

TRABAJO FIN DE GRADO



Técnico Superior en Desarrollo de
Aplicaciones Multiplataforma

Autor: Antonio Pérez Cuéllar

Second Life



Alcalá de Henares, junio de 2025

Índice

0. Introducción.....	3
1. Estudio de mercado.....	4
1.1 Competencia actual.....	4
1.2 Oportunidad en el mercado.....	4
1.3 Propuesta de valor.....	4
2. Metodologías usadas.....	6
2.1 Metodología de desarrollo.....	6
2.2 Metodología de diseño.....	7
2.3 Metodología de gestión.....	8
3. Infraestructura.....	9
3.1 Tecnologías y herramientas utilizadas.....	9
3.2 Flujo de la aplicación.....	11
3.3 Planificación del proyecto.....	13
4. Análisis.....	14
4.1 Diagrama de casos de uso.....	14
4.2 Explicación de Diagrama de casos de uso.....	15
5. Diseño.....	17
5.1 Diseño de la base de datos (ER).....	17
5.2 Diseño del back-end.....	19
5.2 Diseño del front-end.....	26
6. Resultados.....	48
6.1 Objetivos alcanzados.....	48
6.2 Dificultades encontradas.....	49
6.3 Futuras mejoras.....	50
7. Conclusiones.....	51

O. Introducción

Este proyecto consiste en el desarrollo de una aplicación web orientada a la compra y venta de artículos de segunda mano.

El objetivo principal es ofrecer a los usuarios una plataforma donde puedan dar una segunda vida a aquellos objetos que ya no utilizan. La aplicación permite tanto la venta como la compra de productos.

Para acceder a todas las funcionalidades, el único requisito es registrarse mediante un correo electrónico válido.

Funcionalidades

La aplicación permite a los usuarios actuar tanto como compradores como vendedores, de forma que pueden publicar productos que ya no utilicen y, a su vez, adquirir artículos que necesiten por un precio reducido.

Para garantizar una experiencia segura, al subir un producto será obligatorio incluir al menos una fotografía que permita al comprador visualizar el estado del artículo.

¿A qué público va dirigida esta aplicación?

Esta aplicación está dirigida a personas que disponen de objetos en buen estado que ya no utilizan, pero que no desean desechar por motivos económicos, emocionales o medioambientales. También está orientada a quienes buscan productos de segunda mano a precios más accesibles.

Antecedentes

Existen diversas plataformas en el mercado que ofrecen servicios similares, entre ellas destacan:

- **Wallapop:** Aplicación muy popular en España para la compra-venta de productos de segunda mano entre particulares.
- **Vinted:** Plataforma centrada principalmente en ropa y accesorios.

1. Estudio de mercado

1.1 Competencia actual

Actualmente, existen al menos cuatro aplicaciones populares en el ámbito de la compraventa de productos de segunda mano: **Wallapop**, **Vinted**, **Facebook Marketplace** y **Milanuncios**. De todas ellas, las dos que han tenido mayor repercusión en los últimos años son **Wallapop** y **Vinted**, consolidándose como referentes del sector.

Ambas plataformas concentran una gran parte del tráfico de usuarios que buscan tanto vender como adquirir artículos usados. De hecho, es común que las personas interesadas en vender productos utilicen varias plataformas a la vez, subiendo sus anuncios en Wallapop y Vinted simultáneamente con el objetivo de obtener la máxima visibilidad posible. Esta práctica refleja tanto la competitividad del mercado como la demanda real de este tipo de servicios.

1.2 Oportunidad en el mercado

A pesar de la existencia de estas plataformas consolidadas, el número de alternativas sigue siendo limitado, y la mayoría de usuarios suele recurrir siempre a las mismas opciones. Este contexto representa una **oportunidad estratégica** para introducir una nueva aplicación con un enfoque más **simplificado**, capaz de atraer a un público que busca alternativas eficaces a las actuales soluciones.

1.3 Propuesta de valor

Mi propuesta busca aprovechar este hueco, ofreciendo una experiencia de usuario intuitiva y centrada en lo esencial, sin las distracciones o complicaciones que pueden presentar las aplicaciones más grandes. Esto, junto con una base de funcionalidades clave bien diseñadas, posiciona la aplicación como una opción realista y competitiva dentro del mercado.

Tablas comparativas

Plataforma	Puntos fuertes	Puntos débiles
Wallapop	Gran número de usuarios, variedad de productos	Saturación de anuncios, poca moderación
Vinted	Buen sistema de envíos, centrado en moda	Comisiones altas
Second life	Interfaz simplificada, variedad de productos	Nueva en el mercado y sin desarrollo completado

Plataforma	Público objetivo	Detalles de los productos	Chat	Filtros	Búsqueda	Perfiles de usuarios	Ofertas	Valoraciones	Fotos de los productos
Wallapop	Público general	✓	✓	✓	✓	✓	✓	✓	✓
Vinted	Centrado en moda	✓	✓	✓	✓	✓	✓	✓	✓
Second life	Público general	✓	✗	✓	✓	✓	✗	✗	✓

2. Metodologías usadas

2.1 Metodología de desarrollo

Para el desarrollo del proyecto se optó por una metodología ágil basada en Scrum, estableciendo objetivos semanales divididos en tareas concretas. Esta aproximación permitió organizar el trabajo de forma iterativa, facilitando la adaptación continua a los requisitos y la incorporación de mejoras durante el proceso. Gracias a esta metodología, se logró mantener un ritmo constante de progreso, asegurar la calidad del código y responder ágilmente a los posibles obstáculos o cambios necesarios.

Este es el flujo típico a seguir en la metodología ágil Scrum. En este tipo de metodologías, varias personas suelen asumir diferentes roles, como Scrum Master, Product Owner y equipo de desarrollo. Sin embargo, en el caso del autor, al trabajar individualmente, yo asumí todos estos roles.

Aunque esto supone una mayor carga de trabajo y responsabilidad, me permitió aplicar los principios de Scrum y organizar el desarrollo mediante sprints semanales, en los que debía completar un conjunto definido de tareas. Esta práctica favoreció una gestión eficiente del tiempo, contribuyendo a un desarrollo constante y ordenado del proyecto.

2.2 Metodología de diseño

Diseño Entidad-Relación (ER)

El diseño entidad-relación fue uno de los primeros pasos en el desarrollo del proyecto, ya que permite definir desde el inicio cómo se relacionan las distintas entidades del sistema. Esta fase resulta fundamental, ya que un buen diseño garantiza una base sólida sobre la que construir la aplicación. Por el contrario, un diseño incorrecto puede generar inconsistencias y dificultar el desarrollo, arrastrando problemas estructurales a lo largo de todo el proyecto.

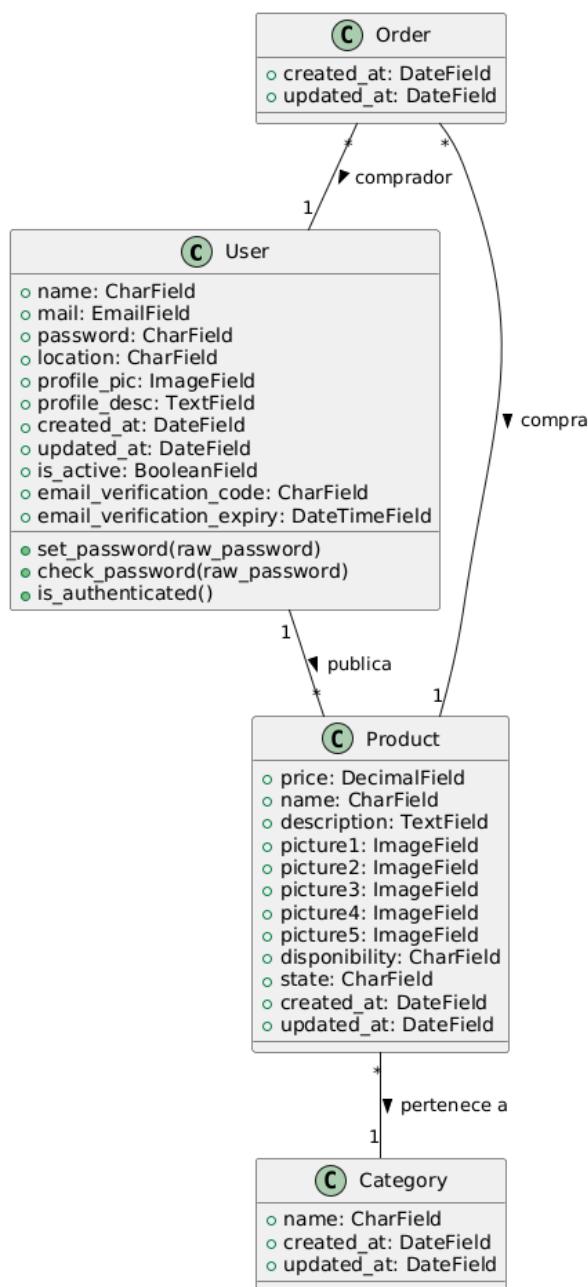
Más información en el apartado **5.1 Diseño de la base de datos**

Diseño basado en modelos (Diagrama de clases)

Una vez definido el modelo entidad-relación, se procedió a implementar los modelos de datos en Django (Python), los cuales permiten generar las migraciones necesarias para crear las tablas en la base de datos. Este enfoque facilita mantener la coherencia entre el diseño lógico y la implementación real.

Este proceso se detalla más adelante en el apartado 3. Infraestructura.

En mi caso el diagrama es bastante parecido al esquema relacional, ya que los modelos creados en python reflejan la base de datos.

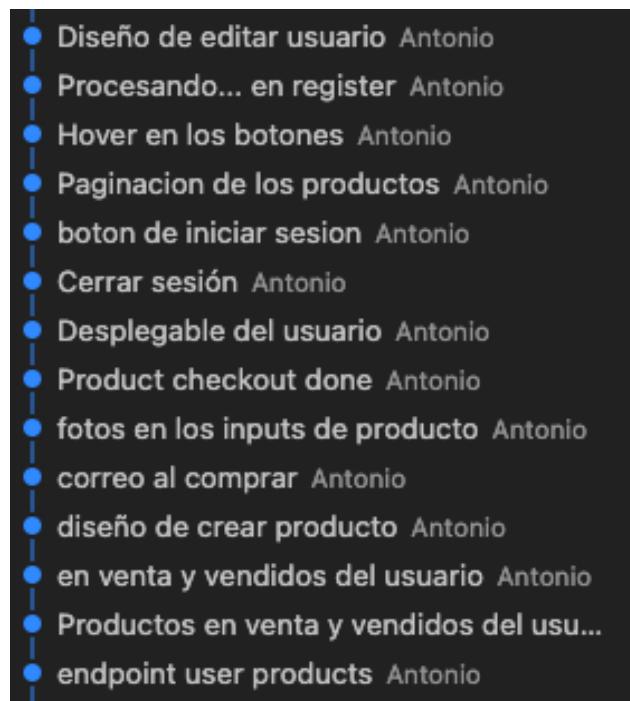


2.3 Metodología de gestión

La gestión de versiones del proyecto se ha realizado utilizando Git como sistema de control de versiones, junto con GitHub como plataforma de almacenamiento remoto.

Aunque lo más recomendable en este tipo de metodologías es trabajar mediante un flujo de ramas por tareas (feature branches), en este caso, al tratarse de un proyecto individual en el que asumía todos los roles, se optó por una estructura más simple.

Los cambios se subían al repositorio una vez completadas tareas específicas, como por ejemplo el desarrollo de la interfaz de la tarjeta de producto. Este enfoque permitió mantener un control claro y ordenado del progreso del proyecto sin añadir una complejidad innecesaria.



Durante el desarrollo, se realizaron múltiples *commits* bien organizados y con mensajes descriptivos, incluso para cambios pequeños. Esta estrategia permite mantener un historial limpio y comprensible del proyecto, facilitando la localización de errores o regresiones. En caso de que surgiera algún problema, se podía volver fácilmente a un estado anterior del proyecto simplemente retrocediendo a un commit previo estable.

3. Infraestructura

3.1 Tecnologías y herramientas utilizadas

En el desarrollo de esta aplicación se han utilizado las siguientes tecnologías y herramientas principales:

- Backend: Django (Python), junto con Django REST Framework para la creación de la API REST. Se utilizaron librerías como djoser y rest_framework_simplejwt para gestionar la autenticación y verificación de usuarios mediante tokens JWT, permitiendo el acceso a ciertas funcionalidades solo si el usuario está autenticado. También se configuró CORS para permitir que el frontend, alojado en un origen diferente, pudiera comunicarse con la API.
- Frontend: Vue.js (JavaScript) fue el framework elegido para desarrollar la interfaz de usuario, utilizando TailwindCSS para aplicar un diseño responsivo de forma ágil y eficiente. Se empleó Pinia para la gestión del estado global de la aplicación y Axios para realizar las peticiones HTTP a la API de forma estructurada y sencilla.
- Base de datos: Se utilizó MySQL como sistema de gestión de bases de datos relacional por su rapidez, fiabilidad y compatibilidad tanto con entornos Unix como Windows.
- Control de versiones: Git fue el sistema elegido para el control de versiones del código fuente, y GitHub se usó como plataforma de alojamiento remoto por su facilidad de configuración y uso. Aunque existen otras alternativas como GitLab o Bitbucket (con las que también me he familiarizado durante mis prácticas), GitHub resultó la más conveniente para este proyecto individual.
- Contenedores: Docker y Docker Compose permitieron contenerizar la aplicación y sus servicios, facilitando la configuración, despliegue y puesta en marcha en entornos locales.

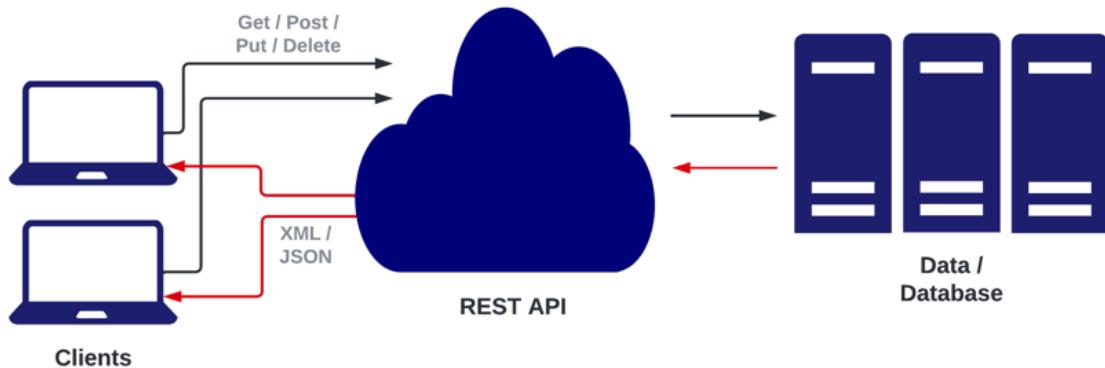
- Otras herramientas:

- MySQL Workbench: se utilizó para diseñar y visualizar el esquema de base de datos mediante diagramas Entidad-Relación, así como para generar automáticamente el código SQL de creación de tablas.
- Visual Studio Code (VSCode): fue el editor de código principal por su versatilidad, amplia comunidad y compatibilidad con múltiples lenguajes y tecnologías.
- Postman: Se utilizó para probar y verificar las llamadas a la API. Con esta herramienta, se introducía la URL correspondiente, el método HTTP deseado (por ejemplo, GET) y en los headers se incluía el token de acceso del usuario autenticado. Si la petición era correcta, la API respondía con un JSON que contenía los datos solicitados.
- También se ha utilizado Gmail para enviar correos de verificación y compras a los usuarios.

Se ha optado por estas herramientas por ser tecnologías modernas y ampliamente utilizadas en el desarrollo web actual. El uso de frameworks como Vue.js (composition API) en el frontend y Django en el backend permite adoptar una arquitectura full-stack basada en componentes, orientada a servicios (API), y alineada con las prácticas profesionales más actuales.

3.2 Flujo de la aplicación

La aplicación sigue un flujo típico cliente-servidor orientado a REST:



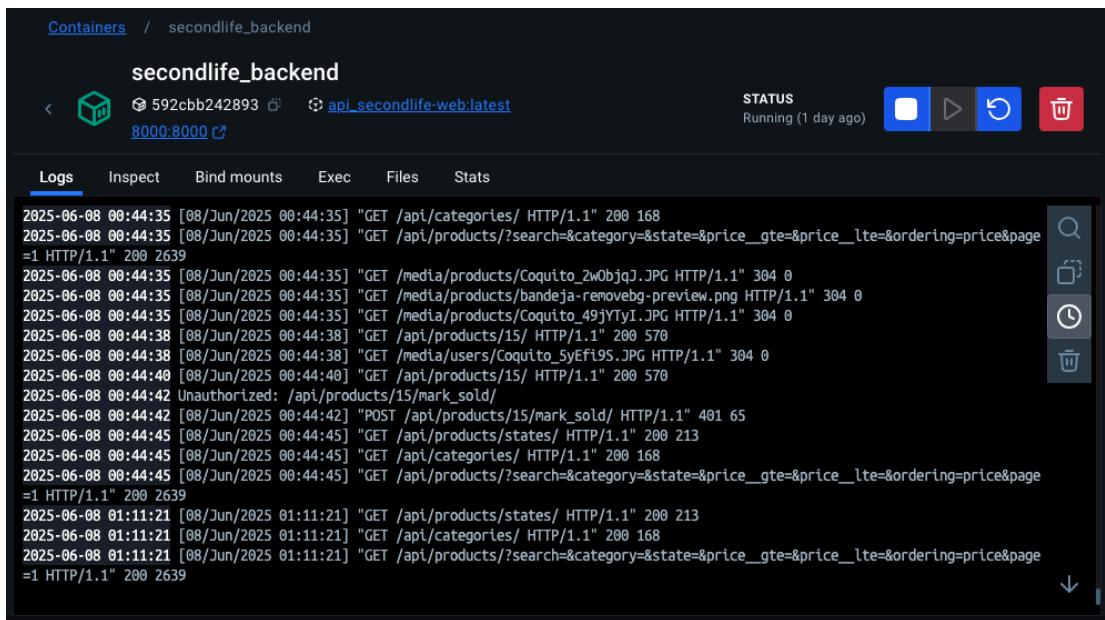
Esquema del flujo cliente – front-end – back-end – API – BBDD

- El **usuario** interactúa con la interfaz web creada en Vue.js, accediendo a funcionalidades como registro, login, búsqueda y filtrado de productos, y gestión de sus publicaciones.
- El **frontend** realiza peticiones HTTP a la API REST creada con Django, enviando y recibiendo datos en formato JSON. Estas peticiones se gestionan mediante endpoints definidos en el backend, que permiten estructurar y ordenar las llamadas de forma clara y eficiente.
- La **API** procesa las solicitudes, interactuando con la base de datos MySQL a través de los modelos definidos en Django, de manera similar a cómo funciona Hibernate en Java. Además, aplica la lógica de negocio, incluyendo la autenticación mediante tokens **JWT**, validaciones y control de permisos.
- Las **imágenes y archivos multimedia** se almacenan en un directorio configurado para tal fin y se sirven directamente desde el backend. Lo ideal sería contar con un servidor remoto especializado para el almacenamiento, guardando en la API solo las URLs de acceso. Sin embargo, en este proyecto se optó por almacenar los archivos localmente, debido a la imposibilidad de disponer de un servidor dedicado para este propósito.

- Todo el sistema se ejecuta en contenedores Docker, lo que garantiza un entorno homogéneo tanto para el desarrollo local como para el despliegue. Los contenedores usan sistemas operativos basados en Unix, por lo que las terminales de Docker donde se realizan instalaciones o consultas de logs utilizan comandos propios de Unix.

	Name ↑	Container ID	Image	Port(s)
□	api_secondlife	-	-	-
□	secondlife_be	592cbb242893	api_second	8000:8000 ↗
□	secondlife_dt	ed0693af5e16	mysql:8	3306:3306 ↗

Ejemplo de logs que se ven en docker de las peticiones a la API



The screenshot shows the Docker interface with the 'Containers' tab selected. A specific container named 'secondlife_backend' is highlighted. The container's status is 'Running (1 day ago)'. The 'Logs' tab is active, displaying the following log entries:

```

2025-06-08 00:44:35 [08/Jun/2025 00:44:35] "GET /api/categories/ HTTP/1.1" 200 168
2025-06-08 00:44:35 [08/Jun/2025 00:44:35] "GET /api/products/?search=&category=&state=&price_gte=&price_lte=&ordering=price&page=1 HTTP/1.1" 200 2639
2025-06-08 00:44:35 [08/Jun/2025 00:44:35] "GET /media/products/Coquito_2wObjqJ.JPG HTTP/1.1" 304 0
2025-06-08 00:44:35 [08/Jun/2025 00:44:35] "GET /media/products/bandeja-removebg-preview.png HTTP/1.1" 304 0
2025-06-08 00:44:35 [08/Jun/2025 00:44:35] "GET /media/products/Coquito_49jYtyI.JPG HTTP/1.1" 304 0
2025-06-08 00:44:38 [08/Jun/2025 00:44:38] "GET /api/products/15/ HTTP/1.1" 200 570
2025-06-08 00:44:38 [08/Jun/2025 00:44:38] "GET /media/users/Coquito_5yEf19S.JPG HTTP/1.1" 304 0
2025-06-08 00:44:40 [08/Jun/2025 00:44:40] "GET /api/products/15/ HTTP/1.1" 200 570
2025-06-08 00:44:42 Unauthorized: /api/products/15/mark_sold/
2025-06-08 00:44:42 [08/Jun/2025 00:44:42] "POST /api/products/15/mark_sold/ HTTP/1.1" 401 65
2025-06-08 00:44:45 [08/Jun/2025 00:44:45] "GET /api/products/states/ HTTP/1.1" 200 213
2025-06-08 00:44:45 [08/Jun/2025 00:44:45] "GET /api/categories/ HTTP/1.1" 200 168
2025-06-08 00:44:45 [08/Jun/2025 00:44:45] "GET /api/products/?search=&category=&state=&price_gte=&price_lte=&ordering=price&page=1 HTTP/1.1" 200 2639
2025-06-08 01:11:21 [08/Jun/2025 01:11:21] "GET /api/products/states/ HTTP/1.1" 200 213
2025-06-08 01:11:21 [08/Jun/2025 01:11:21] "GET /api/categories/ HTTP/1.1" 200 168
2025-06-08 01:11:21 [08/Jun/2025 01:11:21] "GET /api/products/?search=&category=&state=&price_gte=&price_lte=&ordering=price&page=1 HTTP/1.1" 200 2639

```

3.3 Planificación del proyecto

El proyecto se organizó siguiendo una metodología ágil basada en Scrum, adaptada a un entorno de trabajo individual. Se establecieron *sprints* semanales con objetivos concretos divididos en tareas específicas.

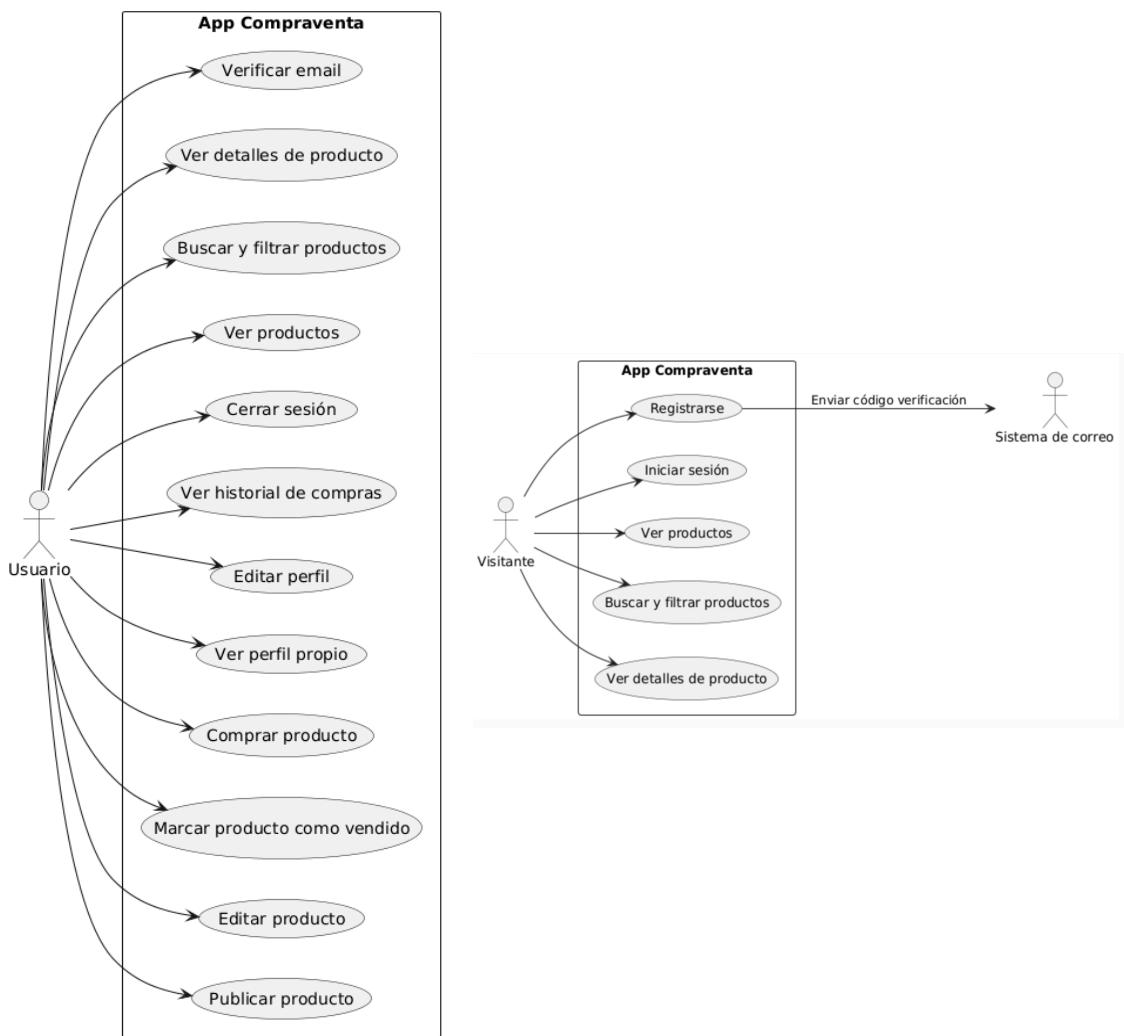
- Estos “**sprints**” se estructuraron en **dos grandes etapas**: la primera centrada en el desarrollo del **backend**, y la segunda en el **frontend**. Esta decisión se tomó con el objetivo de contar con una API funcional desde el principio, lo que permitiría enfocarse más adelante exclusivamente en la parte visual y en la integración con el backend. Sin embargo, durante la implementación del frontend surgieron algunos ajustes necesarios en el backend, algo habitual en el desarrollo de proyectos reales.
- Cada **tarea** completada se **revisaba**, se **probaba** y se **subía** al repositorio en GitHub, lo que permitía mantener un control detallado del progreso. Gracias a una buena organización de los *commits*, fue posible retroceder fácilmente a versiones anteriores en caso de errores o regresiones.
- El uso de **Docker** facilitó la automatización de los entornos de desarrollo, haciendo que la puesta en marcha del proyecto fuese tan sencilla como ejecutar un único comando. Esto garantizó un entorno homogéneo y portable.
- La planificación del proyecto se fue adaptando de forma flexible a lo largo del desarrollo, permitiendo hacer frente a posibles dificultades o cambios en los requisitos sin comprometer el ritmo de trabajo.

Además, para organizar y visualizar las tareas semanales, se fueron utilizando notas sobre lo que se me ocurría que necesitaba y se pasaba a limpio como tareas. Esta planificación estructurada no solo mejoró la eficiencia del desarrollo, sino que también fomentó una mayor conciencia sobre la gestión del tiempo y la priorización de tareas, habilidades clave en cualquier entorno profesional.

4. Análisis

4.1 Diagrama de casos de uso

Así queda el diagrama de casos de uso con ambos tipos de usuarios, el registrado y el no registrado.



4.2 Explicación de Diagrama de casos de uso

A primera vista, el diagrama de casos de uso puede parecer que tiene pocas funcionalidades, pero en realidad refleja de forma concisa las principales acciones disponibles en la aplicación, muchas de las cuales agrupan procesos más complejos.

Por ejemplo, tanto el registro como el inicio de sesión están integrados en la misma vista, mediante un sistema de pestañas que permite al usuario cambiar entre ambas opciones fácilmente. Para registrarse, únicamente se necesita una dirección de correo válida , ya que se enviará un correo con un código de verificación. Una vez verificada la cuenta, el usuario es redirigido automáticamente a la página de inicio de sesión.

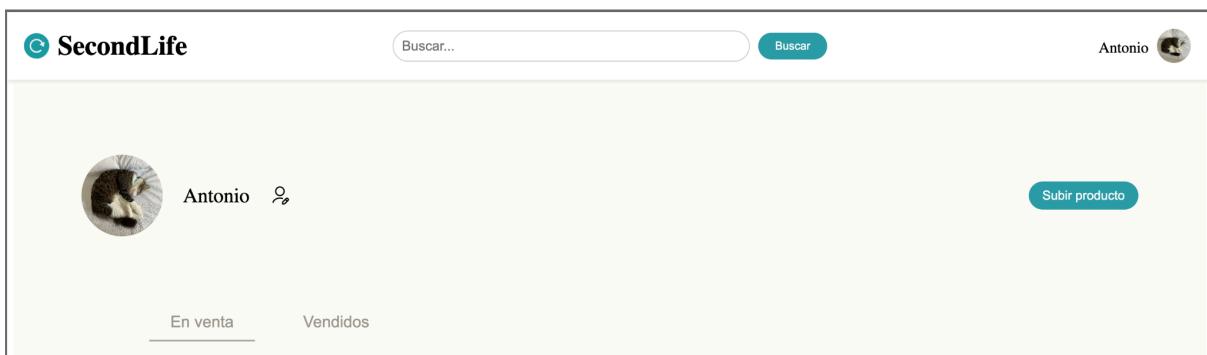
The image contains two screenshots of a login/register form for 'SecondLife'. Both screenshots show a header with the 'SecondLife' logo and navigation links for 'Iniciar sesión' and 'Registrarse'. The first screenshot shows the login section with fields for 'E-mail' and 'Contraseña', and a teal 'Iniciar sesión' button. The second screenshot shows the registration section with fields for 'E-mail', 'Nombre de usuario', 'Contraseña', and 'Repite la contraseña', and a teal 'Continuar' button.

La página principal es de acceso público, incluso para usuarios no autenticados. Esto se ha diseñado así intencionadamente para permitir que cualquier visitante pueda navegar por los productos, aplicar filtros de búsqueda y explorar la plataforma sin necesidad de registrarse. De esta forma, se mejora la accesibilidad y se fomenta el interés antes de pedir ningún dato.

Sin embargo, ciertas acciones están restringidas a usuarios autenticados. Por ejemplo, aunque cualquier visitante puede acceder a la vista detallada de un producto, no podrá realizar acciones como comprar, crear, editar publicaciones o acceder a su perfil personal. Para eso es imprescindible iniciar sesión. En lugar de mostrar el acceso al perfil, la interfaz presentará el botón de “Iniciar sesión” si el usuario aún no está autenticado.



Una vez logueado, el usuario podrá acceder a su perfil, desde donde podrá gestionar sus productos: crear nuevas publicaciones, editarlas o marcarlas como vendidas. Estas acciones están agrupadas dentro del caso de uso del perfil, lo que mantiene el diagrama más limpio y enfocado.



5. Diseño

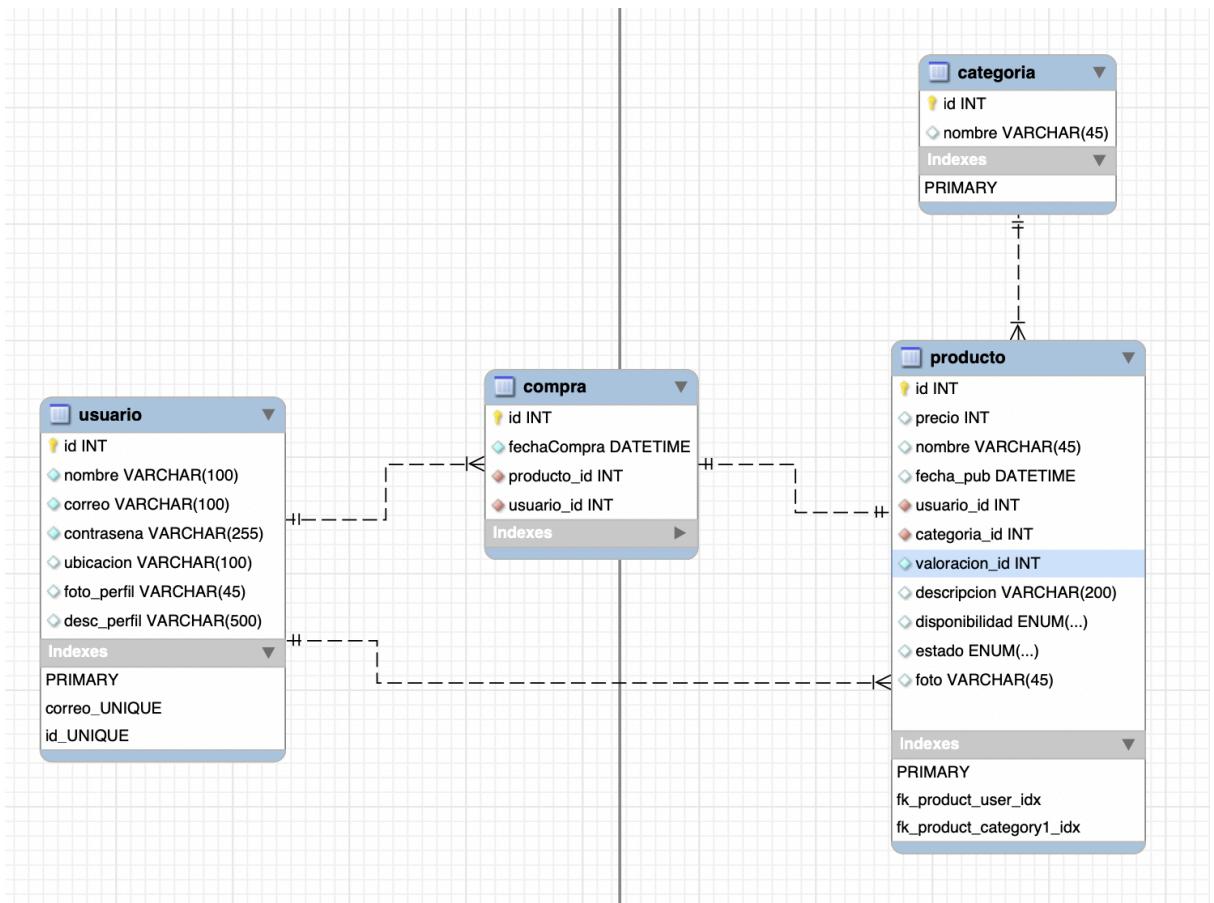
En este apartado se detalla el diseño técnico del sistema, desde el modelo de datos hasta la estructura de clases y módulos que componen tanto el backend como el frontend de la aplicación. Se incluyen el diagrama Entidad-Relación y la descripción de los componentes más relevantes, reflejando la arquitectura general del proyecto.

5.1 Diseño de la base de datos (ER)

La base de datos ha sido modelada mediante un diagrama Entidad-Relación (ER) que representa las entidades principales del sistema y las relaciones entre ellas. Las entidades más relevantes son:

- **Usuario:** contiene los datos personales y de autenticación. Se utiliza un modelo personalizado, que tiene atributos como nombre de usuario, correo (usado como identificador principal), contraseña, foto de perfil, descripción y localización.
- **Producto:** representa los artículos publicados por los usuarios. Incluye atributos como título, descripción, precio, estado, disponibilidad y soporte para múltiples imágenes (hasta un máximo de cinco por producto y un mínimo de una).
- **Categoría:** permite clasificar los productos. Un producto pertenece a una única categoría, y los usuarios pueden filtrar el listado por ellas.
- **Compra:** relación entre el comprador y el producto adquirido. Permite verificar si un usuario puede valorar un producto, y en el back, al realizar una compra se manda un correo con los datos de la compra.

El diagrama ER completo se incluye a continuación:



5.2 Diseño del back-end

El backend está desarrollado en **Django** con **Django REST Framework**. La arquitectura sigue un modelo modular organizado por apps, facilitando el mantenimiento y la escalabilidad del proyecto.

Se implementa una API REST con rutas bien definidas, autenticación basada en JWT y verificación de correo mediante código. Las apps más relevantes son:

- **auth**: maneja el registro, login, logout y verificación por correo. El modelo de usuario está personalizado.

EndPoint del registro: Esta vista se encarga de registrar un nuevo usuario. Al recibir los datos del formulario, se valida el contenido y se guarda el usuario en la base de datos con la cuenta desactivada.

- Se genera un **código de verificación de 4 dígitos** que expira en 15 minutos.
- Se envía un **correo electrónico al usuario** con ese código.

EndPoint del LogIn: La vista de inicio de sesión, le permite al usuario loguearse en la app para tener total acceso a esta. Algunas de las funcionalidades son:

- Se extraen el **correo** y la **contraseña** de la petición.
- Se autentica al usuario.
- Si es correcto, se generan los tokens JWT (access y refresh).
- También se devuelve información adicional del perfil del usuario (nombre, correo, foto, descripción, ubicación).

- **productos:** permite realizar operaciones CRUD (Create, Read, Update, Delete) sobre los productos. Incluye diversas funcionalidades avanzadas como:

- **Subida múltiple de imágenes** (entre 1 y 5 por producto).
- **Filtrado por parámetros en la URL** (`query params`), búsqueda textual y ordenación por precio o fecha.
- **Protección de datos mediante tokens:** al listar productos, no se muestran aquellos que pertenecen al usuario autenticado.
- **Creación de productos** mediante peticiones `POST`.
- **Actualización de productos** mediante `PUT`, tanto para modificar su información como para marcarlos como “vendidos”.

En el modelo se ha normalizado el campo categoría, ya que inicialmente se planteó hacer lo mismo con estado y disponibilidad. Sin embargo, finalmente se optó por mantener estos dos últimos como campos con opciones limitadas (choices), dado que sus valores son más estables y poco propensos a cambios. En cambio, categoría sí se normalizó mediante una tabla independiente, ya que es más probable que sus datos cambien con el tiempo (añadir, editar o eliminar categorías), lo que justifica su separación para facilitar la gestión y escalabilidad del sistema.

Endpoint de productos: este endpoint permite realizar operaciones sobre los productos, incluyendo tanto la consulta (`GET`) como la creación o actualización (`POST` y `PUT`). Se han añadido atributos y funcionalidades específicas para permitir el **filtrado** por diversos campos (categoría, estado, disponibilidad, ubicación, etc.), así como la **ordenación** por fecha o precio.

En las respuestas a las peticiones `GET`, se devuelven todos los productos que **no pertenecen al usuario autenticado**, junto con información relevante como el usuario vendedor, imágenes y detalles del producto.

Endpoint para obtener los productos del usuario autenticado:

Dado que el endpoint general de productos excluye los productos del propio usuario (para evitar que vea sus propias publicaciones en la vista pública), se ha implementado un endpoint adicional que permite al usuario autenticado obtener únicamente sus propios productos.

Este endpoint devuelve todos los productos creados por el usuario, permitiéndole gestionarlos (consultar, editar o marcarlos como vendidos) desde su perfil.

- **usuarios:** expone información pública sobre el perfil de un usuario y sus productos publicados. Además, permite la edición del perfil propio mediante una petición PUT.
 - Permite obtener los productos publicados por un usuario específico.
 - Permite al usuario autenticado editar su propio perfil.

Modelo del usuario: el modelo incluye campos esenciales como nombre, correo electrónico, contraseña, ubicación, descripción y foto de perfil. Además, cuenta con campos comunes en todos los modelos para registrar las fechas de creación y actualización.

Entre sus métodos destacan:

- `set_password`: para encriptar la contraseña antes de guardarla.
- `check_password`: para verificar la contraseña al iniciar sesión.
- `is_authenticated`: para comprobar si un usuario está autenticado.

Endpoint de usuario: en realidad, no hay un solo endpoint para usuarios, sino tres tipos de peticiones diferenciadas según la información que se requiera, estas son algunas de sus funcionalidades:

- Obtener información pública de otro usuario, proporcionando su correo electrónico.
- Obtener los productos publicados por un usuario externo (usando su correo). Obtener y editar el perfil del usuario autenticado (propio).

5.2 Diseño del front-end

El frontend está desarrollado con Vue.js 3, Pinia para el estado global y TailwindCSS para el diseño visual. La estructura está dividida en vistas y componentes reutilizables, organizados por módulos funcionales.

Las vistas principales son:

- **AuthView:** contiene un modal de autenticación con pestañas para login y registro.



Iniciar sesión: este tiene un funcionamiento sencillo, tiene dos inputs, email y contraseña, y un botón para visualizar la contraseña. Cuando le das a iniciar sesión el botón tiene la función de enviar el formulario, donde hago un POST contra la api para que me diga si existe o no el usuario y si esas son sus credenciales.

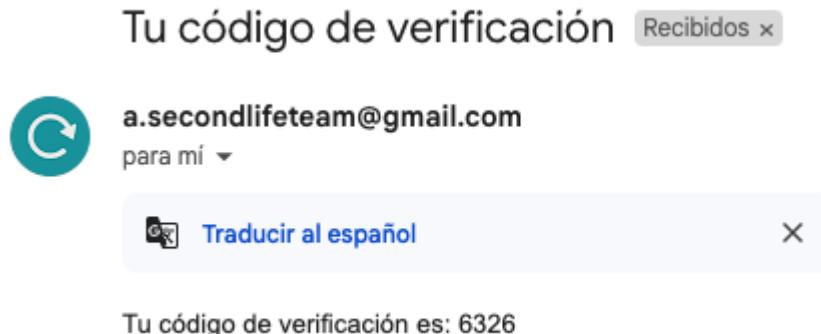
En caso de que no, nos devolverá un mensaje de error

Si todo va bien y conseguimos logearnos se guardará información no delicada en el localStorage del navegador.

access_token	eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ...
description	undefined
lastSearch	
location	Alcalá de Henares, Madrid
mail	antonio.perezcuellar2005@gmail.com
name	Antonio
profile_desc	Descripción de prueba.
profile_pic	http://localhost:8000/media/users/Coqu...
refresh_token	eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ...

Registro: Este tiene alguna funcionalidad más, ya que necesita hacer varias cosas. Tiene la misma funcionalidad de submit en el botón, además de que mientras esperas a que te llegue el correo el botón se deshabilita, para no poder pulsarlo de nuevo.

Una vez hechas las comprobaciones necesarias se procederá a pulsar el botón de continuar, el cual manda un POST a la api con los datos del nuevo usuario, y esta devolverá unos datos y si todo es correcto enviará un correo al mail proporcionado.

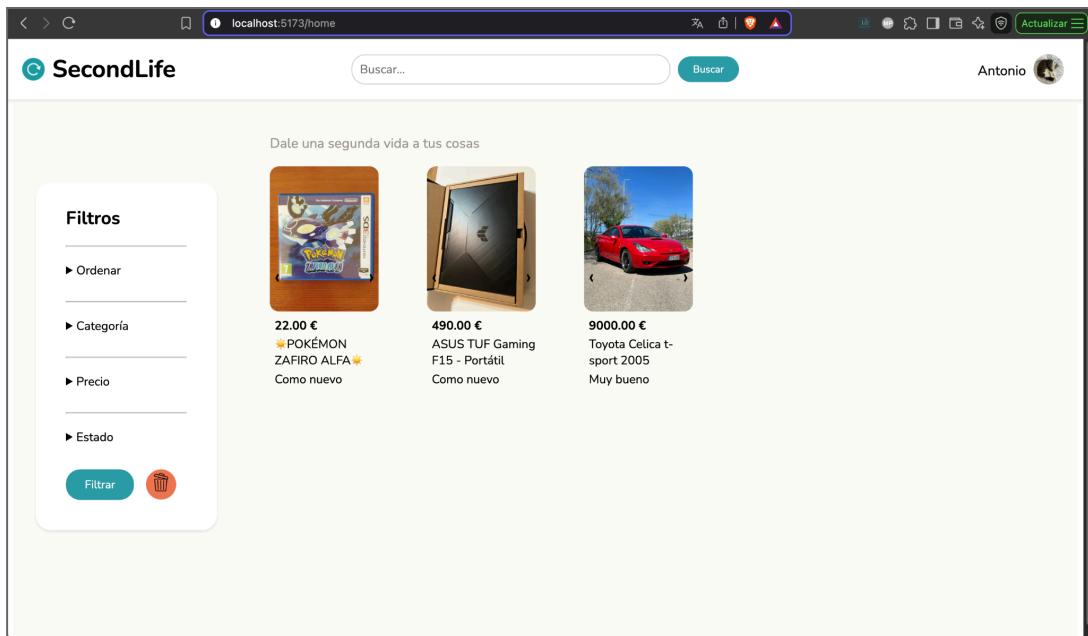


Cuando te llegue deberás meter el código en el nuevo campo desbloqueado, y entonces tu usuario habrá sido verificado.



Una vez puesto el código y registrado con éxito se te redirigirá a la pestaña de iniciar sesión para continuar con el proceso y acceder logueado a la app.

- **HomePage:** vista principal con el listado de productos, filtros por categoría, precio, estado, y una barra de navegación con buscador y perfil de usuario.

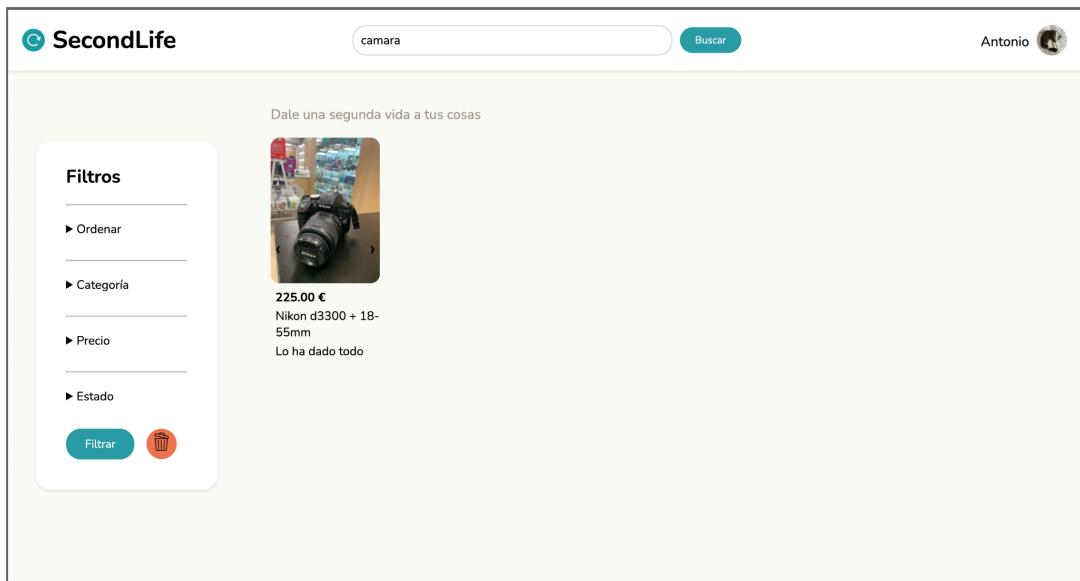


Lo primero es la barra de navegación, este es un componente que se reutiliza en todas las páginas, tiene una funcionalidad básica, cuando scrolleas hacia abajo se esconde, cuando lo haces hacia arriba se muestra.



Esta tiene a la izquierda el nombre de la aplicación, que no es solo decorativo, tiene la utilidad de llevarte a la pantalla de inicio desde cualquier sitio, en el medio una barra de búsqueda, que te permite buscar tanto por nombre como por descripción, por lo que no hace falta un sistema de # para categorizar más específicamente, ya que si quieres en tu anuncio puedes poner palabras clave para las búsquedas.

De esta forma si yo busco la palabra cámara me aparecerá uno aunque su nombre no lo refleje.

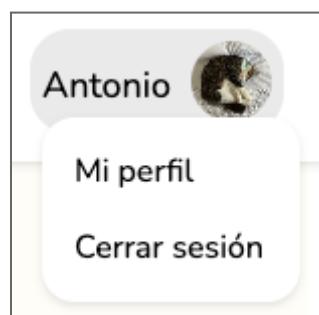


Situado a la derecha se ve el nombre de usuario junto su foto de perfil, en caso de que no tenga foto tendrá una por defecto.

Pero también, recordar que puedes entrar a la aplicación sin logear, por lo que no aparecerá esa información, si no que aparecerá un botón que te redirigirá a la pestaña de autenticación.



Sin embargo, ese no es su único propósito, si tu pulsas tu usuario se desplegarán dos opciones, una para ir a tu perfil y otra para cerrar la sesión.

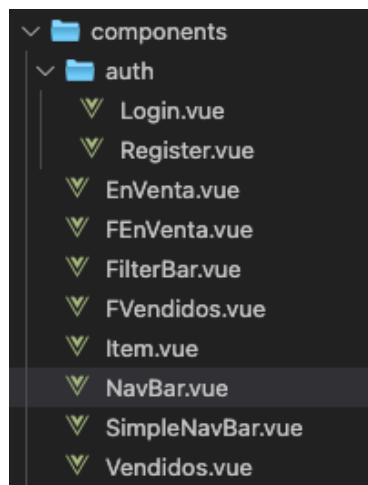


Primero una explicación rápida sobre cómo se cierra la sesión, esta llama al endpoint de logout y posteriormente borra los datos guardados en el localstorage sobre el usuario, como los token, nombre, foto de perfil...

Al borrar el token, ya no habrá restricción de ver sus propios productos, ya que no tendrá la sesión iniciada.

En caso de que le demos a “Mi perfil”, nos llevará a otra vista nueva que explico en el apartado **Profile**.

Aquí se pueden ver los componentes que uso, entre ellos NavBar, del que estamos hablando ahora, al ser un componente lo podemos usar en cualquier template simplemente importandolo.



La funcionalidad del filtro de búsqueda es sencilla, cuando haces enter o le das al botón de buscar, llamará a un método.

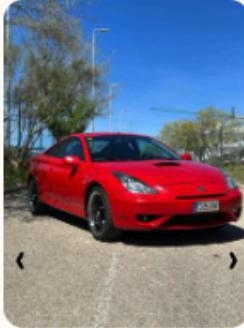
Este método guarda la búsqueda que acabas de hacer en el local storage, para luego aplicarla desde ahí, y esto tiene una razón, cuando tú estás en otra ventana y usas el buscador, te redirige al inicio con el filtro de búsqueda aplicado.

Explicada la barra de navegación continuamos con los **ítems**, los **productos**.

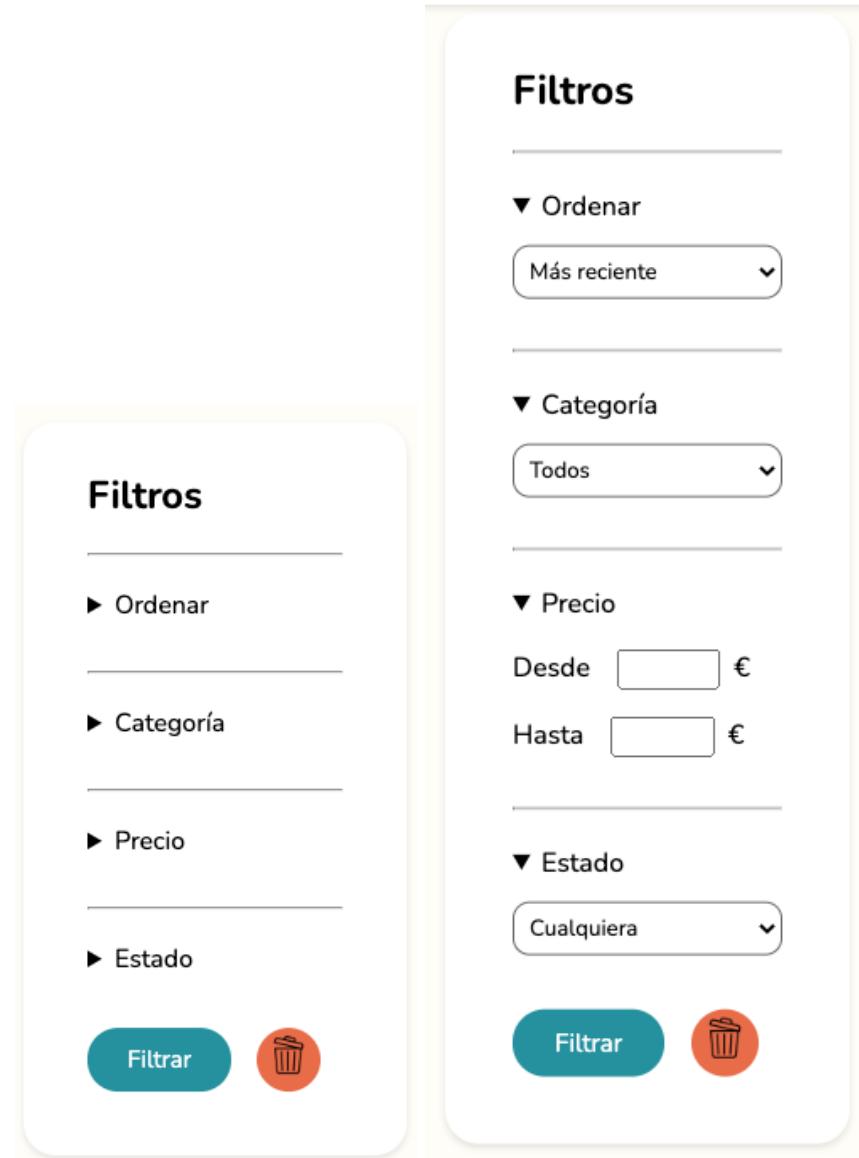
Estos también son componentes al igual que la barra de navegación, el cual se diseña una vez y reutiliza varias para mostrar todos los productos.

Estos ítems contienen una foto principal del producto, con dos botones para pasar al resto de fotos si es que tiene, seguido de un precio, un nombre y el estado del producto.

Dale una segunda vida a tus cosas

		
22.00 € ☀️POKÉMON ZAFIRO ALFA☀️ Como nuevo	490.00 € ASUS TUF Gaming F15 - Portátil Como nuevo	9000.00 € Toyota Celica t- sport 2005 Muy bueno

Por último en la HomePage, los **filtros**, que por defecto, están sin desplegar, para no ocupar demasiado espacio en la pantalla.

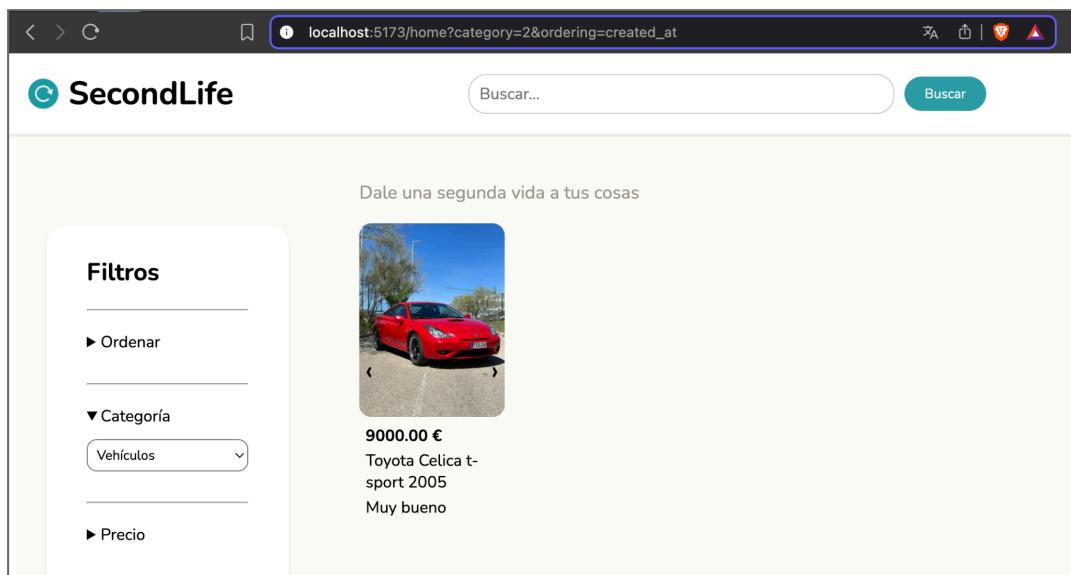


Así se ven todos los filtros desplegados.

En la parte baja se pueden ver dos botones, el de filtrar y otro para borrar los filtros que hay, este segundo botón, borrará los filtros, pero no ejecutará la búsqueda vacía, habría que darle a filtrar para que lo hiciera.

Los filtros funcionan de la siguiente manera, estos al ejecutarse se guardan en las queryparams, en la url, y desde ahí se recogen los datos a filtrar, y con esos datos se hace la llamada a la api de los productos.

La razón por la que se aplican los filtros desde los query params es porque si tu recargas la página, los filtros persisten, y lo mismo si quieras pasarle el enlace de los filtros que has aplicado a otra persona.



- **ProductDetail:** muestra la información detallada de un producto, pudiendo ver información como el vendedor, el precio, el título y la descripción del producto, además de tener también la posibilidad de pasar las fotos con botones en los laterales.

Si lo deseamos podemos meternos en el perfil del vendedor, pero eso se explicará más a fondo en su respectiva vista.

Ivan

22.00 €

☀️POKÉMON ZAFIRO ALFA☀️

- ◆ Pokémon Zafiro Alfa 🎮 para 3Ds y 2Ds.
- ◆ **✓100% ORIGINAL✓**
- ◆ En buenas condiciones 🌟
- ◆ Recién probado y genial 👍
- ◆ Idioma: español. (Pal ESP 🇪🇸)

📦 Hago envíos 📦

⭐⭐ Vendedor con +500 valoraciones ⭐⭐

✗ no leer ✗ Pokemon Diamante Perla Blanca 2 Zafiro Rojo Azul Negra Oro Heartgold Zelda Kirby PS3
PS4 switch Plata Soulsilver Mario Rubí Wii Metroid Mario Misterioso Layton Ranger Dragon Quest Final Fantasy Castlevania Platino Inazuma Fire Emblem PSP Gamecube

Comprar

- **Checkout:** permite al usuario comprar un producto.

Comprar artículo

Pedido



★ POKÉMON ZAFIRO ALFA ★
Como nuevo 22.00 €

Desglose del pedido

Precio del producto:	22.00 €
<input checked="" type="radio"/> En persona <input type="radio"/> Dirección establecida	
Precio de envío:	0 €
Total a pagar:	22.00 €

Dirección

Nombre	Nombre del comprador
Calle y código postal	Dirección a la que llegará el producto
Nº puerta / piso	puerta y piso

En esta ventana podremos ver un desglose de la compra, empezando por el producto a comprar y el precio, justo debajo la dirección de entrega, si es que quieres que se te entregue a tu dirección, ya que en la caja de la derecha podrás elegir si quieres hacer la compra en persona o a la dirección establecida, aunque esa opción sola la podrás elegir si has puesto una dirección, además de que costará por defecto 5 euros el envío, y ya al final te pone el total a pagar.

Una vez le des a pagar empezará el proceso de poner el producto como “vendido” y mandar los correos tanto al comprador como al vendedor, y mientras estará el botón bloqueado para no poder darle de nuevo mientras se hace la “transacción”.

Comprar artículo

Pedido



★ POKÉMON ZAFIRO ALFA ★
Como nuevo 22.00 €

Desglose del pedido

Precio del producto:	22.00 €
<input checked="" type="radio"/> En persona <input type="radio"/> Dirección establecida	
Precio de envío:	0 €
Total a pagar:	22.00 €

Dirección

Nombre	Nombre del comprador
Calle y código postal	Dirección a la que llegará el producto
Nº puerta / piso	puerta y piso

Terminada la transacción llegará el correo con los datos.

Confirmación de compra Recibidos x

 a.secondlifeteam@gmail.com
para mí ▾

Hola Antonio,

Has comprado el producto: ☀️POKÉMON ZAFIRO ALFA☀️

Precio: 22.00€

Dirección de recogida/envío: No especificada

Por favor, ponte en contacto con el vendedor para más detalles.

Gracias por usar SecondLife!

← Responder

→ Reenviar



a.secondlifeteam@gmail.com
para mí ▾

Hola Ivan,

Has vendido el producto: ☀️POKÉMON ZAFIRO ALFA☀️

Precio: 22.00€

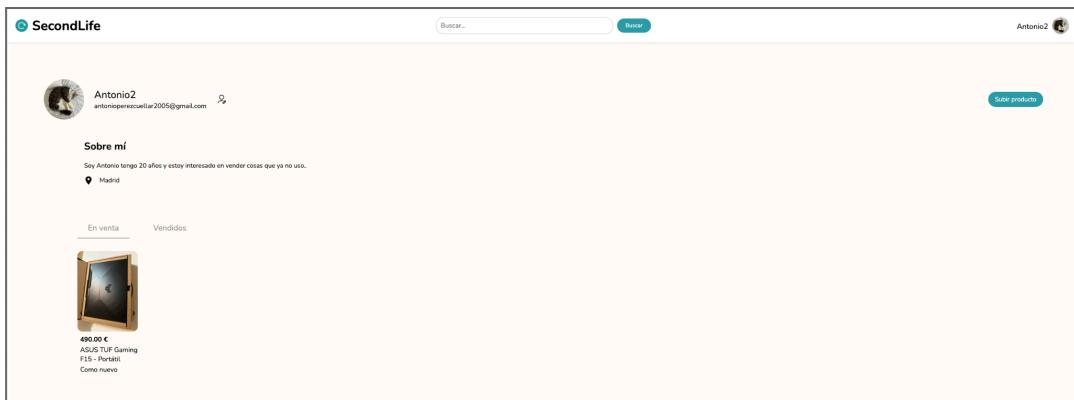
Comprador: Antonio (antonioperezcuellar2005@gmail.com)

Te recomendamos ponerte en contacto para acordar la entrega.

Gracias por usar SecondLife!

Como se ve te dice, el nombre, precio y dirección del usuario, y como el usuario no tiene una localización, aparece como no especificada, por lo que se recomienda el contacto con el vendedor, por eso el correo es público.

- **Profile:** permite al usuario gestionar sus productos, editar publicaciones y editar su perfil.



Como se puede ver, aparece nuestro nombre, correo y foto de perfil junto a un ícono para editar nuestro perfil.

Justo debajo aparece una descripción sobre tu perfil, ya sea sobre lo que vendes o sobre quién eres

Si pulsamos el ícono de editar usuario nos llevará a la siguiente ventana.

This screenshot shows the 'Editar perfil' (Edit profile) form. It includes a circular profile picture placeholder with an 'X' icon, a 'Cambiar foto' (Change photo) button, and a 'Nombre' (Name) field containing 'Antonio2'. Below that is a 'Localización' (Location) field with 'Alcalá de Henares, Madrid'. Underneath is a 'Descripción' (Description) field containing the bio: 'Soy Antonio tengo 20 años y estoy interesado en vender cosas que ya no uso.' At the bottom is a large teal 'Guardar cambios' (Save changes) button.

Aquí se cargan nuestros datos desde el storage y se ponen en los valores de los inputs.

Puedes cambiar cualquiera de los campos, y puedes dejarlos vacíos todos menos el del nombre.

Una vez le demos a guardar cambios podremos ver ya cambios como el nombre o la foto de perfil en la barra de navegación o volver a nuestra pestaña de perfil y ver todos los cambios.

A la derecha se ubica el botón de **subir producto**, que nos lleva también a una página nueva donde podremos subir nuestro producto.

Subir producto

Título del anuncio Pokémo Blanco Ds (inglés)

Descripción del producto Juego Pokémon Blanco DS (versión inglesa). Solo cartucho, sin carátula.
Funciona perfectamente.

Sube hasta 5 fotos, la primera imagen será la que salga como principal



Categoría Videojuegos

Estado Bueno

Precio 40

Subir

Todos los campos son obligatorios a excepción de la descripción y las últimas 4 fotos, ya que una es el mínimo.

Una vez le des a Subir pasará lo mismo que cuando compras, se quedará bloqueado el botón hasta que se suba y cuando se haya subido te redirigirá a tu perfil



Antonio2
antonio.perezcuellar2005@gmail.com

Sobre mí

Soy Antonio tengo 20 años y estoy interesado en vender cosas que ya no uso..

📍 Madrid

[Ver perfil](#)

[En venta](#) [Vendidos](#)

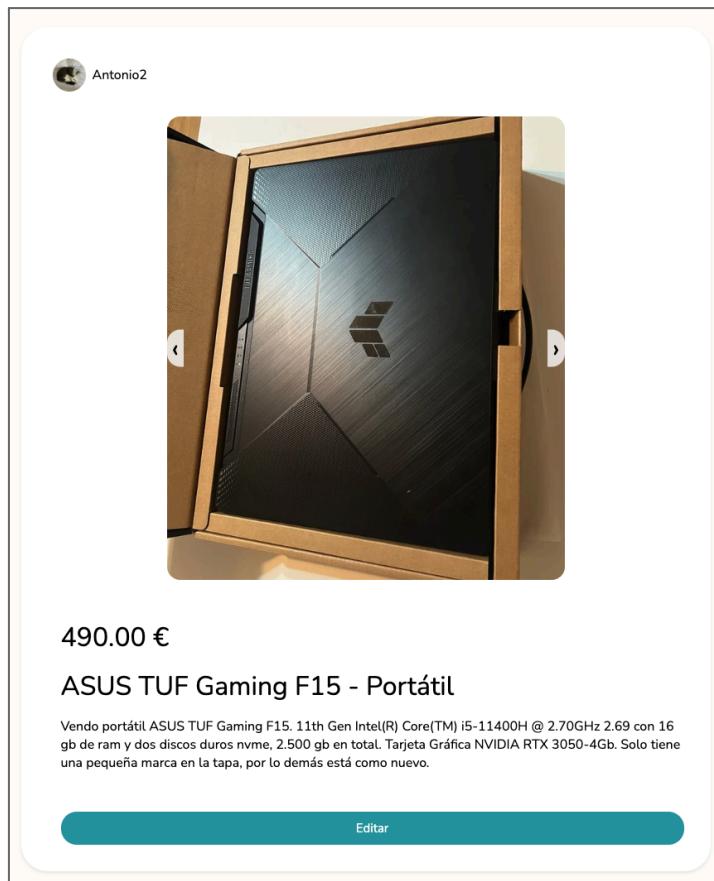


490.00 €
ASUS TUF Gaming
F15 - Portátil
Como nuevo



40.00 €
Pokémon Blanco Ds
(inglés)
Bueno

Otra funcionalidad que tiene nuestro perfil es ver los detalles de nuestros propios productos, lo que puede ser funcional para ver como queda de cara a los compradores de tu producto.



Si le damos al botón de editar nos llevará a otra página en la que podremos ver nuestro producto y los campos que podremos editar sobre él. Esta página es igual a la de subir producto, pero cargando los datos del producto que estamos viendo y en vez de haciendo un POST haciendo un PUT, que es para editar.

Editar producto

Título del anuncio

Descripción del producto

Sube hasta 5 fotos, la primera imagen será la que salga como principal



Categoría

Estado

Precio

Subir

Las restricciones son las mismas, todos los campos obligatorios menos la descripción, las fotos también las podremos cambiar o borrar.

Una vez le des a subir, te llevará a tu página de usuario.

6. Resultados

6.1 Objetivos alcanzados

Se han alcanzado con éxito los objetivos principales establecidos al inicio del proyecto. A partir de esa base, se valoró la incorporación de funcionalidades adicionales, así como la eliminación de algunas ideas en función del tiempo disponible y del orden de prioridades definido durante el desarrollo.

Este enfoque flexible ha sido clave para centrar los esfuerzos en las partes más relevantes de la aplicación. Por ejemplo, se decidió posponer el desarrollo del sistema de valoraciones de productos, ya que se consideró menos prioritario en comparación con funcionalidades esenciales como el correcto funcionamiento de los perfiles de usuario, la posibilidad de editar la información personal, visualizar los productos propios y acceder a los de otros usuarios.

Además, durante el proceso se ha conseguido el importante objetivo de aprender y aplicar nuevos lenguajes de programación y frameworks, como Python con Django para el backend y JavaScript con Vue para el frontend, ampliando así el conjunto de herramientas técnicas manejadas y mejorando significativamente las competencias profesionales.

Esta toma de decisiones basada en prioridades y el aprendizaje continuo han permitido entregar una aplicación funcional, coherente y centrada en la experiencia del usuario, cumpliendo con los objetivos fundamentales del proyecto.

6.2 Dificultades encontradas

Durante el desarrollo del proyecto se han presentado diversas dificultades, muchas de ellas relacionadas con la novedad de las tecnologías utilizadas y el contexto en el que se ha realizado el trabajo.

En primer lugar, uno de los mayores retos ha sido el uso de lenguajes y frameworks que hasta ahora no se habían trabajado en profundidad, como Python con Django para el backend y JavaScript con Vue para el frontend. Ambos han supuesto un cambio significativo con respecto a los entornos de desarrollo habituales en el ciclo de Desarrollo de Aplicaciones Multiplataforma (DAM), en el que predominan tecnologías como Java o entornos de escritorio.

Además, el desarrollo de una aplicación web completa, con separación clara entre frontend y backend, ha implicado enfrentarse a nuevas estructuras de proyecto y arquitecturas desconocidas hasta el momento. Esto ha supuesto un esfuerzo adicional para comprender cómo se comunican las distintas partes de la aplicación, cómo se manejan los datos entre cliente y servidor, y cómo organizar correctamente el código para mantenerlo limpio, modular y escalable.

Otro factor importante ha sido el número de tecnologías nuevas que ha sido necesario aprender y utilizar de forma simultánea: consumo y creación de APIs REST, autenticación basada en tokens JWT, gestión de imágenes, uso de bases de datos relacionales con MySQL desde Django, despliegue con Docker, entre otras.

El uso de tokens de acceso ha supuesto un reto específico, ya que era un concepto completamente nuevo y ha requerido entender cómo funcionan, cómo se generan, almacenan y renuevan, y cómo deben enviarse en las peticiones para mantener la seguridad de la aplicación.

Por último, la gestión del tiempo ha sido uno de los retos más importantes, ya que el desarrollo del proyecto se ha realizado en paralelo a las FCTs (Formación en Centros de Trabajo). Esto ha requerido una buena organización personal y un esfuerzo constante para poder avanzar de forma equilibrada en ambas tareas.

A pesar de todas estas dificultades, superarlas ha supuesto un gran aprendizaje y una mejora notable de las competencias técnicas y organizativas.

6.3 Futuras mejoras

Durante el desarrollo del proyecto, algunas funcionalidades no se han podido implementar por limitaciones de tiempo, pero quedan como opciones interesantes para ampliaciones futuras:

- **Chat entre usuarios:** La idea de incorporar un sistema de mensajería similar al de plataformas como Wallapop o Vinted fue una de las primeras contempladas. Sin embargo, a medida que avanzaba el proyecto y el tiempo disponible disminuía, se decidió posponer esta funcionalidad para garantizar la calidad del núcleo principal de la aplicación y ofrecer la alternativa de hacer públicos los correos electrónicos para su comunicación.
- **Valoraciones de productos:** Poder valorar productos es un mecanismo útil para ofrecer feedback y mejorar la confianza entre usuarios. Aunque no se pudo desarrollar durante esta fase, se implementó una alternativa que permite visualizar los productos vendidos por cada usuario, aportando información relevante sobre su actividad.
- **Favoritos:** La funcionalidad para guardar productos como favoritos es relativamente sencilla de implementar. No obstante, priorizar la estabilidad y buen funcionamiento de las características ya desarrolladas fue esencial, por lo que esta mejora queda pendiente para futuras versiones.
- **Ofertas:** Esta fue una funcionalidad que pensé en implementar al principio, pero al darme cuenta de la carga de trabajo que suponían otras cosas más prioritarias, decidí dejarla como una mejora para el futuro, ya que es una característica muy interesante actualmente.
- **Servidor:** Otra de las mejoras pensadas sería la incorporación de un servidor remoto que permita acceder a la aplicación desde cualquier lugar, mejorando así su disponibilidad y realismo. Además, se consideró la posibilidad de utilizar un servidor independiente para alojar las imágenes, de forma que el backend principal solo almacene las URLs que apuntan a dicho servidor. Esta separación de responsabilidades permite una mayor escalabilidad, una gestión más eficiente del almacenamiento y una mejor preparación para un entorno de producción.

7. Conclusiones

En conclusión, considero que, dadas las circunstancias y el tiempo disponible, he realizado un trabajo del que puedo estar orgulloso. A lo largo del desarrollo del proyecto, he enfrentado diversas dificultades que me han supuesto un reto, pero también una valiosa oportunidad de aprendizaje. He adquirido conocimientos prácticos sobre nuevas tecnologías, lenguajes de programación y metodologías de desarrollo que hasta entonces no había explorado en profundidad.

Además, la experiencia me ha permitido comprender la importancia de una buena organización y gestión del tiempo, especialmente cuando se compagina un proyecto personal con otras responsabilidades como las prácticas profesionales. Esta capacidad de planificación ha sido clave para priorizar funcionalidades y tomar decisiones acertadas sobre el alcance del trabajo.

Sin duda, este proyecto ha supuesto un salto importante en mi formación como desarrollador, brindándome una visión realista de los procesos y desafíos que implica crear una aplicación web completa desde cero. Me llevo no solo un producto funcional, sino también una experiencia que me ayudará a afrontar futuros proyectos con mayor confianza y eficacia.

Por todo ello, este trabajo representa un paso fundamental en mi camino profesional y un motivo de motivación para seguir aprendiendo y mejorando.

8. Bibliografía

8.1 Documentación oficial

Django Software Foundation. *Django documentation.*

<https://docs.djangoproject.com/en/stable/>

Encode. *Django REST Framework documentation.*

<https://www.django-rest-framework.org/>

SimpleJWT. *Simple JWT documentation.*

<https://django-rest-framework-simplejwt.readthedocs.io/en/latest/>

Vue.js. *Vue 3 documentation.* <https://vuejs.org/guide/introduction.html>

Pinia. *Pinia documentation (state management for Vue).* <https://pinia.vuejs.org/>

Docker Inc. *Docker documentation.* <https://docs.docker.com/>

GitHub. *Git documentation.* <https://git-scm.com/doc>

MySQL. *MySQL documentation.* <https://dev.mysql.com/doc/>

Tailwind Labs. *Tailwind CSS documentation.* <https://tailwindcss.com/docs>

Djoser. *Djoser documentation (authentication system for Django REST).*

<https://djoser.readthedocs.io/en/latest/>

8.2 Agradecimientos

También me gustaría agradecer a los compañeros de mi empresa de prácticas, quienes en determinados momentos me ofrecieron su ayuda y consejos técnicos durante el desarrollo del proyecto, especialmente en temas relacionados con la arquitectura del backend y el uso de Docker. Su apoyo ha sido de gran valor para superar algunos bloqueos y seguir aprendiendo en un entorno real.

También me gustaría mencionar a mis compañeros de clase, quienes han sido una fuente constante de ideas, recomendaciones y otros puntos de vista que me han hecho replantearme decisiones y enriquecer el desarrollo del proyecto desde una perspectiva más colaborativa.

Por último, agradecer a los profesores que me han ayudado en este proyecto dando sus recomendaciones desde un punto de vista más experimentado.