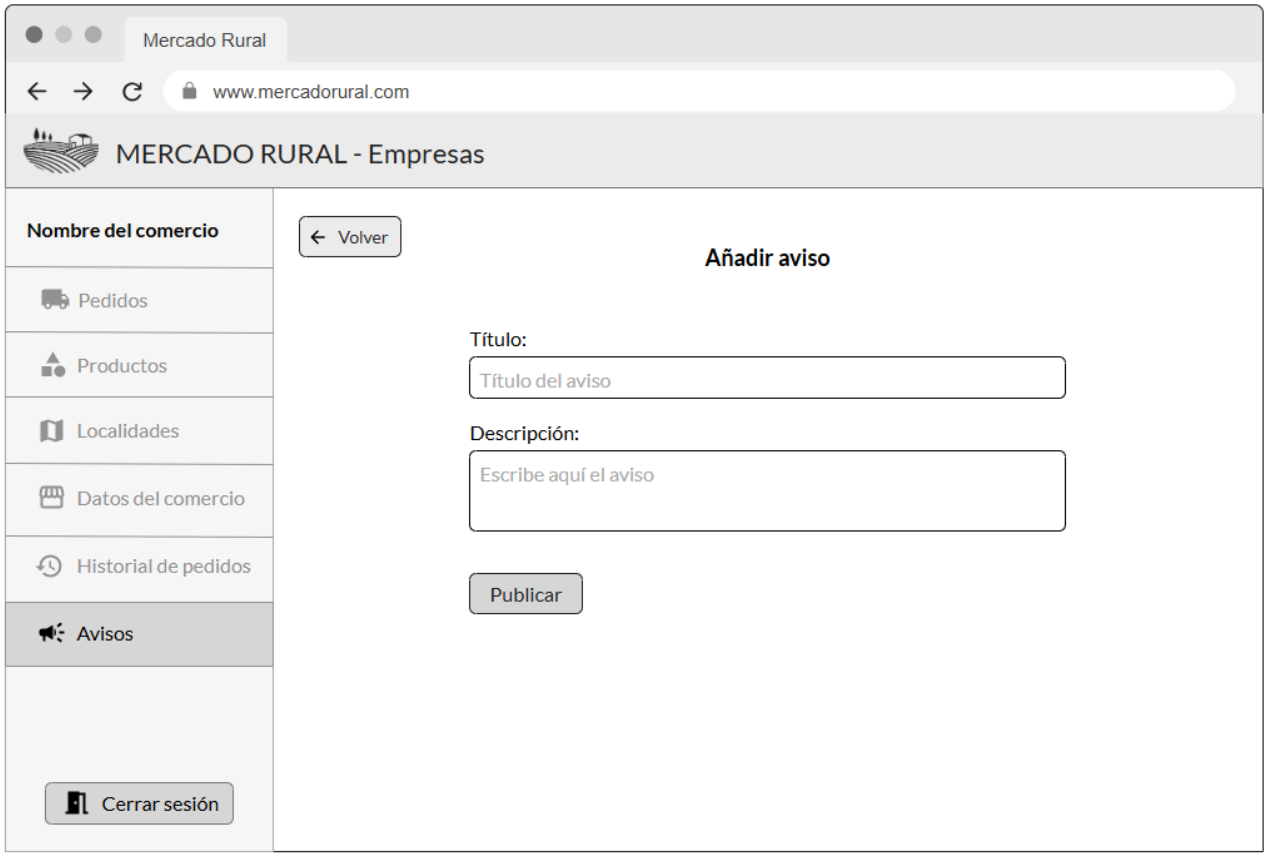

Figura 5.40. Publicar aviso

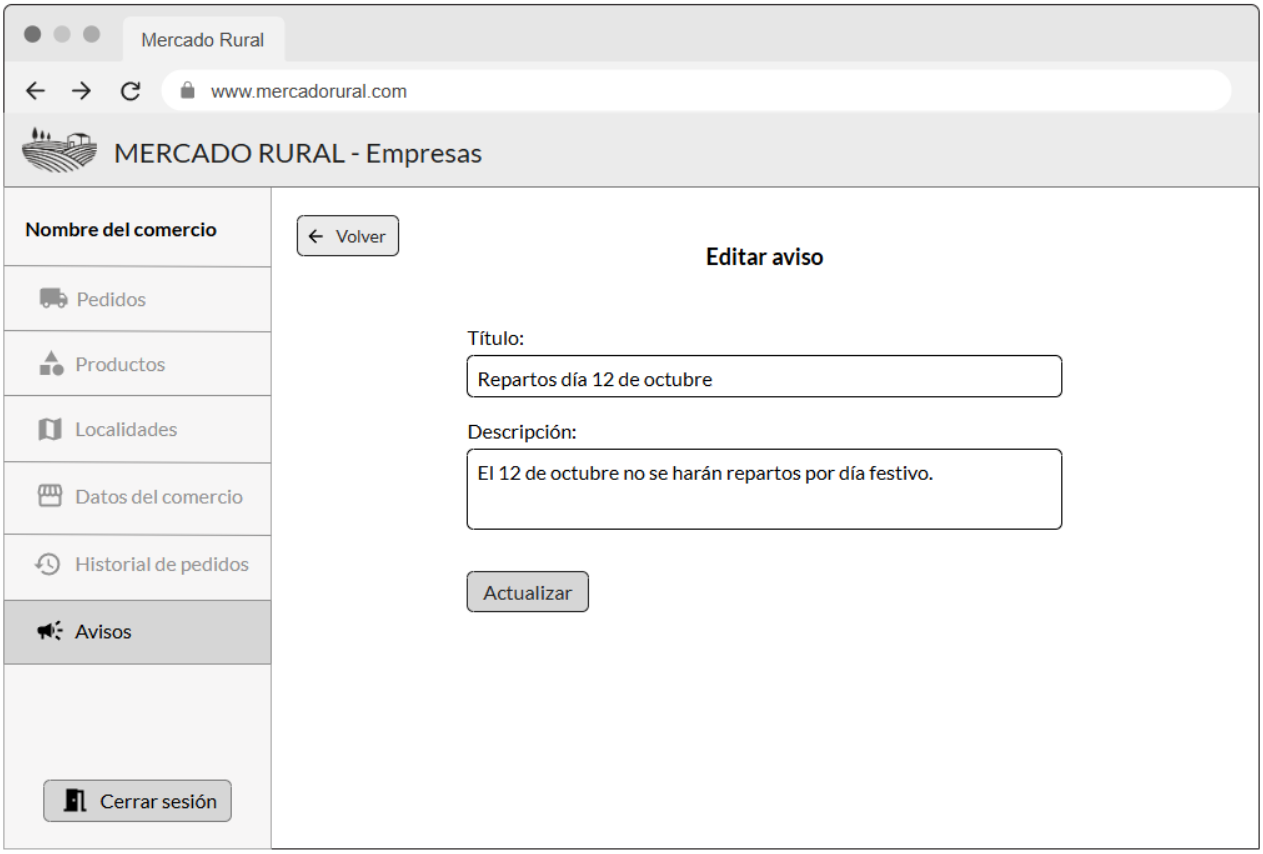

Figura 5.41. Editar aviso

5.5. Diseño de la base de datos

En la Figura 5.42 se muestran las tablas que gestionarán la información necesaria para el correcto funcionamiento del sistema. La base de datos sigue un modelo relacional, donde las distintas tablas almacenan los datos estructurados, y las relaciones entre ellas se definen mediante claves primarias y claves foráneas. Estas claves permiten garantizar la integridad referencial, asegurando que los datos de una tabla se vinculen correctamente con los datos relacionados de otras tablas. Además, se incluyen tablas auxiliares para manejar enumeraciones como las categorías de negocio o los estados de los pedidos, lo que facilita la gestión de los datos constantes dentro del sistema.

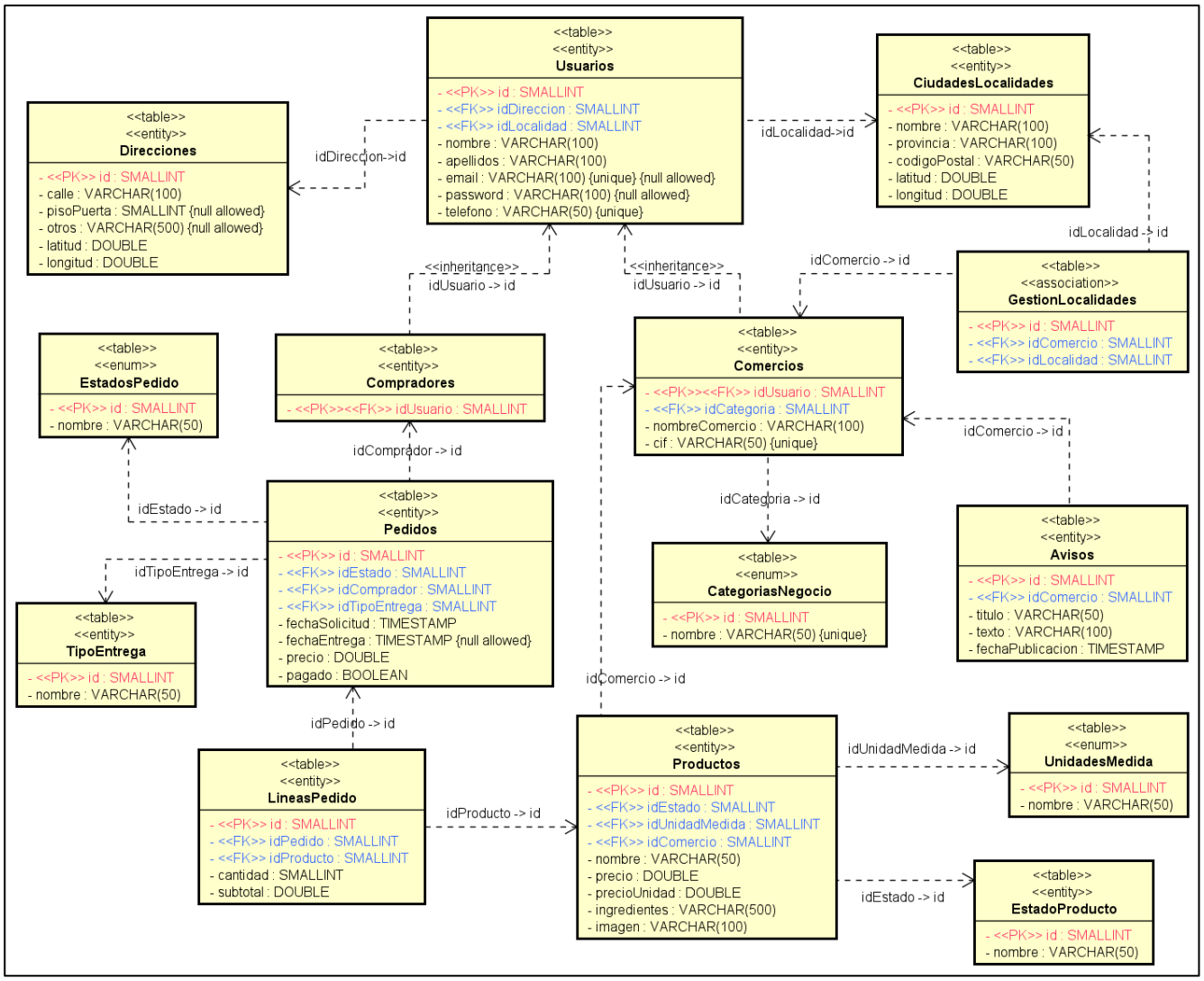

Figura 5.42. Diagrama de la base de datos

Un aspecto importante que mencionar es que, como se muestra en la tabla "Usuarios", los campos *email* y *password* pueden ser nulos. Sin embargo, esto no aplica a los compradores y vendedores registrados en la plataforma, quienes deben proporcionar obligatoriamente estos datos.

Esta decisión se ha tomado para poder representar a aquellos compradores que prefieren no crearse una cuenta en Mercado Rural, pero que aún así desean realizar pedidos. En estos casos, los vendedores pueden registrar sus compras desde su panel de gestión (CU04 Registrar nuevo pedido y boceto de la Figura 5.26). Aunque estos compradores no tienen credenciales de acceso, los datos que se recopilan sobre ellos (especificados en la Tabla 3.4) se almacenan en el sistema para facilitar futuras compras.

5.6. Diseño arquitectónico

El diseño arquitectónico del sistema define la estructura general de la aplicación y la forma en que sus elementos interactúan entre sí. Se ha optado por una arquitectura modular con una separación clara entre el cliente (frontend) y el servidor (backend). Este enfoque mejora la mantenibilidad, escalabilidad y flexibilidad, ya que permite desarrollar, actualizar y escalar cada parte del sistema de manera independiente.

Cliente

El cliente se corresponde con la aplicación Angular, que implementa el patrón MVVM (Model-View-ViewModel) para organizar la lógica de presentación. Este modelo permite una mejor separación entre la interfaz de usuario y la lógica de presentación, facilitando la gestión del estado y la actualización dinámica de los datos en la interfaz.

Servidor

El servidor está implementado en Java con Spring Boot, siguiendo el patrón MVC (Model-View-Controller) para estructurar la lógica de negocio y el acceso a datos.

A diferencia de arquitecturas monolíticas, donde el cliente y el servidor podrían formar una única unidad de despliegue, en este caso, ambos son componentes independientes que solo se comunican mediante peticiones HTTP. No existe una dependencia directa entre ellos en términos de implementación, ya que cada uno puede evolucionar por separado sin afectar al otro, siempre que la API de comunicación se mantenga estable. Por este motivo, en la Figura 5.43 y Figura 5.44, se presentan los diagramas del cliente y del servidor como entidades separadas, reflejando esta independencia estructural.

La explicación detallada de cada paquete del frontend y el backend se verá más adelante en el apartado 6.1.1. Organización del código.

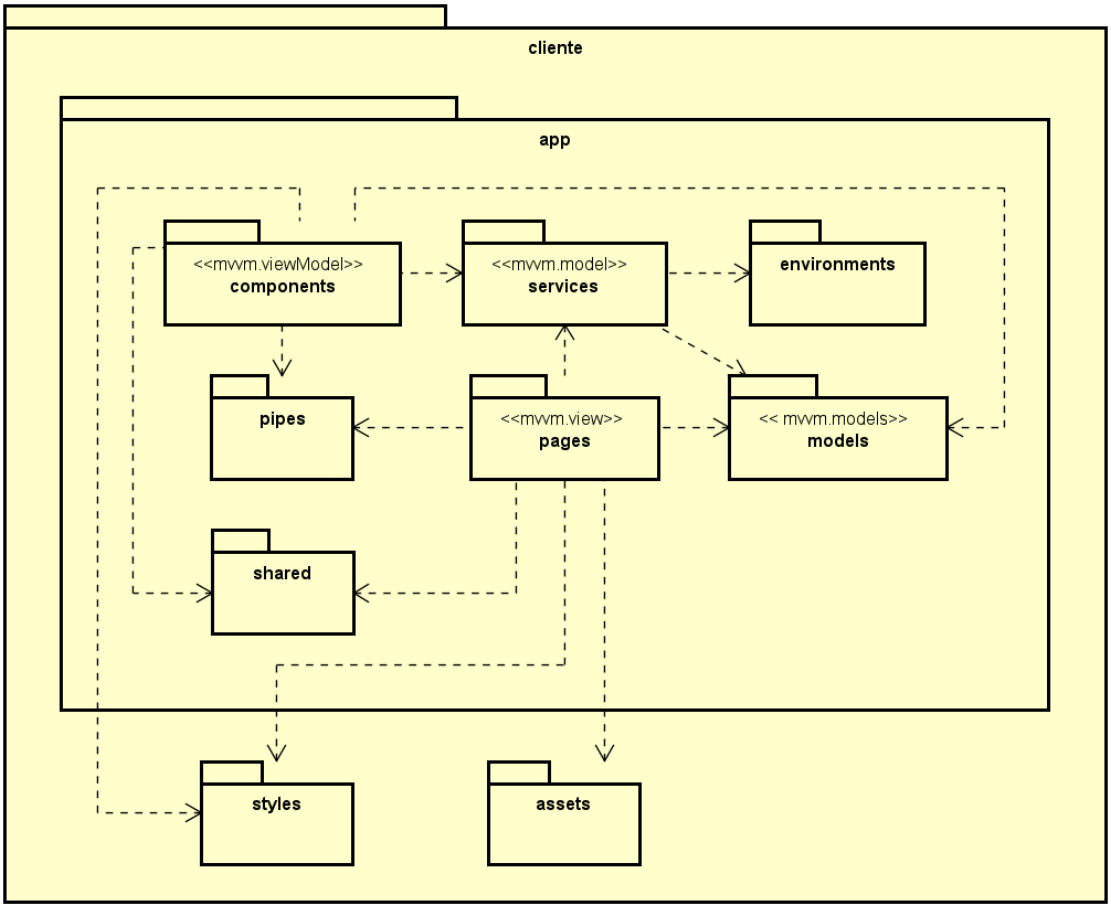

Figura 5.43. Arquitectura en el lado del cliente

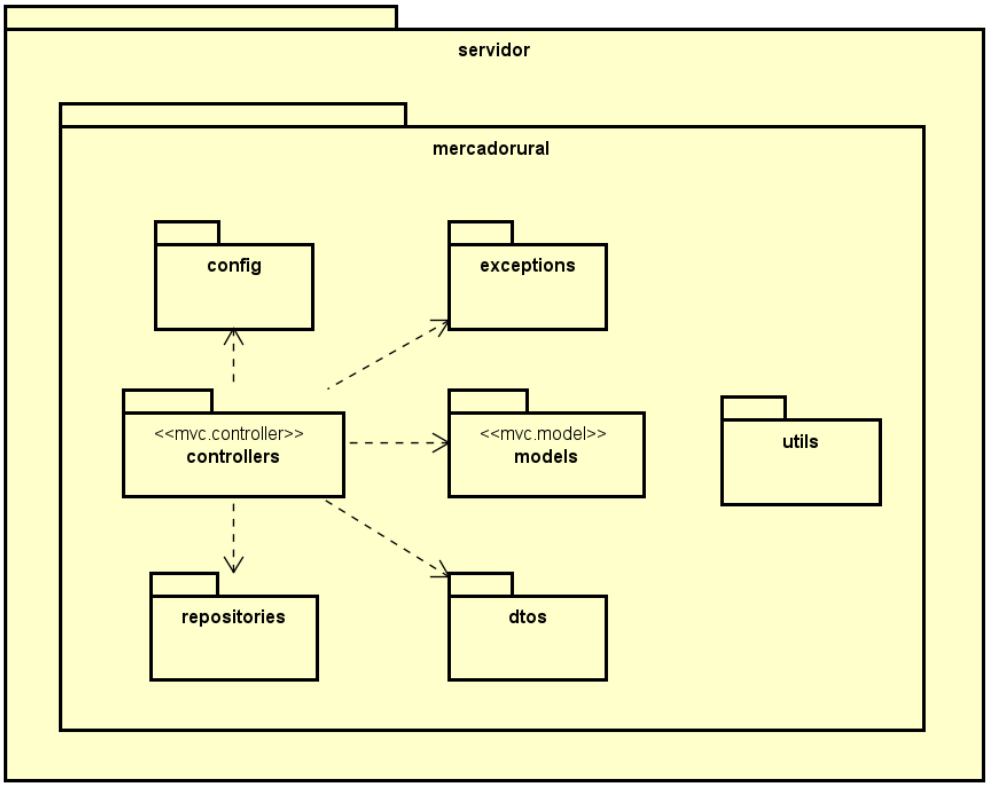

Figura 5.44. Arquitectura en el lado del servidor

5.7. Diseño basado en componentes

El diseño basado en componentes sirve para organizar la interfaz de usuario en elementos reutilizables e independientes, lo que facilita su mantenimiento y escalabilidad. En el caso de este proyecto, el frontend se ha desarrollado con Angular, un framework que fomenta el uso de componentes como unidad fundamental de construcción. Cada componente en Angular sigue el patrón MVVM (Model-View-ViewModel), donde la plantilla (View) está vinculada a un archivo de lógica (ViewModel), permitiendo la interacción con los datos y la presentación dinámica de la información.

Para representar la estructura y relaciones entre los distintos componentes de la aplicación, se utiliza un diagrama de componentes (Figura 5.45), que permite visualizar cómo están organizados, qué dependencias existen entre ellos y cómo se comunican. Este tipo de diagramas es útil para comprender la arquitectura del frontend y garantizar una estructura modular y bien definida.

En el diagrama, los componentes están representados por colores para facilitar su comprensión:

- **Componentes azules.** Páginas, que representan la estructura principal de la aplicación (aquellas con rutas y llamadas a la API).
- **Componentes verdes.** Componentes utilizados por las páginas, que pueden tener o no llamadas a la API.
- **Componentes amarillos.** Componentes auxiliares, como diálogos o cabeceras, que no tienen llamadas a la API pero son necesarios para la interacción del usuario.

Esta explicación se desarrolla con más detalle en el apartado 6.1.1. Organización del código.

5.8. Diseño de la comunicación entre objetos

Para representar la comunicación entre los diferentes objetos del sistema, se han utilizado diagramas de secuencia. Estos diagramas, propios de UML, permiten visualizar el flujo de mensajes entre los distintos elementos del sistema durante la ejecución de un caso de uso concreto, proporcionando una visión detallada de la interacción entre componentes.

En este apartado, se presenta el diagrama de secuencia correspondiente al caso de uso CU17 Iniciar sesión para vendedores, el cual pertenece a la historia de usuario HU01: Acceso de usuarios (login y registro). Se ha elegido este caso de uso porque es un proceso sencillo que permite ilustrar de manera clara la interacción entre el frontend y el backend sin entrar en detalles más complejos. Además, dado que el resto de los casos de uso siguen un esquema de comunicación similar, este diagrama sirve como referencia general para entender la dinámica del sistema.

A continuación, en la Figura 5.46 se muestra el diagrama de secuencia correspondiente. Para mantener la claridad y evitar un diagrama excesivamente extenso, se han desglosado ciertas partes en dos subdiagramas (Figura 5.47 y Figura 5.48), los cuales representan secciones específicas del flujo principal y forman parte del proceso global de inicio de sesión.

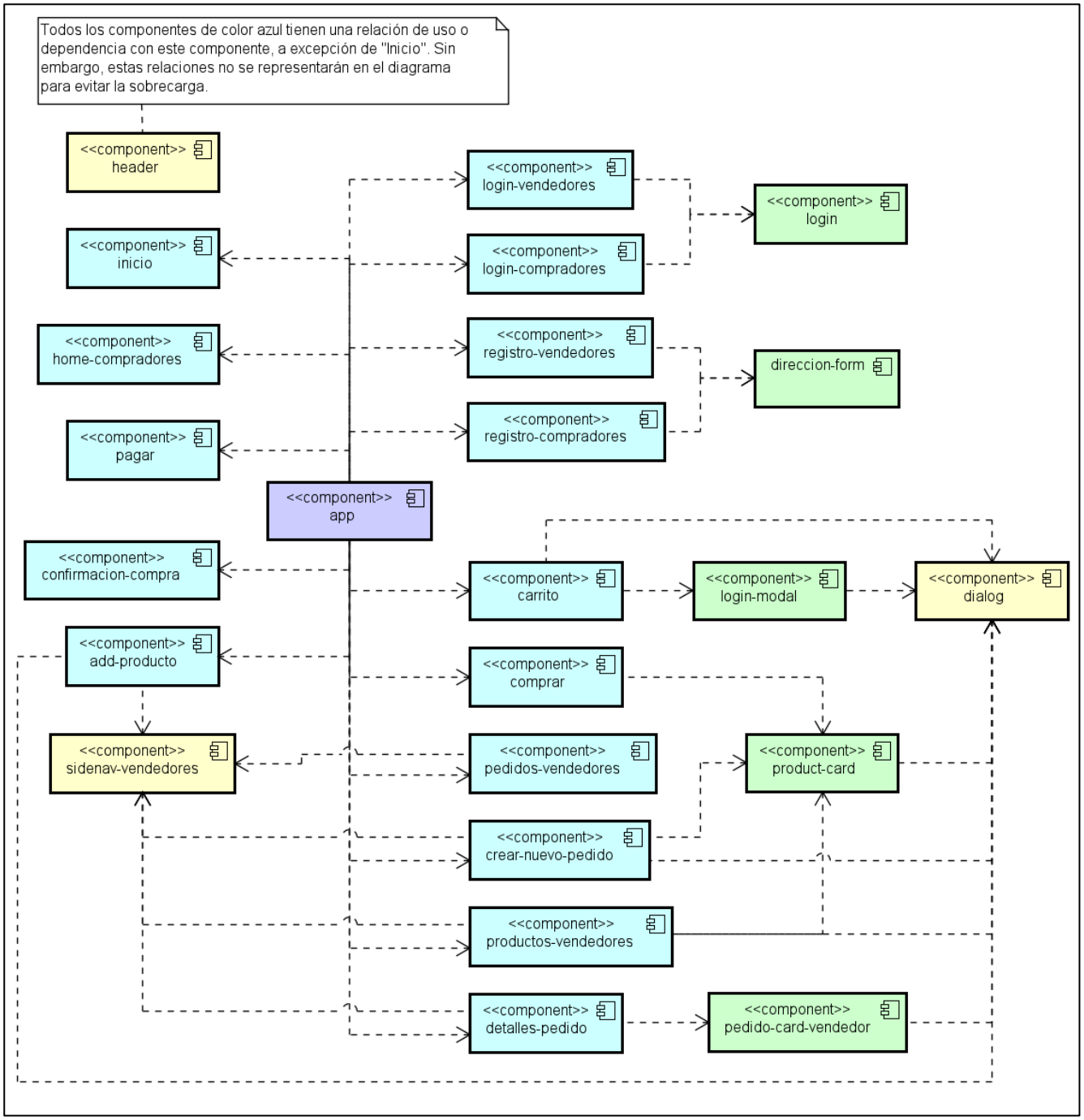

Figura 5.45. Diagrama de componentes del frontend

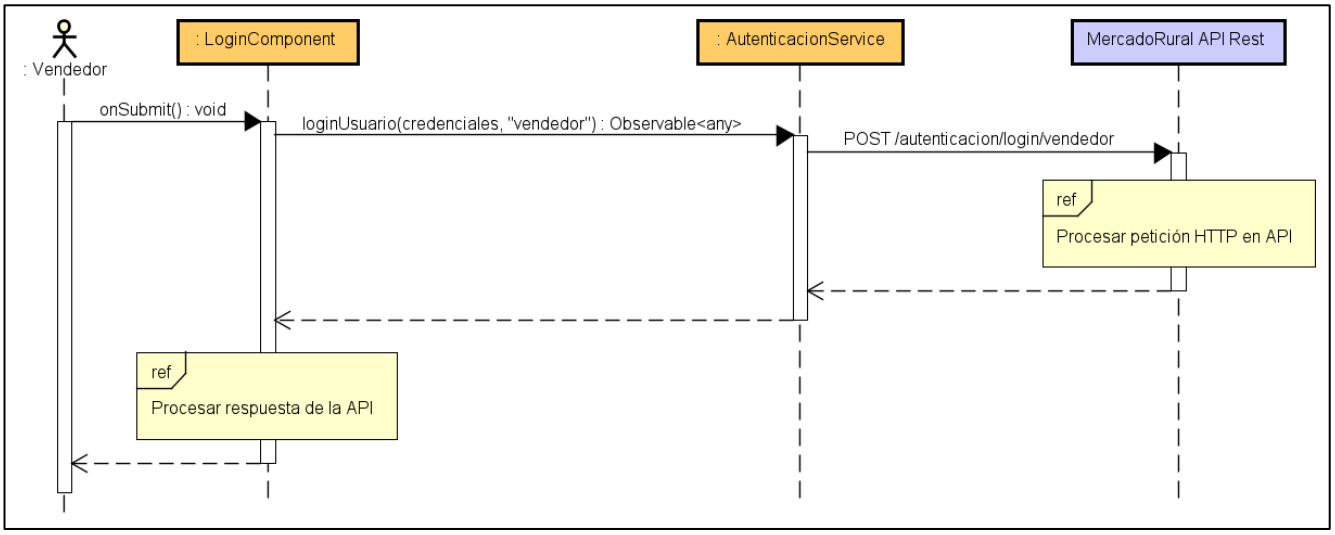

Figura 5.46. Diagrama de secuencia "Iniciar sesión – vendedores"

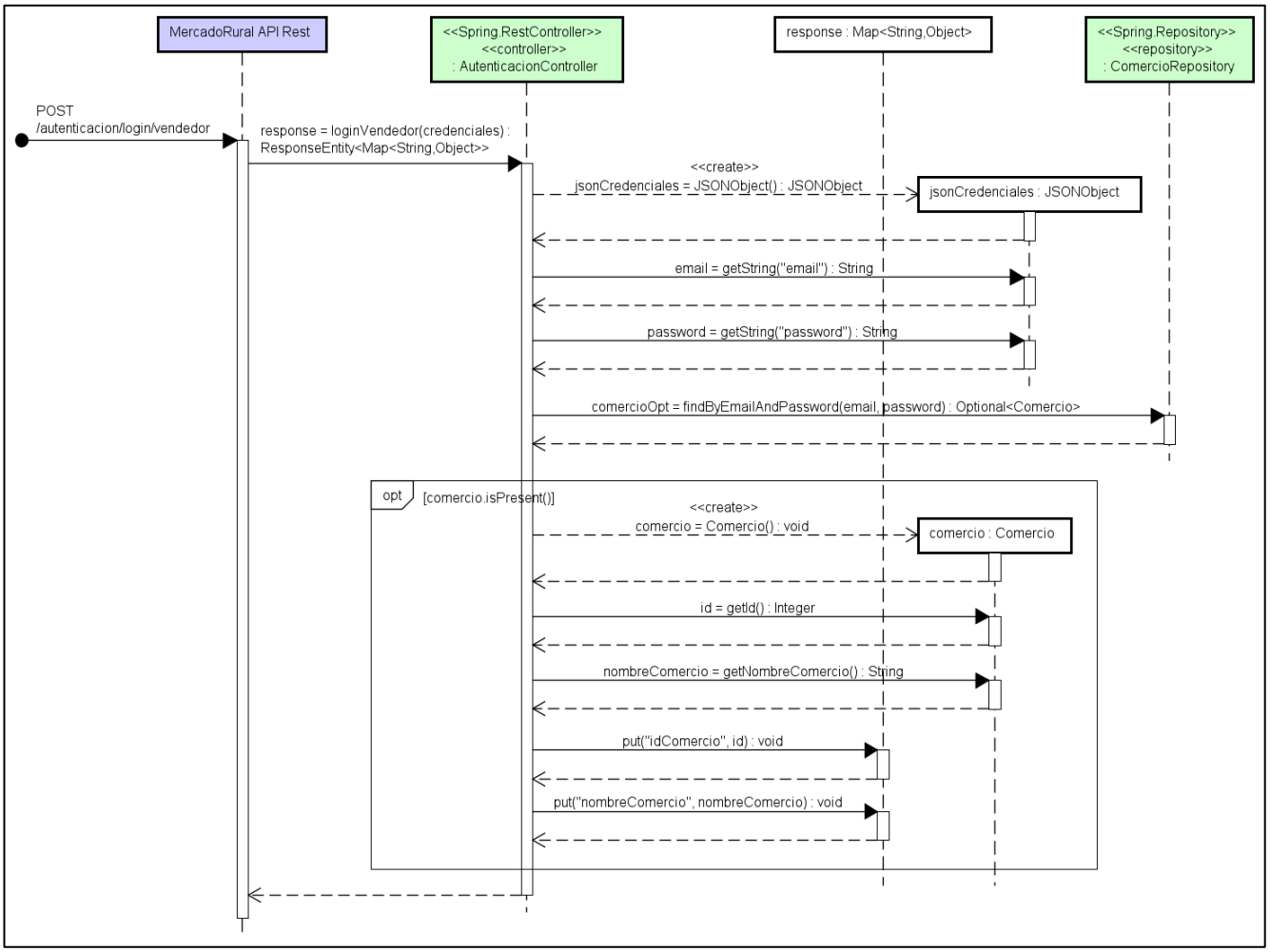

Figura 5.47. Subdiagrama de secuencia "Procesar petición HTTP"

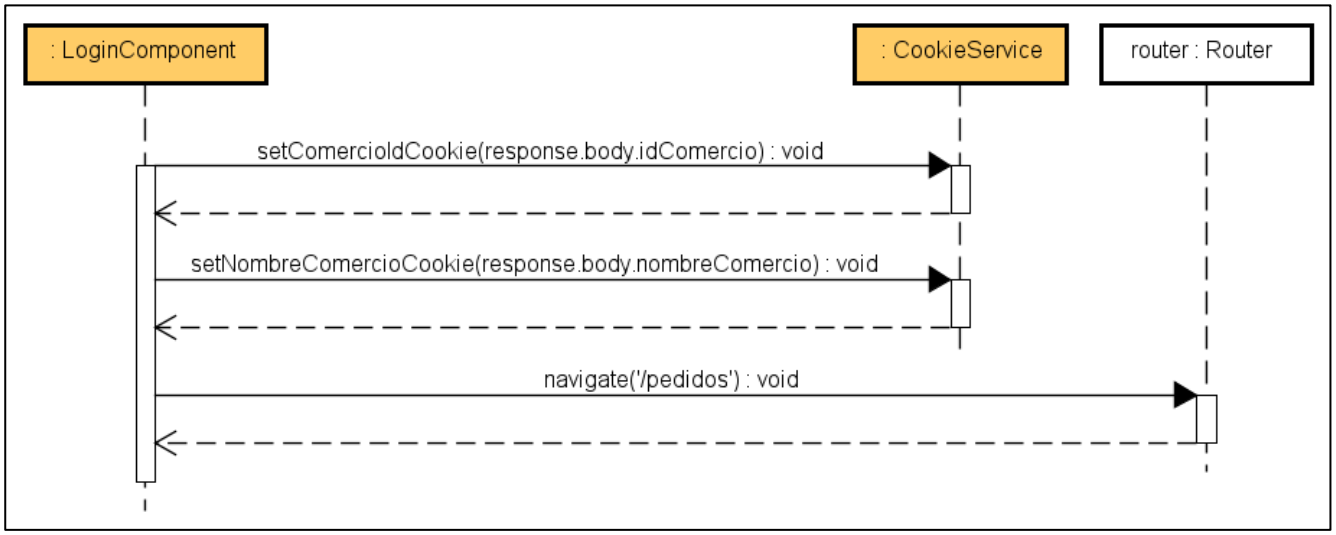

Figura 5.48. Subdiagrama de secuencia "Procesar respuesta de la API"

5.9. Diseño del despliegue

En la Figura 5.49 se muestra el diagrama de despliegue en un entorno local. La aplicación se ejecuta en un navegador que carga los recursos estáticos desde un servidor Node.js. A partir de ahí, se realizan peticiones HTTP a la API, que se ejecuta sobre un servidor Tomcat embebido proporcionado por Spring Boot. Esta API, a su vez, interactúa con la base de datos MySQL mediante consultas SQL a través de JDBC. Todo el sistema se encuentra desplegado en un mismo dispositivo, lo que es adecuado para pruebas y desarrollo.

En un entorno de producción, el frontend estaría alojado de forma independiente en un servidor web o CDN (Content Delivery Network) que es una red de servidores distribuidos que optimiza la entrega de contenido estático para mejorar la velocidad y la experiencia del usuario, que sería accesible desde cualquier dispositivo. La API y la base de datos se desplegarían en servidores separados, típicamente en soluciones en la nube como AWS, Azure o Google Cloud, lo que permitiría una mayor escalabilidad, rendimiento y seguridad. Esta separación facilita la optimización de recursos y la gestión del sistema en entornos de alta demanda.

Fuentes utilizadas: (Hostinger, 2024).

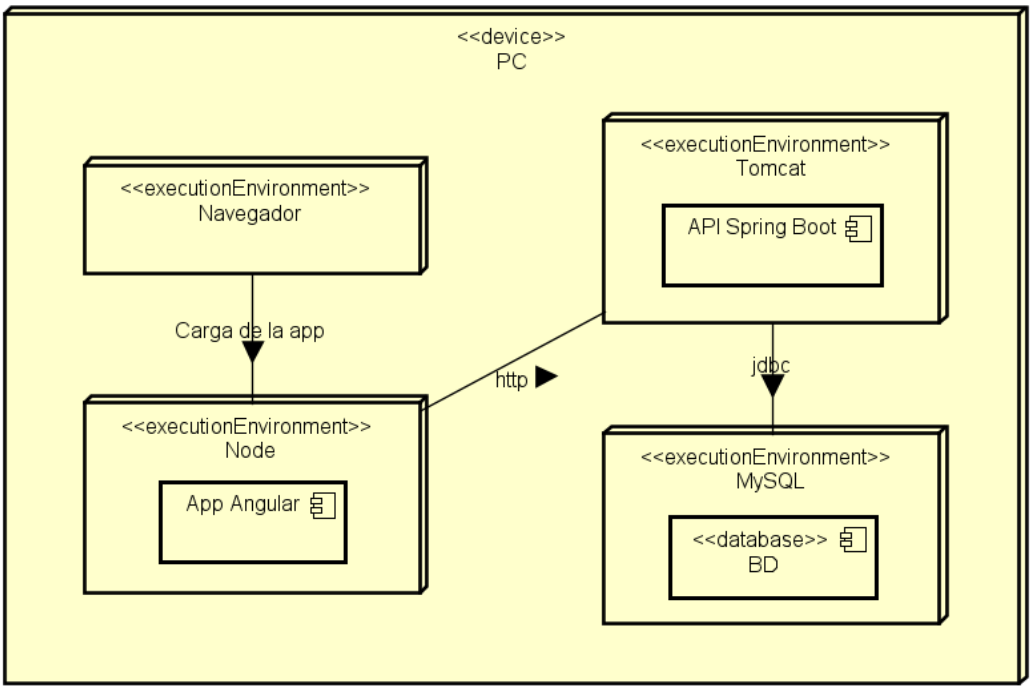

Figura 5.49. Diagrama de despliegue en local

Capítulo 6
Implementación y pruebas

En este capítulo se aborda la fase de implementación, detallando aspectos clave como la estructura del código en el frontend y backend, así como el enfoque seguido para la realización de las pruebas. Además, se describen los problemas encontrados durante esta fase junto con las soluciones implementadas, así como las mejoras o cambios realizados respecto a las decisiones iniciales tomadas en las etapas de análisis y diseño.

6.1. Implementación

6.1.1. Organización del código

En este apartado se describe la organización del código, incluyendo la jerarquía de directorios y archivos, junto con una breve explicación de su contenido y las decisiones tomadas para estructurarlo. Se detalla tanto la organización del frontend como la del backend, destacando cómo esta estructura facilita el desarrollo, la mantenibilidad y la escalabilidad del proyecto.

Organización en el frontend

Uno de los conceptos principales en Angular son los componentes y los módulos. Siguiendo la guía oficial de "Angular Best Practices" (Medium, 2023), la estructura de componentes y archivos utilizada en este proyecto se detalla en las figuras Figura 6.1 y Figura 6.2

A continuación, se describen las carpetas principales:

- ***components***

  Esta carpeta incluye los componentes comunes reutilizables en distintas partes de la aplicación. Como se muestra en la Figura 6.3, contiene:

  - ***dialog.component*** – modales o cuadros de diálogo.
  - ***header.component*** – cabecera de la página web.
  - ***login.component*** y ***direccion-form.component*** – formularios como el de inicio de sesión o el de dirección de domicilio.
  - ***product-card.component*** – cards de productos en la vista de cliente, entre otros.

Estos componentes se reutilizan en varias partes del código, lo que evita la duplicación y mejora la mantenibilidad, claridad y organización del proyecto.

Tal como se menciona en el apartado 5.3.1. Versión de Angular, no se emplean componentes standalone así que todos los componentes definidos en este directorio están agrupados en un único módulo: *components.module.ts*.

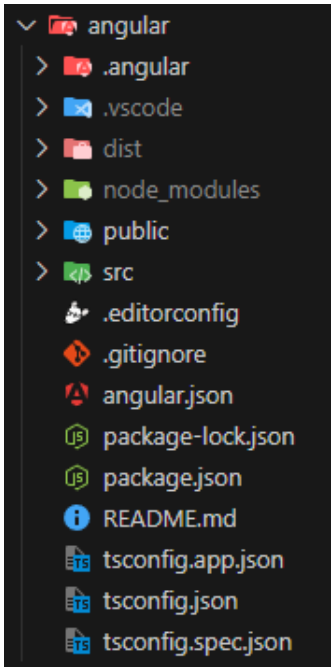

Figura 6.1. Organización en el frontend (general)

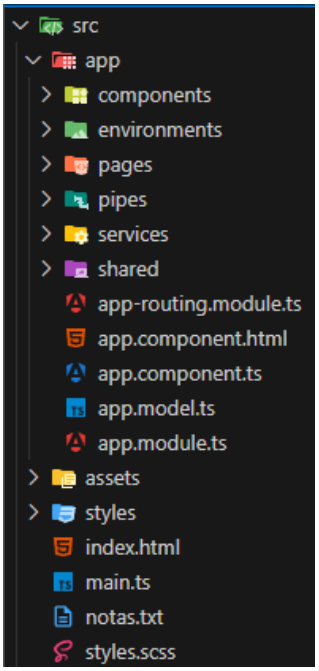

Figura 6.2. Organización del frontend (dentro de *src*)

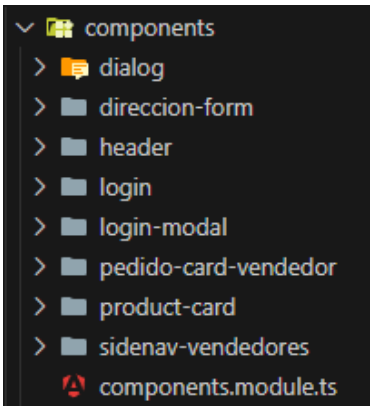

Figura 6.3. Directorio *components*

- ***environments***

  Esta carpeta incluye el fichero de *environment.ts* que en Angular se utiliza para almacenar variables específicas del entorno, como claves de configuración o direcciones de API. En este caso, se emplea para especificar la dirección de la API de backend y para guardar el clientId necesario para configurar las compras en línea con PayPal. Este enfoque centralizado facilita

la gestión segura de credenciales y permite adaptar fácilmente la configuración entre los entornos de desarrollo, prueba y producción.

- ***pages***

  Esta carpeta incluye los componentes que representan las diferentes páginas de la plataforma, como se observa en la Figura 6.4.

  Ejemplos de estas páginas son:

  - ***login-compradores.component***, ***login-vendedores.component***, ***registro-compradores.component*** y ***registro-venedores.component*** – inicio de sesión y registro para compradores y vendedores.
  - ***carrito.component*** – carrito de compras.
  - ***gestion-pedidos.component*** y ***gestion-productos.component*** – gestión de pedidos y productos para vendedores.

  Estos componentes hacen uso de los elementos del directorio *components* mediante una relación padre-hijo para compartir información entre ellos. Todos los componentes de este directorio forman parte del módulo *pages.module.ts.*

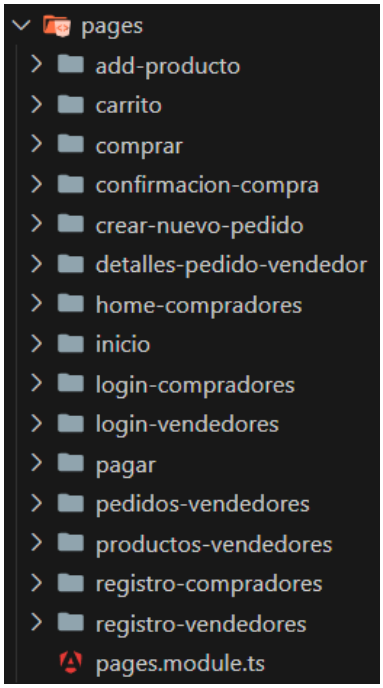

Figura 6.4. Directorio *pages*

- ***pipes***

  Los pipes en Angular permiten transformar datos. En este proyecto se han creado tres pipes (ver Figura 6.5):

  - **Fecha.** Formato *dd/mm/aaaa*
  - **Precio.** Formato *X,XX€*

- **Tiempo.** Formato *hh:mm*

Estos pipes, definidos para estandarizar los formatos utilizados en toda la aplicación, están agrupados en el módulo *pipes.module.ts.*

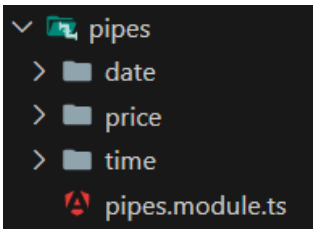

Figura 6.5. Directorio *pipes*

- ***services***

Los servicios en Angular se utilizan para encapsular lógica que debe compartirse entre varios componentes, como peticiones HTTP o gestión de datos.

En la Figura 6.6, se destacan algunos servicios importantes:

- ***compra.service.ts*** – gestiona peticiones relacionadas con el carrito de compras, actualizar la cantidad de productos, vaciarlo u obtener su contenido.
- ***gestion-productos.service.ts*** – maneja las peticiones relacionadas con la gestión de productos y pedidos por parte del vendedor.
- ***dialog.service.ts*** y ***toast.service.ts*** – centralizan la apertura y cierre de componentes de diálogo o mensajes tipo *toast*. Así, los componentes delegan esta responsabilidad a los servicios, mejorando la separación de responsabilidades.

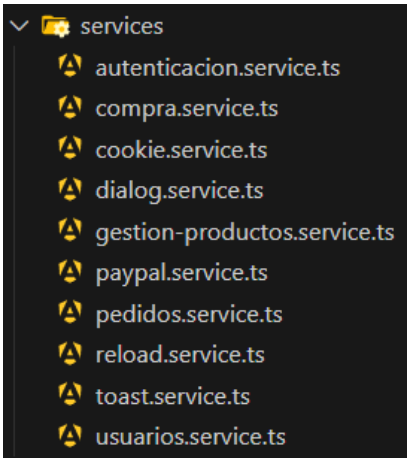

Figura 6.6. Directorio *services*

- ***shared***

Este directorio (ver Figura 6.7) contiene únicamente un archivo: *functions.ts*.
En este archivo se agrupan funciones reutilizadas en diferentes componentes, evitando así duplicar código.

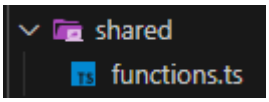

Figura 6.7. Directorio *shared*

- ***assets***

  Este directorio (ver Figura 6.8) contiene los recursos gráficos utilizados en la aplicación, como:

  - Logotipos en diferentes colores
  - Imágenes usadas en la web
  - Una representación de la paleta de colores utilizada

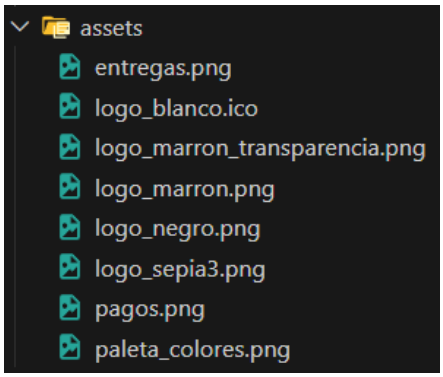

Figura 6.8. Directorio *assets*

- ***styles***

  Aquí se encuentran los archivos SCSS de estilos de la aplicación (ver Figura 6.9). Además de un archivo de estilos por cada componente, se han creado archivos para estilos comunes, como:

  - ***_angularMaterial.scss*** – estilos personalizados para los componentes de Angular Material.
  - ***_colors.scss*** – paleta de colores.
  - ***_constants.scss*** – constantes como márgenes de página.
  - ***_fonts.scss*** – diferentes tamaños y estilos para la tipografía utilizada Lato.
  - ***_general.scss*** – para poder definir mi propio diseño de los elementos HTML.

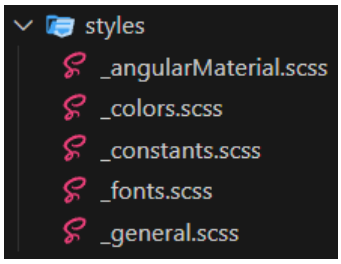

Figura 6.9. Directorio *styles*

Organización en el backend

Para el backend, se ha seguido la estructura recomendada por el framework Spring Boot, como se ilustra en la Figura 6.10.

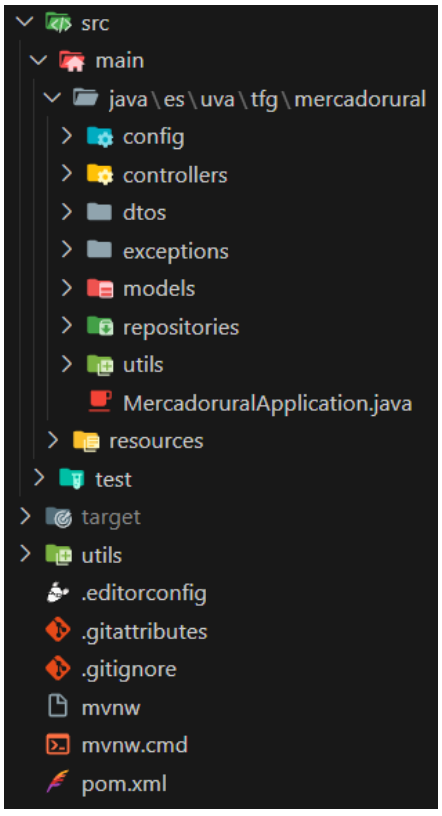

Figura 6.10. Organización en el backend

La organización detallada de carpetas es la siguiente:

- ***config***

  En este directorio solo hay un fichero que es el *WebConfig.java*, el cual sirve para configurar las políticas de CORS (Cross-Origin Resource Sharing) en la aplicación. Permite definir qué orígenes, métodos y encabezados están autorizados para interactuar con el backend, asegurando que las solicitudes desde el frontend, como las realizadas desde `http://localhost:4200`, sean aceptadas de manera segura.

- ***controllers***

  Este directorio contiene los controladores que gestionan las peticiones HTTP. Son responsables de recibir las solicitudes, realizar las acciones correspondientes (como llamar a repositorios) y devolver la información pertinente al cliente.

- ***dtos***

  Tal como se explica en el apartado 5.2.3. Patrón DTO, los DTOs se utilizan para transferir información entre el backend y el frontend de manera estructurada y limpia. Estos objetos permiten abstraer detalles internos de las entidades y asegurar que solo se transmita la información necesaria.

- ***exceptions***

  En este directorio se encuentra un único archivo dedicado a la gestión de excepciones personalizadas. Este archivo permite definir y lanzar excepciones específicas que pueden ser utilizadas por cualquier controlador, proporcionando un manejo centralizado y consistente de errores en toda la aplicación.

- ***models***

  Esta carpeta contiene las clases que representan las tablas de la base de datos. Estas clases incluyen anotaciones como `@Entity` y `@Table`, que son esenciales para mapearlas a las tablas correspondientes en la base de datos.

  Además de su función como representaciones de las tablas, estas clases también forman parte del modelo del dominio, actuando como objetos que encapsulan datos y, en algunos casos, lógica de negocio básica.

- ***repositories***

  Este directorio contiene las interfaces de repositorios, que gestionan las operaciones con la base de datos.

  Se implementa el patrón Repository de Spring Data JPA, permitiendo realizar consultas y operaciones CRUD de manera eficiente. Si bien el patrón DAO (Data Access Object) también se utiliza en algunos contextos, en este caso, la estructura proporcionada por Spring Boot combina ambos patrones para simplificar la interacción con la base de datos.

- ***utils***

  Aquí se agrupan archivos útiles para el desarrollo y las pruebas. Algunos ejemplos incluyen:

  - Un archivo para la población inicial de datos en la base de datos.
  - Un script SQL que permite vaciar la base de datos.

  Estos recursos han sido fundamentales durante las etapas de desarrollo para realizar pruebas y garantizar la consistencia de los datos.

6.1.2. Implementación de pagos

Tal y como se especificó en el requisito funcional RF22, el sistema debe permitir realizar pagos a través de PayPal, complementando así la idea de modernización que busca este proyecto al ofrecer la posibilidad de realizar pagos en línea. Para lograr esta funcionalidad, se ha utilizado una biblioteca de Angular basada en el SDK de JavaScript, llamada `ngx-paypal`, que simplifica la integración de la interfaz de usuario para pagos directamente con la plataforma de PayPal.

La API de PayPal, en su modo sandbox, ofrece dos tipos de pagos: transacciones utilizando cuentas de PayPal y pagos con tarjeta. En este caso, únicamente se ha implementado la funcionalidad de PayPal. Para simular las transacciones, se han creado cuentas personales que representan a los compradores y cuentas de tipo *business* para los vendedores.

La situación en este proyecto presenta una particularidad respecto a los casos más comunes de integración de pagos. Normalmente, los sistemas que utilizan PayPal suelen incluir varios compradores que realizan pedidos a un único vendedor, quien recibe los pagos directamente. Sin embargo, en la plataforma "Mercado Rural", el sistema actúa como intermediario entre compradores y vendedores. Cuando un comprador realiza un pedido, el pago es recibido inicialmente por Mercado Rural, que retiene una comisión especificada en el apartado 1.3.1. Decisión actual de este documento. Posteriormente, el sistema distribuye el importe correspondiente entre los diferentes comercios involucrados en el pedido, ya que un pedido puede incluir productos de varios vendedores.

Para gestionar esta lógica, además de las cuentas personales y *business*, se ha creado una REST API app de tipo *platform*, que simula el funcionamiento de un marketplace. En este caso, Mercado Rural actúa como intermediario encargado de la recepción y el reparto de los pagos.

Implementación técnica

La integración en el proyecto comienza con la instalación de la biblioteca de PayPal mediante el siguiente comando:

```
npm install ngx-paypal
```

Esta biblioteca permite implementar una configuración inicial básica de PayPal desde el frontend mediante un método llamado `initConfig()` que permite especificar:

- El receptor del pago del pedido.
- La moneda de la transacción.
- El importe total del pedido.
- Los productos incluidos en la compra.
- Las acciones a realizar cuando el pago es aprobado o cancelado.

En una implementación estándar, esta configuración sería suficiente para gestionar los pagos, pero en este caso, la lógica de reparto de pagos entre múltiples vendedores no puede resolverse únicamente en el cliente. Por lo tanto, ha sido necesario implementar dicha lógica en el backend.

Desde el backend, se realiza una llamada a la API de PayPal a través de la siguiente URL:

```
https://api-m.sandbox.paypal.com/v1/payments/payouts
```

En esta petición, se especifican los receptores del pago (los vendedores) y las cantidades correspondientes a cada uno, considerando la comisión que retiene Mercado Rural por cada pedido procesado.

Antes de realizar esta operación, es necesario obtener un token de acceso mediante otra llamada a la API de PayPal:

```
https://api-m.sandbox.paypal.com/v1/oauth2/token
```

Este token es imprescindible para autenticar las solicitudes realizadas a la API de pagos.

En conclusión, mientras que la configuración básica de PayPal mediante `initConfig()` es suficiente para casos donde solo hay un vendedor que recibe el pago, en el caso de Mercado Rural, donde se actúa como intermediario y se distribuyen los pagos entre múltiples vendedores, ha sido necesario implementar esta lógica de manera manual desde el backend. Esto asegura que el sistema cumpla con los objetivos definidos y respete las particularidades de funcionamiento de la plataforma como marketplace.

Fuentes utilizadas: (npm Inc, 2022), (Enngage, 2018), (PayPal Developer, 2016), (Developer PayPal, 2021), (Developer PayPal, 2022), (Stackoverflow, 2023), (Developer PayPal, 2022) y (PayPal Community, 2021).

6.2. Pruebas

A continuación, de la Tabla 6.2 a la Tabla 6.44 se detalla la batería de pruebas llevada a cabo para verificar el correcto funcionamiento de la aplicación, junto con los resultados obtenidos. Estas pruebas, de tipo caja negra, se han diseñado en base a las historias de usuario definidas en el apartado 7.2. Fases de implementación, pruebas y mantenimiento, las cuales se enumeran en la siguiente Tabla 6.1.

Para la ejecución de estas pruebas, es necesario que la base de datos cuente con datos previamente cargados. Para ello, se utiliza el script *PoblacionBD.java*, ubicado en el directorio *src/main/java/es/uva/tfg/mercadorural/utils/*, el cual se encarga de poblar la base de datos con la información necesaria para llevar a cabo las verificaciones de manera efectiva.

ID	Historia de usuario	Casos de uso cubiertos
HU01	Acceso de usuarios a la plataforma (login y registro)	CU01: Registrar comercio CU13: Registrarse (comprador) CU17: Iniciar sesión
HU02	Cerrar sesión	CU18: Cerrar sesión
HU03	Consultar productos (compradores)	CU16: Consultar productos
HU04	Gestionar productos (vendedores)	CU07: Añadir producto CU08: Eliminar producto CU09: Modificar producto
HU05	Gestionar pedidos	CU03: Ver pedidos CU05: Actualizar estado de un pedido
HU06	Realizar nuevo pedido (compradores)	CU15: Hacer nuevo pedido
HU07	Crear un pedido (vendedores)	CU04: Registrar nuevo pedido

Tabla 6.1. Historias de usuario

6.2.1. HU01: Acceso de usuarios a la plataforma (login y registro)

CP01	Registrar comercio
Descripción	El usuario vendedor se registra en la plataforma indicando sus datos personales, los datos correspondientes al comercio y los datos de acceso.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García Datos del comercio: ▪ Nombre: La Tahona ▪ CIF: B26162768 ▪ Categoría: Panadería ▪ Teléfono: 987654321 ▪ Email: latahona@mail.com ▪ Dirección: – Calle: C/ La Ermita, 11 – Piso/puerta: (vacío) – Código postal: 47001 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Tahona_12 ▪ Repetición de contraseña: Tahona_12

Resultado esperado	El sistema redirige al login de vendedores (/login_vendedores).

Tabla 6.2. Caso de prueba "Registrar comercio – válido"

CP02	Registrar comercio con un CIF inválido
Descripción	El usuario vendedor introduce un CIF inválido.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García Datos del comercio: ▪ Nombre: La Tahona ▪ CIF: Z1234567H (inicial no válida) ▪ Categoría: Panadería ▪ Teléfono: 987654321 ▪ Email: latahona@mail.com ▪ Dirección: – Calle: C/ La Ermita, 11 – Piso/puerta: (vacío) – Código postal: 47001 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Tahona_12 ▪ Repetición de contraseña: Tahona_12
Resultado esperado	El formulario muestra justo debajo del campo del CIF el mensaje "El CIF no es válido".

Tabla 6.3. Caso de prueba "Registrar comercio – CIF inválido"

CP03	Registrar comercio con un teléfono ya registrado
Descripción	El usuario vendedor introduce un teléfono que ya está asociado a otro usuario del sistema.
Entrada	En primer lugar, se ejecuta el CP01 (Tabla 6.2). A continuación, se introducen los siguientes datos: Datos personales: ▪ Nombre: Pedro ▪ Apellidos: Martínez Benito Datos del comercio: ▪ Nombre: La Parrilla de Oro

	▪ CIF: E94456977 ▪ Categoría: Carnicería ▪ Teléfono: 987654321 ▪ Email: laparrilladeoro@mail.com ▪ Dirección: – Calle: C/ Los Prados, 8 – Piso/puerta: (vacío) – Código postal: 47011 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Parrilla_12 ▪ Repetición de contraseña: Parrilla_12
Resultado esperado	El sistema muestra el mensaje "El teléfono ya está registrado".

Tabla 6.4. Caso de prueba "Registrar comercio – teléfono ya registrado"

CP04	Registrar comercio con email no válido
Descripción	El usuario vendedor proporciona un email sintácticamente inválido.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García Datos del comercio: ▪ Nombre: La Tahona ▪ CIF: E94456977 ▪ Categoría: Panadería ▪ Teléfono: 987654321 ▪ Email: latahona@ ▪ Dirección: – Calle: C/ La Ermita, 11 – Piso/puerta: (vacío) – Código postal: 47001 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Tahona_12 Repetición de contraseña: Tahona_12

Resultado esperado	El sistema muestra el mensaje "El email no es válido".

Tabla 6.5. Caso de prueba "Registrar comercio – email no válido"

CP05	Registrar comercio con email ya registrado
Descripción	El usuario vendedor introduce un email que ya está asociado a otro usuario del sistema.
Entrada	En primer lugar, se ejecuta el CP01 (Tabla 6.2). A continuación, se introducen los siguientes datos: Datos personales: ▪ Nombre: Pedro ▪ Apellidos: Martínez Benito Datos del comercio: ▪ Nombre: La Parrilla de Oro ▪ CIF: B26162768 ▪ Categoría: Carnicería ▪ Teléfono: 987654321 ▪ Email: latahona@mail.com ▪ Dirección: – Calle: C/ Los Prados, 8 – Piso/puerta: (vacío) – Código postal: 47011 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Parrilla_12 ▪ Repetición de contraseña: Parrilla_12
Resultado esperado	El sistema muestra el mensaje "El email ya está registrado".

Tabla 6.6. Caso de prueba "Registrar comercio – email ya registrado"

CP06	Registrar comercio con una contraseña que no cumple el patrón
Descripción	La contraseña introducida no cumple con el patrón definido en la Tabla 3.1 excepción 4g.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García Datos del comercio:

	▪ Nombre: La Tahona ▪ CIF: E94456977 ▪ Categoría: Panadería ▪ Teléfono: 987654321 ▪ Email: latahona@mail.com ▪ Dirección: – Calle: C/ La Ermita, 11 – Piso/puerta: (vacío) – Código postal: 47001 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: 1234 ▪ Repetición de contraseña: 1234
Resultado esperado	El sistema muestra el mensaje "La contraseña ha de tener como mínimo 8 caracteres. Debe contener al menos una mayúscula, una minúscula y un número".

Tabla 6.7. Caso de prueba "Registrar comercio – contraseña no cumple patrón"

CP07	Registrar comercio con contraseñas que no coinciden
Descripción	El usuario vendedor introduce dos contraseñas que no coinciden.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García Datos del comercio: ▪ Nombre: La Tahona ▪ CIF: E94456977 ▪ Categoría: Panadería ▪ Teléfono: 987654321 ▪ Email: latahona@mail.com ▪ Dirección: – Calle: C/ La Ermita, 11 – Piso/puerta: (vacío) – Código postal: 47001 – Ciudad/localidad: Valladolid – Provincia: Valladolid – Otros: (vacío) Datos de acceso: ▪ Contraseña: Tahona_12

	▪ Repetición de contraseña: Tahona_20
Resultado esperado	El sistema muestra el mensaje "Las contraseñas no coinciden".

Tabla 6.8. Caso de prueba "Registro comercio – contraseñas no coincidentes"

CP08	Registrar comprador
Descripción	El usuario comprador se registra en el sistema indicando sus datos personales, datos de domicilio y datos de acceso.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García ▪ Teléfono: 987654321 ▪ Email: juan.hernandogarcia@mail.com Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: Ejemplo_12 ▪ Repetición de contraseña: Ejemplo_12
Resultado esperado	El sistema redirige al login de compradores (/login).

Tabla 6.9. Caso de prueba "Registrar comprador – válido"

CP09	Registrar comprador con un teléfono ya registrado
Descripción	El usuario comprador introduce un teléfono que ya está asociado a otro usuario del sistema.
Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9). A continuación, se introducen los siguientes datos: Datos personales: ▪ Nombre: Pedro ▪ Apellidos: Pérez Pérez ▪ Teléfono: 987654321 ▪ Email: pedro.perezperez@mail.com Dirección de domicilio: ▪ Calle: C/ Madre de Dios, 14

	▪ Piso/puerta: 1C ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: Ejemplo_12 ▪ Repetición de contraseña: Ejemplo_12
Resultado esperado	El sistema muestra el mensaje "El teléfono ya está registrado".

Tabla 6.10. Caso de prueba "Registrar comprador – teléfono ya registrado"

CP10	Registrar comprador con un email no válido
Descripción	El usuario comprador introduce un email sintácticamente inválido.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García ▪ Teléfono: 987654321 ▪ Email: juan.hernandogarcia@ Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: Ejemplo_12 ▪ Repetición de contraseña: Ejemplo_12
Resultado esperado	El sistema muestra el mensaje "El email no es válido".

Tabla 6.11. Caso de prueba "Registrar comprador – email no válido"

CP11	Registrar comprador con un email ya registrado
Descripción	El usuario comprador introduce un email que ya está asociado a otro usuario del sistema.
Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9). A continuación,, se introducen los siguientes datos: Datos personales:

	▪ Nombre: Pedro ▪ Apellidos: Pérez Pérez ▪ Teléfono: 987654321 ▪ Email: juan.hernandogarcia@mail.com Dirección de domicilio: ▪ Calle: C/ Madre de Dios, 14 ▪ Piso/puerta: 1C ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: Ejemplo_12 ▪ Repetición de contraseña: Ejemplo_12
Resultado esperado	El sistema muestra el mensaje "El email ya está registrado".

Tabla 6.12. Caso de prueba "Registrar comprador – email ya registrado"

CP12	Registrar comprador con una contraseña que no cumple el patrón
Descripción	La contraseña introducida no cumple con el patrón definido en la Tabla 3.1 excepción 4g.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García ▪ Teléfono: 987654321 ▪ Email: juan.hernandogarcia@mail.com Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: 1234 ▪ Repetición de contraseña: 1234
Resultado esperado	El sistema muestra el mensaje "La contraseña ha de tener como mínimo 8 caracteres. Debe contener al menos una mayúscula, una minúscula y un número".

Tabla 6.13. Caso de prueba "Registrar comprador – contraseña no cumple patrón"

CP13	Registrar comprador con contraseñas que no coinciden
Descripción	El usuario comprador introduce dos contraseñas que no coinciden.
Entrada	Datos personales: ▪ Nombre: Juan ▪ Apellidos: Hernando García ▪ Teléfono: 987654321 ▪ Email: juan.hernandogarcia@mail.com Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Datos de acceso: ▪ Contraseña: Ejemplo_12 ▪ Repetición de contraseña: Ejemplo_21
Resultado esperado	El sistema muestra el mensaje "Las contraseñas no coinciden".

Tabla 6.14. Caso de prueba "Registrar comprador – contraseñas no coincidentes"

CP14	Iniciar sesión (vendedores)
Descripción	Para iniciar sesión en el sistema se deberá introducir el correo electrónico y contraseña indicados durante el registro.
Entrada	En primer lugar, se ejecuta el CP01 (Tabla 6.2). A continuación, se introducen los siguientes datos: ▪ Email: latahona@mail.com ▪ Contraseña: Tahona_12
Resultado esperado	El sistema redirige a la página de gestión de sus pedidos (/pedidos).

Tabla 6.15. Caso de prueba "Iniciar sesión – válido (vendedores)"

CP15	Iniciar sesión con un email no válido (vendedores)
Descripción	El usuario vendedor introduce un email sintácticamente inválido.
Entrada	En primer lugar, se ejecuta el CP01 (Tabla 6.2). A continuación, se introducen los siguientes datos: ▪ Email: latahona@ ▪ Contraseña: Tahona_12

Resultado esperado	El sistema muestra el mensaje "El email no es válido".

Tabla 6.16. Caso de prueba "Iniciar sesión – email no válido (vendedores)"

CP16	Iniciar sesión con unas credenciales incorrectas (vendedores)
Descripción	El usuario vendedor introduce un par email y contraseña que no se corresponden con ningún comercio registrado en el sistema.
Entrada	En primer lugar, se ejecuta el CP01 (Tabla 6.2). A continuación, se introducen los siguientes datos: ▪ Email: latahona@mail.com ▪ Contraseña: Tahona_21
Resultado esperado	El sistema muestra el mensaje "Credenciales incorrectas".

Tabla 6.17. Caso de prueba "Iniciar sesión – credenciales incorrectas (vendedores)"

CP17	Iniciar sesión (compradores)
Descripción	Para iniciar sesión en el sistema se deberá introducir el correo electrónico y contraseña indicados durante el registro.
Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9). A continuación, se introducen los siguientes datos: ▪ Email: juan.hernandogarcia@mail.com ▪ Contraseña: Ejemplo_12
Resultado esperado	El sistema redirige a la página de home (/home).

Tabla 6.18. Caso de prueba "Iniciar sesión – válido (compradores)"

CP18	Iniciar sesión con un email no válido (compradores)
Descripción	El usuario comprador introduce un email sintácticamente inválido.
Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9). A continuación, se introducen los siguientes datos: ▪ Email: juan.hernandogarcia@ ▪ Contraseña: Ejemplo_12
Resultado esperado	El sistema muestra el mensaje "El email no es válido".

Tabla 6.19. Caso de prueba "Iniciar sesión – email no válido (compradores)"

CP19	Iniciar sesión con unas credenciales incorrectas (compradores)
Descripción	El usuario comprador introduce un par email y contraseña que no se corresponden con ningún comprador registrado en el sistema.

Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9). A continuación, se introducen los siguientes datos: ▪ Email: juan.hernandogarcia@mail.com ▪ Contraseña: Ejemplo_21
Resultado esperado	El sistema muestra el mensaje "Credenciales incorrectas".

Tabla 6.20. Caso de prueba "Iniciar sesión – credenciales incorrectas (compradores)"

6.2.2. HU02: Cerrar sesión

CP20	Cerrar sesión (compradores y vendedores)
Descripción	Al pulsar el botón de "Cerrar sesión", el sistema debe finalizar la sesión activa del usuario, eliminando cualquier información asociada a su autenticación y redirigiéndolo a la página de login.
Entrada	Se pulsa el botón de cerrar sesión.
Resultado esperado	El sistema redirige a los compradores al login para compradores (/login) y a los vendedores al login para vendedores (/login-vendedores).

Tabla 6.21. Caso de prueba "Cerrar sesión"

6.2.3. HU03: Consultar productos (compradores)

CP21	Consultar productos a la venta
Descripción	El usuario comprador puede ver una lista de los productos de cada comercio ordenados por categorías.
Entrada	En la página de home, el usuario pulsa el botón "Empezar a comprar".
Resultado esperado	El sistema redirige al usuario a la página de compra donde podrá visualizar y navegar por el catálogo de productos disponibles en los diferentes comercios.

Tabla 6.22. Caso de prueba "Consultar productos a la venta"

6.2.4. HU04: Gestionar productos (vendedores)

CP22	Añadir un nuevo producto
Descripción	El usuario vendedor puede añadir nuevos productos a su catálogo para que estén disponibles para su venta.
Entrada	▪ Nombre: Hogaza grande ▪ Precio: 2.5 ▪ Precio por unidad: 4

	▪ Unidad: Kilogramo ▪ Ingredientes: Harina, agua, levadura.
Resultado esperado	El sistema muestra una pequeña ventana emergente indicando que el producto que se añadido el producto. Tras cerrar dicha ventana, el sistema no redirige a una nueva página si no que el usuario puede seguir introduciendo nuevos productos. Para comprobar que se ha introducido correctamente, se debe pulsar el botón "Productos" que lleva a la ruta "/productos". Aparecerá al final de la lista de productos.

Tabla 6.23. Caso de prueba "Añadir nuevo producto"

CP23	Añadir un nuevo producto dejando campos sin rellenar
Descripción	Todos los campos del formulario son obligatorios, por lo que se deben rellenar todos para poder registrar un nuevo producto.
Entrada	▪ Nombre: Hogaza grande ▪ Precio: 2.5 ▪ Precio por unidad: 4 ▪ Unidad: Kilogramo ▪ Ingredientes: (vacío)
Resultado esperado	El sistema muestra el mensaje "Campo obligatorio" debajo de los campos vacíos.

Tabla 6.24. Caso de prueba "Añadir nuevo producto – campos vacíos"

CP24	Modificar un producto
Descripción	El usuario vendedor puede modificar cualquier campo de información de un producto.
Entrada	En primer lugar, se ejecuta el CP22 (Tabla 6.23). A continuación, se introducen los siguientes datos: ▪ Nombre: Hogaza grande ▪ Precio: 2.75 (esto es lo que ha cambiado) ▪ Precio por unidad: 4 ▪ Unidad: Kilogramo ▪ Ingredientes: Harina, agua, levadura.
Resultado esperado	El sistema muestra una pequeña ventana emergente indicando que la actualización se ha llevado acabo con éxito y redirige a la página de gestión de productos (/productos) donde se pueden ver los cambios efectuados en la *card* correspondiente al producto recién modificado.

Tabla 6.25. Caso de prueba "Modificar un producto"

CP25	Modificar un producto, pero salir sin guardar cambios
Descripción	El usuario vendedor puede salir sin guardar los cambios efectuados, pero el sistema notificará de ello y pedirá confirmación.
Entrada	En primer lugar, se ejecuta el CP22 (Tabla 6.23). A continuación, se introducen los siguientes datos: ▪ Nombre: Hogaza grande ▪ Precio: 2.75 (esto es lo que ha cambiado) ▪ Precio por unidad: 4 ▪ Unidad: Kilogramo ▪ Ingredientes: Harina, agua, levadura.
Resultado esperado	El sistema muestra una ventana de confirmación indicando que se han detectado cambios sin guardar. Si el usuario sale sin guardar el sistema redirige a la página de gestión de productos y el usuario podrá comprobar que los cambios no se han efectuado. Sin embargo, si no decide volver atrás, la ventana emergente se cierra y el usuario puede continuar modificando la información del producto.

Tabla 6.26. Caso de prueba "Modificar un producto – salir sin guardar cambios"

CP26	Eliminar un producto
Descripción	Como se detalla en el apartado 7.2.3. Sprint 3 (26/11/2024 – 10/12/2024), el CU08 Eliminar producto ha sido redefinido para que en lugar de poder eliminarlo completamente del sistema se pueda ocultar, de modo que deja de estar disponible para la venta.
Entrada	El usuario pulsa el botón de ocultar en el card de uno de sus productos.
Resultado esperado	El sistema muestra una ventana de confirmación en la que, si el usuario confirma la acción, el producto cambia a un estado de "no disponible" dejando de estar a la venta, y se muestra además un mensaje en la parte inferior de la pantalla confirmando la acción. Por otro lado, si el usuario cancela la acción, la ventana de confirmación simplemente se cierra y el estado del producto permanece sin cambios.

Tabla 6.27. Caso de prueba "Ocultar un producto"

CP27	Mostrar un producto
Descripción	Con la redefinición del caso de uso "eliminar un producto" y la introducción de la opción para ocultarlo, se incorpora también la funcionalidad opuesta: volver a mostrar el producto. Esto permite que el producto pase nuevamente al estado de "disponible" y esté habilitado para la venta.
Entrada	El usuario pulsa el botón de mostrar en el card de uno de sus productos ocultos.
Resultado esperado	El sistema muestra una ventana de confirmación en la que, si el usuario confirma la acción, el producto cambia a un estado de "disponible" volviendo a estar a la venta, y se muestra además un mensaje en la parte inferior de la pantalla

	confirmando la acción. Por otro lado, si el usuario cancela la acción, la ventana de confirmación simplemente se cierra y el estado del producto permanece sin cambios.

Tabla 6.28. Caso de prueba "Mostrar un producto"

CP28	Mostrar u ocultar todos los productos
Descripción	El sistema permite al usuario vendedor mostrar u ocultar todos sus productos con un solo botón.
Entrada	El usuario pulsa el botón de "Mostrar todos" u "Ocultar todos".
Resultado esperado	El sistema muestra una ventana de confirmación en la que si el usuario confirma la acción todos sus productos pasan al estado contrario.

Tabla 6.29. Caso de prueba "Mostrar u ocultar todos los productos"

6.2.5. HU05: Gestionar pedidos

CP29	Ver nuevos pedidos
Descripción	Cuando un vendedor recibe un nuevo pedido, este aparece en la página de gestión de pedidos. En primer lugar, se muestra un resumen general de los productos solicitados organizados por localidad. Si el vendedor desea consultar los detalles de los pedidos específicos dentro de cada localidad, puede hacerlo mediante el botón "Ver detalles". Al acceder a los detalles de un pedido en una localidad concreta, el vendedor podrá visualizar la siguiente información: el nombre y apellidos del comprador, su número de teléfono (para posibles necesidades de contacto), la dirección de entrega, el tipo de entrega seleccionado (a domicilio o al locker), el estado de pago del pedido (pagado o pendiente), la lista de productos solicitados, la fecha y hora del pedido, así como el precio total.
Entrada	Desde la página de gestión de pedidos (/pedidos), el usuario vendedor puede ver directamente el resumen de los pedidos y para ver el detalle de cada uno pulsa el botón "Ver detalles".
Resultado esperado	En el resumen de pedidos, además de mostrar la información correspondiente, se incluirá un icono de alerta en el panel de navegación lateral izquierdo, junto al botón de "Pedidos", siempre que el vendedor tenga pedidos nuevos (es decir, aquellos que no estén marcados como preparados ni entregados). Al hacer clic en el botón "Ver detalles", el sistema redirige al usuario a la página donde se visualizan los pedidos específicos de la localidad seleccionada.

Tabla 6.30. Caso de prueba "Ver nuevos pedidos"

CP30	Marcar un pedido como preparado
Descripción	Cuando el usuario vendedor ha preparado un pedido puede indicarlo a través de la opción "Marcar como preparado", de manera que el pedido pasará a estar en estado "preparado" y se podrá visualizar en la estaña de "Preparados".
Entrada	El usuario vendedor pulsa el botón "Marcar como preparado" en una de las cards de un pedido.
Resultado esperado	El sistema presenta una ventana de confirmación en la que, al confirmar el cambio, se muestra un mensaje en la parte inferior de la pantalla indicando que la acción ha sido realizada con éxito. Además, el estado del pedido se actualiza a "preparado".

Tabla 6.31. Caso de prueba "Marcar pedido como preparado"

CP31	Marcar un pedido como entregado
Descripción	Cuando el usuario vendedor ha entregado un pedido puede indicarlo a través de la opción "Marcar como entregado", de manera que el pedido pasará a estar en estado "entregado".
Entrada	El usuario vendedor pulsa el botón "Marcar como entregado" en una de las cards de un pedido.
Resultado esperado	El sistema presenta una ventana de confirmación en la que, al confirmar el cambio, se muestra un mensaje en la parte inferior de la pantalla indicando que la acción ha sido realizada con éxito. Además, el estado del pedido se actualiza a "entregado".

Tabla 6.32. Caso de prueba "Marcar pedido como entregado"

6.2.6. HU06: Realizar nuevo pedido (compradores)

CP32	Añadir un producto al carrito
Descripción	El usuario comprador añade un producto al carrito desde el catálogo de productos.
Entrada	El usuario pulsa el botón de "Añadir" de uno de los productos.
Resultado esperado	El sistema muestra un mensaje en la parte inferior de la pantalla indicando que se ha añadido una unidad de producto y el botón "Añadir" del producto pasa a ser un selector de cantidad. Dicho selector muestra un botón con el icono de una papelera y otro botón con el signo "+" para seguir añadiendo unidades al carrito.

Tabla 6.33. Caso de prueba "Añadir un producto al carrito"

CP33	Eliminar producto del carrito
Descripción	El usuario comprador elimina un producto del carrito (desde el catálogo o desde el carrito).

Entrada	Con la cantidad de 1 seleccionada, el usuario pulsa el botón con el icono de la papelera o independientemente de la cantidad seleccionada escribe en el campo de texto, donde aparece escrita dicha cantidad, un 0.
Resultado esperado	El sistema muestra un mensaje de confirmación de la acción y el selector pasa a ser nuevamente un botón con texto "Añadir".

Tabla 6.34. Caso de prueba "Eliminar un producto del carrito"

CP34	Modificar cantidad seleccionada de producto
Descripción	El usuario puede modificar la cantidad de producto seleccionada.
Entrada	El usuario pulsa el botón "+" para añadir unidades de uno en uno o escribe en el campo de texto la cantidad deseada.
Resultado esperado	El sistema actualiza la cantidad de producto y, tras la aparición de un spinner de carga, se ve en el input la nueva cantidad.

Tabla 6.35. Caso de prueba "Modificar cantidad seleccionada de producto"

CP35	Seleccionar una cantidad excesiva de producto
Descripción	El usuario selecciona una cantidad muy grande de producto.
Entrada	El usuario, teniendo seleccionadas 50 unidades de producto, pulsa el botón "+". El usuario escribe una cantidad superior a 50 en el campo de texto de la cantidad.
Resultado esperado	Si el usuario, teniendo seleccionadas 50 unidades de producto, pulsa el botón "+" aparece un mensaje indicando que se ha alcanzado el límite permitido. Si por el contrario, escribe una cantidad superior a 50 en el campo de texto de la cantidad, el sistema automáticamente lo sustituye por 50 (cantidad máxima permitida) y notifica de ello con un mensaje.

Tabla 6.36. Caso de prueba "Seleccionar cantidad excesiva de producto"

CP36	Vaciar el carrito
Descripción	El usuario comprador estando en el carrito puede vaciarlo.
Entrada	El usuario pulsa el botón de "Vaciar carrito".
Resultado esperado	El sistema muestra una ventana de confirmación en la que, si se acepta el vaciado del carrito, este se vacía por completo y se muestra el carrito vacío, acompañado de un mensaje indicando que no hay productos seleccionados. Si el usuario decide no vaciar el carrito, la ventana se cierra sin realizar ningún cambio.

Tabla 6.37. Caso de prueba "Vaciar el carrito"

CP37	Realizar pedido pagando en efectivo
Descripción	El usuario comprador realiza un pedido y selecciona la opción de pagar con efectivo.
Entrada	Desde el carrito, el usuario pulsa el botón de "Comprar".

	En la nueva página, selecciona la opción "En efectivo" y, por tanto, obligatoriamente la opción de entrega a domicilio y pulsa el botón de "Confirmar compra".
Resultado esperado	Desde el carrito, cuando el usuario pulsa el botón de "Comprar" el sistema le redirige a la página de pagar. En esta nueva página, se muestra la información de domicilio del comprador, la forma de pago y el tipo de entrega deseada. Tras seleccionar las opciones correspondientes y confirmar la compra, el sistema muestra durante unos segundos un spinner de carga para evitar que el usuario pulse cualquier botón de la página y, tras ello, le redirige a una página de confirmación de que el pedido se ha realizado correctamente.

Tabla 6.38. Caso de prueba "Realizar pedido pagando en efectivo"

CP38	Realizar pedido pagando con PayPal
Descripción	El usuario comprador realiza un pedido y selecciona la opción de pagar con PayPal.
Entrada	Desde el carrito, el usuario pulsa el botón de "Comprar". En la nueva página, selecciona la opción "Paypal" como método de pago y la opción que desee del tipo de entrega.
Resultado esperado	Desde el carrito, cuando el usuario pulsa el botón de "Comprar" el sistema le redirige a la página de pagar. En esta nueva página, se muestra la información de domicilio del comprador, la forma de pago y el tipo de entrega deseada. Tras seleccionar el tipo de pago y el tipo de entrega, el botón de "Confirmar compra" ya no es visible, ya que el proceso de pago será gestionado por PayPal. Al pulsar el botón "PayPal", se abre una ventana emergente para que el usuario inicie sesión y elija el método de pago deseado (tarjeta o saldo). Una vez confirmado el pago, la ventana emergente desaparece, se muestra un spinner de carga durante unos segundos y, finalmente, el sistema redirige al usuario a la página de confirmación del pedido realizado.

Tabla 6.39. Caso de prueba "Realizar pedido pagando con PayPal"

CP39	Cancelar pedido pagando con PayPal
Descripción	El usuario interrumpe el proceso de compra habiendo elegido el método de pago de PayPal.
Entrada	Cuando la ventana emergente de PayPal se muestra, el usuario cierra la ventana en cualquier momento del proceso.
Resultado esperado	Si el usuario decide cerrar la ventana emergente de PayPal, la compra se cancela. El sistema no procesará el pago, no se registrará el pedido y no redirigirá al usuario a ningún sitio. En su lugar, se mostrará nuevamente la página de pago, permitiendo al usuario decidir si desea realizar el pago en otro momento.

Tabla 6.40. Caso de prueba "Cancelar pedido pagando con PayPal"

6.2.7. HU07: Crear un pedido (vendedores)

CP40	Crear un pedido como vendedor
Descripción	Otra manera de registrar pedidos en el sistema es a través de una llamada telefónica por parte de un comprador que no desea utilizar la plataforma, pero sí quiere encargar pedidos al comercio. El vendedor, por tanto, tiene la opción de registrar nuevos pedidos.
Entrada	Datos del cliente: ▪ Teléfono: 987654321 ▪ Nombre: Juan ▪ Apellidos: Hernando García Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid ▪ Otros: (vacío) Después de revisar el resumen del pedido para asegurarse de que todo es correcto, el vendedor confirma la creación del pedido.
Resultado esperado	Cuando el usuario pulsa el botón de "Crear pedido", el sistema muestra una ventana emergente de confirmación. Si el usuario confirma, se muestra un mensaje indicando que el pedido se ha registrado correctamente y el sistema redirige a la página de los pedidos.

Tabla 6.41. Caso de prueba "Crear un pedido como vendedor"

CP41	Cancelar pedido como vendedor habiendo datos sin guardar
Descripción	Durante la creación de un pedido, el vendedor puede cancelar en cualquier momento el proceso.
Entrada	Datos del cliente: ▪ Teléfono: 987654321 ▪ Nombre: Juan ▪ Apellidos: Hernando García Dirección de domicilio: ▪ Calle: C/ La Paloma, 12 ▪ Piso/puerta: 3A ▪ Código postal: 47001 ▪ Ciudad/localidad: Valladolid ▪ Provincia: Valladolid

	▪ Otros: (vacío) El usuario pulsa el botón de "Volver" estando en el formulario.
Resultado esperado	Se mostrará una ventana emergente que informará al usuario que se han detectado cambios y que, si decide salir, estos se perderán. Si el usuario confirma la salida, el pedido se cancela. En caso contrario, podrá continuar con el proceso de creación del pedido.

Tabla 6.42. Caso de prueba "Cancelar pedido como vendedor – datos sin guardar"

CP42	Cancelar pedido como vendedor sin haber escrito datos
Descripción	Durante la creación de un pedido, el vendedor puede cancelar en cualquier momento el proceso.
Entrada	Datos del cliente: ▪ Teléfono: (vacío) ▪ Nombre: (vacío) ▪ Apellidos: (vacío) Dirección de domicilio: ▪ Calle: (vacío) ▪ Piso/puerta: (vacío) ▪ Código postal: (vacío) ▪ Ciudad/localidad: (vacío) ▪ Provincia: (vacío) ▪ Otros: (vacío) El usuario pulsa el botón de "Volver" estando en el formulario.
Resultado esperado	El sistema redirige directamente a la página de pedidos.

Tabla 6.43. Caso de prueba "Cancelar pedido como vendedor – sin datos"

CP43	Crear un pedido autocompletando información del comprador
Descripción	Los clientes que realizan pedidos a través del vendedor no son considerados usuarios del sistema, ya que no cuentan con un correo electrónico y contraseña como los usuarios registrados. Sin embargo, su información se guarda en la base de datos debido a que tienen pedidos asociados, lo que permite reutilizar sus datos en futuras interacciones. Por ejemplo, al crear un pedido, el vendedor puede ingresar únicamente el número de teléfono del cliente, y si este ya está registrado en la base de datos, el sistema autocompletará automáticamente el resto de los campos con la información almacenada del comprador, tanto si tiene un usuario en el sistema como si no.
Entrada	En primer lugar, se ejecuta el CP08 (Tabla 6.9) o el CP40 (Tabla 6.41). A continuación, se introducen los siguientes datos:

	Datos del cliente: ▪ Teléfono: 987654321 ▪ Nombre: (vacío) ▪ Apellidos: (vacío) Dirección de domicilio: ▪ Calle: (vacío) ▪ Piso/puerta: (vacío) ▪ Código postal: (vacío) ▪ Ciudad/localidad: (vacío) ▪ Provincia: (vacío) ▪ Otros: (vacío) Y se pulsa el botón "Siguiente".
Resultado esperado	El sistema autocompleta con la información del comprador en caso de existir. Si el teléfono introducido no corresponde con ningun comprador del sistema, este muestra un mensaje informando del error.

Tabla 6.44. Caso de prueba "Crear un pedido – autocompletar datos comprador"

6.3. Licencia

Este proyecto cuenta con una licencia de software de código abierto de tipo **Academic Free License 3.0 (AFL 3.0)** que permite a cualquier persona utilizar, modificar y distribuir el código libremente, siempre que se respeten ciertos principios básicos.

Algunos de sus puntos más importantes son:

- **Libertad de uso y modificación.** Cualquier persona puede usar el software para cualquier propósito, modificarlo y adaptarlo a sus necesidades.
- **Distribución con condiciones claras.** Si alguien redistribuye el software (modificado o no), debe incluir el aviso de derechos de autor original y la licencia AFL 3.0.
- **Protección legal.** Ofrece términos más claros en cuanto a derechos y responsabilidades, protegiendo tanto a los desarrolladores como a los usuarios.
- **Compatibilidad con otras licencias.** Se puede combinar con software bajo otras licencias, siempre que no haya restricciones incompatibles.

Capítulo 7
Seguimiento del proyecto

En este capítulo se presenta el seguimiento detallado de las actividades realizadas durante el desarrollo del proyecto, registrando los tiempos reales de cada tarea en comparación con los tiempos estimados previamente en la fase de planificación (ver apartado 2.9. Plan de proyecto). La finalidad de este seguimiento es evaluar cómo se ha ajustado la ejecución del proyecto a las estimaciones iniciales, analizar las posibles desviaciones y justificar los cambios realizados a lo largo del proceso.

El capítulo se organiza en dos apartados que corresponden a las fases del proyecto según la metodología empleada. En primer lugar, se abordan las fases inicial, de análisis y de diseño, que siguen la metodología en cascada. Posteriormente, se agrupan las fases de implementación, pruebas y mantenimiento, que se desarrollan bajo el marco de trabajo ágil Scrum.

7.1. Fases inicial, de análisis y de diseño

En este apartado se muestran las tablas con las estimaciones de tiempo previstas para cada fase junto con los tiempos reales empleados. A pesar de las variaciones en los tiempos dedicados, las fechas de inicio y fin de las diferentes fases coinciden con las inicialmente previstas.

En la fase inicial (Tabla 7.1), se estimaron 18 horas y 30 minutos, aunque finalmente se requirieron 25 horas y 45 minutos para completarla. La fase de análisis (Tabla 7.2) tuvo una desviación menor, con 8 horas previstas frente a 11 horas reales. En cuanto a la fase de diseño (Tabla 7.3), el tiempo real fue ligeramente inferior al estimado ya que se previeron 38 horas y finalmente se necesitaron 34 horas y 30 minutos.

En general, aunque hubo pequeñas desviaciones en las horas dedicadas a algunas fases, el proyecto se ha desarrollado de acuerdo con la planificación inicial, manteniendo los plazos establecidos.

Tarea	Estimación			Real		
	Fecha inicio	Fecha fin	Tiempo empleado	Fecha inicio	Fecha fin	Tiempo empleado
Contexto	14/09/2024	22/09/2024	2h	14/09/2024	20/09/2024	3h 30m
Motivación	16/09/2024	22/09/2024	2h	14/09/2024	20/09/2024	3h
Análisis de alternativas	16/09/2024	22/09/2024	3h	22/09/2024	22/09/2024	3h

Objetivos	14/09/2024	22/09/2024	1h 30m	14/09/2024	22/09/2024	1h 45m
Identificación de stakeholders	20/09/2024	20/09/2024	30m	20/09/2024	25/09/2024	1h
Análisis de requisitos funcionales	16/09/2024	22/09/2024	1h	16/09/2024	25/09/2024	3h 30m
Análisis de requisitos no funcionales	16/09/2024	22/09/2024	30m	16/09/2024	22/09/2024	1h
Análisis de riesgos	20/09/2024	22/09/2024	3h	20/09/2024	22/09/2024	3h
Planificación	25/09/2024	29/09/2024	2h 30m	25/09/2024	29/09/2024	4h
Presupuesto	25/09/2024	29/09/2024	2h 30m	26/09/2024	29/09/2024	2h
Resumen de la fase	14/09/2024	29/09/2024	18h 30m	**14/09/2024**	**29/09/2024**	**25h 45m**

Tabla 7.1. Tiempo empleado en fase inicial

Tarea	**Estimación**			**Real**		
	Fecha inicio	**Fecha fin**	**Tiempo empleado**	**Fecha inicio**	**Fecha fin**	**Tiempo empleado**
Definición de los casos de uso	30/09/2024	6/10/2024	5h	29/09/2024	5/10/2024	7h 30m
Diagrama de CU	6/10/2024	6/10/2024	30m	6/10/2024	6/10/2024	30m
Modelo del dominio	6/10/2024	6/10/2024	2h	6/10/2024	6/10/2024	2h 30m
Diagrama de estados	6/10/2024	6/10/2024	30m	6/10/2024	6/10/2024	30m
Resumen de la fase	30/09/2024	6/10/2024	8h	**29/09/2024**	**6/10/2024**	**11h**

Tabla 7.2. Tiempo empleado en fase de análisis

Tarea	Estimación			Real		
	Fecha inicio	**Fecha fin**	**Tiempo empleado**	**Fecha inicio**	**Fecha fin**	**Tiempo empleado**
Especificación de las tecnologías utilizadas	7/10/2024	9/10/2024	3h	7/10/2024	9/10/2024	3h
Diseño arquitectónico	11/10/2024	13/10/2024	5h	10/10/2024	26/10/2024	6h
Patrones de diseño	11/10/2024	13/10/2024	5h	11/10/2024	26/10/2024	5h 30m
Diseño de la BD	12/10/2024	13/10/2024	5h	20/10/2024	25/10/2024	3h 30m
Diseño de la interfaz de usuario	18/10/2024	27/10/2024	20h	12/10/2024	20/10/2024	16h 30m
Resumen de la fase	7/10/2024	27/10/2024	38h	**7/10/2024**	**26/10/2024**	**34h 30m**

Tabla 7.3. Tiempo empleado en fase de diseño

7.2. Fases de implementación, pruebas y mantenimiento

7.2.1. Sprint 1 (30/10/2024 – 12/11/2024)

Sprint planning

Para este primer sprint, me centraré exclusivamente en la **historia de usuario** que permite el **acceso a la plataforma**, abarcando tanto el inicio de sesión como el registro de usuarios. Esto incluye la implementación de acceso para dos tipos de usuarios: compradores y vendedores/comercios, contemplando todos los escenarios de éxito y error posibles.

En cuanto al inicio de sesión, solo se requerirán el correo electrónico y la contraseña. En el caso de los vendedores, el correo proporcionado deberá ser el asociado al comercio. Para el registro, se solicitarán los datos especificados en el RF01 para los vendedores y en el RF14 para los compradores. En caso de éxito, los usuarios serán redirigidos a la página de inicio correspondiente según su rol, mientras que en caso de error se mostrarán mensajes claros y específicos que guíen al usuario.

Esta historia de usuario se tratará de desarrollar en su totalidad en este sprint cubriendo tanto el backend como el frontend, incluyendo la base de datos, la lógica de acceso y registro, y las interfaces de usuario en Angular para los formularios y las páginas de redirección.

En los próximos sprints se seguirá la misma dinámica de trabajo dividiendo las funcionalidades en historias de usuario para facilitar el desarrollo y seguimiento de la implementación.

En la Tabla 7.4 se detalla en profundidad la historia de usuario, incluyendo su estimación en puntos que se multiplica por un factor de 5 para calcular el tiempo estimado en horas. También se especifican las tareas que componen esta historia de usuario, con el tiempo estimado y el tiempo real empleado en cada una. Al final del sprint, se registra el estado final de cada tarea y de la propia historia de usuario, facilitando así la identificación de cualquier tarea pendiente o incompleta.

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU01	Acceso de usuarios a la plataforma (login y registro)		8	40h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	Configuración de la BD	Creación de todas las tablas para tener la BD lo más refinada posible desde un comienzo. También se incluye la creación de los repositorios.	5h 30m	5h	Finalizada
T02	Población de la BD	Poblar la BD con la información mínima como para poder hacer las pruebas de esta historia de usuario.	4h	2h	Finalizada
T03	Login de compradores	Creación de la UI, validaciones y la lógica general incluyendo escenarios de éxito y de error.	5h	5h	Finalizada
T04	Login de comercios	Creación de la UI, validaciones y la lógica general incluyendo escenarios de éxito y de error.	5h	5h	Finalizada
T05	Registro de compradores	Creación de la UI, validaciones y la lógica general incluyendo	6h	6h	Finalizada

		escenarios de éxito y de error.			
T06	Registro de comercios	Creación de la UI, validaciones y la lógica general incluyendo escenarios de éxito y de error.	6h	6h	Finalizada
T07	Interfaz de la página principal de compradores (home)	Creación de la página principal a la que se le redirige al usuario cuando el login y/o registro es exitoso.	3h	4h	Finalizada
T08	Interfaz de la página principal de vendedores (home)	Creación de la página principal a la que se le redirige al usuario cuando el login y/o registro es exitoso. Únicamente lo que es estático (cabecera y botones laterales)	3h	4h	Finalizada
T09	Pruebas	Para el frontend se realizarán pruebas manuales para comprobar que las validaciones funcionan correctamente en las situaciones esperadas. Para el backend se realizarán tests JUnit con al menos 3 operaciones sobre las tablas implicadas en esta HU.	2h	2h	Finalizada
T10	Documentación	Redacción del sprint planning, review, retrospective y un apartado para mencionar el logo, tipografía y paleta de colores elegida.	30m	1h	Finalizada
Resumen			**40h**	**40h**	**10/10**

Tabla 7.4. HU01: Acceso de usuarios (login y registro)

Sprint Review

Se ha logrado completar con éxito la historia de usuario prevista para este primer sprint.

A lo largo del sprint, se encontraron algunos desafíos, especialmente relacionados con errores de código en la integración entre el backend y el frontend, lo que consumió más tiempo del previsto en la resolución de estos problemas. No obstante, estas incidencias se abordaron con eficacia, permitiendo finalizar a tiempo la historia de usuario. Como resultado, todas las tareas relacionadas con el acceso de usuarios están completadas y probadas.

En cuanto a las tareas de documentación, se realizaron modificaciones en la lista de casos de uso (apartado 3.1. Casos de uso). Se añadió un nuevo caso de uso "Consultar productos" para el comprador, el cual formará parte de la historia de usuario del siguiente sprint. Se actualizó por tanto el diagrama de CU (Figura 3.1) y se añadió la definición correspondiente en la Tabla 3.16.

Sprint Retrospective

Durante este sprint, se ha identificado la importancia de asignar tiempo adicional en la planificación para la resolución de errores de integración entre el backend y el frontend, ya que estos problemas consumieron más tiempo de lo previsto. Esto permitirá enfrentar posibles incidencias de manera más organizada en los próximos sprints, minimizando el riesgo de retrasos.

7.2.2. Sprint 2 (12/11/2024 – 26/11/2024)

Sprint planning

Este segundo sprint incluirá dos historias de usuario:

- **Funcionalidad de cerrar sesión.** Esta historia de usuario se corresponde con el CU18 (definición en la Tabla 3.18) y permitirá a los compradores y vendedores cerrar sesión de forma segura una vez que la hayan iniciado.
- **Consultar productos y añadirlos al carrito.** La funcionalidad de consulta de productos, correspondiente al CU16 (definición en la Tabla 3.16), permitirá a los compradores visualizar la lista completa de productos disponibles de los comercios que distribuyen en su localidad. Los productos estarán organizados por categorías de comercio, tal y como se diseñó en el boceto de la Figura 5.8. Además, los compradores podrán añadir productos a su cesta directamente desde esta lista.

En la Tabla 7.5 y Tabla 7.6 se detallan las tareas que forman parte de cada historia de usuario.

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU02	Cerrar sesión		2	10h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	Cerrar sesión compradores	Implementar la funcionalidad correspondiente al tipo de usuario "comprador".	3h	3h	Finalizada
T02	Cerrar sesión vendedores	Implementar la funcionalidad correspondiente al tipo de usuario "vendedor".	3h	2h 45m	Finalizada
T03	Pruebas y resolución de errores	Esta tarea incluye pruebas de la funcionalidad creada es esta historia de usuario y tareas de corrección de errores que seguramente surjan durante el desarrollo.	3h 30m	4h	Finalizada
T04	Documentación	Redacción del sprint planning, review y retrospective.	30m	15m	Finalizada
Resumen			**10h**	**10h**	**4/4**

Tabla 7.5. HU02: Cerrar sesión

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU03	Consultar productos y añadirlos al carrito		6	30h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	Interfaz de la página de compra + lógica backend correspondiente	Esto incluye la barra lateral con las diferentes categorías que se pueden seleccionar y las "producto card"	14h	13h	Finalizada
T02	Interfaz de la página del carrito + lógica backend correspondiente	Mostrar los productos seleccionados.	10h	9h	Finalizada

T03	Población de la base de datos	Poblar con más registros la BD para tener más ejemplos con que probar la funcionalidad.	1h	2h	Finalizada
T04	Resolución de errores	Tiempo dedicado para solucionar posibles errores que surjan de añadir esta nueva funcionalidad o refactorización en el código para mejorarlo.	5h	5h	Finalizada
T05	Documentación	Redacción del sprint planning, review y retrospective.	1h	1h	Finalizada
Resumen			**30h**	**30h**	**5/5**

Tabla 7.6. HU03: Consultar productos (compradores)

Sprint Review

El objetivo principal de este sprint se ha logrado con éxito, completando las dos historias de usuario previstas.

Sprint Retrospective

No se han identificado aspectos destacables que requieran cambios o mejoras significativas. Considero que el flujo de trabajo y las decisiones tomadas durante este sprint han sido adecuadas, lo que indica que el proceso actual sigue funcionando de manera eficiente.

7.2.3. Sprint 3 (26/11/2024 – 10/12/2024)

Sprint planning

En este sprint me centraré en desarrollar las funcionalidades relacionadas con la **gestión de productos** por parte de los vendedores. El objetivo principal será completar la historia de usuario correspondiente, que incluye las tareas de añadir, modificar y "eliminar" productos. Estas tareas se alinean con los casos de uso CU07 Añadir producto, CU09 Modificar producto y CU08 Eliminar producto.

Es importante destacar que he decidido redefinir el concepto de "eliminar producto". En lugar de permitir la eliminación definitiva, optaré por implementar una funcionalidad para ocultar el producto, de manera que este deje de estar disponible para la venta. Esta decisión responde a dos motivos principales:

1. Evitar problemas de borrados en cascada en la base de datos, que aunque pueden controlarse técnicamente, representan una complejidad adicional.
2. Seguir una práctica común en muchas plataformas y aplicaciones reconocidas, donde los datos no se eliminan por completo, sino que se conservan en la base de datos para posibles usos futuros.

Considero que esta solución no solo es técnicamente eficiente, sino que también puede resultar interesante desde el punto de vista funcional, ya que mejora la experiencia de usuario del vendedor en caso de que quiera volver a añadir un producto que había retirado.

En cuanto a la planificación, he adoptado un enfoque pesimista ya que he completado esta historia de usuario antes del tiempo estimado. Por ende, con el tiempo restante he comenzado a trabajar en la siguiente historia de usuario, relacionada con la **gestión de pedidos**. Esto abarca los casos de uso CU03 Ver pedidos y CU05 Actualizar estado de un pedido. Por el momento, he decidido postergar la funcionalidad de crear pedidos por parte del vendedor (CU04 Registrar nuevo pedido), ya que se considera una característica adicional que podría implementarse en futuras iteraciones.

Esta nueva historia de usuario permitirá a los vendedores ver un resumen de los pedidos organizados por localidades, con la posibilidad de explorar los detalles específicos de cada pedido según lo definido en los bocetos (ver apartado 5.4. Diseño de la interfaz de usuario). Además, incluirá la capacidad de actualizar el estado de los pedidos conforme a lo establecido en el RF04.

A continuación, en las tablas Tabla 7.7 y Tabla 7.8 detallo las tareas que conforman las historias de usuario definidas para este sprint junto con los tiempos estimados y empleados.

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU04	Gestionar productos (vendedor)		4	20h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	Ver productos	Incluye la interfaz de usuario y la lógica necesaria para recoger los datos de la BD.	4h	3h	Finalizada
T02	Añadir producto	Incluye la UI y la dinámica correspondiente.	4h	5h 30m	Finalizada
T03	Modificar producto	Como se reutilizará la UI creada para añadir un producto, se deberá	4h	2h	Finalizada

		adaptar para este nuevo caso de añadir un producto, junto con la lógica oportuna.			
T04	Ocultar producto	Incluye la UI y la lógica correspondiente para ocultar y volver a mostrar los productos.	4h	5h	Finalizada
T05	Otras tareas	Tiempo dedicado a errores, mejoras de código, etc.	3h 30m	2h	Finalizada
T06	Documentación	Redacción del sprint planning, review y retrospective.	30m	30m	Finalizada
Resumen			**20h**	**18h**	**6/6**

Tabla 7.7. HU04: Gestionar productos (vendedor)

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU05	Gestionar pedidos		4	20h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	IU para mostrar los pedidos (sin lógica)	Interfaz gráfica estática para mostrar la información de los pedidos.	3h	2h	Finalizada
T02	Lógica correspondiente para recoger los datos de la BD y mostrar los pedidos organizados según corresponda	Agregar dinámica a la tarea anterior para mostrar los datos de la BD.	8h	10h	Finalizada
T03	Cambiar estado a los pedidos	Implementar la lógica correspondiente para manejar el cambio de estado de un pedido.	4h 30m	2h	Finalizada
T04	Otras tareas	Tiempo dedicado a errores, mejoras de código, mejoras visuales que mejoren la	4h	5h	Finalizada

		experiencia de usuario como un icono de alerta que indique cuándo hay pedidos nuevos, etc.			
T05	Documentación	Redacción del sprint planning, review y retrospective.	30m	30m	Finalizada
Resumen			**20h**	**19h 30m**	**5/5**

Tabla 7.8. HU05: Gestionar pedidos

Sprint review

Como se puede observar en el desglose de tareas de la historia de usuario de gestionar productos (Tabla 7.7), la mayoría de las tareas fueron completadas en menos tiempo del estimado, en comparación con aquellas que requirieron más tiempo del previsto. Esto permitió ganar tiempo suficiente para adelantar la siguiente historia de usuario de gestión de pedidos (Tabla 7.8), cuya ejecución muestra un mayor equilibrio entre los tiempos estimados y los realmente empleados. Esta experiencia subraya la importancia de ajustar las estimaciones en futuros sprints para reflejar mejor el esfuerzo real necesario en las tareas.

Sprint retrospective

Considero que esta situación fue puntual, ya que este sprint coincidió con unos días de puente, lo que me permitió dedicar más tiempo al proyecto. De cara al futuro, al menos en los dos próximos sprints que coinciden con las vacaciones de Navidad, no aumentaré la carga de trabajo, ya que preveo disponer de menos tiempo para el TFG. Sin embargo, tras las vacaciones, en los sprints 5 y 6 podría plantearme asumir una mayor carga de trabajo y adoptar un enfoque más optimista en la planificación de tareas, ajustando los tiempos de manera más ambiciosa.

7.2.4. Sprint 4 (10/12/2024 – 24/12/2024)

Sprint planning

En este sprint, me centraré en desarrollar una nueva historia de usuario, la número 6, que corresponde a la creación de un nuevo pedido por parte del comprador. Esta funcionalidad abarca el CU15 Hacer nuevo pedido y consiste en permitir que, una vez el comprador haya seleccionado los productos deseados en el carrito de la compra, pueda completar la transacción eligiendo tanto el tipo de entrega como el método de pago, siguiendo lo especificado en los requisitos funcionales RF20 y RF22, así como la regla de negocio RN01.

Dado que este sprint coincide con fechas cercanas a Navidad y tengo otros compromisos académicos de mayor prioridad en estas semanas, no iniciaré una nueva historia de usuario una vez finalizada esta. El tiempo restante lo dedicaré a avanzar en la documentación del proyecto.

A continuación, se detallan en la Tabla 7.9 las tareas que conforman esta historia de usuario.

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU06	Realizar un nuevo pedido (compradores)		4	20h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	Investigación sobre la API de PayPal	Creación de las cuentas en modo sandbox en PayPal y comprender cómo funciona.	5h	4h	Finalizada
T02	UI de la página de pagos	Incluye información sobre la dirección de envío del usuario que está registrado y opciones para elegir el método pago y el tipo de entrega.	4h	4h	Finalizada
T03	Implementación de la API de PayPal	Instalación de las librerías necesarias, implementación del código correspondiente para gestionar la lógica, particionar los pagos entre los diferentes comercios…	8h	10h	No finalizada
T04	UI de redirección tras el pago	Una vez que el pago se ha efectuado con éxito se crea la página de redirección y se comprueba que el pedido se ha registrado correctamente en la BD.	1h	1h	Finalizada
T05	Documentación	Redacción del sprint planning, review y retrospective.	2h	2h	Finalizada
Resumen			**20h**	**21h**	**4/5**

Tabla 7.9. HU06: Realizar un nuevo pedido (compradores)

Sprint review

Como se muestra en la Tabla 7.9, todas las tareas planificadas se han completado, excepto una: la T03 – Implementación de la API de PayPal. Esta tarea no se ha finalizado debido a la complejidad técnica que ha representado. He logrado implementar correctamente el envío y recepción de dinero entre un comprador y un vendedor, pero aún no he conseguido que funcione cuando hay varios vendedores involucrados en el mismo pedido. Por lo tanto, esta tarea queda pendiente y se abordará en próximos sprints, probablemente en el sprint 6.

Sprint retrospective

No se han identificado aspectos que requieran decisiones concretas o cambios significativos para futuros sprints. Aunque queda una tarea pendiente por su complejidad, esta ya se encuentra planificada para ser abordada en siguientes sprints, por lo que considero que el flujo de trabajo y las decisiones tomadas durante este sprint han sido apropiadas y continúan siendo efectivas.

7.2.5. Sprint 5 (24/12/2024 – 7/01/2025)

Sprint planning

Dado que este sprint coincide directamente con las vacaciones de Navidad, planeo dedicar menos tiempo al desarrollo. Por ello, me centraré en tareas más livianas que incluyen finalizar ciertas funcionalidades pendientes y realizar mejoras en el diseño de la página web.

Una de las tareas clave para completar el ciclo de la funcionalidad principal del proyecto, "comprar productos", es añadir una validación en el modo invitado. Actualmente, cuando un usuario en este modo (que solo ha ingresado su código postal para ver los comercios que reparten en su localidad) intenta finalizar una compra tras llenar el carrito, no se le solicita iniciar sesión. Implementaré una ventana emergente (pop-up o modal) que informará al usuario que debe iniciar sesión o registrarse en el sistema para continuar con el pago. Esta ventana incluirá el formulario de inicio de sesión y un botón que redirija al registro en caso de que el usuario no tenga una cuenta. Una vez iniciada la sesión, el usuario podrá proceder con el pago de la compra.

Además, como parte de las mejoras en el diseño, planeo incluir información adicional en el formulario de inicio de sesión de vendedores. Esto servirá para destacar las funcionalidades disponibles para ellos en la plataforma, como la gestión de pedidos, productos y localidades de reparto, entre otras. Estas mejoras buscan hacer la experiencia más informativa y atractiva para los usuarios.

ID tarea	Tarea	Descripción	Tiempo estimado	Tiempo empleado	Estado final
T01	Compras para invitados	Esto incluye el modal de inicio de sesión en caso de	5h	8h	Finalizada

		que se quiera realizar la compra sin estar registrado, las correspondientes redirecciones y la lógica interna del carrito y el usuario.			
T02	Añadir más información sobre los pedidos	Falta indicar si el pedido ya está pagado o no, así que añado esa información.	1h	30m	Finalizada
T03	Rediseño del login de vendedores y el home de compradores	Añadir algo de información en el login de vendedores indicando qué se puede hacer en la plataforma.	3h	2h	Finalizada
T04	Otras tareas	Crear un componente para el formulario de la dirección ya que puede que lo necesite en futuras tareas de futuros sprints.	4h	2h 30m	Finalizada
T05	Documentación	Redacción del sprint planning, review y retrospective.	1h	1h	Finalizada
Resumen			**14h**	**14h**	**5/5**

Tabla 7.10. Tareas del sprint 5

Sprint review

El objetivo de este sprint se ha logrado con éxito completando todas las tareas previstas.

Sprint retrospective

No se han identificado aspectos destacables que requieran cambios o mejoras significativas. Considero que el flujo de trabajo y las decisiones tomadas durante este sprint han sido adecuadas, lo que indica que el proceso actual sigue funcionando de manera eficiente.

7.2.6. Sprint 6 (7/01/2025 – 21/01/2025)

Sprint planning

En este sprint me voy a centrar en la última funcionalidad que completa el ciclo de la función principal de Mercado Rural: la creación de pedidos, pero por parte del vendedor. Esta tarea se activa cuando un vendedor registra un pedido tras recibir una llamada telefónica de un cliente. La funcionalidad está pensada especialmente para personas mayores o aquellos que no desean usar la plataforma digitalmente, permitiéndoles realizar pedidos y aprovechar las ventajas que ofrece el proyecto. Para ello, defino la nueva historia de usuario "Crear un pedido (vendedores)", que cubre el CU04 Registrar nuevo pedido en la Tabla 7.11.

Además, voy a finalizar la tarea T03, que dejé pendiente en el 7.2.4. Sprint 4 (10/12/2024 – 24/12/2024). En esta tarea, debo implementar la distribución de los pagos realizados a través de PayPal entre los diferentes comercios.

Finalmente, en este sprint redactaré la batería de pruebas que completa la fase de pruebas del proyecto (véase apartado 6.2. Pruebas).

Estas dos últimas tareas, se desglosan en la Tabla 7.12.

ID	**Nombre de la historia de usuario**		**Puntos estimados**	**Tiempo estimado**	**Estado final**
HU07	Crear un pedido (vendedores)		3	15h	Finalizada
ID tarea	**Tarea**	**Descripción**	**Tiempo estimado**	**Tiempo empleado**	**Estado final**
T01	UI Crear pedidos (vendedor)	Crear la interfaz de usuario que permita crear un pedido. Cuenta con un formulario donde introducir los datos del cliente que realiza el pedido y los productos y la cantidad que desea de los mismos.	3h	2h	Finalizada
T02	Lógica crear pedidos	Crear la lógica correspondiente para la creación de un pedido por parte del vendedor. Además, se añadirá a mayores una funcionalidad y es la de	11h	14h	Finalizada

		autocompletar la información del cliente introduciendo únicamente su número de teléfono.			
T03	Documentación	Redacción del sprint planning, review y retrospective.	1h	1h	Finalizada
Resumen			**15h**	**17h**	**3/3**

Tabla 7.11. HU07: Crear un pedido (vendedores)

ID tarea	Tarea	Descripción	Tiempo estimado	Tiempo empleado	Estado final
T01	Reparto de pagos PayPal	Repartir el dinero de los pedidos entre los diferentes vendedores implicados en un pedido.	10h	8h	No finalizada
T02	Batería de pruebas		10h	10h	Finalizada
T03	Documentación	Otras tareas de documentación (completar apartados, añadir los que faltan, revisar).	1h	1h	Finalizada
Resumen			**21h**	**19h**	**2/3**

Tabla 7.12. Tareas del sprint 6

Sprint review

Con este sprint se completa la funcionalidad principal que se presentará en el producto mínimo viable (MVP). Sin embargo, la tarea T01: Reparto de pagos no ha podido finalizarse y se trasladará al siguiente sprint. Esto se debe a la limitada documentación oficial disponible sobre la implementación del reparto de pagos con PayPal. A pesar de este inconveniente, mantengo un enfoque optimista y confío en que la tarea se podrá completar durante el último sprint, junto con el resto de las actividades relacionadas con la documentación.

Sprint retrospective

No se han identificado aspectos destacables que requieran cambios o mejoras significativas. Considero que el flujo de trabajo y las decisiones tomadas durante este sprint han sido las adecuadas.

7.2.7. Sprint 7 (21/01/2025 – 4/02/2025)

Sprint planning

En este último sprint, me he centrado en la última tarea pendiente de programación: el reparto de pagos con PayPal, una funcionalidad crucial que finalmente he logrado implementar con éxito. Con la finalización de esta tarea, doy por concluida la fase de implementación del proyecto. Además, he completado el resto de la documentación pendiente, como las conclusiones (Capítulo 8), los apéndices y otros detalles que requerían estar finalizados tras este capítulo.

Los tiempos estimados y empleados se detallan en la Tabla 7.13.

ID tarea	Tarea	Descripción	Tiempo estimado	Tiempo empleado	Estado final
T01	Reparto de pagos PayPal	Repartir el dinero de los pedidos entre los diferentes vendedores implicados en un pedido.	10h	8h	Finalizada
T02	Documentación	Terminar de documentar la memoria.	12h	12h	Finalizada
Resumen			**20h**	**20h**	**2/2**

Tabla 7.13. Tareas del sprint 7

Sprint review

La complejidad del reparto de los pagos de PayPal se ha debido a varios factores. En primer lugar, era la primera vez que trabajaba con la integración de PayPal, y la situación para este proyecto presentaba desafíos adicionales. A diferencia de un caso típico en el que hay un único vendedor, mi plataforma debía gestionar múltiples vendedores, como se detalla en el apartado 6.1.2. Implementación de pagos. Esto implicaba una lógica de negocio más compleja para distribuir los pagos. Además, gran parte del tiempo se invirtió en la búsqueda de documentación adecuada, ya que la mayoría de los recursos disponibles en línea provenían de foros no oficiales, donde las respuestas a menudo eran insuficientes o no garantizaban una solución funcional. Finalmente, gracias a la persistencia y al apoyo de mi tutora que me orientó en la investigación, logré superar este reto.

Sprint retrospective

A lo largo de esta fase de implementación, considero que he hecho un buen trabajo, superando los desafíos técnicos y completando las funcionalidades esenciales del proyecto. Este último sprint ha sido especialmente significativo, ya que me permitió cerrar una tarea compleja que había arrastrado durante varios sprints. En general, estoy satisfecha con mi progreso y con los resultados obtenidos en esta etapa.

7.3. Resumen de la ejecución del proyecto

Una vez finalizado el proyecto, resulta relevante realizar un contraste entre la planificación inicial y su desarrollo real. En esta sección, se comparan diversos aspectos clave para analizar las diferencias y similitudes.

7.3.1. Calendarización planificada y real

En el apartado 2.9. Plan de proyecto se presentó una estimación de los tiempos que se consideraron necesarios para completar las tareas del proyecto, especificando las fechas de inicio y finalización de cada fase, así como el tiempo estimado para cada una de ellas. Esta planificación se dividió en dos secciones: la fase inicial, de análisis y diseño (apartado 2.9.1. Adaptación del modelo de cascada al proyecto y calendarización), y las fases de implementación y pruebas (apartado 2.9.2. Adaptación de Scrum al proyecto y calendarización).

En el apartado 7.1. Fases inicial, de análisis y de diseño se puede observar que las tres primeras fases del proyecto se completaron dentro del período de tiempo previsto, desde el 14 de septiembre hasta el 26 de octubre. Aunque las tareas de la fase inicial requirieron más tiempo del estimado, esto se puede atribuir al hecho de que, al ser el comienzo del proyecto, me encontraba aún en una etapa de adaptación. La fase de análisis y diseño, por otro lado, aunque experimentó algunas variaciones, se desarrolló de manera más equilibrada en comparación con la estimación inicial.

En cuanto a la fase de implementación, detallada en el apartado 7.2. Fases de implementación, pruebas y mantenimiento, se trabajó bajo el marco de trabajo Scrum, estructurado en sprints de dos semanas, tal como se había planificado en el apartado 2.9.2. Adaptación de Scrum al proyecto y calendarización. En este caso, no fue necesario realizar una replanificación, ya que cada sprint se desarrolló conforme a lo esperado. Aunque se produjeron variaciones en las horas estimadas y las realmente empleadas, se logró cumplir con el alcance del proyecto dentro del tiempo estipulado. Es importante destacar que dichas variaciones generalmente no implicaron superar las 40 horas previstas por sprint; en su lugar, las tareas resultaron ser menos complejas de lo estimado, lo que permitió utilizar de manera más eficiente el tiempo disponible. Tal como se indicó en las retrospectivas de cada sprint, el enfoque y la metodología adoptados durante el proceso fueron efectivos, sin que fuera necesario implementar medidas excepcionales. En términos generales, el desarrollo de esta fase fue muy satisfactorio.

Por otro lado, durante la fase de implementación, se realizaron algunos ajustes y cambios en los diseños inicialmente definidos en los bocetos del apartado 5.4. Diseño de la interfaz de usuario. Estos cambios, documentados a lo largo de cada sprint, respondieron principalmente a la necesidad de adaptar ciertos detalles para optimizar tanto la estética como la funcionalidad del proyecto. Por ejemplo, en el caso del carrito de compras (Figura 5.16), inicialmente se había diseñado un formato de "cards" alargadas horizontalmente para mostrar los productos. Sin embargo, para aprovechar la card

ya definida en la página de compra (Figura 5.8), que era más cuadrada, decidí modificar el diseño reutilizando el formato de card de la página anterior. Un cambio similar se produjo en la creación del pedido por parte del vendedor (Figura 5.28 y Figura 5.29), donde también se rediseñó la visualización de los productos para mantener la lógica del sistema de cards, aunque sin alterar su funcionamiento principal.

Además de estos ajustes, se implementaron mejoras estéticas y de usabilidad en la gestión de pedidos. En un principio, había previsto mostrar todos los cards de los pedidos, mezclando los pedidos nuevos con los ya preparados, tal como se observa en la Figura 5.25. Sin embargo, durante el desarrollo de la página, se me ocurrió que usar el componente "tabs" de Angular Material podría mejorar significativamente la visibilidad y la experiencia de usuario, permitiendo organizar la información de manera más clara y ordenada.

En términos generales, la ejecución del proyecto siguió de manera fiel la planificación establecida, cumpliendo con los plazos previstos y alcanzando la mayoría de los objetivos definidos. A continuación, se presenta un resumen comparativo entre la planificación inicial y los resultados obtenidos:

- **Fecha de inicio estimada:** 14/09/2024
- **Fecha de inicio real:** 14/09/2024
- **Fecha de finalización estimada:** 04/02/2025
- **Fecha de finalización real:** 04/02/2025
- **Historias de usuario completadas:** 7/7 (Tabla 6.1)
- **Requisitos funcionales cumplidos:** 15/22 (apartado 2.3. Requisitos funcionales)
- **Casos de uso desarrollados:** 16/19 (apartado 3.1. Casos de uso)

Cabe destacar que los requisitos funcionales y casos de uso no cubiertos en este proyecto, se debe a que no forman parte del núcleo esencial del proyecto. Se ha priorizado el desarrollo de aquellas características que constituyen la base mínima necesaria para que la plataforma cumpla su propósito principal. Las funcionalidades restantes quedan, por tanto, recogidas en el apartado 8.2. Líneas de trabajo futuras como posibles mejoras o evoluciones futuras.

7.3.2. Tiempo estimado y empleado

Según la guía docente, el tiempo estimado para la realización del Trabajo de Fin de Grado es de 300 horas, aunque en la práctica se suele calcular en torno a las 350 horas debido a la falta de experiencia del estudiante. En mi caso, la estimación total basada en las planificaciones presentadas en el apartado 2.9. Plan de proyecto es de aproximadamente **345 horas** sin tener en cuenta un posible sprint de refuerzo. Si se incluye este sprint adicional, el total ascendería a unas 385 horas.

7.3.3. Coste real

Dado que las horas realmente empleadas en la elaboración del TFG se han mantenido muy próximas a las estimaciones realizadas en el análisis de costes del apartado 2.10. Estimación de costes, el coste simulado y el coste real final no experimentarán variaciones significativas respecto a lo inicialmente estimado.

Coste estimado final

Los elementos utilizados son los mismos que se detallaron en el apartado 2.10.1. Coste simulado, aunque con una modificación en el tiempo estimado, que se ajusta a 345 horas en lugar de las 350 horas originalmente planteadas. El coste final se presenta en la Tabla 7.14.

Concepto	Precio	Tiempo	Total
Desarrollador software	28.226€/año	345 horas	4.830,00€
Portátil	850€	345 horas	69,82€
Licencia Microsoft Teams	3,70€/mes	345 horas	14,58€
Licencia Moqups	9€/mes	345 horas	35,50€
Licencia Astah Professional	11,15€/mes	345 horas	43,96€
Oficina	595€/mes	345 horas	2.346,00€
Total			7.339,85€
Total extendido (+20%)			**8.807,85€**

Tabla 7.14. Coste estimado final

Como resultado de este ajuste, el valor pasa de ser de 8.935,50€ a **8.808,85€**.

Coste real final

Los elementos están detallados en el apartado 2.10.2. Coste real, por lo que el coste real final se presenta tal como se indica en la Tabla 7.15.

Concepto	Precio	Tiempo	Total
Desarrollador software	0€/año	345 horas	0€
Portátil	850€	345 horas	69,82€
Licencia Microsoft Teams	0€/mes	345 horas	0€
Licencia Moqups	0€/mes	345 horas	0€
Licencia Astah Professional	0€/mes	345 horas	0€
Total			**69,82€**

Tabla 7.15. Coste real final

El coste real inicialmente estimado en 70,83€ ha experimentado una ligera reducción, pasando a ser **69,82€**.

Capítulo 8
Conclusiones

8.1. Conclusiones

Los objetivos del proyecto, definidos en el apartado 1.4. Objetivos, se han cumplido satisfactoriamente, logrando una plataforma que facilita el comercio en zonas rurales, especialmente en áreas con menor acceso a la tecnología. Se ha abordado tanto a los jóvenes, modernizando las relaciones comerciales, como a las personas mayores, con funcionalidades específicas como el pedido telefónico, implementado en el Sprint 6 (7/01/2025 – 21/01/2025). La plataforma incluye funcionalidades clave como la entrega flexible y herramientas de gestión para pequeños comerciantes, lo que favorece la inclusión tecnológica en comunidades con diferentes niveles de conocimiento digital.

A nivel personal, el proyecto ha sido un reto que me ha permitido adquirir experiencia en desarrollo full-stack con Angular y Spring Boot, así como trabajar con metodologías como el modelo en cascada y Scrum. Además, ha consolidado mis conocimientos en la gestión y desarrollo de software, aplicando lo aprendido durante la carrera en todas las etapas del proyecto.

En resumen, los objetivos se han alcanzado, y la experiencia adquirida ha sido fundamental para mi crecimiento profesional. Este proyecto demuestra el potencial de la digitalización en zonas rurales y establece una base sólida para futuras mejoras que sigan promoviendo la inclusión tecnológica en estas comunidades.

8.2. Líneas de trabajo futuras

El producto desarrollado en este proyecto incluye las funcionalidades principales y básicas de la plataforma propuesta. No obstante, la visión completa de la idea abarca muchas más funcionalidades que no se han implementado. Estas son las **funcionalidades** planteadas para ser desarrolladas en el futuro:

Para los vendedores:

- **Gestión de localidades.** Tal como se describe en el requisito funcional RF11, el vendedor podrá gestionar las localidades a las que desea vender. Esto incluye la posibilidad de seleccionar a qué localidades realizar entregas mediante una interfaz visual, como un mapa, que permita identificar localidades cercanas al comercio. Además, el vendedor podrá dejar de vender en una localidad en cualquier momento.

- **Ver y actualizar los datos del comercio.** Una sección donde el vendedor pueda consultar los datos introducidos al registrarse y actualizar información, como su nombre, apellidos, teléfono o contraseña, de acuerdo con lo establecido en el RF09.
- **Historial de pedidos.** Acceso a un registro histórico de los pedidos atendidos en el pasado, permitiendo una mejor gestión y control del negocio según lo especificado en el RF10.
- **Publicación de avisos.** Una funcionalidad adicional que permitiría a los vendedores publicar avisos visibles para los compradores registrados en la plataforma. Esto facilitaría la comunicación de información importante, como promociones, cambios en horarios de atención o productos destacados (RF12).
- **Modificar pedidos.** Según lo establecido en la regla de negocio RN04, el vendedor podrá modificar pedidos que estén en estado "nuevo". Por ejemplo, si la cantidad solicitada por el comprador no es viable, el vendedor podrá contactar al cliente por teléfono para acordar una nueva cantidad y, posteriormente, actualizar el pedido desde su área de gestión.

Para los compradores:

- **Historial de pedidos.** Al igual que los vendedores, los compradores podrán acceder a un histórico de sus pedidos (RF21). Esta funcionalidad les será especialmente útil para consultar el estado actual de los pedidos y verificar si han sido preparados o están en reparto.
- **Consultar y actualizar datos personales.** De acuerdo con el RF17, los compradores podrán modificar la información que proporcionaron durante el registro, como su nombre, dirección, teléfono o contraseña, desde esta sección.
- **Programar pedidos.** Una funcionalidad que surge tras el desarrollo de la plataforma consiste en permitir a los compradores programar pedidos regulares. Esto les permitirá automatizar pedidos recurrentes, como productos que necesitan semanalmente, evitando la necesidad de solicitarlos manualmente cada vez.

Además de considerar nuevas funcionalidades, las líneas de trabajo futuras también incluyen **mejoras a nivel de código**. Estas mejoras no están relacionadas directamente con grandes funcionalidades visibles para el usuario, sino con pequeños detalles técnicos que impactan significativamente en la experiencia de usuario y en la eficiencia de la plataforma. A continuación, se detallan algunas propuestas:

- **Diseño responsive.** Desde el inicio, el diseño responsive fue clave para asegurar que la página web se adapte a diferentes tamaños de pantalla, especialmente a dispositivos móviles, que son los más utilizados por los usuarios. Actualmente, en la sección de vendedores, la barra lateral se muestra como en la Figura A.12. Como mejora, se propone implementar un menú flotante accesible desde un botón en la cabecera (Figura 8.1),

utilizando componentes de Angular Material para facilitar su integración y mejorar la usabilidad en móviles.

- **Accesibilidad.** La accesibilidad es un aspecto esencial que a menudo se pasa por alto pero que aporta un gran valor añadido. Se podría implementar el uso de ARIA labels (Accessible Rich Internet Applications), etiquetas que permiten a los lectores de pantalla describir los elementos de la interfaz de usuario. Estas etiquetas facilitan la navegación de personas con discapacidades visuales, mejorando significativamente la inclusión y usabilidad de la plataforma.
- **Soporte para varios idiomas.** Ya que la plataforma está diseñada para el comercio rural en España, incorporar soporte para idiomas como gallego, vasco, catalán o valenciano puede fomentar la inclusión lingüística. Este soporte podría implementarse utilizando herramientas de Angular como TranslateModule o TranslateService. Estas permiten definir los textos de la aplicación en archivos JSON separados para cada idioma. Al configurar los literales de la aplicación en estos archivos, se facilita la traducción a múltiples idiomas sin necesidad de modificar manualmente los textos en el código.
- **Uso del local storage.** El uso del local storage del navegador es una estrategia eficaz para mejorar el rendimiento, permitiendo almacenar información temporal y no confidencial localmente, evitando llamadas innecesarias a la base de datos.
- **Imágenes en los productos.** Actualmente, las tarjetas de los productos muestran únicamente el nombre, los ingredientes y el selector de cantidad. Incluir imágenes de los productos sería una mejora significativa, ya que las imágenes facilitan a los compradores identificar los productos de manera más visual e intuitiva. Esto no solo mejoraría la estética de la plataforma, sino también la experiencia de navegación además de brindar mejor servicio a vendedores y compradores.
- **Barras de búsqueda.** Agregar una barra de búsqueda sería una funcionalidad útil tanto para compradores como para vendedores. Para los compradores, facilitaría la selección de productos, especialmente en casos donde haya un catálogo amplio. Para los vendedores, una barra de búsqueda en la sección de gestión de productos ayudaría a encontrar rápidamente productos específicos. También podría ser muy útil en el historial de pedidos.
- **Filtros por fecha.** Incorporar filtros por fecha en la sección de pedidos permitiría a los usuarios visualizar pedidos realizados en un rango de tiempo determinado. Esto facilitaría la gestión de pedidos tanto para vendedores como para compradores, ofreciendo un acceso rápido a información relevante.
- **Seguridad.** Para mejorar la seguridad, se propone implementar tokens JWT para la autenticación, permitiendo gestionar sesiones sin enviar credenciales en cada solicitud. Además, se podrían aplicar medidas como hashing de contraseñas con bcrypt o Argon2, expiración y renovación de tokens, protección contra ataques comunes (fuerza bruta, XSS, inyecciones SQL), uso de HTTPS, entre otros. Estas mejoras podrían garantizar una mayor protección de los datos de los usuarios.

Además de las funcionalidades previamente mencionadas, hay ciertos aspectos clave que podrían desarrollarse en el futuro para mejorar la plataforma y adaptarla a un uso más amplio.

Uno de los puntos a considerar es la **integración real de lockers** para la recogida de pedidos. Actualmente, la funcionalidad está pensada como una opción de entrega, pero la implementación física de estos sistemas implicaría nuevos requisitos y reglas de negocio. Por ejemplo, sería necesario establecer un tiempo máximo de permanencia del pedido en el locker, que podría fijarse en un día para evitar problemas con productos perecederos. También habría que definir qué ocurre si el tiempo máximo se supera y el vendedor debe gestionar la retirada del pedido. Otra posible mejora sería permitir que el vendedor seleccione un locker disponible desde la plataforma en el momento de preparar el pedido, optimizando así la logística y asegurando un uso eficiente del espacio disponible.

Además, en un desarrollo más avanzado, se podría contemplar la incorporación de un sistema de hardware y software embebido para gestionar estos lockers de manera automática, permitiendo que la aplicación se comunique con ellos para la asignación de espacios, la generación de códigos de recogida y la monitorización del estado de los pedidos.

Por otro lado, otra línea de trabajo futura podría centrarse en la colaboración con ayuntamientos. En municipios pequeños, los ayuntamientos podrían desempeñar un papel clave en la validación de negocios que quieran unirse a la plataforma, asegurando que cumplen con ciertos requisitos legales o administrativos. También podrían gestionar el alta de direcciones dentro del sistema, por ejemplo, en zonas rurales donde la numeración de calles no es clara o donde existen dificultades para la localización de ciertos domicilios. Esto permitiría mejorar la precisión del servicio y facilitar el uso de la plataforma en entornos donde las direcciones no están siempre bien definidas en los mapas digitales.

Estas mejoras, junto con las ya planteadas, permitirían ampliar el alcance del proyecto y dotarlo de una mayor robustez para su implementación en escenarios reales.

Figura 8.1. Diseño responsive para la barra lateral de los vendedores

Bibliografía

Angular Material. 2020. Componentes de Angular Material. [En línea] 31 de mayo de 2020. [Citado el: 26 de octubre de 2024.]

Angular University. 2024. SPA: What are the benefits? [En línea] 17 de enero de 2024. [Citado el: 29 de septiembre de 2024.] https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/.

Arquitectura Java. 2021. ¿JPA vs Hibernate? . [En línea] 19 de junio de 2021. [Citado el: 29 de septiembre de 2024.] https://www.arquitecturajava.com/jpa-vs-hibernate/.

—. 2023. DAO vs. Repository y sus diferencias . [En línea] 12 de abril de 2023. [Citado el: 26 de octubre de 2024.] https://www.arquitecturajava.com/dao-vs-repository-y-sus-diferencias/.

Arsys. 2024. ¿Qué es MySQL? Explicación y características. [En línea] 7 de junio de 2024. [Citado el: 8 de febrero de 2025.] https://www.arsys.es/blog/mysql#:~:text=Qu%C3%A9%20es%20MySQL%3F-,MySQL%20es%20un%20sistema%20de%20gesti%C3%B3n%20de%20bases%20de%20datos,recuperar%20datos%20de%20manera%20eficiente..

Astah. 2024. Individual Licensing Options. [En línea] julio de 2024. [Citado el: 28 de septiembre de 2024.] https://astah.net/pricing/individual/.

—. 2011. Make software design more productive, rich and fun. [En línea] 29 de junio de 2011. [Citado el: 8 de febrero de 2025.] https://astah.net/products/astah-professional/.

Atlassian. 2020. Qué es Git. [En línea] 7 de marzo de 2020. [Citado el: 8 de febrero de 2025.] https://www.atlassian.com/es/git/tutorials/what-is-git.

—. 2013. Qué es Scrum y cómo empezar. [En línea] 26 de octubre de 2013. [Citado el: 21 de septiembre de 2024.] https://www.atlassian.com/es/agile/scrum.

AWS. 2022. ¿Qué es una API RESTful? [En línea] 7 de mayo de 2022. [Citado el: 25 de octubre de 2024.] https://aws.amazon.com/es/what-is/restful-api/.

Blancarte, Oscar. 2018. Singleton (Patrón de diseño creacional). [En línea] 14 de junio de 2018. [Citado el: 26 de octubre de 2024.] https://reactiveprogramming.io/blog/es/patrones-de-diseno/singleton.

Campus MVP. 2021. La API de persistencia de Java: ¿Qué es JPA? [En línea] 23 de marzo de 2021. [Citado el: 29 de septiembre de 2024.] https://www.campusmvp.es/recursos/post/la-api-de-persistencia-de-java-que-es-jpa-jpa-vs-hibernate-vs-eclipselink-vs-spring-jpa.aspx.

ComponentSource. 2013. Acerca de Astah. [En línea] 30 de enero de 2013. [Citado el: 29 de septiembre de 2024.] https://www.componentsource.com/es/product/astah-uml/about.

Desarrollo Web. 2016. Angular Material. [En línea] 16 de febrero de 2016. [Citado el: 26 de octubre de 2024.] https://desarrolloweb.com/articulos/angular-material.html.

Developer PayPal. 2022. Authentication . [En línea] 18 de mayo de 2022. [Citado el: 26 de enero de 2025.] https://developer.paypal.com/api/rest/authentication/.

—. 2022. Get an access token . [En línea] 19 de mayo de 2022. [Citado el: 26 de enero de 2025.] https://developer.paypal.com/reference/get-an-access-token/.

—. 2021. Payouts . [En línea] 7 de octubre de 2021. [Citado el: 26 de enero de 2025.] https://developer.paypal.com/docs/api/payments.payouts-batch/v1/#payouts_post.

Docker. 2019. The industry-leading container runtime. [En línea] 30 de abril de 2019. [Citado el: 29 de septiembre de 2024.] https://www.docker.com/products/container-runtime/.

Enngage. 2018. Angular PayPal. [En línea] 12 de marzo de 2018. [Citado el: 9 de enero de 2025.] https://enngage.github.io/ngx-paypal/.

Fotocasa. 2024. Oficinas de alquiler en Valladolid Capital. [En línea] 2024. [Citado el: 5 de octubre de 2024.] https://www.fotocasa.es/es/alquiler/oficinas/valladolid-capital/todas-las-zonas/l.

Ganttpro. 2024. Modelo en cascada (Waterfall): qué es y cuándo conviene usarlo. [En línea] enero de 2024. [Citado el: 21 de septiembre de 2024.] https://blog.ganttpro.com/es/metodologia-de-cascada/.

Genbeta. 2014. Node.js y npm. [En línea] 14 de octubre de 2014. [Citado el: 29 de septiembre de 2024.] https://www.genbeta.com/desarrollo/node-js-y-npm.

GitLab. 2020. Issue boards. [En línea] 15 de noviembre de 2020. [Citado el: 26 de octubre de 2024.] https://docs.gitlab.com/ee/user/project/issue_board.html.

Glassdoor. 2024. Sueldos de Desarrollador de Software en España. [En línea] 26 de septiembre de 2024. [Citado el: 28 de septiembre de 2024.] https://www.glassdoor.es/Sueldos/desarrollador-de-software-sueldo-SRCH_KO0,25.htm.

Hiberus. 2021. ¿Qué es Angular y para qué sirve? [En línea] 13 de octubre de 2021. [Citado el: 29 de septiembre de 2024.] https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/.

Hostinger. 2025. ¿Qué es Gitlab y para qué sirve? [En línea] 29 de enero de 2025. [Citado el: 8 de febrero de 2025.] https://www.hostinger.es/tutoriales/que-es-gitlab#:~:text=GitLab%20es%20una%20plataforma%20de,ramas%20para%20el%20desarrollo%20as%C3%ADncrono..

—. 2024. ¿Qué es un CDN? Content Delivery Network. [En línea] 27 de marzo de 2024. [Citado el: 9 de febrero de 2025.] https://www.hostinger.es/tutoriales/que-es-cdn.

HubSpot. 2023. ¿Qué es Angular? Características y ventajas. [En línea] 20 de enero de 2023. [Citado el: 29 de septiembre de 2024.] https://blog.hubspot.es/website/que-es-angular.

IBM. 2021. ¿Qué es Java Spring Boot? [En línea] 19 de diciembre de 2021. [Citado el: 29 de septiembre de 2024.] https://www.ibm.com/es-es/topics/java-spring-boot.

Idealista. 2024. Alquiler oficinas Valladolid. [En línea] 2024. [Citado el: 5 de octubre de 2024.] https://www.idealista.com/alquiler-oficinas/valladolid-valladolid/.

Imagina Formación. 2024. ¿Qué es el Patrón de Arquitectura (MVVM)? [En línea] 29 de mayo de 2024. [Citado el: 11 de octubre de 2024.] https://imaginaformacion.com/tutoriales/que-es-el-patron-de-arquitectura-mvvm.

IONOS. 2019. El modelo en cascada: desarrollo secuencial de software. [En línea] 3 de marzo de 2019. [Citado el: 21 de septiembre de 2024.] https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/.

KeepCoding. 2024. ¿Qué es el patrón de arquitectura MVVM? . [En línea] 22 de julio de 2024. [Citado el: 11 de octubre de 2024.] https://keepcoding.io/blog/que-es-el-patron-de-arquitectura-mvvm/.

—. 2024. ¿Qué es material design? . [En línea] 24 de abril de 2024. [Citado el: 20 de octubre de 2024.] https://keepcoding.io/blog/que-es-material-design/.

Material Design. 2016. Material Design. [En línea] 20 de octubre de 2016. [Citado el: 20 de octubre de 2024.] https://m3.material.io/.

Medium (Fernando Andrade). 2024. Spring - Uso de DTO vs Modelo . [En línea] 18 de marzo de 2024. [Citado el: 26 de octubre de 2024.] https://medium.com/@ingenieroandrade/uso-de-dto-vs-modelo-c1ae1d1797ea.

Medium. 2023. Angular Best Practices: Tips for Project Structure and Organization. [En línea] 22 de noviembre de 2023. [Citado el: 8 de enero de 2025.] https://medium.com/@marketing_26756/angular-best-practices-tips-for-project-structure-and-organization-490ca7950829.

—. 2016. Data Access Objects vs Repositories . [En línea] 2 de octubre de 2016. [Citado el: 26 de octubre de 2024.] https://medium.com/@jotauribe/data-access-objects-vs-repositories-b1497565a873.

Microsoft. 2023. Modelo-Vista-Modelo de vista (MVVM) . [En línea] 30 de mayo de 2023. [Citado el: 11 de octubre de 2024.] https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm.

—. 2024. Planes empresariales. [En línea] 2024. [Citado el: 4 de octubre de 2024.] https://www.microsoft.com/es-es/microsoft-teams/compare-microsoft-teams-business-options.

Microsoft Support. 2020. Introducción a Microsoft Teams. [En línea] 5 de junio de 2020. [Citado el: 8 de febrero de 2025.] https://support.microsoft.com/es-es/office/introducci%C3%B3n-a-microsoft-teams-b98d533f-118e-4bae-bf44-3df2470c2b12.

Moqups. 2024. Precios moqups. [En línea] 10 de octubre de 2024. [Citado el: 28 de septiembre de 2024.] https://moqups.com/pricing/?src=up.

MySQL. 2011. MySQL Installer 8.0.41. [En línea] 12 de agosto de 2011. [Citado el: 25 de enero de 2025.] https://dev.mysql.com/downloads/installer/.

NFON. 2022. ¿Qué es y cómo funciona Microsoft Teams? [En línea] 12 de julio de 2022. [Citado el: 29 de septiembre de 2024.] https://blog.nfon.com/es/que-es-y-como-funciona-microsoft-teams/.

NodeJS. 2016. Node JS. [En línea] 28 de julio de 2016. [Citado el: 25 de enero de 2025.] https://nodejs.org/es.

npm Inc. 2022. Angular PayPal. [En línea] 19 de diciembre de 2022. [Citado el: 9 de enero de 2025.] https://www.npmjs.com/package/ngx-paypal.

NTT Data. 2019. ¿Qué es Hibernate? ¿Por qué usarlo? [En línea] 6 de agosto de 2019. [Citado el: 29 de septiembre de 2024.] https://ifgeekthen.nttdata.com/s/post/que-es-java-hibernate-por-que-usarlo-MC5FU56AIPGBGIHNJ677RBIXUHOI?language=es.

OpenWebinars. 2018. [En línea] 16 de octubre de 2018. [Citado el: 11 de octubre de 2024.] https://openwebinars.net/blog/que-patron-usa-angular-mvc-o-mvvm/.

—. 2019. Qué es Apache Maven . [En línea] 29 de abril de 2019. [Citado el: 29 de septiembre de 2024.] https://openwebinars.net/blog/que-es-apache-maven/.

—. 2019. Qué es MySQL: características y ventajas. [En línea] 24 de septiembre de 2019. [Citado el: 8 de febrero de 2025.] https://openwebinars.net/blog/que-es-mysql/.

—. 2022. Visual Studio Code: Editor de código para desarrolladores. [En línea] 22 de julio de 2022. [Citado el: 29 de septiembre de 2024.] https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/.

Oracle. 2023. Java SE 21 Archive Downloads. [En línea] 19 de septiembre de 2023. [Citado el: 25 de enero de 2025.] https://www.oracle.com/java/technologies/javase/jdk21-archive-downloads.html.

Oscar Blancarte. 2024. Arquitectura en Capas. [En línea] 12 de abril de 2024. [Citado el: 25 de octubre de 2024.] https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas.

—. 2018. Data Transfer Object (DTO) . [En línea] 30 de noviembre de 2018. [Citado el: 26 de octubre de 2024.] https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/.

—. 2020. Data Transfer Object (DTO) . [En línea] 22 de marzo de 2020. [Citado el: 26 de octubre de 2024.] https://reactiveprogramming.io/blog/es/patrones-arquitectonicos/dto.

OurAcademy. 2019. El patrón Repository: implementación y buenas prácticas. [En línea] 11 de agosto de 2019. [Citado el: 26 de octubre de 2024.] https://our-academy.org/posts/el-patron-repository:-implementacion-y-buenas-practicas.

PayPal Community. 2021. "Token signature verification failed" when creating an order . [En línea] 16 de marzo de 2021. [Citado el: 26 de enero de 2025.] https://www.paypal-community.com/t5/Sandbox-Environment/quot-Token-signature-verification-failed-quot-when-creating-an/td-p/2638197.

PayPal Developer. 2016. Sandbox test accounts. [En línea] 30 de diciembre de 2016. [Citado el: 9 de enero de 2025.] https://developer.paypal.com/dashboard/accounts.

Platzi. 2023. ¿Qué es el Patrón Repository para Arquitecturas limpias? [En línea] 16 de junio de 2023. [Citado el: 26 de octubre de 2024.] https://platzi.com/blog/patron-repository/.

—. 2020. Patrón arquitectónico de Capas/Layers. [En línea] 20 de mayo de 2020. [Citado el: 25 de octubre de 2024.] https://platzi.com/tutoriales/1248-pro-arquitectura/5439-patron-arquitectonico-de-capas-layers/.

—. 2023. Patrón DAO y Repository . [En línea] 19 de septiembre de 2023. [Citado el: 12 de diciembre de 2024.] https://platzi.com/clases/8048-java-sql/64346-patron-dao-y-repository/.

PMBC. 2025. Gestión de riesgos en proyectos: identificación, evaluación y mitigación. [En línea] 2025. [Citado el: 6 de febrero de 2025.] https://pmbc.es/gestion-de-riesgos-en-proyectos-identificacion-evaluacion-y-mitigacion/?utm_source=chatgpt.com.

Proyectos Ágiles. 2015. Qué es SCRUM. [En línea] 25 de octubre de 2015. [Citado el: 21 de septiembre de 2024.]

RedHat. 2023. ¿Qué es una API de REST? [En línea] 31 de julio de 2023. [Citado el: 25 de octubre de 2024.] https://www.redhat.com/es/topics/api/what-is-a-rest-api.

Refactoring Guru. 2020. Singleton. [En línea] 15 de julio de 2020. [Citado el: 26 de octubre de 2024.] https://refactoring.guru/es/design-patterns/singleton.

SalySEO. 2024. [En línea] 27 de junio de 2024. [Citado el: 20 de octubre de 2024.] https://salyseo.com/diseno-ux-ui/que-es-material-design/#:~:text=A%20trav%C3%A9s%20de%20una%20estructura,y%20agradables%20para%20los%20usuarios..

Seguridad Social. 2024. Bases y tipos de cotización 2024. [En línea] 1 de enero de 2024. [Citado el: 28 de septiembre de 2024.] https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#:~:text=Desde%20el%201%20de%20enero%20de%202024%2C%20el%20tipo%20de,ciento%20a%20cargo%20del%20empleado..

SmartSheet. 2023. Cómo crear un plan de gestión de riesgos de un proyecto. [En línea] 27 de febrero de 2023. [Citado el: 6 de febrero de 2025.] https://es.smartsheet.com/content/project-risk-management-plan?utm_source=chatgpt.com.

Stackoverflow. 2023. How do I send a fetch request to Paypal's Oath API (v2)? [En línea] 16 de julio de 2023. [Citado el: 26 de enero de 2025.] https://stackoverflow.com/questions/74779027/how-do-i-send-a-fetch-request-to-paypals-oath-api-v2.

UXPin. 2023. [En línea] 13 de marzo de 2023. [Citado el: 20 de octubre de 2024.] https://www.uxpin.com/studio/blog/what-is-progressive-disclosure/#:~:text=Progressive%20disclosure%20is%20a%20technique,interface%20of%20a%20digital%20product..

Valladolid Movilidad Sostenible S.L. 2017. Vamos ya! Tu movilidad económica y funcional. [En línea] 4 de julio de 2017. [Citado el: 6 de febrero de 2025.] https://www.vamos-ya.com/.

Vallecerca. 2021. Vallecerca. [En línea] 12 de febrero de 2021. [Citado el: 21 de septiembre de 2024.] https://vallecerca.es/.

Virtual Project Manager. 2022. Crear mejores wireframes y prototipos con Moqups. [En línea] 12 de mayo de 2022. [Citado el: 29 de septiembre de 2024.] https://yourvirtualprojectmanager.com/es/crear-mejores-wireframes-y-prototipos-con-moqups/.

We are marketing. 2024. Qué es Scrum y cómo funciona esta metodología de trabajo. [En línea] 4 de septiembre de 2024. [Citado el: 21 de septiembre de 2024.] https://www.wearemarketing.com/es/blog/scrum-que-es-como-funciona.html#.

Working Days. 2024. Días laborables. [En línea] 2024. [Citado el: 28 de septiembre de 2024.] https://www.dias-laborables.es/Castilla%20y%20Le%C3%B3n.htm.

Apéndice A
Manuales

A.1. Manual de instalación y despliegue

Este manual proporciona las instrucciones necesarias para preparar el entorno de trabajo y realizar el despliegue del proyecto Mercado Rural, abarcando tanto los requisitos de instalación previos como los pasos para ejecutar correctamente el frontend y el backend.

A.1.1. Requisitos previos

Para garantizar el correcto funcionamiento del proyecto, es necesario cumplir con las siguientes especificaciones mínimas de sistema y software.

Requisitos del sistema:

- Windows 10 o superior

Requisitos de software:

- Frontend:
  - **Angular CLI** versión 18.2.9
  - **npm** versión 10.0.0
  - **Node.js** versión 20.18.0
- Backend:
  - **Java Development Kit** (JDK) versión 21 (debe estar configurado en el sistema)
  - **Gestor de base de datos:** MySQL Installer Community 8.0.40.0
  - **Extensión de MySQL** para Visual Studio Code (opcional para la gestión visual de la base de datos)

A.1.2. Instalación de herramientas y configuración

Instalación de Node.js y npm

Node.js se puede instalar desde la página oficial (NodeJS, 2016).

Para instalar Angular CLI, se debe utilizar el siguiente comando en la terminal:

```
npm install -g @angular/cli@18.2.9
```

Instalación de JDK 21

Se descarga en la página oficial de Oracle (Oracle, 2023).

Tras la instalación, hay que configurar la variable de entorno `JAVA_HOME` apuntando al directorio de instalación del JDK.

Instalación de MySQL

Se descarga desde el sitio oficial la versión *community* (MySQL, 2011).

Tras la instalación, se debe configurar un usuario administrador (no hace falta crear una base de datos ya que el proyecto la crea automáticamente).

Instalación de la extensión de MySQL para VS Code (opcional)

En Visual Studio Code, se puede instalar dicha extensión desde el Marketplace.

A.1.3. Despliegue del proyecto

En primer lugar, es necesario **clonar el repositorio** del proyecto desde GitLab ejecutando el siguiente comando:

```
git clone https://gitlab.inf.uva.es/tfg-mercado-rural/mercado-rural.git
```

A continuación, hay que navegar al directorio raíz del proyecto donde se encuentra el backend y ejecutar el siguiente comando para **iniciar la aplicación**:

```
& 'C:\Program Files\Java\jdk-21\bin\java.exe'
'@C:\Users\<usuario>\AppData\Local\Temp\cp_3ocdfetthe5psn6xoybykpuuz.argfile'
'es.uva.tfg.mercadorural.MercadoruralApplication'
```

En este comando, algunos parámetros pueden variar en función de la ubicación del JDK en el sistema o la ruta del archivo MercadoruralApplication. Alternativamente, si no se desea escribir el comando manualmente, se puede abrir el archivo MercadoruralApplication.java desde el entorno de desarrollo y seleccionar la opción "Run Java". Esto ejecutará automáticamente el comando anterior con los parámetros correctos según la configuración del equipo.

Finalmente, hay que acceder al directorio del proyecto correspondiente al frontend desarrollado en Angular e **instalar las dependencias** necesarias ejecutando:

```
npm install
```

Tras ello, **iniciar el servidor de desarrollo** con el siguiente comando:

```
ng serve
```

Verificación del despliegue:

- **Frontend.** La aplicación estará disponible en el navegador en la URL `http://localhost:4200`.
- **Backend.** El servidor REST estará activo en `http://localhost:8080`.

A.2. Manual de usuario

Una vez desplegada la aplicación, la primera pantalla que se presenta es la página de inicio de la Figura A.1. En esta página, se encuentra a la izquierda el logotipo de la plataforma acompañado de una breve descripción que introduce los objetivos y funcionalidades de la web. A la derecha, se muestran las dos opciones de roles disponibles en la plataforma: comprador y vendedor, permitiendo al usuario seleccionar el rol con el que desea interactuar en la aplicación.

Figura A.1. Página de inicio

A.2.1. Rol de comprador

Si se selecciona el rol de comprador, se accede a la página de inicio de sesión de la figura Figura A.2.

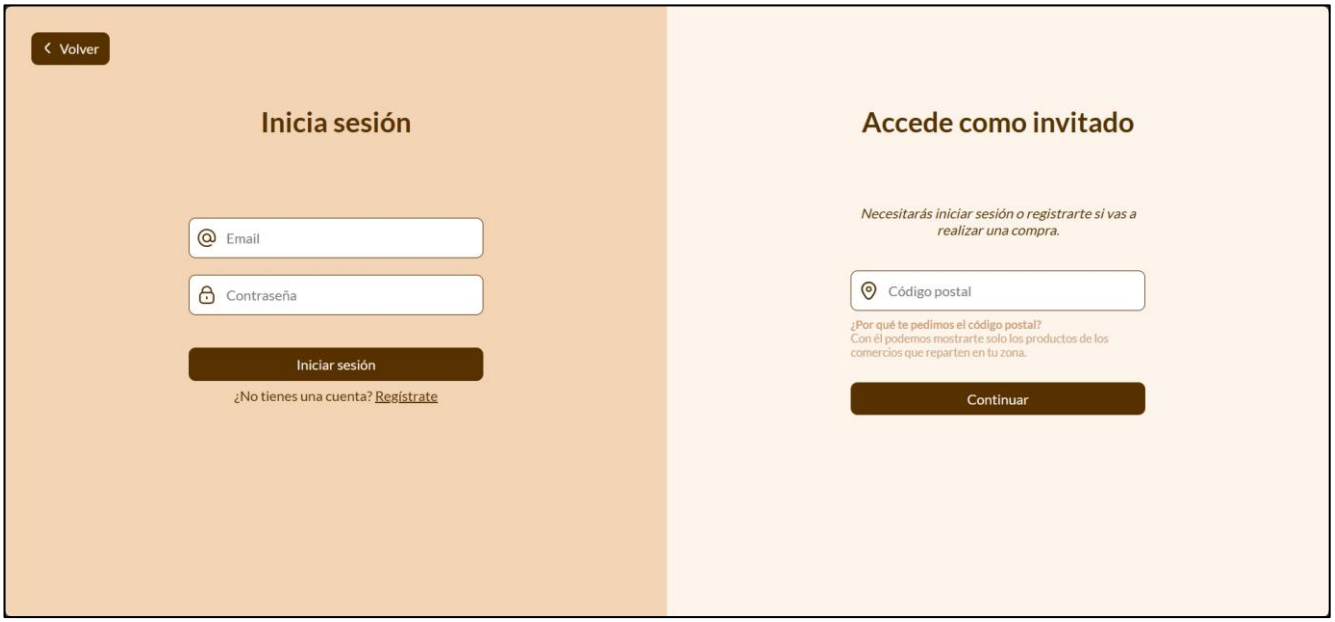

Figura A.2. Inicio de sesión de compradores

En esta página, se ofrecen dos opciones:

- **Iniciar sesión con una cuenta existente.** Si el usuario ya dispone de una cuenta, puede introducir sus credenciales en el formulario correspondiente para acceder al sistema.
- **Iniciar sesión como invitado.** Esta opción permite al usuario ingresar únicamente su código postal, con el objetivo de mostrar los comercios que realizan entregas en su área. Cabe destacar que esta funcionalidad no se encuentra implementada actualmente, pero está contemplada en el apartado 8.2. Líneas de trabajo futuras.

Si el usuario no dispone de una cuenta, puede registrarse pulsando la opción "Regístrate", ubicada debajo del formulario de inicio de sesión. Al seleccionar esta opción, se accede al formulario de registro (Figura A.3), donde se deben completar los campos requeridos. Una vez introducidos todos los datos, el sistema procede a registrar al nuevo usuario en la plataforma.

Tras haber iniciado sesión, el usuario accede a la página principal (home), como se muestra en la Figura A.4.

Una vez en la página de home, para comenzar a realizar compras e introducir productos en el carrito, es necesario pulsar el botón "Empezar a comprar", lo que redirige al catálogo de productos de los comercios, tal y como se observa en la Figura A.5. En esta sección, el usuario puede seleccionar las cantidades deseadas de cada producto. Una vez completada la selección, se puede acceder al carrito de compras para proceder con el pedido.

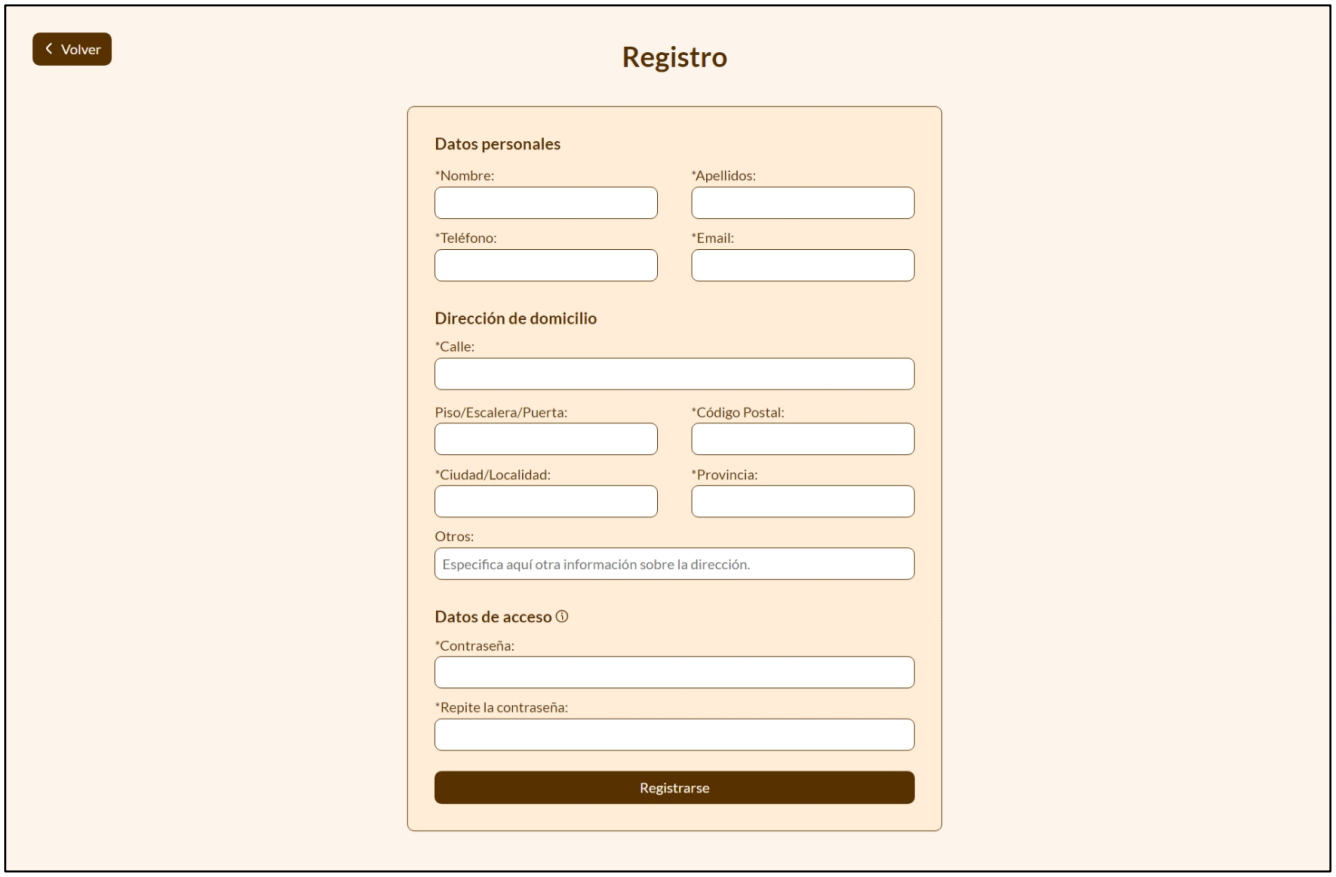

Figura A.3. Registro de compradores

Figura A.4. Página home de compradores

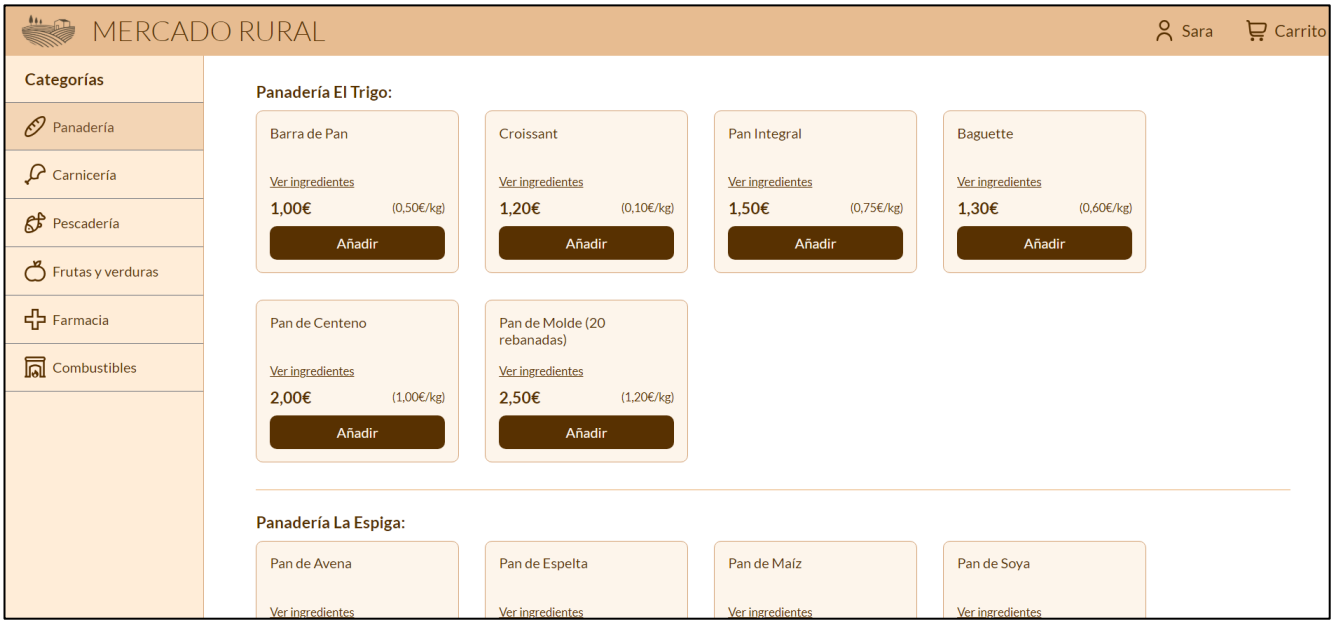

Figura A.5. Catálogo de productos

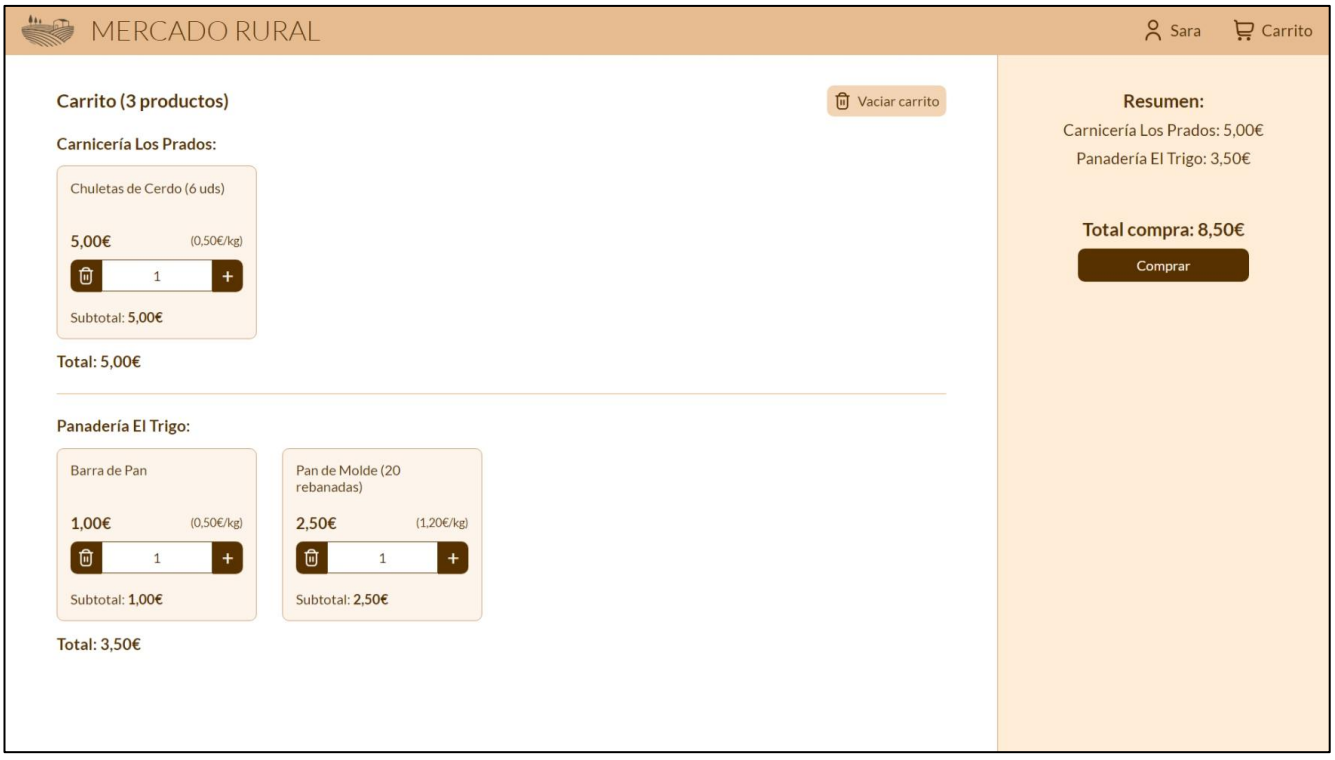

Figura A.6. Carrito de la compra

En el carrito (Figura A.6), el usuario tiene la posibilidad de modificar las cantidades seleccionadas previamente o eliminar productos si lo desea. Para finalizar la compra, se debe pulsar el botón "Comprar", lo que lleva a la pantalla de confirmación de pedido (Figura A.7). En esta etapa, el usuario debe elegir tanto la forma de pago como el tipo de envío que desea.

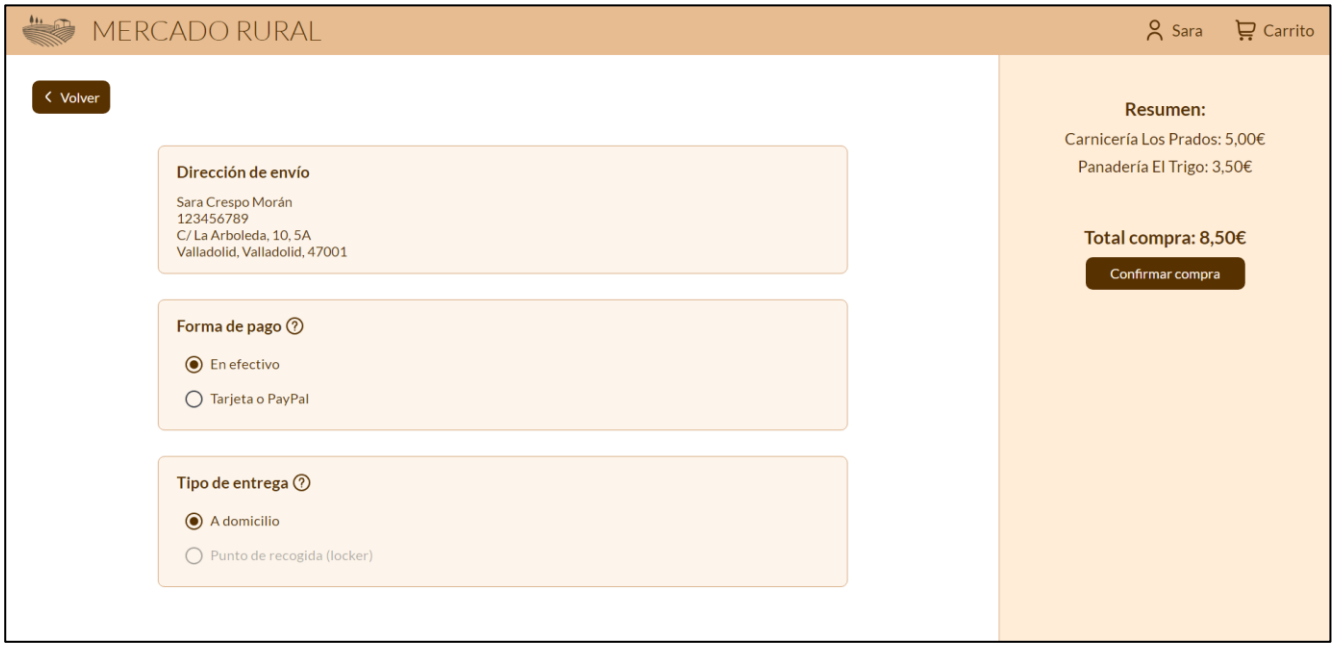

Figura A.7. Realizando un pedido

Hay dos métodos de pago disponibles:

- **Pago en efectivo.** Si se selecciona esta opción, basta con pulsar el botón "Confirmar compra" de la Figura A.7 y el pedido quedará registrado en el sistema.
- **Pago con PayPal.** Si se elige esta modalidad, aparece el botón de PayPal (Figura A.8). Al pulsarlo, se abre una ventana emergente (Figura A.9) donde el usuario debe iniciar sesión en su cuenta de PayPal y seguir los pasos indicados para completar el pago.

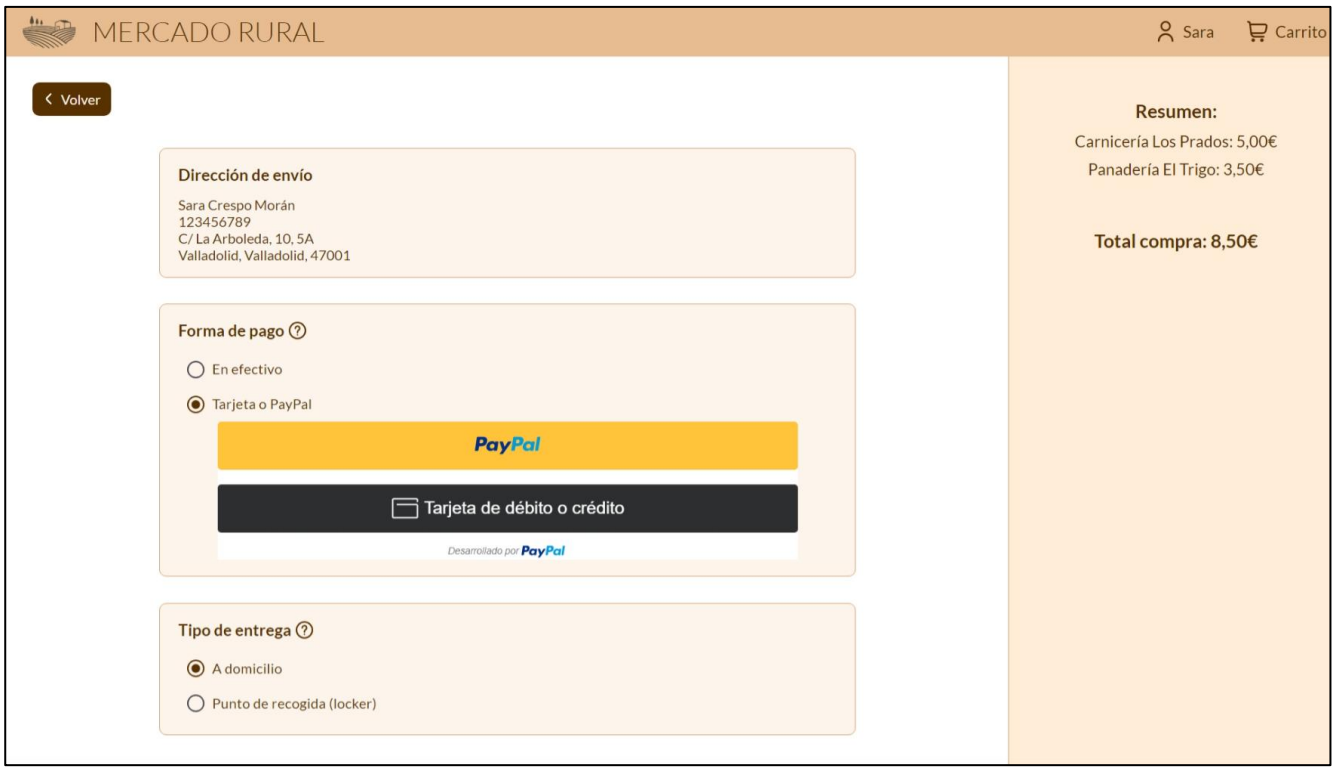

Figura A.8. Realizando un pedido con PayPal

Figura A.9. Ventana emergente de PayPal

Una vez confirmada la transacción, el pedido se procesa y el sistema registra el estado del mismo según el flujo establecido.

A.2.2. Rol de vendedor

Si en la Figura A.1 se selecciona el rol de vendedor, el sistema redirige a la página mostrada en la Figura A.10. Esta página presenta la misma funcionalidad que en el caso del comprador: se deben introducir las credenciales correspondientes si ya se dispone de una cuenta, o bien registrarse a través del formulario que se muestra en la Figura A.11 si no se cuenta con una.

Una vez que el vendedor inicia sesión, el sistema redirige a la página de gestión de pedidos, como se observa en la Figura A.12. En esta sección, se ofrece un resumen de los pedidos organizados por localidad, permitiendo al vendedor tener una visión clara de las áreas a las que debe atender.

Al pulsar en los detalles de cada localidad, se despliega una lista de los pedidos específicos asociados a esa área (Figura A.13). Desde esta lista, el vendedor tiene la capacidad de gestionar los pedidos, incluyendo la posibilidad de cambiar su estado según el flujo establecido (por ejemplo, de "solicitado" a "preparado", o de "preparado" a "entregado"). Esta funcionalidad permite al vendedor llevar un control eficiente de los pedidos y garantizar su correcta entrega.

Figura A.10. Inicio de sesión de vendedores

Figura A.11. Registro de vendedores

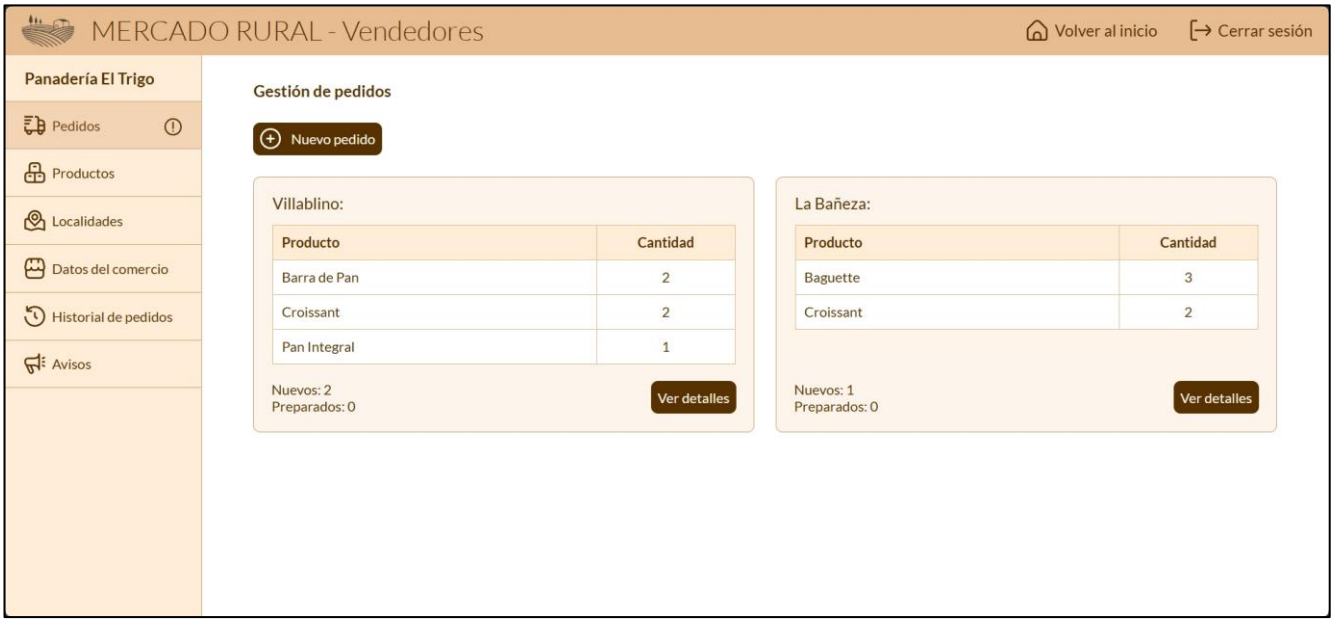

Figura A.12. Gestión de pedidos

Figura A.13. Detalles de los pedidos de una localidad

Para registrar un nuevo pedido que se haya recibido a través de una llamada telefónica, el vendedor debe hacer clic en el botón "Nuevo pedido" de la Figura A.12. Esto abrirá un formulario, tal y como se muestra en la Figura A.14, donde el vendedor deberá completar la información del cliente. Una vez introducidos los datos del cliente, el sistema guiará al vendedor a través de los siguientes pasos necesarios para finalizar el registro del pedido. Estos pasos están reflejados en las Figura A.15 y Figura A.16, permitiendo así al vendedor completar el pedido.

Figura A.14. Crear nuevo pedido como vendedor (paso 1/3)

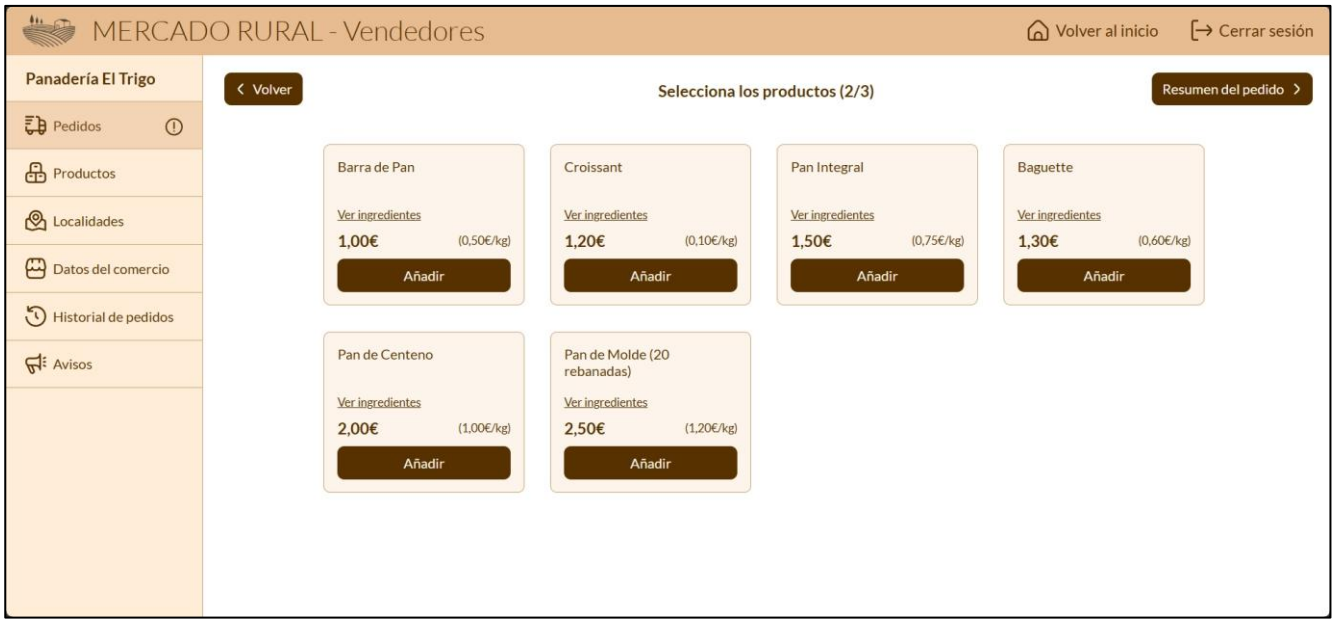

Figura A.15. Crear nuevo pedido como vendedor (paso 2/3)

Figura A.16. Crear nuevo pedido como vendedor (paso 3/3)

Por último, para gestionar sus productos, el vendedor debe seleccionar la opción "Productos" en la barra lateral, como se muestra en la Figura A.17. Desde esta sección, puede añadir productos nuevos mediante un formulario (Figura A.18), editar información de productos existentes y controlar la disponibilidad, marcándolos como disponibles o no disponibles según desee. Esta funcionalidad permite administrar el inventario de forma sencilla y eficiente.

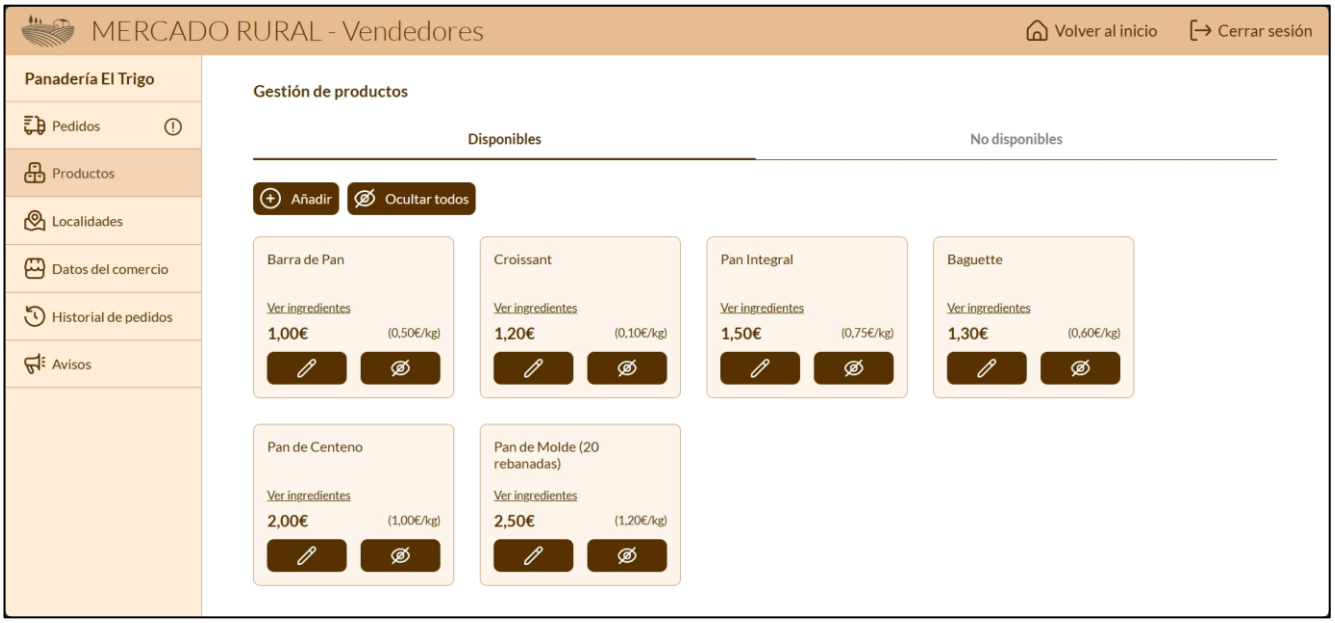

Figura A.17. Gestión de productos

Figura A.18. Añadir nuevo producto como vendedor

Apéndice B
Resumen de enlaces adicionales

El enlace de interés en este Trabajo de Fin de Grado es:

- Repositorio del código del proyecto:
  `https://gitlab.inf.uva.es/tfg-mercado-rural/mercado-rural`

  Nota: El repositorio está alojado en GitLab con visibilidad interna, lo que significa que solamente pueden acceder a él los usuarios autenticados dentro de la plataforma.