Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención Ingeniería del Software

SecondHandChain: Aplicación web descentralizada orientada a la venta de artículos de segunda mano

---

Autor:
D. Darío Gago Pérez

Tutor Académico:
D. Manuel Barrio Solorzano

Tutores de empresa:
D. Daniel López Palomo
Dña. Julia Zuara Jiménez

Agradecimientos

En primer lugar, agradecer a mi familia y amigos, quienes me han apoyado durante todo el transcurso de la carrera, tanto en los buenos como en los malos momentos.

De la misma forma, a mis compañeros de carrera, que, a lo largo de estos años, han sido mi día a día y apoyo para afrontar las dificultades.

Por último, pero no menos importante, a mis tutores de empresa, quienes me han supervisado y aconsejado durante todo este periodo, y a mi tutor académico por la ayuda y revisiones proporcionadas.

Resumen

En este proyecto se presenta el desarrollo de una aplicación descentralizada cuya finalidad es aportar confianza y transparencia en la compra y venta de artículos de segunda mano.

La motivación principal que lleva al desarrollo del presente Trabajo de Fin de Grado es permitir conocer la "historia" detrás de un artículo, desde su origen hasta los cambios de titular y modificaciones que ha ido sufriendo a lo largo de su puesta en venta. Por tal razón, se hace uso de las tecnologías emergentes de la blockchain, concretamente desarrollando el proyecto sobre la red local de Ganache, con el objetivo de mantener su historial inmutable y accesible.

En este documento se presenta la planificación del mismo, la cual está basada en la metodología ágil Scrum, adaptándola a las necesidades del autor y sus tutores.

Posteriormente, se presenta el proceso seguido para el desarrollo de la aplicación, la implementación de contratos inteligentes, utilizando el lenguaje Solidity, y el desarrollo del front-end, haciendo uso del lenguaje JavaScript, junto con la librería React.

Abstract

This project presents the development of a decentralised application whose purpose is to provide confidence and transparency in the purchase and sale of second-hand items.

The main motivation behind the development of this Final Degree Project is to make it possible to know the "history" behind an item, from its origin to the changes in ownership and modifications it has undergone throughout its sale. For this reason, use is made of emerging blockchain technologies, specifically by developing on the local Ganache network, with the aim of keeping its history immutable and accessible.

This paper presents the planning of the project, which is based on the agile Scrum methodology, adapting it to the needs of the author and his tutors.

Subsequently, the process followed for the development of the application, the implementation of smart contracts, using the Solidity language, and the development of the front-end, making use of the JavaScript language, together with the React library, are presented.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Introducción

1.1.1. Contexto

La compraventa de artículos de segunda mano es una práctica cada vez más común en la sociedad actual debido a sus numerosos beneficios, como el ahorro económico o la reducción del impacto ambiental. Sin embargo, los compradores se enfrentan a un desafío significativo: la falta de información fiable sobre el pasado del artículo que desean adquirir. Esta incertidumbre puede generar desconfianza y dificultades en la toma de decisiones informadas.

Esta falta de información sobre el pasado del artículo también puede plantear preguntas sobre su autenticidad. Por ejemplo, en el mercado de artículos de lujo, es común encontrar productos falsificados que se venden como auténticos. Los compradores interesados en adquirir estos artículos deben ser especialmente cautelosos y buscar formas de verificar su autenticidad antes de realizar la compra.

Para afrontar estos desafíos, han surgido varias soluciones en el mercado. Una de ellas es la aparición de plataformas en línea especializadas en la compraventa de artículos de segunda mano, donde los vendedores pueden proporcionar información detallada sobre el estado y la historia del artículo. Estas plataformas pueden incluir características como la posibilidad de adjuntar fotografías del artículo, descripciones detalladas, informes de inspección o certificados de autenticidad.

Otra opción es recurrir a servicios de verificación de terceros. Estas empresas se encargan de realizar inspecciones exhaustivas de los artículos de segunda mano y proporcionar informes detallados sobre su estado y autenticidad. Los consumidores pueden solicitar estos informes antes de realizar una compra, lo que les brinda una mayor tranquilidad y confianza en la transacción.

Todas estas soluciones poseen un denominador común, existe cierta dependencia de la confianza en el operador centralizado de la plataforma o servicio. Los compradores deben confiar en que la información proporcionada por el vendedor o el informe de verificación de terceros sea precisa y confiable. Esto puede generar preocupaciones sobre la manipulación de datos o la posibilidad de información falsa. Así mismo, la centralización de los datos puede ser problemática en términos de censura o vulnerabilidad a ataques o fallas en el sistema.

Para abordar estos problemas, se ha llevado a cabo el presente proyecto como Trabajo de Fin de Grado de Ingeniería Informática, mención en Ingeniería del Software. El objetivo principal es desarrollar una solución que proporcione a los compradores información confiable y completa sobre los artículos de segunda mano. En este documento se recogen todas las fases, procedimientos y herramientas utilizadas en su desarrollo.

1.1.2. Motivación

La idea principal de este proyecto, y la aplicación que resulta del mismo, nace de la necesidad de aportar solución al desconocimiento del pasado de un artículo utilizado en venta.

En este trabajo, se propone el desarrollo de una plataforma que solucione este problema, aportando confianza y valor a las ventas de segunda mano. El desarrollo de una plataforma que permita conocer el historial de un artículo de segunda mano se relaciona directamente con los principios de la blockchain (Ver 2.1.2.), para garantizar la transparencia y la auditabilidad de la información relacionada con los artículos usados. Esta tecnología permite crear registros inmutables y seguros, lo que hace prácticamente imposible la modificación eliminación fraudulenta de la información registrada sobre un artículo.

De este modo, se puede crear un sistema confiable en el que la identificación e historial de cada producto sean únicos y puedan ser verificados mediante la cadena de bloques. Los vendedores podrán registrar los artículos que desean vender, proporcionando detalles como su estado actual, reparaciones previas, historial de uso y cualquier otra información relevante. Además, podrán adjuntar documentos o imágenes para respaldar la descripción del artículo.

Por otro lado, los consumidores podrán ver la información completa y transparente sobre un artículo en particular, lo que les permitirá tomar decisiones informadas antes de realizar una compra. Por ejemplo, podrán verificar si el artículo deseado ha sufrido daños importantes, si ha sido reparado anteriormente o si ha tenido múltiples propietarios.

Además de proporcionar información detallada, la plataforma también permitirá a los usuarios realizar transacciones de manera segura. Al utilizar la cadena de bloques, se pueden registrar y verificar las transacciones de compra y venta, lo que brinda un nivel adicional de confianza y seguridad para ambas partes involucradas.

1.1.3. Objetivos

El objetivo de este proyecto es la realización de una aplicación descentralizada que facilite a los usuarios una plataforma donde poder adquirir y consultar toda la información dada de un artículo desde su dada de alta en la misma. De esta forma, los objetivos principales del proyecto son los siguientes:

- Realizar el análisis de las tecnologías existentes con el fin de seleccionar aquellas que beneficien al desarrollo de la aplicación.
- Desarrollar la funcionalidad que permita a los usuarios consultar todos los eventos relacionados con un artículo dentro de la plataforma, desde su registro inicial hasta las transacciones posteriores y cambios de estado.
- Permitir al usuario realizar sus operaciones mediante el uso de una cartera Metamask.
- Desplegar la aplicación dentro de una red de prueba Ethereum.
- Diseñar una interfaz de usuario minimalista, intuitiva y comprensible, centrándose en la experiencia del usuario y facilitando el acceso a la información relevante sobre los artículos en venta.

1.1.4. Objetivos personales

Los objetivos personales del autor, en relación con la realización de este proyecto, residen en:

- Comprender la creación de una aplicación descentralizada desde su inicio.
- Explorar el uso de tecnologías blockchain y contratos inteligentes en casos de uso reales.
- Adquirir experiencia en la implementación y despliegue de la aplicación en una red de prueba Ethereum, para comprender el proceso y las consideraciones necesarias para su funcionamiento.
- Aprender a diseñar contratos inteligentes utilizando el lenguaje de programación Solidity.
- Aplicar metodologías ágiles en la planificación y desarrollo de un proyecto extenso.

- Aprender a utilizar nuevos frameworks útiles e interesantes para el futuro, como Angular. Al igual que mejorar las habilidades de desarrollo de aplicaciones web.

1.1.5. Usuarios objetivos

El proyecto propuesto puede beneficiar a un amplio abanico de usuarios con diferentes objetivos y características:

En primera instancia, la aplicación está destinada a ser utilizada por particulares interesados en la compra de todo tipo de bienes de segunda mano. Estos usuarios pueden beneficiarse de toda información adicional que proporciona la plataforma, antes de realizar una compra. De esta manera, puede facilitar la toma de una decisión sobre la compra basándose en el historial y el estado actual del artículo, al igual que proporciona una sensación de confianza y fiabilidad, crucial al tratarse de ventas de artículos usados.

De la misma forma, puede favorecer a vendedores dispuestos a vender sus artículos. Como ya se ha dicho, tener un historial limpio del artículo facilita venderlo mejor y a un precio más ajustado al estado del producto. Así también podemos destacar a las personas coleccionistas, quienes pueden disponer de mayor información sobre artículos de valor como sus propietarios anteriores antes de ampliar su colección.

Adicionalmente, esta plataforma puede beneficiar a organizaciones y empresas que comercien con artículos de ocasión, facilitando a los potenciales compradores el historial de los bienes, haciendo que el proceso de venta sea más fluido y transparente. Así, las empresas podrían hacer un seguimiento de su inventario de bienes de segunda mano, lo que facilitaría su gestión y venta.

1.1.6. Estructuración de la memoria

El presente documento está estructurado en los siguientes capítulos:

- **Capítulo 2 Blockchain**, en este capítulo se recoge la investigación previa sobre las tecnologías blockchain y las herramientas utilizadas con relación a esta tecnología.
- **Capítulo 3 Planificación del proyecto**, en este capítulo se explica el procedimiento de planificación seguido, así como el análisis de riesgos y las desviaciones sobre la planificación inicial.
- **Capítulo 4 Análisis**, en este capítulo se presenta el análisis de requisitos, los casos de uso y los modelos iniciales.
- **Capítulo 5 Diseño**, en este capítulo se presenta el diseño de la aplicación y el prototipo de interfaz inicial.
- **Capítulo 6 Implementación**, en este capítulo se explican las tecnologías utilizadas y la implementación de la aplicación.
- **Capítulo 7 Pruebas**, en este capítulo se recogen las pruebas realizadas para la comprobación del correcto funcionamiento de la aplicación.
- **Capítulo 8 Conclusiones y trabajo futuro**, en este capítulo se explican las conclusiones finales sobre el proyecto, así como las líneas de trabajo futuro que podría seguir.

Al final del documento se proporciona la bibliografía utilizada, así como una serie de anexos, donde se encuentran la Guía de instalación (Ver A) y el Manual de usuario (Ver B).

Capítulo 2

Blockchain

2.1. Fundamentos de la blockchain

En este segundo capítulo abordaremos el marco teórico sobre la tecnología detrás de conseguir las ventajas atribuidas a este proyecto. Introduciremos el concepto de "blockchain", su funcionamiento, principios y arquitectura, y se profundizará en detalle en las tecnologías utilizadas y sus principales ventajas.

2.1.1. Introducción a la blockchain

Blockchain, también conocida como cadena de bloques, es una tecnología innovadora que ha revolucionado la forma en que se almacena y se transfiere la información. Aunque se popularizó inicialmente gracias a su aplicación en las criptomonedas, como el Bitcoin, su alcance se ha extendido a diversos sectores, transformando la manera en que se realizan transacciones y se garantiza la seguridad de los datos.

En esencia, es un registro digital distribuido, descentralizado e inmutable que permite a múltiples participantes mantener una copia actualizada y sincronizada de la información en tiempo real. A diferencia de los sistemas tradicionales, en los que la información se almacena en un único servidor o base de datos centralizada, la cadena de bloques se basa en una red de nodos interconectados que trabajan en conjunto para validar y registrar cada transacción o evento (Zhejiang University Press, 2020).

El funcionamiento de esta tecnología se fundamenta en la creación de bloques, unidades de información que contienen datos y registros de transacciones. Estos se enlazan entre sí de forma cronológica, formando una cadena continua de bloques, de ahí su nombre. Cada uno contiene un identificador único llamado hash, que lo vincula al anterior, creando así una integridad y una secuencia inalterable.

Una de sus características más destacadas es su naturaleza descentralizada. En lugar de depender de una entidad central, como un banco o una autoridad gubernamental, el consenso en la red se logra mediante algoritmos matemáticos y la participación de múltiples nodos. Esto asegura la transparencia y la seguridad de las transacciones, ya que cualquier modificación o intento de fraude requeriría el consenso de la mayoría de los nodos, lo cual resulta extremadamente difícil debido a la distribución de la red.

Además, la Blockchain ofrece gran resistencia a la censura y la manipulación de datos. Debido a que cada participante de la red tiene una copia actualizada de la cadena completa, resulta altamente improbable que un ataque malicioso o un fallo en un nodo afecte la integridad general de la red. Esto lo convierte en una herramienta confiable para garantizar la seguridad y la autenticidad de las transacciones, así como para agilizar procesos, eliminar intermediarios y reducir costos en diversos sectores.

Se introdujo por primera vez en 2008, como la tecnología detrás de la moneda digital Bitcoin. Sin embargo, en la actualidad, sus aplicaciones potenciales se han ampliado mucho más allá de este tipo de monedas.

2.1.2. Principios de la blockchain

De acuerdo con IBM (2023), para permitir realizar transacciones seguras, trasparentes e inmutables, esta tecnología se basa en los siguientes tres principios:

La **criptografía** es la técnica utilizada para proteger los datos almacenados en la cadena de bloques. Cada transacción se cifra con un hash criptográfico, un identificador único que hace prácticamente imposible alterar la transacción una vez se registra.

La **descentralización** es otro principio fundamental. En lugar de tener una autoridad central, la cadena de bloques es una red distribuida de nodos que trabajan juntos para validar las transacciones. Cada nodo de la red tiene una copia de la cadena de bloques, lo que la hace muy resistente a los ataques y garantiza que no haya un único punto de fallo.

El **consenso** es el tercer principio de blockchain. Para que una transacción se añada a la cadena de bloques, todos los nodos de la red deben estar de acuerdo en que es válida. Esto se consigue mediante un mecanismo de consenso, que puede variar dependiendo de la implementación de la cadena de bloques. Por ejemplo, Bitcoin utiliza un mecanismo de consenso de prueba de trabajo, mientras que Ethereum (Ver 2.2.1) utiliza un mecanismo de consenso de prueba de participación.

La combinación de estos principios hace que esta tecnología sea altamente segura y fiable para almacenamiento y transferencia de datos. También elimina la necesidad de intermediarios, como bancos u otros terceros, lo que puede reducir costes y aumentar la eficiencia.

2.1.3. Arquitectura de un sistema Blockchain

De acuerdo con Valbuena, Sanabria y Góngora (2021), un sistema Blockchain se basa en una arquitectura multicapa, estas capas trabajan en conjunto para brindar una plataforma segura y descentralizada para el intercambio de información y valor. A continuación, se describirán las diferentes capas y su importancia en el ecosistema:

La primera es la **capa de datos** como pilar fundamental de un sistema blockchain. En ella, se almacenan todos los registros de transacciones en bloques enlazados mediante criptografía. Como hemos visto, cada bloque contiene un conjunto de transacciones verificadas y una referencia al bloque anterior, lo que garantiza la integridad y la inmutabilidad de los datos. Gracias a esta capa, los participantes de la red pueden acceder a un historial completo y transparente de transacciones.

La **capa de red** es responsable de la conectividad en la red. Utiliza protocolos de comunicación especializados, como peer-to-peer, para permitir que los nodos de la red se comuniquen entre sí y compartan información. La descentralización es uno de los principales pilares de esta capa, ya que no existe una autoridad centralizada que controle o regule la red. En su lugar, todos los nodos participantes tienen igual importancia y pueden colaborar en la validación y propagación de transacciones.

La **capa de consenso** es esencial para garantizar la validez de las transacciones. Aquí es donde se lleva a cabo el proceso de consenso, que puede variar dependiendo del algoritmo utilizado. Su objetivo principal es lograr un acuerdo entre los nodos sobre el estado actual de la cadena de bloques y la secuencia de transacciones válidas. El consenso puede basarse en métodos como la prueba de trabajo (PoW), la prueba de participación (PoS), propia de la plataforma Ethereum (Ver 2.2.1.), o algoritmos de consenso más nuevos y eficientes.

Una de las claves para motivar a los participantes a actuar de manera honesta y en beneficio del sistema es la **capa de incentivos**. En este tipo de sistemas, los incentivos suelen estar ligados a recompensas económicas otorgadas a los nodos que realizan la validación de las transacciones y aseguran la red. Estas recompensas pueden ser en forma de token nativas del sistema o mediante otros mecanismos establecidos en el protocolo.

La **capa de contrato** introduce la capacidad de ejecutar contratos inteligentes en la red. Este tipo de contratos son programas informáticos autónomos escritos en lenguajes de programación específicos, como Solidity (Ver 2.2.4.), que se ejecutan de forma automática cuando se cumplen las reglas y condiciones predefinidas, lo que garantiza que todas las partes involucradas cumplan con sus obligaciones. De esta forma, permiten la automatización de acuerdos y transacciones, eliminando la necesidad de intermediarios y brindando transparencia y confianza en las interacciones.

Esta capa de lógica programable forma parte de la creación y ejecución de aplicaciones descentralizadas (DApps) y servicios basados en contratos inteligentes. También proporciona la interoperabilidad entre diferentes sistemas blockchain, lo que facilita la creación de redes y ecosistemas descentralizados más amplios y conectados.

Por último, la **capa de aplicación** es donde se desarrollan las aplicaciones y servicios que aprovechan las características de la cadena de bloques. Pueden incluir billeteras digitales, plataformas de intercambio de tokens, o cualquier otro servicio que se beneficie de la transparencia, la seguridad y la descentralización proporcionadas por el blockchain.

Cada una de estas capas desempeña un papel importante en el funcionamiento global de la red. Esta arquitectura única ha llevado a la adopción de blockchain en diversos sectores y ha abierto la puerta a la innovación tecnológica en todo el mundo.

2.2. Tecnologías blockchain utilizadas

Tras esta breve introducción a esta innovadora tecnología, en este apartado se profundizará en el concepto, características y ventajas de las diversas herramientas relacionadas con la tecnología blockchain utilizadas en el desarrollo de este proyecto.

2.2.1. Ethereum

Ethereum es una plataforma descentralizada de código abierto que utiliza la tecnología blockchain para permitir la creación y ejecución de contratos inteligentes y aplicaciones descentralizadas. Fue fundada en 2015 por Vitalik Buterin, y desde entonces ha crecido hasta convertirse en una de las plataformas de blockchain más importantes y populares en el mundo (Ethereum community, 2023).

El Ether (ETH) es la criptomoneda nativa de esta plataforma. Se utiliza como método de pago de las transacciones y los servicios en la red Ethereum. Además, se utiliza para pagar las tarifas de transacción y los costos de ejecución de los contratos inteligentes en la red (Ethers v5, 2023).

Una de las principales ventajas de Ethereum es su capacidad para ejecutar contratos inteligentes (Ver 2.2.3.). Estos son verificados por toda la red, lo que garantiza que el código sea seguro y que las transacciones sean confiables.

2.2.4. Solidity

Solidity es un lenguaje de programación de alto nivel utilizado para escribir contratos inteligentes. Está diseñado específicamente para ser utilizado con la Máquina Virtual de Ethereum (EVM), responsable de la ejecución de los smart contracts en la plataforma Ethereum (The Solidity Authors, 2023).

Es un lenguaje de tipado estático, por lo que los tipos de datos de las variables se determinan en tiempo de compilación, ayudando a evitar errores y aumentar la seguridad. También admite la herencia, permitiendo a los desarrolladores reutilizar el código y hacer que los contratos inteligentes sean más modulares. También se basa en una sintaxis similar a la de JavaScript y C++, lo que facilita su adopción para aquellos familiarizados con esos lenguajes.

Cuenta con una amplia variedad de bibliotecas y herramientas a disposición, lo que resulta beneficioso para el desarrollo de aplicaciones complejas. Como lenguaje de código abierto, posee una comunidad grande y activa de desarrolladores, lo que implica una abundancia de recursos y soporte.

2.2.2. Billetera Metamask

Una billetera es un software diseñado para almacenar y gestionar tokens de manera segura. Esta herramienta es esencial para poder interactuar con la red blockchain, dado que permite almacenar, transferir y controlar los activos poseídos (Cryptocurrency wallet, 2023).

Existen diferentes tipos de billeteras, cada una con sus propias características y niveles de seguridad. Las billeteras de software son las más comunes y se pueden descargar en un ordenador o dispositivo móvil. También existen billeteras de hardware, dispositivos físicos que almacenan las claves privadas de forma segura y pueden ser conectados a un ordenador para realizar transacciones.

La billetera Metamask es una de las opciones más populares en el mercado de billeteras virtuales para tokens. Creada en 2016 por la compañía ConsenSys, es una extensión de navegador web que funciona con Chrome, Firefox y Brave. Esta herramienta permite a los usuarios almacenar, enviar y recibir tokens de manera segura y fácil (Lee, 2019).

Un factor primordial es su seguridad, utiliza un sistema de claves privadas y públicas para proteger las transacciones de los usuarios. Además, es posible habilitar la autenticación de dos factores (2FA) para una mayor protección. De la misma forma, su facilidad de uso la hace ideal tanto para usuarios nuevos como experimentados (MetaMask, 2023).

2.2.5. Ganache

Según Truffle Suite (2022), Ganache es una herramienta de blockchain personal que permite a los desarrolladores trabajar en un entorno de pruebas seguro y sin riesgos, ofreciendo la simulación de una red Ethereum privada.

Su funcionamiento se basa en la emulación de una red blockchain local. Cuando se inicia, se genera una blockchain privada en el equipo, lo que le permite interactuar con contratos inteligentes, simular transacciones y explorar el comportamiento de la aplicación sin necesidad de utilizar una red real. También crea una serie de cuentas predefinidas con direcciones y claves privadas asociadas, las cuales se pueden importar en una billetera para simular diferentes roles de usuario, así como utilizar sus tokens ficticios en las transacciones.

También es posible integrar este software con otros marcos de desarrollo blockchain, como la herramienta Truffle que veremos a continuación, lo que permite una mayor flexibilidad y personalización en el desarrollo de aplicaciones descentralizadas.

2.2.6. Truffle

Según Truffle Suite (2022), Truffle es un entorno de desarrollo, un marco de pruebas y una tubería de activos para blockchain que utilizan la Máquina Virtual Ethereum. Este software proporciona un conjunto de herramientas de desarrollo que simplifican el proceso de construcción, prueba y despliegue de aplicaciones descentralizadas.

Proporciona una gran variedad de herramientas de desarrollo, incluyendo una red de blockchain privada para pruebas, un compilador de contratos inteligentes y un marco de pruebas automatizadas. Todo esto se combina con una variedad de mecanismos de seguridad para garantizar que los contratos inteligentes y las aplicaciones que se construyen sean seguros y resistentes a ataques maliciosos. Estos mecanismos incluyen el análisis de contratos inteligentes para detectar vulnerabilidades y errores, así como la integración con herramientas de auditoría.

Capítulo 3

Planificación del proyecto

3.1. Introducción

En este capítulo se presenta el plan de desarrollo seguido en la elaboración de la aplicación descentralizada **SecondHandChain**.

El proyecto realizado por el alumno Darío Gago Pérez, ha sido desarrollado utilizando una de las metodologías ágiles más populares, adaptándola a sus necesidades y a las de sus tutores.

3.2. Glosario

En la tabla 3.1 se pueden observar diferentes siglas que se nombrarán a lo largo del documento, junto a su significado.

Sigla	Significado
VS Code	Visual Studio Code
MS Project	Microsoft Project
Astah	Astah Professional
JS	JavaScript
IPFS	InterPlanetary File System
DApp	Aplicación descentralizada

Tabla 3.1: Glosario

3.3. Artefactos del proyecto

- Plan de desarrollo software
- Análisis de requisitos y casos de uso
- Diseño
- Implementación
- Informe de pruebas
- Conclusiones y trabajo futuro
- Guía de instalación
- Manual de usuario

3.4. Restricciones del proyecto

La única restricción del proyecto es la fecha definida para la entrega en periodo ordinario del Trabajo de Fin de Grado, 23 de junio de 2023.

3.5. Metodología utilizada

3.5.1. Scrum

Scrum es un marco de trabajo ágil utilizado para la gestión de productos utilizado habitualmente en el desarrollo de software. Fue creado por Ken Schwaber y Jeff Sutherland en la década de 1990. La gerencia y los equipos de Scrum trabajan juntos alrededor de requisitos y tecnologías para entregar productos funcionando de manera incremental usando el empirismo (Turley & Nader, 2019).

Esta es una metodología altamente adaptable y se puede ajustar según las necesidades del proyecto y del equipo. El marco de trabajo se enfoca en el valor que se entrega y en la colaboración constante entre los miembros del equipo. Esto permite adaptarse a los cambios del proyecto de forma rápida y eficiente, para poder garantizar la satisfacción del cliente.

3.5.2. Organización del equipo

Este marco de trabajo está diseñado para equipos de diez o menos miembros que dividen su trabajo en objetivos que pueden completarse en iteraciones de tiempo limitado, denominados sprint (Ver 3.5.3.). En la tabla 3.2. se presentan los tres roles principales de un equipo Scrum y sus responsabilidades.

Rol	Responsabilidad
Scrum Master	Encargado de optimizar y maximizar el valor del producto, siendo la persona encargada de gestionar el flujo de valor del producto a través del Product Backlog (Ver 3.5.4.).
Product Owner	Dueño del producto, responsable de definir los criterios de aceptación sobre el producto y asegurar que se cumplan.
Development Team	Responsables del desarrollo del producto. Su principal responsabilidad es crear un incremento funcional y de calidad al final de cada sprint.

Tabla 3.2: Roles en Scrum

3.5.3. Eventos

A lo largo del desarrollo se utilizan una serie de eventos clave para gestionar el proceso de desarrollo y entregar un producto de calidad. Estos están diseñados para ayudar al equipo a trabajar de manera eficiente y mantener una comunicación fluida con los interesados del proyecto.

- **Sprint**, marco de tiempo establecido para la entrega de un incremento de producto potencialmente entregable por parte del equipo de Scrum. Su duración típica es de 2 a 4 semanas.

- **Sprint planning**, planificación previa al comienzo de un sprint, con el fin de establecer el objetivo y el plan para alcanzarlo.
- **Daily Scrum**, reunión diaria, de no más de 15 minutos, realizada para sincronizar el trabajo del equipo y verificar el progreso hacia el objetivo del sprint.
- **Sprint review**, reunión al final de cada Sprint con objetivo de revisar el trabajo realizado y determinar qué elementos del Product Backlog (Ver 3.5.4.) han sido completados.
- **Sprint Retrospective**, posterior a la revisión del sprint, reflexión sobre el sprint completado para determinar las dificultades presentadas y propuestas de mejora.
- **Product Backlog refinement,** revisión y actualización del Product Backlog por parte del Development Team y el Product Owner.

3.5.4. Artefactos

Al igual que los eventos, los artefactos se utilizan para garantizar que el equipo trabaje de manera coordinada, manteniendo siempre la comunicación entre los miembros y los interesados. Estos son dinámicos y evolucionan a lo largo del proyecto, siendo responsabilidad de todos su revisión y actualización constante.

- **Product Backlog**, lista ordenada por prioridad de los requisitos necesarios del producto. Sus elementos son definidos por el Product Owner y estimados por el Equipo de Desarrollo para incluirlos en el sprint.
- **Sprint Backlog**, lista de elementos seleccionados del Product Backlog para ser completados durante el sprint actual. Está conformado por historias de usuario y, una vez establecido, es inmutable durante el desarrollo.
- **Incremento**, resultado del trabajo del Equipo de Desarrollo durante un sprint, incluyendo los elementos completados en los anteriores. Se debe haber completado con éxito la implementación, pruebas e integración de todas las funcionalidades necesarias para que sea usable y funcional.

En la Figura 3.1 se presenta una visión general del proceso completo dentro de esta metodología ágil.

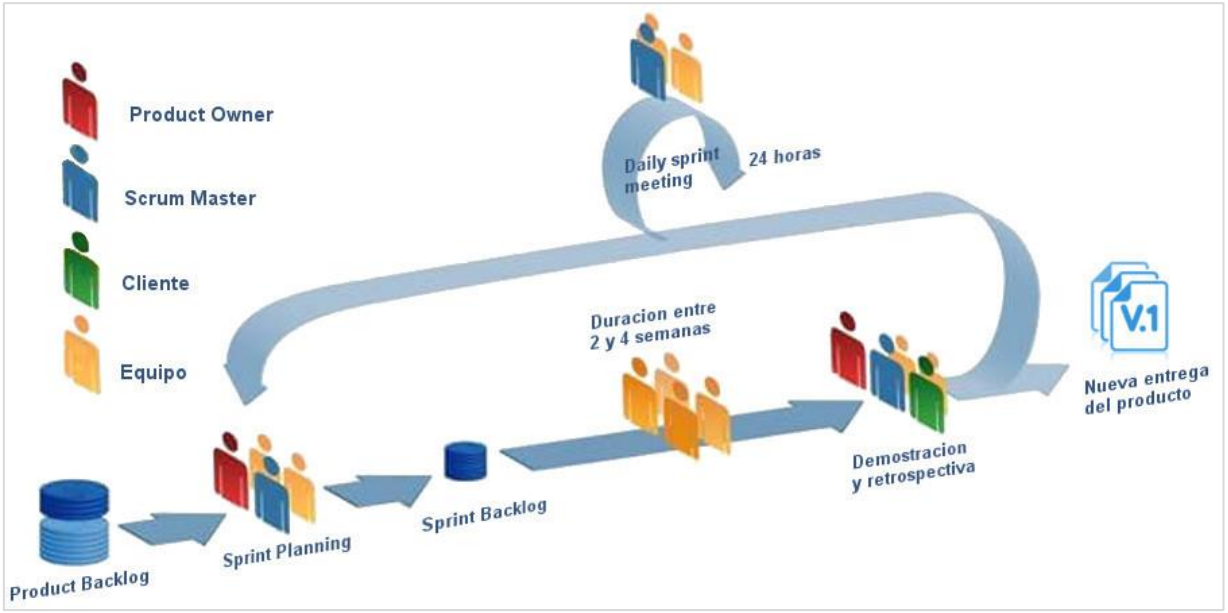


Figura 3.1: Eventos y artefactos Scrum. Obtenido de Welcomedevelopers, por Domínguez (2018).

3.6. Planificación

3.6.1. Adaptación del Scrum al proyecto

Otro factor importante que ha influido en la elección de esta metodología es la libertad que ofrece. Al basarse en sprints de tiempo fijo, se tiene una visión global clara del trabajo a realizar, favoreciendo la organización y adaptación a las circunstancias externas.

En primer lugar, se establecen los roles del equipo, donde el alumno será el Product Owner y, al mismo tiempo, único miembro del Equipo de Desarrollo. Los Scrum Masters e interesados del proyecto, serán los propios tutores del alumno, tanto de empresa como académico.

Todos los eventos referentes a cada sprint, omitiendo el Daily Scrum, se realizarán a través de una reunión de seguimiento establecida mediante videoconferencia entre los tutores de empresa y el alumno. En caso de imposibilidad por alguna de las partes, se establecerá una nueva fecha donde realizar la reunión, con el fin de no retrasar las fechas previstas de los futuros sprints. Del mismo modo, se establece la comunicación semanal vía correo para la necesidad de cualquiera de las partes.

Se realizará un sprint 0, con motivo del desconocimiento y dudas del alumno sobre las tecnologías y objetivos del proyecto. La reunión de inicio se establecerá para la fecha 22/02/2023.

3.6.2. Planificación inicial de los sprints

Se ha establecido la duración de los sprints en 2 semanas, tiempo aceptado por el alumno para el correcto desarrollo de cada incremento. La finalización del último sprint se ha fijado en la fecha más próxima al final del mes de junio acorde a la fecha límite de presentación del presente Trabajo de Fin de Grado en el periodo ordinario. De esta forma, cada sprint tendrá una carga aproximada de 34 horas, dando un resultado final aproximado de 300 horas, conforme a la guía docente de la asignatura.

Sprint	Comienzo	Finalización
Sprint 1	22/02/2023	08/03/2023
Sprint 2	08/03/2023	22/03/2023
Sprint 3	22/03/2023	05/03/2023
Sprint 4	05/04/2023	19/04/2023
Sprint 5	19/04/2023	03/05/2023
Sprint 6	03/05/2023	17/05/2023
Sprint 7	17/05/2023	31/05/2023
Sprint 8	31/05/2023	14/06/2023
Sprint 9	14/06/2023	28/06/2023

Tabla 3.3: Planificación inicial de sprints

3.7. Análisis de riesgos

En el siguiente apartado se presentan los riesgos propuestos como obstáculos en el transcurso normal del desarrollo del proyecto, así como su probabilidad de aparición, impacto, consecuencia, nivel de riesgo, plan de prevención y plan de respuesta. Se presenta en la tabla 3.4 el cálculo del nivel de riesgo según la probabilidad y el impacto, que se utilizará en las tablas de los riesgos establecidos desde la tabla 3.5 hasta la tabla 3.16.

Prob/Imp	Baja	Media	Alta
Bajo	Bajo	Bajo	Medio
Medio	Bajo	Medio	Alto
Alto	Medio	Alto	Alto

Tabla 3.4: Nivel de riesgo

R0	Falta de disponibilidad
Descripción	El alumno no se dispone de suficiente tiempo o sufre una baja temporal
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	El alumno no sigue la planificación
Plan de mitigación	Modificar la planificación según se prevé una falta de disponibilidad
Plan de contingencia	Replanificar o compensar con tiempo extra

Tabla 3.5: Riesgo 0- Falta de disponibilidad

R1	Problemas en la formación
Descripción	El alumno presenta dificultades en su formación en las tecnologías del proyecto
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	Ralentización respecto a la planificación
Plan de mitigación	Búsqueda de información y formación con los tutores
Plan de contingencia	Revisar conocimientos

Tabla 3.6: Riesgo 1- Problemas de formación

R2	Falta de experiencia en la planificación
Descripción	El alumno desarrolla incorrectamente la planificación del proyecto
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	No cumplimiento de las fechas propuestas
Plan de mitigación	Revisión de la planificación por parte de los tutores
Plan de contingencia	Refinamiento del Sprint Backlog

Tabla 3.7: Riesgo 2- Falta de experiencia en la planificación

R3	Pérdida de entregables
Descripción	Se pierde la documentación y/o código del proyecto
Probabilidad	Baja
Impacto	Alta
Nivel de riesgo	Medio
Consecuencia	Volver a realizar el entregable perdido
Plan de mitigación	Uso del sistema de control de versiones GitLab
Plan de contingencia	Búsqueda de posibles copias de seguridad

Tabla 3.8: Riesgo 3- Pérdida de entregables

R4	Imposibilidad de seguimiento
Descripción	Imposibilidad por alguna de las partes de realizar reuniones de seguimiento
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Consecuencia	Falta de revisión en las actualizaciones del proyecto
Plan de mitigación	No aplica
Plan de contingencia	Establecer una nueva fecha de reunión

Tabla 3.9: Riesgo 4- Imposibilidad de seguimiento

R5	Falta de medios
Descripción	Los medios necesarios para el desarrollo del proyecto sufren problemas técnicos y se imposibilita su uso
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	Aplazamiento de la planificación del proyecto
Plan de mitigación	Recursos de reserva
Plan de contingencia	Búsqueda de nuevas herramientas

Tabla 3.10: Riesgo 5- Falta de medios

R6	Diseño incorrecto
Descripción	El diseño de la aplicación se realiza de forma incorrecta o incoherente
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	Replanteamiento del diseño
Plan de mitigación	Revisiones al diseño por parte de los tutores
Plan de contingencia	Reformulación del diseño

Tabla 3.11: Riesgo 6- Diseño incorrecto

R7	Nuevas especificaciones
Descripción	Se incorporan nuevas funcionalidades al proyecto
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Consecuencia	Nuevas implementaciones a realizar
Plan de mitigación	Previsión de todas las especificaciones al inicio de la planificación
Plan de contingencia	Implantación y adaptación de las nuevas especificaciones

Tabla 3.12: Riesgo 7- Nuevas especificaciones

R8	Falta de experiencia en planificación de riesgos
Descripción	El alumno desarrolla incorrectamente los planes de mitigación o contingencia en alguno de los riesgos
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Consecuencia	Consecuencias no previstas ante el riesgo sufrido
Plan de mitigación	No aplica
Plan de contingencia	Reanálisis del riesgo

Tabla 3.13: Riesgo 8- Falta de experiencia en planificación de riesgos

R9	Insuficiente planificación de riesgos
Descripción	No se han planteado todos los posibles riesgos
Probabilidad	Media
Impacto	Medio
Nivel de riesgo	Medio
Consecuencia	Aparición de un riesgo no previsto
Plan de mitigación	Planificación de todos los posibles riesgos
Plan de contingencia	Análisis del riesgo

Tabla 3.14: Riesgo 9- Insuficiente planificación de riesgos

R10	Baja calidad de código
Descripción	La aplicación está mal codificada, de forma poco eficiente o produce fallas
Probabilidad	Baja
Impacto	Alto
Nivel de riesgo	Medio
Consecuencia	Ralentización respecto a la planificación
Plan de mitigación	Revisión de la calidad de código y funcionalidades
Plan de contingencia	Reestructuración del código y arreglo de errores

Tabla 3.15: Riesgo 10- Baja calidad de código

R11	No cumplimiento de la fecha límite
Descripción	La finalización del proyecto no se cumple en el plazo previsto
Probabilidad	Baja
Impacto	Medio
Nivel de riesgo	Bajo
Consecuencia	No cumplimiento con la planificación prevista
Plan de mitigación	No aplica
Plan de contingencia	Replanificación del proyecto y entrega en periodo extraordinario

Tabla 3.16: Riesgo 11- No cumplimiento de la fecha límite

3.8. Estimación de costes

Para estimar el costo del proyecto, se tomará como referencia la remuneración correspondiente a un ingeniero informático en España, calculada en función de las horas necesarias para llevar a cabo el proyecto. De este modo, el costo del proyecto comprenderá tanto el importe necesario para cubrir las horas de trabajo del programador, como los gastos adicionales requeridos por el uso de las herramientas de software necesarias para el desarrollo, junto con el hardware necesario para su implementación y otros costes.

Se supondrá la duración del proyecto en las 300 horas estimadas por la Universidad de Valladolid, para el Trabajo de Fin de Grado en la carrera de Ingeniería Informática.

3.8.1. Coste de mano de obra

Según el Ministerio de Trabajo y economía social (2023), un ingeniero informático recibe de media unos 30.300€ brutos al año, realizando aproximadamente unas 1.664 horas. Por lo que, estaría cobrando de media unos 18,21 euros/hora, dando un total de 5.463€ euros por 300 horas de trabajo.

3.8.2. Costes de hardware

En el desarrollo del proyecto se hará uso únicamente de un ordenador portátil y una pantalla auxiliar. Para la estimación de su coste, se tendrá en cuenta el coste real de cada dispositivo, en proporción con las horas de uso desde su adquisición. Ver tabla 3.17.

Dispositivo	Precio	Horas de uso	Precio/hora	Horas en proyecto	Coste
Ordenador portátil	850€	4380h (3años, 4h/día)	0,19€/hora	300h	57€
Pantalla auxiliar	90€	2920h (2años, 4h/día)	0,03€/hora	270h	8,10€
Periféricos	30€	2920h (2años, 4h/día)	0,01€/hora	270h	2,70€

Tabla 3.17: Costes hardware

3.8.3. Costes de software

Gran parte de las herramientas que serán utilizadas son software gratuito, sin embargo, se usarán ciertas herramientas de pago. La licencia de la mayoría de estas herramientas es proporcionada por la Universidad de Valladolid para el alumno, sin coste alguno. Por lo que, realmente no supondrán un gasto personal. Ver tabla 3.18.

Software	Coste
Windows 11 home	145€
VS Code	0€
Ganache	0€
GitLab	0€
Astah	30€/año
Microsoft 365	69€/año

Tabla 3.18: Costes software

3.8.4. Otros costes

La conexión a internet que será utilizada durante el periodo de trabajo se obtiene del paquete Movistar de fibra óptica de 300 Mb por un total de 29,99€ mensuales. Dando un total, en 4 meses, de 119,96€.

Suponemos el coste de la electricidad insignificante, dado que el único consumo eléctrico será la carga del dispositivo portátil y router.

3.8.5. Presupuesto del proyecto

La estimación total del trabajo se realizará sumando todos los costes anteriores. Dando un presupuesto total para el proyecto de 5.894,76€. Ver tabla 3.19.

Software	Coste
Coste de mano de obra	5.463€
Coste de hardware	67,80€
Coste de software	244€
Otros costes	119,96€
Presupuesto total	5.894,76€

Tabla 3.19: Presupuesto total

3.9. Seguimiento del proyecto

A continuación, se presentan los desvíos en la planificación respecto al comienzo y finalización estimados y reales de cada uno de los sprints (Ver tabla 3.20).

Sprint	Comienzo estimado	Finalización estimada	Comienzo real	Finalización real	Desvío
Sprint 1	22/02/2023	08/03/2023	22/02/2023	22/03/2023	14 días
Sprint 2	08/03/2023	22/03/2023	22/03/2023	12/04/2023	21 días
Sprint 3	22/03/2023	05/04/2023	12/04/2023	19/04/2023	14 días
Sprint 4	05/04/2023	19/04/2023	19/04/2023	03/05/2023	14 días
Sprint 5	19/04/2023	03/05/2023	03/05/2023	17/05/2023	14 días
Sprint 6	03/05/2023	17/05/2023	17/05/2023	31/05/2023	14 días
Sprint 7	17/05/2023	31/05/2023	31/05/2023	14/06/2023	14 días
Sprint 8	31/05/2023	14/06/2023	14/06/2023	28/06/2023	14 días
Sprint 9	14/06/2023	28/06/2023	-	-	0 días

Tabla 3.20: Desvío respecto a la planificación inicial

La tabla muestra como la mayoría de los sprints experimentaron retrasos en su inicio y finalización en comparación con la planificación inicial, con desvíos que varían entre los 14 y 21 días adicionales a lo planeado. Esta falta de cumplimiento con el cronograma original del proyecto ha requerido ajustes en la organización y gestión del proyecto para minimizar los retrasos futuros.

En el caso del Sprint 1, el retraso se debió a la aparición de los riesgos R0, falta de disponibilidad, y R1, problemas de formación. Del mismo modo, en el Sprint 2, la demora fue debida de nuevo a la aparición del riesgo R0, falta de disponibilidad, referente a las vacaciones escolares de Semana Santa. Este retraso fue reducido en el siguiente sprint, aplicando uno de los planes de contingencia descritos en R0, compensar con tiempo extra. Se presentan los riesgos descritos en el apartado 3.7. Análisis de riesgos.

Capítulo 4

Análisis

En este capítulo se presenta un análisis detallado del funcionamiento deseado de la aplicación. Se llevará a cabo un minucioso análisis de los requisitos deseados, así como un estudio de los casos de uso en los que se puedan hallar los actores encontrados.

4.1. Análisis de requisitos

El análisis de requisitos se refiere al proceso de comprender, identificar y documentar las necesidades y expectativas de las partes interesadas en este proyecto. Es un paso crítico que establece las bases para el diseño, implementación y prueba del sistema.

4.1.1. Requisitos funcionales

En la tabla 4.1 se presentan todos los requisitos funcionales del proyecto, los cuales describen las funcionalidades y características que la aplicación debe proporcionar y especifican cómo los usuarios podrán interactuar y qué resultados deben esperar del sistema.

ID	Requisito	Descripción
RF01	Confirmación de transacción	El sistema debe permitir confirmar las transacciones a través de una billetera Metamask
RF02	Registro producto	El sistema debe permitir registrar un producto mediante sus características, imágenes y documentos
RF03	Búsqueda producto	El sistema debe permitir buscar un producto a través de su nombre o identificador
RF04	Colección productos	El sistema debe permitir visualizar el listado de productos pertenecientes a la cuenta utilizada
RF05	Editar producto	El sistema debe permitir la edición de las características de los productos pertenecientes a la cuenta utilizada
RF06	Eliminar producto	El sistema debe permitir la edición de la eliminación de los productos pertenecientes a la cuenta utilizada
RF07	Productos en venta	El sistema debe permitir visualizar el listado de productos no pertenecientes a la cuenta utilizada
RF08	Compra de un producto	El sistema debe permitir la compra de productos no pertenecientes a la cuenta utilizada
RF09	Visión del producto	El sistema debe permitir la visualización de las características e imágenes de un producto
RF10	Descargar documentos	El sistema debe permitir la descarga de los documentos de un producto
RF11	Historia del producto	El sistema debe permitir visualizar los eventos sufridos por un producto en orden cronológico

RF12	Características eventos historia	El sistema debe permitir visualizar los detalles de cada evento sufrido por un producto
RF13	Filtrar historia por tipo	El sistema debe permitir filtrar el listado de eventos de un producto por el tipo evento
RF14	Filtrar historia por rango de fechas	El sistema debe permitir filtrar el listado de eventos de un producto por la fecha en los que se emitieron
RF15	Sobre nosotros	El sistema debe permitir visualizar la información referida al presente proyecto
RF16	Atrás	El sistema debe permitir volver atrás en cualquier momento
RF17	Actualización	El sistema debe tener actualizados los listados de productos y sus características en función de la cuenta utilizada

Tabla 4.1: Requisitos funcionales

4.1.2. Requisitos de información

En la tabla 4.2 se presentan todos los requisitos de información del proyecto, centrados en la información que el sistema debe gestionar y procesar, tanto en términos de entrada como de salida.

ID	Requisito	Descripción
RI01	Red blockchain	El sistema deberá estar integrado en una cadena de bloques para garantizar la transparencia y la auditabilidad de la información
RI02	Producto	El sistema deberá almacenar información detallada del producto, como su nombre, descripción y precio
RI03	Información adicional	El sistema deberá permitir a los usuarios agregar información adicional, como imágenes y documentos relacionados con el producto
RI04	Historial del producto	El sistema deberá almacenar toda la información relevante del historial del producto, incluyendo propietarios anteriores, modificaciones y puesta en alta

Tabla 4.2: Requisitos de información

4.1.3. Requisitos no funcionales

En la tabla 4.3 se presentan todos los requisitos no funcionales del proyecto, referentes a las características y cualidades que debe poseer este sistema, más allá de su funcionalidad.

ID	Descripción	Importancia
RNF01	El sistema debe desarrollarse sobre una red de pruebas Ethereum	Crítico
RNF02	El sistema debe permitir la realización de transacciones a través de una billetera Metamask	Crítico
RNF03	El front-end del sistema debe consistir en una aplicación web	Crítico
RNF04	El sistema debe proporcionar un mecanismo seguro de registro, compra, edición, eliminación y visualización de productos	Crítico
RNF05	El sistema debe proporcionar todos sus servicios a través del despliegue de contratos inteligentes en una red blockchain	Crítico
RNF06	El sistema debe permitir la edición y eliminación de un producto solamente al propietario	Crítico
RNF07	El sistema debe utilizar el formato UTF-08	Crítico
RNF08	El sistema debe proveer una interfaz sencilla y clara de utilizar	Deseable
RNF9	El sistema debe ser accesible desde los principales navegadores	Deseable
RNF10	El sistema debe ser eficiente en su uso	Deseable

Tabla 4.3: Requisitos no funcionales

4.1.4. Reglas de negocio

En la tabla 4.4 se presentan todas las reglas de negocio del proyecto, los cuales definen las políticas y restricciones que gobiernan el funcionamiento del sistema resultado ante la comunidad de usuarios.

ID	Regla	Descripción
RDN01	Actualización del historial del producto	Los usuarios deben mantener actualizado el historial del producto, incluyendo cualquier uso previo, mantenimiento, reparaciones, etc.
RDN02	Uso responsable de la plataforma	Los usuarios deben utilizar la plataforma de manera responsable y no hacer un mal uso de la información proporcionada
RDN03	Prohibición de productos ilegales o peligrosos	La plataforma no permitirá la venta de productos ilegales o peligrosos

Tabla 4.4: Reglas de negocio

4.2. Casos de uso

4.2.1. Actores principales

Únicamente se ha hallado como actor principal el propio usuario que interactúa con el sistema, quien representa una dirección de cuenta dentro de la billetera Metamask conectada a la red blockchain.

4.2.2. Diagrama de casos de uso

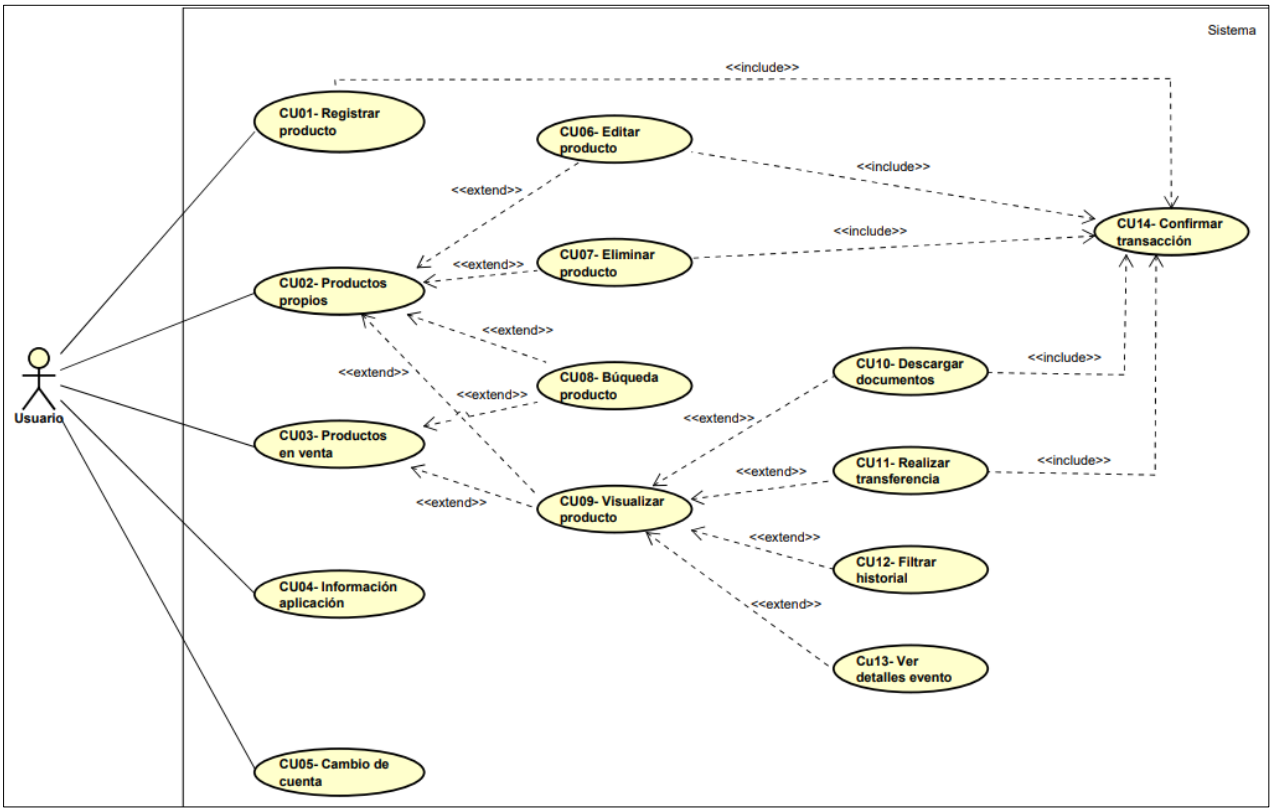


Figura 4.1: Diagrama de casos de uso

4.2.3. Descripción de los casos de uso

A continuación, se describen la secuencia de eventos que ocurren en la interacción entre el usuario y el sistema para lograr el objetivo de cada caso de uso. Así como las situaciones alternativas que pueden ocurrir durante su ejecución.

ITEM	VALUE
UseCase	CU01- Registrar producto
Summary	El usuario registra un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask.
Postcondition	El usuario añade el producto como propietario actual del mismo.
Base Sequence	1. El usuario selecciona el botón 'Registrar producto'. 2. El sistema muestra el formulario de registro de producto. 3. El usuario rellena al menos los campos obligatorios y pulsa el botón 'Registrar'. 4. Se ejecuta el CU13- Confirmar transacción. 5. El sistema redirige a la página 'Mis productos'.
Branch Sequence	3a. El usuario no rellena los campos obligatorios, se informa al usuario de los campos que falt an por rellenar.
Exception Sequence	2,3- Si el usuario vuelve hacia atrás, el caso de uso queda sin efecto
Sub UseCase	CU14- Confirmar transacción
Note	

Figura 4.2: Caso de Uso 1- Registrar producto

UseCase	CU02- Productos propios
Summary	El usuario consulta el listado de los productos pertenecientes a su cuenta.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask.
Postcondition	El sistema muestra un lista de productos pertenecientes a la cuenta del usuario actualmente.
Base Sequence	1. El usuario selecciona el botón 'Mis productos'. 2. El sistema muestra un listado de los productos.
Branch Sequence	2a. El usuario no posee productos propios, no se muestran productos
Exception Sequence	
Sub UseCase	
Note	

Figura 4.3: Caso de Uso 2- Productos propios

UseCase	CU03- Productos en venta
Summary	El usuario consulta el listado de los productos en venta.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask.
Postcondition	El sistema muestra un lista de productos no pertenecientes a la cuenta del usuario actualmente.
Base Sequence	1. El usuario accede a la página principal de la aplicación. 2. El sistema muestra un listado de los productos.
Branch Sequence	1a. El usuario selecciona el logo de la apliacación, el caso de uso sigue en el paso 2.
Exception Sequence	
Sub UseCase	
Note	

Figura 4.4: Caso de Uso 3- Productos en venta

UseCase	CU04- Información aplicación
Summary	El usuario accede a la sección de información.
Actor	Usuario
Precondition	
Postcondition	El sistema muestra la página de información sobre la aplicación.
Base Sequence	1. El usuario selecciona el botón 'Sobre nosotros". 2. El sistema muestra la información sobre la aplicación.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figura 4.5: Caso de Uso 4- Información aplicación

UseCase	CU05- Cambio de cuenta
Summary	El usuario cambia de cuenta dentro de su billetera metamask.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask.
Postcondition	El sistema actualizará la información referente a la cuenta elegida.
Base Sequence	1. El usuario cambia de cuenta dentro de su billetera metamask. 2. El sistema recarga la página donde se encontraba el usuario.
Branch Sequence	1a. Se produce un error en la conexión de la nueva cuenta con la red, el caso de uso queda sin efecto.
Exception Sequence	
Sub UseCase	
Note	

Figura 4.6: Caso de Uso 5- Cambio de cuenta

UseCase	CU06- Editar producto
Summary	El usuario edita mediante un formulario las características de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la sección 'Mis productos'.
Postcondition	Se actualizan las carterísticas del producto.
Base Sequence	1. El usuario selecciona el icono de edición de un producto. 2. El sistema muestra un formulario con las características del producto. 3. El usuario edita los campos del producto. 4. Se ejecuta el CU13- Confirmar transacción. 5. El sistema muestra el listado actualizado.
Branch Sequence	3a. El usuario deja vacío alguno de los campos obligatorios, se informa al usuario de los campos que faltan por rellenar.
Exception Sequence	2a,3a- El usuario cierra el formulario de edición, el caso de uso queda sin efecto 2b,3b- Si el usuario vuelve hacia atrás, el caso de uso queda sin efecto
Sub UseCase	CU14- Confirmar transacción
Note	

Figura 4.7: Caso de Uso 6- Editar producto

UseCase	CU07- Eliminar producto
Summary	El usuario elimina un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la sección 'Mis productos'.
Postcondition	Se elimina el producto.
Base Sequence	1. El usuario selecciona el icono de edición de un producto. 2. El sistema muestra una alerta de eliminación del producto. 3. El usuario pulsa el botón 'aceptar'. 4. Se ejecuta el CU13- Confirmar transacción. 5.El sistema muestra el listado actualizado.
Branch Sequence	2a. El usuario pulsa el botón 'cancelar', el caso de uso queda sin efecto.
Exception Sequence	2b- Si el usuario vuelve hacia atrás, el caso de uso queda sin efecto
Sub UseCase	CU14- Confirmar transacción
Note	

Figura 4.8: Caso de Uso 7- Eliminar producto

UseCase	CU08- Búqueda producto
Summary	El usuario busca el nombre o ID de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de inicio o en la sección 'Mis productos'
Postcondition	Se muestra el listado actualizado con los productos cuyo nombre o ID contienen la entrada de l usuario.
Base Sequence	1. El usuario escribe una entrada en la barra de búsqueda. 2. El sistema muestra el listado de productos actualizado.
Branch Sequence	2a. No hay coincidencia con ningún producto, no se muestran productos
Exception Sequence	
Sub UseCase	
Note	

Figura 4.9: Caso de Uso 8- Búsqueda producto

UseCase	CU09- Visualizar producto
Summary	El usuario accede a la información de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de inicio o en la sección 'Mis productos'.
Postcondition	Se muestran las imágenes, características, documentos e historial del producto seleccionado.
Base Sequence	1. El usuario selecciona un producto. 2. El sistema muestra la información del producto seleccionado.
Branch Sequence	2a- El producto pertenece a la cuenta, no se muestra el botón 'Transferir'.
Exception Sequence	
Sub UseCase	
Note	

Figura 4.10: Caso de Uso 9- Visualizar producto

UseCase	CU10- Descargar documentos
Summary	El usuario descarga un documento perteneciente a un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de información de un producto. Existen documentos registrados para ese producto.
Postcondition	Se descarga un documento .pdf
Base Sequence	1. El usuario selecciona el botón 'Descargar' de alguno de los documentos. 2. El sistema descarga el documento seleccionado.
Branch Sequence	2a- Se produce un error en la descarga, el sistema informa de que se ha producido un error e n la descarga.
Exception Sequence	
Sub UseCase	CU14- Confirmar transacción
Note	

Figura 4.11: Caso de Uso 9- Descargar documentos

UseCase	CU11- Realizar transferencia
Summary	El usuario realiza la transferencia de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de información de un producto no propio.
Postcondition	El propietario se añade como propietario actual del producto.
Base Sequence	1. El usuario selecciona el botón 'Transferir'. 2. Se ejecuta el CU13- Confirmar transacción. 3. El sistema redirige a la página 'Mis productos'.
Branch Sequence	
Exception Sequence	
Sub UseCase	CU14- Confirmar transacción
Note	

Figura 4.12: Caso de Uso 11- Realizar transferencia

UseCase	CU12- Filtrar historial
Summary	El usuario filtra el listado de eventos del historial de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de información de un producto.
Postcondition	Se filtra el listado de eventos del historial del producto.
Base Sequence	1. El usuario utiliza el filtro de tipo de evento y/o el filtro de rango de fechas. 2. El sistema muestra el listado de eventos actualizado.
Branch Sequence	2a- No se encuentran eventos, el sistema la frase 'No se tiene información para ese periodo'.
Exception Sequence	
Sub UseCase	
Note	

Figura 4.13: Caso de Uso 12- Filtrar historial

UseCase	Cu13- Ver detalles evento
Summary	El usuario accede a los detalles de un evento del historial de un producto.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El usuario se encuentra en la página de información de un producto. Hay al menos un evento en el historial del producto.
Postcondition	El sistema muestra los detalles del evento seleccionado.
Base Sequence	1. El usuario selecciona uno de los eventos del historial de un producto. 2. El sistema abre una ventana emergente con los detalles del evento seleccionado.
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

Figura 4.14: Caso de Uso 13- Ver detalles evento

UseCase	CU14- Confirmar transacción
Summary	La billetera del usuario solicita confirmar una transacción.
Actor	Usuario
Precondition	La cartera del usuario está conectada a la red blockchain a través de Metamask. El sistema ha enviado una transacción a la billetera del usuario.
Postcondition	Se produce la transacción.
Base Sequence	1. El sistema muestra una notificación de Metamask solicitando la confirmación de la transacci ón. 2. El usuario pulsa el botón 'confirmar' en la notificación. 3. El sistema ejecuta la función del contrato que implementa el servicio solicitado.
Branch Sequence	2a. El usuario rechaza la transacción, el sistema informa de que la transacción ha sido rechaz ada. 3a. Se produce un error en la transacción, el sistema informa de que se ha producido un erro r en la transacción.
Exception Sequence	
Sub UseCase	
Note	

Figura 4.15: Caso de Uso 14- Confirmar transacción

4.3. Modelo de dominio

El modelo de dominio incluye entidades como Usuario, Producto y Evento, con asociaciones que conectan a los usuarios con los productos a través de eventos específicos de transacción. Esto permite representar las interacciones entre el usuario y los productos existentes, como son el registro de un nuevo producto, la modificación de sus atributos o la transferencia del mismo a otro usuario.

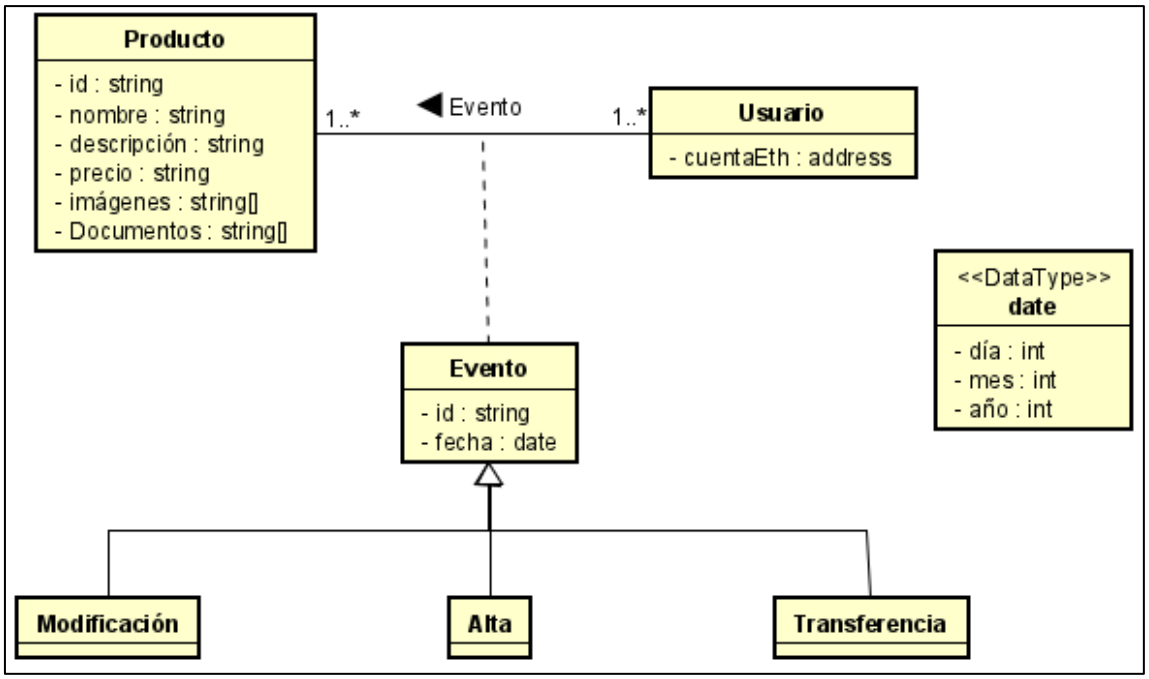


Figura 4.16: Modelo conceptual de dominio

En los siguientes puntos se describen cada una las clases representadas, así como la funcionalidad de sus atributos.

4.3.1. Usuario

- **Descripción:** Clase que modela la dirección de una cuenta Metamask.
- **Atributos:**
    - cuentaEth: Dirección de la cuenta de la billetera Metamask (Ver 2.2.2.).

4.3.2. Producto

- **Descripción:** Clase que modela un producto registrado.
- **Atributos:**
    - id: Identificador del producto.
    - nombre: Nombre del producto.
    - descripción: Descripción del producto

- precio: Precio del producto.
- imágenes: Lista de hashes de acceso a las imágenes del producto dentro de la red IPFS (Ver 6.2.5.).
- documentos: Lista de hashes de acceso a los documentos del producto dentro de la red IPFS.

4.3.3. Evento

- **Descripción:** Clase que modela un evento emitido dentro del contrato inteligente.
- **Atributos:**
  - id: Identificador del evento.
  - fecha: Fecha de emisión del evento en la red.

4.3.4. Alta

- **Descripción:** Clase que modela el evento de alta de un producto.

4.3.5. Modificación

- **Descripción:** Clase que modela el evento de modificación de un producto.

4.3.6. Transferencia

- **Descripción:** Clase que modela el evento de transferencia de un producto a otra cuenta.

Capítulo 5

Diseño

Como paso sucesivo al Análisis y previo al inicio de la implementación de la aplicación, resulta necesario la descripción de las decisiones de diseño que la rigen. En las siguientes secciones se exponen las decisiones de diseño adoptadas, así como la arquitectura empleada y la organización interna del diseño de la aplicación.

5.1. Patrón de diseño

5.1.1. Model-View-ViewModel

El patrón MVVM es un patrón de diseño arquitectónico que se utiliza en el desarrollo de aplicaciones. Permite separar limpiamente la lógica de negocios de su interfaz de usuario. Se divide, de acuerdo con Stonis, Jain y Pine (2022), en tres capas principales:

- Vista (View), capa de presentación con la que los usuarios interactúan para realizar acciones y ver los resultados.
- Modelo (Model), capa de datos que representa la estructura de datos y la lógica de negocio relacionada con el acceso y manipulación de datos.
- Modelo de vista (ViewModel), capa intermedia que proporciona la lógica necesaria para actualizar la vista cuando cambian los datos del modelo y puede contener lógica adicional, como la validación de datos o la gestión de eventos.

En la figura 5.1. se muestra una representación de la comunicación entre las diferentes capas del patrón.

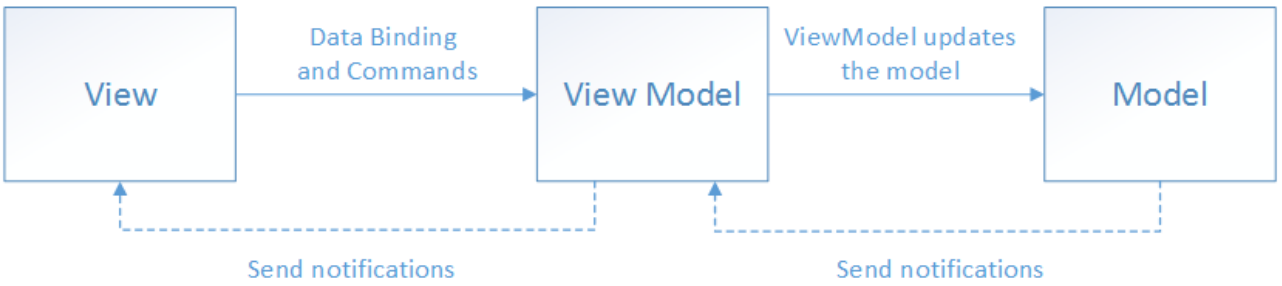


Figura 5.1: Patrón MVVM. Obtenido de Microsoft Corporation, por Stonis, Jain, & Pine, (2022).

Se observa como las tres capas se comunican entre sí de manera unidireccional, siguiendo el principio de separación de intereses. Esta separación hace que sea más fácil realizar cambios en la lógica de negocio o en la interfaz de usuario sin afectar al resto de la aplicación.

5.1.2. NVVM en el proyecto

En el marco de la presente aplicación descentralizada, este patrón puede ser especialmente útil para separar la interfaz de usuario de la lógica del contrato y facilitar la integración entre ambas partes. De esta forma, podemos diferenciar las capas vistas anteriormente en el contexto de la aplicación:

- Vista (View), la biblioteca React (Ver 6.2.1.) se encargará de la parte de la vista.
- Modelo (Model), incluirá los datos almacenados en la red blockchain mediante el despliegue de los smart contracts.
- Modelo de vista (ViewModel), interactuará con los contratos inteligentes a través de la tecnología Web3 (Ver 6.2.4.), para acceder y manipular datos dentro de la red.

A continuación, se describen algunas de las ventajas de este patrón dentro de la aplicación:

- Separación de intereses, facilita el mantenimiento y cambios en la aplicación a medida que se agregan nuevas funcionalidades y se modifican los requisitos.
- Reutilización de código, permite que cada componente pueda ser reutilizado en otras partes de la aplicación que utilicen la misma lógica de contrato inteligente.
- Integración con la blockchain, favorece la integración con la cadena de bloques y la gestión de las transacciones y eventos.
- Pruebas unitarias, creación de pruebas unitarias para cada componente, verificando su correcto funcionamiento de forma independiente.

5.2. Diseño web

5.2.1. Diseño basado en páginas

El planteamiento de estructura para el diseño web es el enfoque basado en páginas. En este enfoque, se construyen componentes que se corresponden con cada una de las páginas que compondrán la aplicación. Cada componente está diseñado para contener todos los elementos necesarios para mostrar y operar en esa página. También se incluyen componentes secundarios, como encabezados o pies de página, que se comparten entre el resto. Estas son algunas de las ventajas que aporta este enfoque de implementación dentro del cliente:

- Simplicidad, facilita el entendimiento e implementación de la estructura de la aplicación, ideal para aplicaciones de complejidad moderada.
- Separación, permite desglosar las diferentes áreas en componentes individuales. Esto ayuda a separar su diseño, funcionalidad y gestión de la del resto.
- Eficiencia, de acuerdo con Oracle (2022) el sistema virtual DOM de React permite actualizar solo los componentes que han cambiado de estado en lugar de actualizar todo el árbol de componentes, lo que mejora la eficiencia y el rendimiento.
- Reutilización de código, facilita la creación de componentes reutilizables, acelerando el proceso de desarrollo y reduciendo la cantidad de código.

5.2.2. Componentes del proyecto

A continuación, se presentará la lista de los componentes que se desarrollarán en la implementación (Ver 6.5.):

- **App:** Componente padre del que cuelgan el resto de componentes. Además, contiene el enrutamiento general entre estos.
- **ProductList:** Componente reutilizado para dos de las páginas, la página de inicio y la página de productos en propiedad.
- **Register:** Componente que contiene el formulario de registro de un nuevo producto.
- **ShowProduct:** Componente que muestra la información e historia de un producto existente.
- **Header:** Barra superior de navegación.
- **Footer:** Marca de derechos reservados.
- **AboutUs:** Componente que muestra la información introductoria a la aplicación web.

5.3. Diagrama de componentes

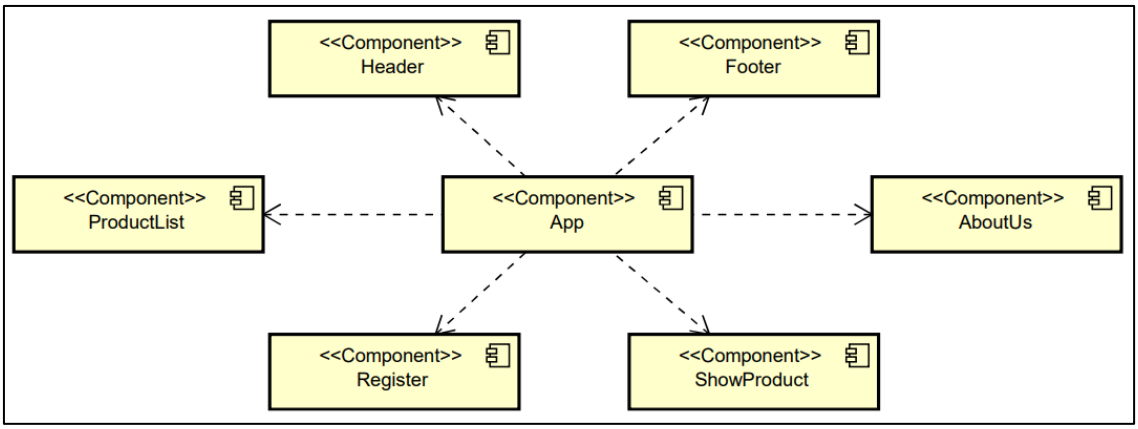


Figura 5.2: Diagrama de componentes

5.4. Prototipo de la interfaz

En el presente apartado se presenta el prototipo de la interfaz de usuario creado con la herramienta de diseño gráfico Canva (s.f.).

En la Figura 5.3 se muestra un prototipo detallado de la página de inicio de la aplicación. Se dispone de una cabecera con distintas opciones para el usuario como volver a la página de inicio, ver sus productos en propiedad o registrar un nuevo producto. En el apartado de título se muestra una barra de búsqueda para poder filtrar el listado de productos que aparece en la parte inferior.

Figura 5.3: Página de inicio

En la Figura 5.4 se ha navegado al apartado de la cabecera “Mis Productos”. Se dispone, al igual que en la página de inicio, de una barra de búsqueda y un listado de los productos, todos los pertenecientes al usuario. Adicionalmente, se muestran dos botones en cada producto, uno de edición y otro de borrado del mismo.

Figura 5.4: Página de productos en propiedad

En la Figura 5.5 se ha navegado al apartado “Registrar Producto” de la cabecera. En esta página se muestra un formulario rellenable para el registro de un nuevo producto, donde se pueden importar imágenes y documentos, al igual que registrar características del artículo, como su nombre, descripción o precio.

Figura 5.5: Registro de un producto

En la Figura 5.6 se presenta la visualización de un producto al ser accedido por el usuario, este acceso se producirá al pulsar sobre uno de los productos dentro de los listados vistos en las figuras 5.3 y 5.4.

En la parte superior se dispone de las imágenes del producto junto a la información registrada del mismo. También se muestra el botón "Transferir", que dará acción a la transferencia de ese producto a la propiedad del usuario. Destacar que este botón no estará disponible para el propietario actual del producto.

En la parte inferior, se dispone una línea cronológica con los eventos sufridos por el producto desde su dada de alta en la aplicación. Cada evento muestra un título indicando el tipo de evento, la fecha en la que se realizó y un botón "Ver detalles" que mostrará todos los detalles respectivos a ese evento, como las modificaciones realizadas en caso de un evento de edición.

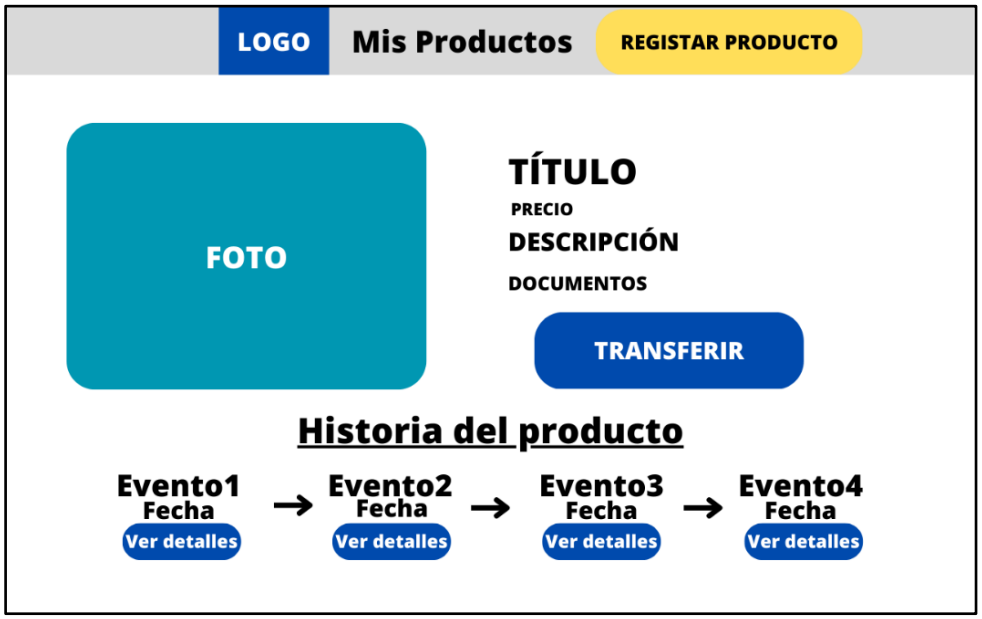


Figura 5.6: Visualización de un producto

Capítulo 6

Implementación

En este capítulo se proporciona una descripción detallada de la estructura general del código en la aplicación, así como de las herramientas utilizadas en su desarrollo.

6.1. Herramientas de desarrollo

Para el desarrollo de la aplicación se ha utilizado Visual Studio Code, un editor de código ampliamente utilizado. Este entorno ha sido utilizado en un equipo con sistema operativo Windows 11.

6.2. Tecnologías utilizadas

Además de las herramientas relacionadas con la tecnología blockchain, ya mencionadas en el segundo capítulo del presente documento (Ver 2.2.), se proceden a explicar el resto de herramientas utilizadas en el desarrollo de este proyecto.

6.2.1. React

De acuerdo con Thakkar (2020), React es una biblioteca de JavaScript de código abierto desarrollada por la actual empresa tecnológica Meta Open Source. Se utiliza para construir interfaces de usuario interactivas y reactivas, basándose en el concepto de componentes, bloques de construcción reutilizables que encapsulan el comportamiento y la presentación de una parte específica de la interfaz de usuario.

Los componentes poseen un ciclo de vida que consta de diferentes etapas, desde su creación hasta su eliminación. Durante estas etapas, estos pueden realizar acciones específicas y responder a cambios en su estado o propiedades. Para gestionar su estado y ciclo de vida, se utilizan los Hooks, funciones especiales que permiten utilizar características de React en componentes funcionales.

Según Meta Open Source (2023), uno de los Hooks más utilizados es useState. Este permite añadir estado a un componente funcional. Al llamar a useState con un valor inicial, se devuelve una tupla que contiene el estado actual y una función para actualizarlo, así los componentes funcionales pueden mantener y modificar su estado interno.

Otro Hook fundamental es useEffect, el cual se utiliza para realizar efectos secundarios en un componente, como la suscripción a eventos, llamadas a API o actualizaciones del DOM, entre otros. Este recibe una función de efecto y se ejecuta después de que el componente se haya renderizado. También puede aceptar una lista de dependencias, que indica a React que debe volver a ejecutar el efecto si alguna de esas dependencias cambia.

6.2.2. Node.js

Node.js es un entorno de ejecución de JavaScript basado en el motor V8 de Google Chrome. A diferencia de otros entornos de ejecución de JavaScript que se limitan a ejecutar código en el navegador, Node.js permite ejecutar código JavaScript en el servidor. Esto lo convierte en una herramienta poderosa para construir aplicaciones web escalables y de alto rendimiento (OpenJS Foundation, 2023).

Una de las ventajas clave de Node.js es su capacidad para manejar una gran cantidad de conexiones concurrentes de manera eficiente. En contraposición con los servidores tradicionales basados en hilos, que pueden ser costosos en términos de memoria y rendimiento, Node.js utiliza un modelo de E/S no bloqueante y orientado a eventos. Esto significa que en lugar de esperar a que una operación de E/S se complete, puede continuar ejecutando otras tareas y manejar múltiples solicitudes simultáneamente. Esto hace que sea especialmente adecuado para aplicaciones web en tiempo real.

6.2.3. npm

npm es el administrador de paquetes predeterminado para el entorno de ejecución de JavaScript basado en Node.js. Proporciona una amplia gama de paquetes y bibliotecas de código abierto que los desarrolladores pueden utilizar en sus proyectos. Con más de un millón de paquetes disponibles, npm se ha convertido en el ecosistema central para compartir y distribuir código JavaScript (npm, Inc., 2023).

Cuando trabajamos con React, esta herramienta se utiliza principalmente para gestionar las dependencias de un proyecto y facilitar la instalación de bibliotecas y módulos adicionales necesarios para el desarrollo. Al crear una nueva aplicación, npm es fundamental para instalar y mantener las dependencias clave, como React y ReactDOM.

Además de las dependencias principales, se puede utilizar para instalar otras bibliotecas y paquetes que proporcionen características adicionales y funcionalidad específica para la aplicación. En el presente proyecto se han incluido nuevas bibliotecas npm cuya utilidad se describirá a continuación:

- **React-router-dom**: Proporciona herramientas de enrutamiento. Permite crear aplicaciones de página única (SPA) con múltiples vistas y navegación sin necesidad de recargar la página completa.
- **Web3-utils**: Suministra métodos y herramientas para el manejo de direcciones Ethereum, la conversión de valores en unidades wei o Ether, y otras funcionalidades esenciales para el desarrollo en el entorno blockchain de Ethereum.
- **Ipfs-http-client**: Simplifica la integración de aplicaciones con la red IPFS, como veremos en el siguiente punto (Ver 6.2.5.), proporciona una interfaz de programación de aplicaciones (API) para interactuar con el protocolo a través de HTTP.
- **React-bootstrap**: Provee una colección de componentes de interfaz de usuario preestilizados y responsivos para facilitar la creación de interfaces.
- **React-icons/fa**: Permite importar y utilizar iconos como componentes para integrarlos en la interfaz de usuario sin la necesidad de cargar recursos adicionales.

6.2.4. Web3JS

Web3.js es una biblioteca de JavaScript que proporciona una API para interactuar con la red Ethereum. Esta biblioteca es esencial para el desarrollo de dApps que utilizan la tecnología de blockchain de Ethereum.

Dentro de una aplicación descentralizada, se utiliza para conectar y comunicarse con un nodo de Ethereum, cliente que se conecta a la red, y almacena una copia completa o parcial de la cadena de bloques. De esta forma, permite establecer esta conexión y proporcionar una serie de métodos y funciones para interactuar con contratos inteligentes.

Además de esta interacción, también facilita el manejo de billeteras y la firma de transacciones. Proporciona métodos para acceder a la billetera del usuario y firmar transacciones de forma segura, lo que garantiza que las transacciones enviadas a la cadena de bloques sean válidas y autorizadas por el usuario (ChainSafe Systems, 2023).

6.2.5. IPFS

De acuerdo con Infura (2023), el InterPlanetary File System (IPFS) es un protocolo de comunicación y sistema de archivos distribuido diseñado para transformar la manera en que se almacenan, distribuyen y acceden a los archivos en Internet. Con su enfoque descentralizado y basado en contenido, ofrece una solución innovadora para el almacenamiento y recuperación de elementos en la red.

Dentro de la red IPFS, los archivos se almacenan y se acceden a través de hashes criptográficos únicos, lo que garantiza la integridad de los datos. Cuando se añade un nuevo archivo, este se divide en bloques más pequeños y se le asigna un hash único basado en su contenido.

6.2.6. Jira

Jira es una plataforma de gestión de proyectos desarrollada por Atlassian basada en metodología ágil, como Scrum. Facilita la planificación de flujos de trabajo al permitir la creación de conjuntos de tareas a partir de historias de usuarios, las cuales son asignadas a los miembros del equipo. Además, la herramienta permite el seguimiento de problemas o incidencias que puedan surgir en estas tareas, como incumplimientos u otras dificultades.

De esta forma, esta herramienta ha sido de gran utilidad al haber permitido organizar y priorizar las historias de usuario asignadas a cada sprint. Cada historia se ha transformado en una serie de tareas necesarias para lograr la funcionalidad objetivo del proyecto.

6.2.7. Astah

Astah es una herramienta de modelado ampliamente utilizada en el campo de la ingeniería de software y otros sectores relacionados. Su enfoque principal se centra en la creación y representación de diagramas UML, lo que permite a los profesionales de la ingeniería de software analizar y visualizar de manera efectiva los sistemas y procesos.

La utilización de esta herramienta ha sido fundamental en el desarrollo de diagramas UML. Gracias a estos, ha sido posible documentar los requisitos y la estructura de este sistema en el capítulo Análisis presente en el documento (Ver Cap. 4.).

6.2.8. Control de versiones

El control de versiones se ha llevado a cabo utilizando la herramienta Git, así como la plataforma GitLab proporcionada por la empresa tecnológica HP para almacenar las versiones de forma online y compartir las nuevas actualizaciones con los tutores de empresa del autor.

Git ofrece una solución sencilla y conveniente para el control de versiones, permitiendo la creación de ramas de desarrollo separadas para cada funcionalidad del software. Esto permite un enfoque modular, donde cada rama se dedica a una característica específica y se pueden fusionar los cambios de manera controlada (Software Freedom Conservancy, 2023).

6.3. Implementación de la red blockchain

La implementación de la red blockchain viene dada por la herramienta Ganache, presentada en el segundo capítulo de esta memoria (Ver 2.2.5.). Como vimos en sus beneficios, ha sido elegida por facilitar la configuración de una red blockchain local personalizada, permitiendo mayor flexibilidad y control sobre el entorno de desarrollo. Además, su capacidad para emular transacciones y el estado de la red de manera rápida y segura ha agilizado el proceso de desarrollo.

6.4. Implementación del smart contract

En este apartado se explicará la estructura general del contrato inteligente y todos los componentes involucrados en su correcto funcionamiento.

6.4.1. Estructuras y mapeos

```
pragma solidity ^0.8.12;

contract StoreProduct{
    struct Producto {
        uint id;
        string[] imagenes;
        string[] documentos;
        string nombre;
        string descripcion;
        uint256 precio;
    }

    Producto[] public productos; //Almacena todos los productos de la aplicación

    mapping (uint => address) public productOwner; //Mapeo que relaciona el identificador del producto con la dirección de su propietario actual

    mapping (address => uint) private ownerCount; //Mapeo que relaciona la dirección de un propientario con el número de productos que posee

    mapping (uint => bool) private activeProducts; // Mapeo que indica si un producto está activo o eliminado
```

Figura 6.1: Estructuras y mapeos del smart contract

Dentro del contrato StoreProduct, se define una estructura llamada “Producto” que representa las características de un producto. A continuación, se explicará cada uno de sus atributos:

- **uint id:** Identificador único para cada producto.
- **string[] imagenes:** Matriz dinámica de cadenas que almacena los hashes de acceso en IPFS a las imágenes relacionadas con el producto.
- **string[] documentos:** Matriz dinámica de cadenas que almacena hashes de acceso en IPFS a los documentos relacionados con el producto.

- **string nombre:** Nombre del producto.
- **string descripcion:** Descripción del producto.
- **uint256 precio:** Precio del producto.

Posterior a la estructura de datos, se declaran los diferentes mapeos del contrato:

- **Producto [] public productos:** Declara una matriz pública que almacena todos los productos de la aplicación.
- **mapping (uint => address) public productOwner:** Define un mapeo que relaciona el identificador de un producto con la dirección del propietario actual. Así, permite rastrear quién es el propietario de cada producto.
- **mapping (address => uint) private ownerCount:** Establece un mapeo privado que relaciona la dirección de un propietario con el número de productos que posee. De esta forma, permite contar cuántos productos tiene cada propietario.
- **mapping (uint => bool) private activeProducts:** Establece un mapeo que indica si un producto está activo o eliminado. Utiliza el identificador del producto como clave y un valor booleano (true o false) para indicar si el producto está activo o eliminado.

6.4.2. Eventos

Un evento dentro de un smart contract es un mecanismo que permite comunicar y registrar información relevante en la cadena de bloques. En este contrato, permiten a los usuarios y aplicaciones que interactúan con el contrato mantener un historial de los sucesos ocurridos en relación con los productos, lo que brinda transparencia y trazabilidad tanto en la red como en la propia interfaz de la aplicación como veremos en el siguiente apartado (Ver 6.5.5.).

Se han implementado cuatro tipos de eventos, que se relacionan con las funcionalidades disponibles para los usuarios en relación con los productos. Estos eventos son: “NewProduct” (Nuevo Producto), “ProductModified” (Producto Modificado), “ProductTransferred” (Producto Transferido) y “DeletedProduct” (Producto Eliminado). Cada uno de ellos almacena información relevante del hecho específico, incluyendo siempre el identificador del producto y la hora y fecha exactas en la que sucedió.

Cabe destacar, que el evento emitido al eliminar un producto “DeletedProduct” no posee uso en esta versión de la aplicación. Su emisión está pensada para ser utilizada en líneas de trabajo futuro de la misma (Ver 8.2.).

6.4.3. Métodos y funciones

A continuación, se explicará el funcionamiento y utilidad de todas los métodos y funciones implementadas en el contrato inteligente:

- **createProduct(imagenesHash, documentosHash, nombre, descripción, precio):** Crea un nuevo objeto producto y lo agrega a la lista de productos, comprobando que el precio sea superior a cero. Al crearlo, emite un nuevo evento “NewProduct”.
- **getProduct(productoID):** Devuelve un producto a partir de su identificador, en caso de que exista y esté activo.
- **getProducts():** Devuelve una lista con todos los productos activos.
- **getProductsByOwner(direccionDueño):** Devuelve una lista con todos los productos activos pertenecientes a una dirección ETH.

- **deleteProduct(productoID):** Elimina un producto a partir de su identificador, en caso de que exista y esté activo. Al eliminarlo, emite un nuevo evento "DeletedProduct".
- **modifyProduct(productoID, nuevoNombre, nuevaDescripción, nuevoPrecio):** Actualiza los atributos nombre, descripción y precio de un producto a través de su identificador, comprobando que exista y el precio sea superior a cero. Al modificarlo, emite un nuevo evento "ProductModified".
- **transferProduct(productoID, direccionNuevoDueño):** Realiza el cambio de propietario de un producto a una nueva dirección ETH, comprobando que exista, la dirección no coincida con su propietario actual y la cantidad transferida sea igual o superior al precio actual. Al transferirlo, emite un nuevo evento "ProductTransferred".

6.5. Implementación del cliente

En este apartado se explicará la estructura general y partes relevantes de su funcionamiento dentro de la parte cliente de la aplicación. Como se ha explicado anteriormente, su implementación se ha hecho uso de la biblioteca React.

6.5.1. Carga inicial de la aplicación

El archivo App.js define el componente principal de la aplicación que interactúa con la red Ethereum a través del contrato inteligente. Administra el estado, las funciones y las rutas de la aplicación, y renderiza los diferentes componentes según la URL actual. También se conecta a la extensión MetaMask para obtener la cuenta del usuario que interactúa con la aplicación. A continuación, se explicarán los procedimientos dentro del hook useEffect de este componente que presenta la carga inicial de la aplicación:

```
useEffect(() => {
  async function load() {
    if (window.ethereum) {
      window.web3 = new Web3(window.ethereum);
      await window.ethereum.enable();
    } else if (window.web3) {
      window.web3 = new Web3(window.web3.currentProvider);
    } else {
      window.alert(
        "No se detectó ningún proveedor de Ethereum. Por favor, instala MetaMask
      );
    }

    const web3 = window.web3;
    const accounts = await web3.eth.requestAccounts();
    const ownerAddress = accounts[0];
    const storeProductContract = await getContract(web3);
    const allProducts = await storeProductContract.methods
      .getProducts()
      .call();
    const ownProducts = await storeProductContract.methods
      .getProductsByOwner(ownerAddress)
      .call();

    setWeb3(web3);
    setAccount(ownerAddress);
    setStoreProductContract(storeProductContract);
    setAllProducts(allProducts);
    setProducts(
      allProducts.filter(
        (product) =>
          !ownProducts.some((otherProduct) => otherProduct.id === product.id)
      )
    );
    setOwnerProducts(ownProducts);

    window.ethereum.on("accountsChanged", (newAccounts) => {
      console.log("Cambio de cuenta detectado:", newAccounts[0]);
      setAccount(newAccounts[0]);
      load();
    });
  }

  load();
}, []);
```

Figura 6.2: Hook useEffect en App.js

El bloque de código dentro de useEffect se ejecuta cuando el componente App se monta. Dentro de este hook encontramos una sola función denominada “load”, que será la encargada de realizar la carga inicial de todos los datos necesarios para comenzar a interactuar con la aplicación.

En primer lugar, la función verifica si window.ethereum está disponible para interactuar con la extensión MetaMask de Ethereum. Si lo está, crea una instancia de Web3 y se habilita la cuenta del usuario. En caso de no estarlo, saltará una alerta pidiendo al usuario que instale la extensión Metamask en su navegador.

Una vez obtenida la dirección de la cuenta del usuario y la instancia del contrato StoreProduct, se obtienen dos listas: una con todos los productos existentes y otra con aquellos pertenecientes a la dirección del usuario conectado. Tras la llamada a estos métodos, se filtra la lista de todos los productos existentes para eliminar aquellos que ya pertenecen al propietario.

Finalmente, se actualizan las variables de estado con los datos obtenidos y se establece un evento de escucha para detectar cambios de cuenta en MetaMask. En caso de cambio de cuenta, se volverá a ejecutar la función “load”, de forma que se actualizarán todos los datos anteriores con la nueva dirección de cuenta.

6.5.2. Enrutamiento de la aplicación

```
<BrowserRouter basename="/">
  <Routes>
    <Route path="/" element={<ProductsList products={products} />} />
    <Route path="*" element={<ProductsList products={products} />} />
    <Route
      path="/registro"
      element={<Register handleCreateProduct={handleCreateProduct} />
    />
    <Route path="/sobreNosotros" element={<AboutUs />} />
    <Route
      path="/misProductos"
      element={
        <ProductsList
          products={ownerProducts}
          isOwner={true}
          handleDeleteProduct={handleDeleteProduct}
          handleEditProduct={handleEditProduct}
        />
      }
    />
    <Route
      path="/producto/:id"
      element={
        <ShowProduct
          products={allProducts}
          account={account}
          ownIDs={ownerProducts.map((producto) => producto.id)}
          handleGetEvents={handleGetEvents}
          handleTransferProduct={handleTransferProduct}
        />
      }
    />
  </Routes>
</BrowserRouter>
```

Figura 6.3: Enrutamiento en App.js

Dentro del archivo App.js, se utiliza la biblioteca “react-router-dom” (Ver 6.2.3.) para configurar el enrutamiento de la aplicación. Cada ruta definida utilizando el componente Route renderizará un componente específico según la URL actual. Los componentes ProductsList, Register, AboutUs, y ShowProduct, que vimos en el capítulo 5 (Ver 5.2.2.) se pasan como elementos secundarios de las rutas para su renderizado. Al mismo tiempo, se envían varias propiedades y funciones a estos componentes para permitir la interacción y visualización de los datos dentro de ellos. En este caso, se definen las siguientes rutas y su funcionalidad:

- **Ruta raíz (“/”):** Renderiza el componente ProductsList con la lista de todos los productos no pertenecientes a la dirección de cuenta establecida como argumento. Será la página de inicio de la aplicación.
- **Ruta comodín (“*”):** Esta ruta captura cualquier URL que no coincida con las rutas establecidas y navega a la ruta raíz, redirigiendo a la página de inicio.

- **Ruta "/registro":** Renderiza el componente Register con la función "handleCreateProduct" como argumento para la creación de los nuevos productos.
- **Ruta "/sobreNosotros":** Renderiza el componente AboutUs.
- **Ruta "/misProductos":** Renderiza el componente ProductsList con la lista de los productos pertenecientes a la dirección de cuenta establecida como argumento. Del mismo modo, se pasan las funciones "handleEditProduct" y "handleDeleteProduct" para su edición y eliminación respectivamente.
- **Ruta "/producto/:id":** Renderiza el componente ShowProduct con varios argumentos para la correcta visualización de la información del producto, como la lista de todos los productos para poder comprobar su existencia a partir del identificador dentro de la URL. También se envía una lista con los identificadores de los productos en propiedad del usuario para comprobar si el especificado pertenece, y las funciones "handleGetEvents" y "handleTransferProduct" para obtener los eventos respectivos a ese producto y poder transferirlo a otro propietario respectivamente.

6.5.3. Registro de un producto

El componente Register es el formulario de registro de nuevos productos, permite al usuario ingresar información sobre un producto, como imágenes, archivos, nombre, descripción y precio (Ver Figura 6.4.). Tras ello, se procesa la información ingresada y se envía como argumentos al método "handleCreateProduct" para su posterior creación.

Figura 6.4: Formulario de registro

El formulario presenta una serie de limitaciones en la información registrada por el usuario, como son un máximo de cinco imágenes (.png, .jpg, .jpeg) y documentos (.pdf), un límite de 50 y 500 caracteres en el nombre y descripción respectivamente, y un precio cuyo valor varía entre 0,00001 hasta 999.99999 ETH.

Una vez rellenados todos los campos y pulsado el botón "Registrar producto", se obtienen los hashes de acceso en la red IPFS a las imágenes y documentos registrados. Esto se consigue inicializando una instancia de este protocolo utilizando la biblioteca "ipfs-http-client" y se crea un objeto File a partir del blob e información del archivo original. Con todo esto, se agrega el archivo a la red IPFS utilizando la llamada "ipfs.add". El hash devuelto será el almacenado posteriormente en el registro del producto.

Cabe destacar que dentro del método "handleCreateProduct", el argumento precio es convertido a su valor equivalente en unidades wei (la unidad más pequeña de Ethereum) mediante la función "Web3Utils.toWei()" para poder almacenar precios con valores decimales.

6.5.4. Obtención de documentos e imágenes

Los métodos de obtención de documentos e imágenes son llamados desde el hook useEffect del componente ShowProduct, donde se carga y visualiza toda la información específica de un producto. En estos métodos se hace de nuevo uso de una instancia de la biblioteca "ipfs-http-client", con la que se llama a la función "ipfs.cat" para obtener el contenido del archivo a partir de su hash. Este contenido se devuelve en formato buffer.

En caso de las imágenes, el buffer se transforma en una cadena en formato base64 para poder ser mostrada posteriormente en la interfaz. De forma similar, mediante el buffer de cada documento se crea un archivo Blob en formato pdf que luego será utilizo para descargarlo. También se establece un nombre al documento del tipo "AdjuntoX", donde X es el número del documento adjunto más uno.

6.5.5. Obtención de eventos

```
async function handleGetEvents(productID, tipoEvento) {
  try {
    const events = await storeProductContract.getPastEvents(tipoEvento, {
      filter: { id: productID },
      fromBlock: 0,
      toBlock: "latest",
    });
    return events;
  } catch (e) {
    console.error("Error al obtener los eventos: " + e);
  }
}
```

Figura 6.5: Función handleGetEvents en App.js

Además de las funciones ya mencionadas en el apartado Implementación del smart contract (Ver 6.4.3.), en la función asíncrona handleGetEvents mostrada en la Figura 6.5, se hace uso de un método no implementado, "getPastEvents", utilizado para obtener los eventos emitidos dentro del contrato storeProduct. Como vemos, recibe los eventos relacionados con el producto en base a su identificador y el tipo de evento ("ProductTransferred", "NewProduct", "ProductModified").

```
  if (modifyEvents) {
    modifyModifications = modifyEvents.map((event) => {
      const fecha = new Date(event.returnValues[7] * 1000);
      fecha.setHours(0, 0, 0);
      return {
        fecha,
        hora: new Date(event.returnValues[7] * 1000).toLocaleTimeString(),
        descripcionEvento: "Modificación",
        titulo: event.returnValues[3],
        descripcion: event.returnValues[4],
        precio: event.returnValues[5] / 10 ** 18,
        propietario: event.returnValues[6],
      };
    });
  }

  const allModifications = newProductModifications
    .concat(transferModifications)
    .concat(modifyModifications);

  allModifications.sort((a, b) => {
    const dateA = a.fecha.toISOString();
    const dateB = b.fecha.toISOString();

    const timeA = a.fecha.toLocaleTimeString();
    const timeB = b.fecha.toLocaleTimeString();

    if (dateA === dateB) {
      return timeA.localeCompare(timeB);
    } else {
      return dateA.localeCompare(dateB);
    }
  });
  setProductModifications(allModifications);
}
```

Figura 6.6: Fragmento función getProductModifications en ShowProduct.js

Una vez obtenidos todos los eventos dentro de la función “getProductsModifications”, mostrada en la Figura 6.5, se comprueba el tipo de cada evento y se mapea para crear objetos modificaciones con la información relevante al tipo de evento. Estos objetos contienen propiedades como la fecha y hora de emisión, la descripción del evento o el propietario del producto, entre otras. Finalmente, se concatenan todas las listas de eventos y se ordenan en base a su fecha y hora de forma ascendente.

Figura 6.7: Historia de un producto

De esta forma, se obtiene una línea cronológica de todos los eventos emitidos para ese producto, que posteriormente será visualizada en el cliente, como se aprecia en la Figura 6.6. También es posible filtrar la historia del producto por uno o varios tipos de evento y la fecha de emisión, al igual ver sus detalles pulsando en cada uno de los eventos.

Todas las vistas y funcionalidades adicionales no mencionadas en este capítulo están recogidas en el Manual de usuario del presente documento (Ver Apéndice B).

Capítulo 7

Pruebas

En el presente capítulo se exponen las pruebas efectuadas para verificar el correcto funcionamiento del sistema. Aunque dichas pruebas se llevaron a cabo durante todo el proceso de implementación, en este capítulo se presentan aquellas que revisten una relevancia especial en términos de funcionamiento, así como sus resultados finales.

Estos tests se dividen en dos categorías: las pruebas unitarias realizadas sobre el contrato inteligente y las pruebas de aceptación realizadas en el cliente.

7.1. Pruebas sobre el smart contract

Para la validación de la correcta funcionalidad del contrato inteligente se ha construido el archivo storeProduct.js dentro de la carpeta "test" del proyecto. Este archivo apoyado en la herramienta Truffle nos permite mediante el comando ***truffle test*** ejecutar las 22 pruebas sobre el smart contract de forma automática.

Al desplegarse el test se realizan las pruebas unitarias para cada función del contrato. A continuación, se presentan cada una de estas pruebas explicando su funcionamiento, la salida esperada y su salida obtenida (Ver Tablas 7.1 – 7.22):

Psc01	Crear nuevo producto
Descripción	Comprueba la correcta creación de un nuevo producto
Resultado esperado	El producto creado posee los valores introducidos
Resultado obtenido	Resultado esperado

Tabla 7.1: Prueba sobre el contrato - Crear nuevo producto

Psc02	Crear nuevo producto vacío
Descripción	Comprueba si se puede crear un nuevo producto sin proporcionar imágenes, documentos, nombre o descripción
Resultado esperado	El producto creado posee los valores introducidos y se ha emitido el evento "NewProduct" correctamente
Resultado obtenido	Resultado esperado

Tabla 7.2: Prueba sobre el contrato - Crear nuevo producto vacío

Psc03	Error al crear producto con precio cero
Descripción	Comprueba si se produce un error al intentar crear un producto con un precio igual a cero
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.3: Prueba sobre el contrato - Error al crear producto con precio cero

Psc04	Obtener todos los productos
Descripción	Comprueba si se pueden obtener todos los productos almacenados en el contrato
Resultado esperado	El número de productos devueltos coincide con la cantidad esperada
Resultado obtenido	Resultado esperado

Tabla 7.4: Prueba sobre el contrato - Obtener todos los productos

Psc05	Error al obtener producto no existente
Descripción	Comprueba si se produce un error al intentar obtener un producto que no existe
Resultado esperado	Se lanza una excepción con el mensaje de error correcto.
Resultado obtenido	Resultado esperado

Tabla 7.5: Prueba sobre el contrato - Error al obtener producto no existente

Psc06	Obtener todos los productos de una dirección
Descripción	Comprueba si se pueden obtener todos los productos de una dirección específica
Resultado esperado	El número de productos devueltos coincide con la cantidad esperada
Resultado obtenido	Resultado esperado

Tabla 7.6: Prueba sobre el contrato - Obtener todos los productos de una dirección

Psc07	Obtener productos de una dirección sin productos
Descripción	Comprueba si se pueden obtener los productos de una dirección que no tiene ningún producto
Resultado esperado	La lista de productos devuelta está vacía
Resultado obtenido	Resultado esperado

Tabla 7.7: Prueba sobre el contrato - Obtener productos de una dirección sin productos

Psc08	Eliminar un producto
Descripción	Comprueba si se puede eliminar un producto existente.
Resultado esperado	El número de productos de la dirección del propietario se actualiza correctamente
Resultado obtenido	Resultado esperado

Tabla 7.8: Prueba sobre el contrato - Eliminar un producto

Psc09	Error al obtener producto eliminado
Descripción	Comprueba si se produce un error al intentar obtener un producto eliminado
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.9: Prueba sobre el contrato - Error al obtener producto eliminado

Psc10	Error al eliminar producto no existente
Descripción	Comprueba si se produce un error al intentar eliminar un producto que no existe
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.10: Prueba sobre el contrato - Error al eliminar producto no existente

Psc11	Error al eliminar producto eliminado
Descripción	Comprueba si se produce un error al intentar eliminar un producto que ya ha sido eliminado
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.11: Prueba sobre el contrato - Error al eliminar producto eliminado

Psc12	Error al eliminar producto sin ser propietario
Descripción	Comprueba si se produce un error al intentar eliminar un producto sin ser el propietario actual
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.12: Prueba sobre el contrato - Error al eliminar producto sin ser propietario

Psc13	Modificar un producto
Descripción	Comprueba si se puede modificar un producto existente y se ha emitido el evento " ProductModified" correctamente
Resultado esperado	Los valores del producto modificado coinciden con los valores actualizados
Resultado obtenido	Resultado esperado

Tabla 7.13: Prueba sobre el contrato – Modificar un producto

Psc14	Error al modificar producto eliminado
Descripción	Comprueba si se puede modificar un producto que ha sido eliminado
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.14: Prueba sobre el contrato - Error al modificar producto eliminado

Psc15	Error al modificar producto no existente
Descripción	Comprueba si se puede modificar un producto que no existente
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.15: Prueba sobre el contrato - Error al crear producto no existente

Psc16	Error al modificar producto con precio cero
Descripción	Comprueba si se puede modificar un producto con un nuevo precio igual a cero
Resultado esperado	Se lanza una excepción con el mensaje de error correcto
Resultado obtenido	Resultado esperado

Tabla 7.16: Prueba sobre el contrato - Error al crear producto con precio cero

Psc17	Error al modificar producto sin ser propietario
Descripción	Comprueba si se puede modificar un producto sin ser propietario del mismo
Resultado esperado	Se lanza una excepción con el mensaje correcto
Resultado obtenido	Resultado esperado

Tabla 7.17: Prueba sobre el contrato - Error al crear producto sin ser propietario

Psc18	Transferir un producto
Descripción	Comprueba si se puede transferir la propiedad de un producto a otra cuenta y se ha emitido el evento "ProductTransferred" correctamente
Resultado esperado	Se ha realizado la transferencia correctamente
Resultado obtenido	Resultado esperado

Tabla 7.18: Prueba sobre el contrato – Transferir un producto

Psc19	Error al transferir producto eliminado
Descripción	Comprueba si se puede transferir un producto que ha sido eliminado
Resultado esperado	Se lanza una excepción con el mensaje correcto
Resultado obtenido	Resultado esperado

Tabla 7.19: Prueba sobre el contrato - Error al transferir producto eliminado

Psc20	Error al transferir producto no existente
Descripción	Comprueba si se puede transferir un producto no existente
Resultado esperado	Se lanza una excepción con el mensaje correcto
Resultado obtenido	Resultado esperado

Tabla 7.20: Prueba sobre el contrato - Error al transferir producto no existente

Psc21	Error al transferir producto al propietario actual
Descripción	Comprueba si se puede transferir un producto a su propietario actual.
Resultado esperado	Se lanza una excepción con el mensaje correcto
Resultado obtenido	Resultado esperado

Tabla 7.21: Prueba sobre el contrato - Error al transferir producto al propietario actual

Psc22	Error al transferir producto con cantidad insuficiente
Descripción	Comprueba si se puede transferir un producto con una cantidad insuficiente
Resultado esperado	Se lanza una excepción con el mensaje correcto
Resultado obtenido	Resultado esperado

Tabla 7.22: Prueba sobre el contrato - Error al crear producto con cantidad insuficiente

7.2. Pruebas sobre el cliente

Después de llevar a cabo las pruebas unitarias, se procede a analizar el comportamiento simultáneo de múltiples componentes que conforman el cliente junto con el funcionamiento del contrato. De esta forma, se verifica que la interacción entre los componentes se ajusta a las expectativas establecidas.

A continuación, se mostrarán varias tablas con todas las pruebas de alto nivel realizadas en el cliente de la aplicación, una vez establecida la conexión con la cuenta usuario:

7.2.1 Test de navegabilidad

En esta sección se presenta una tabla de las pruebas de navegabilidad una vez dentro de la aplicación, junto con su resultado esperado y su resultado obtenido.

N.º	Descripción	Entrada url	Resultado esperado	Resultado obtenido
P01	Usuario accede a la página de inicio	/	Redirección a la página de inicio	Resultado esperado
P07	Usuario accede a página de información	/sobreNosotros	Redirección a la página de información	Resultado esperado
P02	Usuario accede a la página de registro de productos	/registro	Redirección a la página de registro de productos	Resultado esperado
P03	Usuario accede a la página de productos en propiedad	/misProductos	Redirección a la página de productos en propiedad	Resultado esperado
P04	Usuario accede a la página de visualización de un producto existente	/producto/id	Redirección a la página de visualización el producto con id especificado	Resultado esperado
P05	Usuario accede a la página de visualización de un producto no existente o eliminado	/producto/id	Redirección a la página de inicio	Resultado esperado
P06	Usuario accede a cualquier url no contemplada	/*	Redirección a la página de inicio	Resultado esperado

Tabla 7.23: Pruebas sobre el cliente – Test de navegación

7.2.2 Test de registro de productos

En estas pruebas de nivel superior se busca evaluar la funcionalidad al momento de crear un nuevo producto y las limitaciones de los campos del formulario de registro.

N.º	Descripción	Resultado esperado	Resultado obtenido
P07	Usuario pulsa registrar con todos los campos vacíos	Mensaje de aviso “Completa este campo”	Resultado esperado
P08	Usuario pulsa registrar con el campo nombre vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P09	Usuario pulsa registrar con el campo descripción vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P10	Usuario pulsa registrar con campo el precio vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P11	Usuario pulsa registrar con un precio menor o igual a cero	Mensaje de aviso “El valor debe ser superior o igual a 0,00001”	Resultado esperado

P12	Usuario pulsa registrar con los campos imágenes y documentos vacíos	Notificación Metamask con la validación de la operación	Resultado esperado
P13	Usuario pulsa registrar con todos los campos rellenos	Notificación Metamask con la validación de la operación	Resultado esperado
P14	Usuario rechaza la operación de registro	Redirección a la página de productos en propiedad	Resultado esperado
P15	Usuario realiza el registro del producto	Redirección a la página de productos en propiedad actualizada	Resultado esperado

Tabla 7.24: Pruebas sobre el cliente – Test de registro de productos

7.2.3 Test de búsqueda, edición y eliminación de productos

En esta sección se presenta una lista de las pruebas de alto nivel llevadas a cabo para la búsqueda, edición y eliminación de los productos en propiedad.

N.º	Descripción	Resultado esperado	Resultado obtenido
P16	Usuario realiza la búsqueda de un nombre en la lista	Se actualiza la lista con los productos cuyos nombres contienen la sucesión de caracteres	Resultado esperado
P17	Usuario realiza la búsqueda de un nombre no coincidente en la lista	Aparece la lista de productos vacía	Resultado esperado
P18	Usuario pulsa el botón de edición de uno de sus productos	Ventana emergente con un formulario con la información actual	Resultado esperado
P19	Dentro de edición, pulsa “Guardar cambios” con el campo nombre vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P20	Dentro de edición, pulsa “Guardar cambios” con el campo descripción vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P21	Dentro de edición, pulsa “Guardar cambios” con campo el precio vacío	Mensaje de aviso “Completa este campo”	Resultado esperado
P22	Dentro de edición, pulsa “Guardar cambios” con un precio menor o igual a cero	Mensaje de aviso “El valor debe ser superior o igual a 0,00001”	Resultado esperado
P23	Dentro de edición, pulsa “Guardar cambios” con todos los campos rellenos	Notificación Metamask con la validación de la operación	Resultado esperado
P24	Usuario rechaza la operación de edición dentro de Metamask	Recarga de la página	Resultado esperado
P25	Usuario realiza la edición del producto dentro de Metamask	Recarga de la página con la lista actualizada	Resultado esperado
P26	Usuario pulsa el botón de eliminar de uno de sus productos	Alerta de validación de eliminación	Resultado esperado
P27	Usuario pulsa “cancelar” en la alerta de aviso de eliminación	Sin efecto	Resultado esperado
P28	Usuario pulsa “aceptar” en la alerta de aviso de eliminación	Notificación Metamask con la validación de la operación	Resultado esperado
P29	Usuario rechaza la operación de eliminación dentro de Metamask	Recarga de la página	Resultado esperado
P30	Usuario realiza acepta la operación de eliminación dentro de Metamask	Recarga de la página con la lista actualizada	Resultado esperado

Tabla 7.25: Pruebas sobre el cliente – Test de búsqueda, edición y eliminación de productos

7.2.4 Test de transferencia, descarga de documentos y filtros

En estos tests de alto nivel se pretende evaluar la funcionalidad al visualizar un producto, descargar sus documentos adjuntos, transferirlo y filtrar los eventos de su historia.

N.º	Descripción	Resultado esperado	Resultado obtenido
P31	Usuario accede a un producto en propiedad	Se visualizan todas las características del producto menos el botón "Realizar transacción"	Resultado esperado
P32	Usuario accede a un producto sin imágenes registradas	En la sección de las imágenes aparece una imagen de no imagen disponible	Resultado esperado
P33	Usuario accede a un producto sin documentos registrados	No aparece la sección de documentos adjuntos	Resultado esperado
P34	Usuario pulsa el botón "Descargar" de un documento adjunto	Se descarga un documento en formato .pdf cuyo contenido coincide con el documento registrado	Resultado esperado
P35	Usuario filtra por uno o varios tipos de evento existente	Se filtran los eventos de ese tipo	Resultado esperado
P36	Usuario filtra por uno o varios tipos de evento no existente	Aparece el mensaje "No se tiene información para este periodo"	Resultado esperado
P37	Usuario filtra por el calendario izquierdo una fecha	Aparecen todos los eventos con una fecha igual o posterior y el calendario derecho deshabilita las fechas anteriores a esta	Resultado esperado
P38	Usuario filtra por el calendario izquierdo una fecha posterior a la fecha del último evento	Aparece el mensaje "No se tiene información para este periodo" y el calendario derecho deshabilita las fechas anteriores a esta	Resultado esperado
P39	Usuario filtra por el calendario derecho una fecha	Aparecen todos los eventos con una fecha igual o anterior y el calendario izquierdo deshabilita las fechas posteriores a esta	Resultado esperado
P40	Usuario filtra por el calendario derecha una fecha anterior a la fecha del primer evento	Aparece el mensaje "No se tiene información para este periodo" y el calendario izquierdo deshabilita las fechas posteriores a esta	Resultado esperado
P41	Usuario filtra por ambos calendarios la misma fecha	Aparecen todos los eventos cuya fecha coincide con esta	Resultado esperado
P42	Usuario filtra por ambos calendarios un rango de fechas	Aparecen todos los eventos con una fecha perteneciente a ese rango	Resultado esperado
P43	Usuario filtra por ambos calendarios y por uno o varios tipos de fecha	Aparecen todos los eventos con una fecha perteneciente a ese rango y cuyo tipo coincida con alguno de los tipos filtrados	Resultado esperado
P44	Usuario pulsa el botón "Limpiar filtros"	Se eliminan los filtros establecidos y se restauran los filtros de tipo y fecha	No se restauran los filtros de fecha

Tabla 7.26: Pruebas sobre el cliente – Test de transferencia, descarga de documentos y filtros

Se concluyen todos los tests como válidos, incluyendo la prueba P44. Aunque el resultado esperado no se haya obtenido en su totalidad, la parte no resultante no se considera significativa.

7.2.5 Test de cambio de cuenta

En esta sección se presentan los tests de alto nivel utilizados para evaluar el cambio de cuenta de Metamask dentro de las páginas de la aplicación donde se debería producir un cambio en la información mostrada.

N.º	Descripción	Resultado esperado	Resultado obtenido
P45	Usuario cambia de cuenta en la página de inicio	Se recarga la página actualizando la lista de productos en venta	Resultado esperado
P46	Usuario cambia de cuenta en la página de productos en propiedad	Se recarga la página actualizando la lista de productos en propiedad	Resultado esperado
P47	Usuario cambia de cuenta en la página de visualización de un producto en propiedad de la nueva cuenta	Se recarga la página de visualización del producto sin el botón “Realizar transferencia”	Resultado esperado
P48	Usuario cambia de cuenta en la página de visualización de un producto en propiedad que no es propiedad de la nueva cuenta	Se recarga la página de visualización del producto añadiendo el botón “Realizar transferencia”	Resultado esperado

Tabla 7.27: Pruebas sobre el cliente – Test de cambio de cuenta

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

En relación al desarrollo del presente Trabajo de Fin de Grado, se han cumplido satisfactoriamente los objetivos planteados al inicio del mismo (Ver 1.1.3.), exceptuando el despliegue de la aplicación en una red de prueba Ethereum, que quedará como una de las líneas de trabajo futuro planteadas en el siguiente punto. De esta manera, este trabajo ha resultado en el completo desarrollo de una aplicación web descentralizada desplegada sobre una red blockchain local, la cual ha cumplido con los requisitos propuestos desde su inicio.

El desarrollo del proyecto se ha realizado de manera continua y en constante comunicación con los tutores, permitiendo que estos puedan verificar el adecuado progreso del mismo.

Durante la realización de este proyecto he podido aplicar los conocimientos adquiridos en las asignaturas durante estos cuatro años de carrera, sobre todo los adquiridos en asignaturas como Servicios y Sistemas Web, donde me inicié en el desarrollo de aplicaciones y servicios web, y aquellas que tratan sobre el desarrollo de proyectos y fases de diseño del mismo, tales como Modelado de Software, Diseño de Software y Planificación y Gestión de Proyectos. También ha sido de utilidad, para la elección de este proyecto, la asignatura de Análisis y Diseño de Bases de Datos, donde se ampliaron mis conocimientos sobre conceptos de la tecnología blockchain.

Espero que este proyecto sirva de utilidad a las próximas generaciones de alumnos de la escuela, para ayudarles a entender mejor los conceptos y beneficios que ofrece esta innovadora tecnología, que estoy seguro que veremos florecer en un futuro próximo.

8.2. Líneas de trabajo futuro

Aunque este proyecto ha finalizado satisfactoriamente, implementando la funcionalidad requerida por el Product Owner y autor del mismo, ante la limitación del tiempo es posible seguir mejorando este trabajo final tanto en sus funcionalidades como objetivo, lo que permitirá hacer de esta aplicación una herramienta más completa. Algunas de las posibles formas de seguir ampliando su utilidad son las siguientes:

- Permitir a los usuarios suscribirse a los eventos sobre un producto específico u otro usuario, de forma que pueda recibir notificaciones en tiempo real cuando se creen nuevos productos, se modifiquen, se transfieran o se eliminen.
- Desplegar la aplicación sobre una red de prueba Ethereum.
- Transformar la aplicación en un sistema con un backend intermediario que proporcione una API para que otras plataformas se conecten y registren sus productos en la blockchain.
- Permitir negociar con el propietario del producto, así los usuarios podrán comunicarse entre sí para discutir detalles de la transacción, hacer preguntas sobre los artículos o acordar los términos de la venta.
- Implementar la búsqueda de los productos mediante un código QR único para cada uno, que sea creado al ser registrado y se permita su descarga.

- Permitir establecer los artículos como no en venta para que no poder ser transferidos por otros usuarios.
- Permitir a los usuarios dejar calificaciones y comentarios sobre artículos adquiridos y sus vendedores.
- Permitir la edición de las imágenes y documentos pertenecientes a un producto, de manera que sea posible añadir, eliminar y cambiar el orden de visualización de los mismos.

Bibliografía

Cai, W., Wang, Z., Ernst, B., Hong, Z., Freng, C., & Leung, V. C. (2018). Decentralized Applications: The Blockchain-Empowered Software System. *IEEE Access, 6*, 53019-53033. doi:10.1109/ACCESS.2018.2870644

Canva. (s.f.). https://www.canva.com/

ChainSafe Systems. (2023). *Web3JS*. https://web3js.org/#/

Cryptocurrency wallet. (2023). *The Free Encyclopedia Wikipedia*. https://en.wikipedia.org/wiki/Cryptocurrency_wallet

Domínguez, P. (2018). El ciclo de producción en Scrum. *Welcomedevelopers*. https://welcomedevelopers.es/?s=scrum

Ethereum community. (5 de mayo de 2023). *Etherum*. https://ethereum.org/en/about/

Ethers v5. (2023). Documentation. *Creative Commons License* . https://docs.ethers.org/v5/

Feliu, J. (2018). Smart Contract. Concepto, ecosistema y principales cuestiones de Derecho privado. *La Ley mercantil*(47). https://dialnet.unirioja.es/servlet/articulo?codigo=6435058

IBM. (2023). *What is blockchain technology?* IBM: https://www.ibm.com/topics/blockchain

Infura. (2023). *Ipfs*. https://www.infura.io/product/ipfs

Jiménez Valbuena, J. A., Osorio Sanabria, M. A., & Maestre Góngora, G. P. (2021). Análisis del estado de adopción de la tecnología Blockchain en el sector Fintech y otras industrias. *Revista Cies*. http://revista.escolme.edu.co/index.php/cies/article/view/372

Lee, M. W. (2019). Using the MetaMask Chrome Extension. En *Beginning Ethereum Smart Contracts Programming* (págs. 93-126). Berkeley: Apress. https://doi.org/10.1007/978-1-4842-5086-0_5

Martinez, A. D. (2020). Consensys. *Diario Bitcoin*. https://www.diariobitcoin.com/empresas/consensys/

Meta Open Source. (2023). *React*. https://es.react.dev/

MetaMask. (2023). *Integrate with the MetaMask wallet.* Obtenido de MetaMask: https://docs.metamask.io/wallet/

Millán, V. (22 de marzo de 2022). Quién es Vitalik Buterin: el fundador de Ethereum nacido en Rusia que muchos ven como un genio. *ElEconomista*. https://www.bing.com/ck/a?!&&p=e801e3c5db47d9f7JmltdHM9MTY4MzY3NjgwMCZpZ3VpZD0xNDY1MDM5MC1kN2RkLTY3NDMtM2M4NC0xMDk3ZDY3NTY2NzQmaW5zaWQ9NTMzMg&ptn=3&hsh=3&fclid=14650390d7dd67433c841097d6756674&psq=Vitalik_Buterin&u=a1aHR0cHM6Ly93d3cuZWxlY29ub21pc3RhLm

Ministerio de Trabajo y economía social. (2023). *Anexo II. Tabla salarial inicial. Año 2023 (0,8%). [Repositorio de Boletín Oficial del Estado].* https://www.boe.es/boe/dias/2023/03/10/pdfs/BOE-A-2023-6347.pdf

npm, Inc. (2023). *npm*. https://www.npmjs.com/

OpenJS Foundation. (2023). *nodejs*. https://nodejs.org/es

Oracle. (26 de mayo de 2022). *Developer Resource Center*. Obtenido de React JS: https://developer.oracle.com/es/learn/technicalarticles/reactjs#:~:text=El%20DOM%20virtual%20(modelo%20de,componentes%20principales%20(o%20principales).

Pérez, I. (2022). Las User Stories en Scrum. https://www.linkedin.com/pulse/las-user-stories-en-scrum-ivan-p%C3%A9rez-torres/?trk=pulse-article_more-articles_related-content-card&originalSubdomain=es

RadhaKrishna, K. S., & Teja, M. C. (2018). Blockchain based Smart Contract deployment on Ethereum Platform using Web3.js and Solidity. *ICFAI Foundation for Higher Education*. https://www.researchgate.net/publication/356988224_Blockchain_based_Smart_Contract_deployment_on_Ethereum_Platform_using_Web3js_and_Solidity

Software Freedom Conservancy. (2023). *Git*. https://git-scm.com/

Stonis, M., Jain, T., & Pine, D. (11 de abril de 2022). Model-View-ViewModel (MVVM). *Microsoft Corporation*. https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

Thakkar , M. (2020). Introducing React.js. En *Building React Apps with Server-Side Rendering* (págs. 41-91). Apress, Berkeley, CA. doi:https://doi.org/10.1007/978-1-4842-5869-9_2

The Solidity Authors. (2023). *Solidity*. GitHub v0.8.19: https://docs.soliditylang.org/en/v0.8.19/

Truffle Suite. (2022). *Documentation*. Truffle Suite: https://trufflesuite.com/docs/

Turley, F., & Nader, K. (2019). *Los Fundamentos de Agile Scrum.* Van Haren Publishing. https://books.google.es/books?id=yX3DwAAQBAJ&lpg=PR4&ots=un3b7rcrVr&dq=Scrum%20&lr&hl=es&pg=PR4#v=onepage&q=Scrum&f=false

Type system. (2023). *The Free Encyclopedia Wikipedia*. https://en.wikipedia.org/wiki/Type_system#Static_typing

Vanzo, A. (2023). *Kant on Empiriscm and rationalism* (Vol. 30). https://www.jstor.org/stable/43488058

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). Blockchain Technology Overview. *NIST*, 1-66. https://doi.org/10.6028/NIST.IR.8202

Zhejiang University Press. (2020). Blockchain: Research and Applications. https://www.scopus.com/sourceid/21101101317

Zou, W., Lo, D., Kochhar, P. S., Le, X.-B. D., Xia, X., Feng, Y., . . . Xu, B. (2021). Smart Contract Development: Challenges and Opportunities. *IEEE Transactions on Software Engineering*. doi:10.1109/TSE.2019.2942301

Anexo A

Guía de instalación

A.1. Contenido del repositorio

El repositorio del proyecto se encuentra almacenado en el grupo de GitLab "HP-SCDS", donde se almacenan Trabajos de Fin de Grado del Observatorio de la empresa tecnológica HP. A través del nombre de repositorio "SecondHandChain", se encuentran los siguientes componentes:

- **Proyecto:** Directorio raíz del proyecto donde se encuentra el código fuente de la aplicación. En este se hallan diferentes directorios y documentos:
  - **build/contracts:** Directorio donde se almacena el contrato inteligente compilado en formato JSON.
  - **client:** Directorio donde se encuentra el código fuente del cliente.
  - **contracts:** Directorio que contiene el contrato inteligente escrito en lenguaje de programación Solidity.
  - **migrations:** Directorio que almacena archivos de migración utilizados para desplegar el contrato en la cadena de bloques.
  - **node_modules:** Directorio donde se almacenan las dependencias del proyecto.
  - **test:** Directorio donde se encuentran los archivos de prueba del contrato inteligente.
  - **truffle-config.js:** Archivo de configuración principal de Truffle.
- **Licencia:** Declaración de derechos de autor sobre el proyecto.
- **Guía de instalación:** Documento de copia de este Anexo.
- **Manual de usuario:** Documento de copia del Anexo B.

A.2. Instalación

En la presente sección se explica el procedimiento de instalación de las herramientas necesarias para ejecución la aplicación. Para la descarga del código fuente, es necesario acceder al repositorio para descargar y descomprimir el zip con todos los directorios.

Todos los pasos seguidos son explicados para la instalación en un equipo con sistema operativo Windows.

A.2.1. Instalación de NVM

Se procede a explicar cada paso del proceso de instalación de Node.js y npm, requerido para el funcionamiento de Truffle y React.

1. Descargar e instalar Node.js desde el sitio web oficial: https://nodejs.org/es/. Se recomienda la instalación de la última versión LTS (Long-Term Support).
2. Abrir una ventana de línea de comandos o terminal y verificar la correcta instalación de Node.js y npm, ejecutando los siguientes comandos:

```
node -v
npm -v
```

3. Instalar Truffle ejecutando el siguiente comando:

```
npm install -g truffle
```

4. Una vez finalizada la instalación, verificar que ha sido correcta ejecutando el siguiente comando:

```
truffle version
```

A.2.2. Instalación y configuración de Ganache

El siguiente paso es la instalación y configuración de la red local dentro de la herramienta Ganache.

1. Descargar e instalar Ganache desde el sitio web oficial: https://trufflesuite.com/ganache/.
2. Iniciar la aplicación y crear un nuevo espacio de trabajo, pulsando en “NEW WORKSPACE”.
3. Insertar el nombre deseado y agregar el archivo “Truffle-config.js” del proyecto (Ver A.1.), pulsando en el botón “ADD PROJECT”.

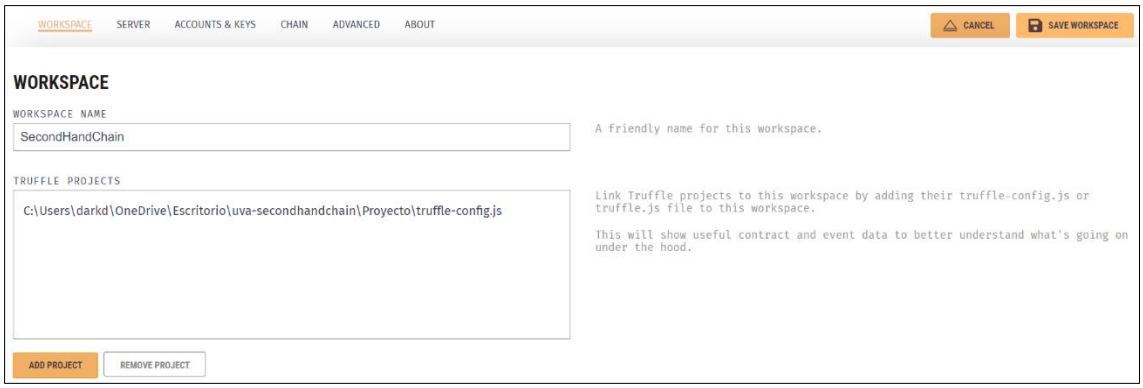


Figura A.1: Creación Workspace en Ganache

4. Pulsar sobre el botón “SAVE WORKSPACE” para guardar los cambios. De esta forma, se genera una nueva red local Ethereum con hasta diez cuentas con 100 ETH válidos dentro de esta red, para poder ser importadas posteriormente (Ver A.2.4).

A.2.3. Despliegue del contrato inteligente

Ahora se desarrollará el procedimiento de despliegue del smart contract dentro de la red local anteriormente creada.

1. Abrir una ventana de línea de comandos o terminal y navegar hasta el directorio raíz del proyecto, Proyect (Ver A.1.).

2. Ejecutar el siguiente comando para migrar el contrato inteligente a la red local de Ganache:

```
truffle migrate
```

3. Comprobar que el contrato está desplegado en la red, accediendo al apartado “CONTRACTS” dentro de Ganache.

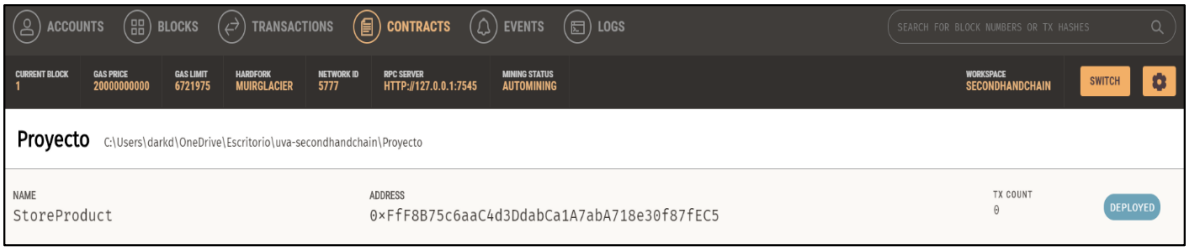


Figura A.2: Apartado de contratos en Ganache

4. Otra forma de comprobar el despliegue, es dentro de la ventana de línea de comandos o terminal el siguiente comando para verificar el correcto funcionamiento de todos los tests del contrato inteligente:

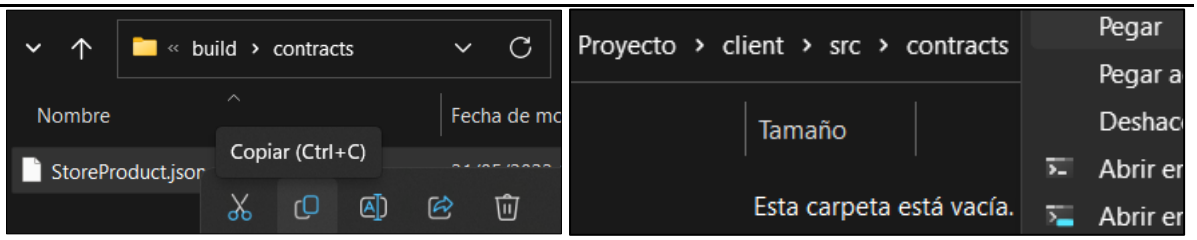
```
truffle test
```

```
  ✓ Error al crear producto con precio cero (194ms)
  ✓ Obtener todos los productos (148ms)
  ✓ Error al obtener producto no existente (87ms)
  ✓ Obtener todos los productos de una dirección (96ms)
  ✓ Obtener productos de una dirección sin productos (111ms)
  ✓ Eliminar un producto (495ms)
  ✓ Error al obtener producto eliminado (100ms)
  ✓ Error al eliminar producto no existente (114ms)
  ✓ Error al eliminar producto eliminado (99ms)
  ✓ Error al eliminar producto sin ser propietario (101ms)
  ✓ Modificar un producto (891ms)     ✓ Error al modificar producto
  ✓ Error al modificar producto no existente (115ms)
  ✓ Error al modificar producto con precio igual a cero (92ms)
  ✓ Error al moficar producto sin ser propietario (115ms)
  ✓ Transferir un producto (720ms)
  ✓ Error al transferir producto eliminado (83ms)
  ✓ Error al transferir producto no existente (108ms)
  ✓ Error al transferir producto al propietario actual (147ms)
  ✓ Error al transferir producto con cantidad insuficiente (126ms)


22 passing (7s)
```

Figura A.3: Resultado correcto de los tests

5. Una vez comprobado, se debe copiar el nuevo archivo creado “StoreProduct.json” almacenado en el directorio build/contracts (Ver A.1.) al directorio client/src/contracts. Esto se puede hacer dentro de un entorno de desarrollo o en el explorador de archivos de Windows.



Figuras A.4: Copiado y pegado del archivo StoreProduct.json

También es posible hacerlo mediante comandos. Por ejemplo, dentro de una ventana de línea de comandos o terminal, navegar hasta el directorio raíz del proyecto, Proyect (Ver A.1.) y ejecutar el siguiente comando para copiar y pegar el archivo descrito:

```
copy .\build\contracts\StoreProduct.json .\client\src\contracts\
```

En caso de ya existir otro archivo "StoreProduct.json" en el directorio client/src/contracts, simplemente sobrescribirlo con el nuevo generado en build/contracts.

```
¿Sobrescribir .\client\src\contracts\StoreProduct.json? (Sí/No/Todo): Sí
```

A.2.4. Instalación y configuración de Metamask

Una vez desplegado el contrato inteligente, se procede a instalar y configurar la extensión de navegador Metamask.

1. Instalar la extensión Metamask en un navegador Google Chrome o Firefox, a través de la tienda de extensiones o el sitio web oficial: https://metamask.io/.
2. Tras iniciar la extensión, crear una nueva cuenta o importar una, en caso de disponer de una cuenta existente.
3. Una vez dentro de la cuenta, pulsar sobre el apartado de "Configuración".

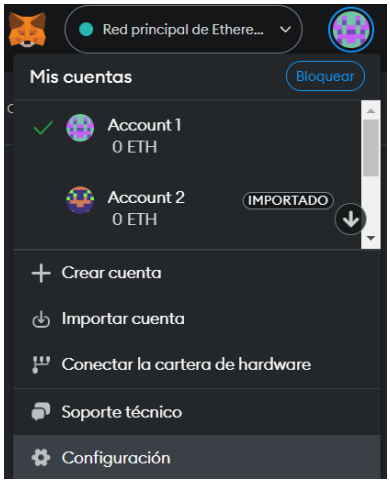


Figura A.5: Apartado configuración en Metamask

4. Dentro de la configuración, pulsar sobre "Avanzado", y activar la opción "Mostrar redes de prueba".

5. Salir de configuración, abrir en el desplegable de redes y pulsar sobre el botón “Agregar red”.
6. Dar a la opción inferior “Agregar una red manualmente” e introducir lo datos de la red local.

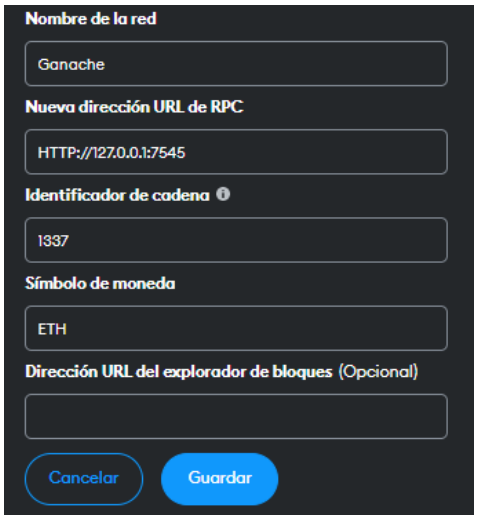


Figura A.6: Formulario registro red en Metamask

7. Rellenar el campo “Nueva dirección URL de RPC” con la información proporcionada de la red.

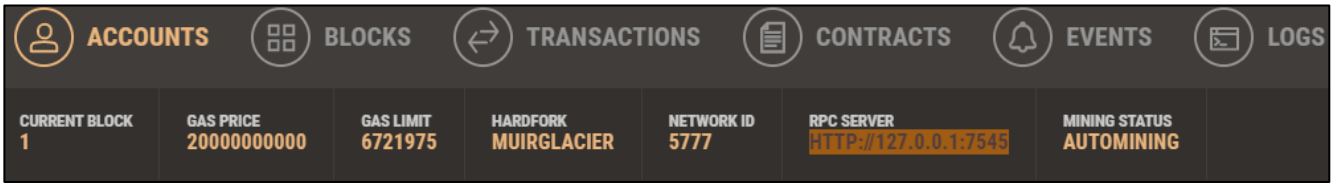


Figura A.7: Datos de la red local en Ganache

Una vez conectada la extensión Metamask con la red local, es posible importar nuevas cuentas para poder usar los 100 ETH ficticios que Ganache proporciona para cada cuenta de prueba.

8. Dentro del apartado “ACCOUNTS” de Ganache dar sobre el botón en forma de llave de cualquier cuenta, exceptuando la primera. Dado que esta es la utilizada para pagar los gastos de las transacciones al desplegar los contratos inteligentes.

Figura A.8: Información de una cuenta en Ganache

9. Copiar la clave privada de la cuenta.

Figura A.9: Clave privada de una cuenta en Ganache

10. Dentro de Metamask acceder a la opción “Importar cuenta” y agregar la clave privada anteriormente copiada.

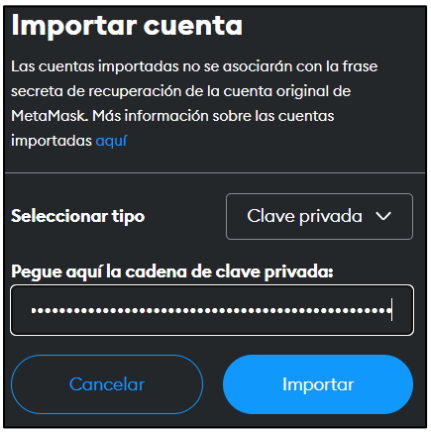


Figura A.10: Importación de cuenta en Metamask

Siguiendo estos últimos pasos, es posible agregar tantas cuentas como se desee para simular distintos usuarios reales dentro de la aplicación.

A.2.5. Ejecución y conexión

Tras la instalación y configuración de todas las herramientas necesarias para el correcto funcionamiento de la aplicación, se explicarán los procedimientos de ejecución de la misma.

1. Abrir una ventana de línea de comandos o terminal y navegar hasta el directorio Client del proyecto (ver A.1.).
2. Descargar las dependencias del proyecto React, ejecutando el siguiente comando:

```
npm install
```

3. En el mismo directorio, ejecutar el siguiente comando para iniciar la aplicación:

```
npm start
```

4. Tras unos segundos, se abrirá en el navegador la página de inicio de la aplicación. Es necesario conectar una de las cuentas importadas anteriormente (Ver A.2.4.) para comenzar a interactuar correctamente con la aplicación.

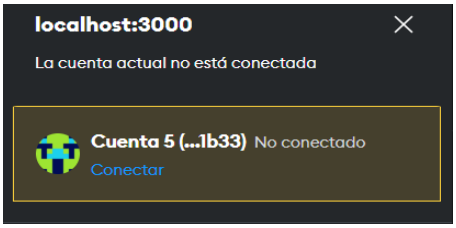


Figura A.11: Conexión de una cuenta en Metamask

Anexo B

Manual de usuario

En este apéndice se describirá el proceso de creación, búsqueda, modificación, eliminación y transferencia de productos desde el punto de vista de un usuario real. Así como todas las funcionalidades adicionales incorporadas en la página web. De esta forma, servirá como manual para cualquier usuario que desee utilizar la aplicación.

B.1. Página de inicio

Una vez conectada nuestra cuenta Metamask, aparecerá la pantalla de inicio de la aplicación (Ver Figura B.1.). En la parte superior se encuentra la cabecera de navegación con hasta cuatro botones, cuyas funcionalidades se describirá a continuación. Debajo de esta, se hayan las opciones de búsqueda de productos, por nombre y código QR; y en la parte inferior, aparece la lista de productos en venta, con información del título, descripción y precio de cada uno de los artículos. Al hacer clic sobre cualquiera de estos se navegará a la visualización de sus características (Ver B.4.).

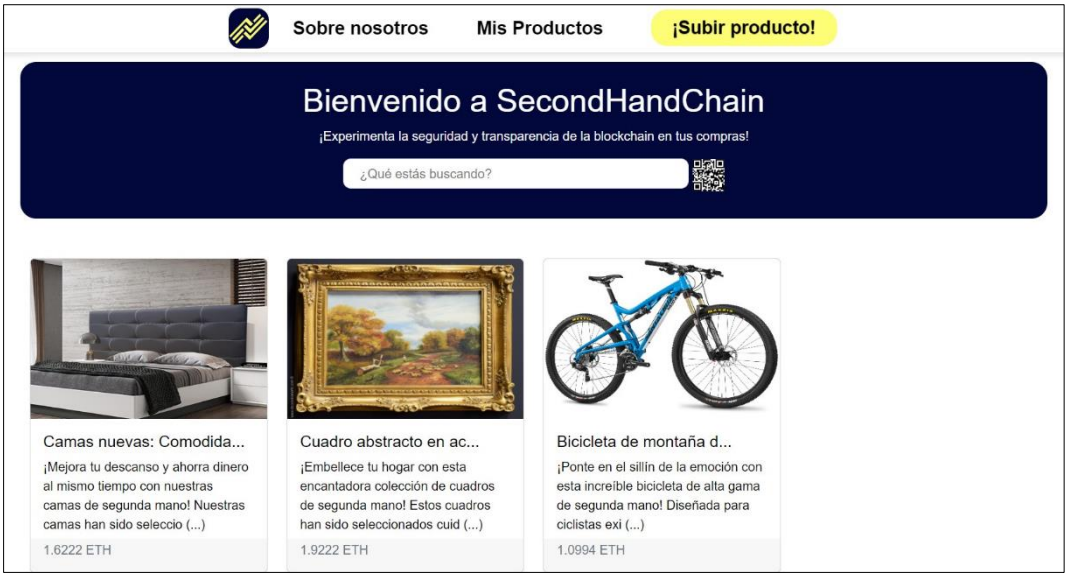


Figura B.1: Página de inicio

B.1.1. Cabecera

A continuación, se explicará el funcionamiento de cada componente de la barra de navegación, común para todas las páginas de la aplicación:

- Icono de la aplicación: Navegación a la pantalla de inicio, donde se encuentran la lista de productos en venta.

- Sobre nosotros: Navegación al apartado de información de la aplicación.
- Mis productos: Navegación a la lista de productos en propiedad del usuario.
- ¡Subir producto!: Navegación al formulario de registro de un producto.

B.1.2. Búsqueda de productos

Tanto en la página de inicio como en la de productos en propiedad, se encuentra una barra de búsqueda con la que es posible filtrar los artículos visibles a partir de la contención del texto en su nombre. La opción de búsqueda mediante código QR no ha sido implementada en la actualidad.

B.2. Registro de productos

El formulario de registro permite crear un nuevo producto para la propiedad del usuario. La información obligatoria a rellenar es el nombre, la descripción y el precio del mismo. El nombre y descripción poseen limitación en el número de caracteres, con un máximo de 50 y 500 caracteres respectivamente. El precio debe ser mayor a cero, con hasta cinco decimales, y un máximo de 999.99999 ETH.

Dentro de los no obligatorios, el campo Imágenes acepta los formatos .png, .jpg y .jpeg, mientras que el campo Archivos solamente acepta documentos en formato .pdf. Con hasta un máximo de cinco archivos para cada uno de estos dos campos.

Una vez rellenado el formulario, se tendrá que pulsar en el botón “Registrar producto”. Tras unos instantes, aparecerá una notificación en Metamask con la validación de la operación. Si no se desea realizar la transacción se podrá rechazar dando al botón “Cancelar” en la notificación. Al realizarse o cancelarse la operación se navegará a la página “Mis productos”, con la lista de productos en propiedad actualizada.

Figura B.2: Página de registro de un producto

B.3. Productos en propiedad

En la página “Mis productos” se observan los productos que el usuario tiene en su propiedad actualmente. Al hacer clic sobre cualquiera de ellos se navegará a la visualización de sus características e información (Ver B.4.). De la misma forma, en la parte inferior de cada artículo aparecen dos botones, uno grisáceo con icono de edición, y uno de color rojizo con icono de papelera, para su eliminación.

Figura B.3: Página de productos en propiedad

B.3.1. Edición

Una vez pulsado el botón de edición de cualquiera de los artículos, se abrirá una ventana emergente con los campos Nombre, Descripción y Precio (ETH) rellenos con la información actual del producto (Ver Figura B.4.). Estos serán campos serán editables con las mismas limitaciones presentes en el formulario de registro.

Una vez rellenado el formulario, se tendrá que dar en el botón “Guardar cambios”. Tras unos instantes, aparecerá una notificación en Metamask con la validación de la operación. Al realizarse o cancelarse la operación se recargará la página “Mis productos”, con la información del producto actualizada.

Figura B.4: Ventana de edición

B.3.2. Borrado

Una vez pulsado el botón de borrado de cualquiera de los artículos, aparecerá una alerta para verificar la operación (Ver Figura B.5.). Una vez pulsado el botón “Aceptar”, aparecerá una notificación en Metamask con la validación de la operación. Al realizarse o cancelarse la operación se recargará la página “Mis productos”, con la lista de productos en propiedad actualizada.

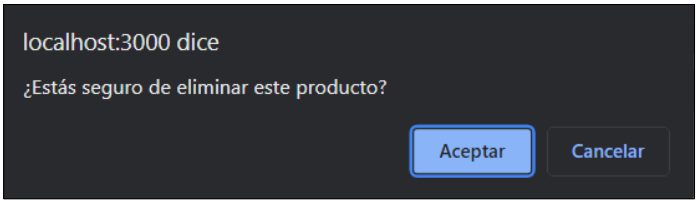


Figura B.5: Notificación de eliminación de un producto

B.4. Visualización de un producto

Una vez accedido a cualquiera de los productos, se navegará a la página de información del mismos. Dentro de ella, se tendrá la información completa del artículo. Al lado izquierdo, se presentan las imágenes adjuntadas en su registro, en caso de haber, con dos flechas a sus lados para poder pasar de una a otra. De forma similar, en el lado derecho se muestra el título y precio del mismo, junto con dos desplegables: Descripción, con la descripción completa, y Documentos adjuntos, donde aparecerán, en caso de haber, los documentos registrados con opción a descargar cada uno de ellos, pulsando en el botón “Descargar” alineado con el nombre de cada uno.

En la parte inferior de las características, se encuentra el botón “Realizar transacción”, con el que se podrá efectuar la operación de compra del artículo. Al pulsarlo, aparecerá una notificación, con la validación de la operación, en la que aparecerá la cantidad total de la transacción (Precio del producto + Tarifa de gas). Al realizarse o cancelarse, se recargará la página de visualización del presente producto.

Cabe destacar que el botón de transferencia del producto solo será visible en caso de acceder a un producto no perteneciente al usuario en cuestión, dado que no es posible transferirse a uno mismo un producto en propiedad.

Figura B.6: Página de visualización de un producto

B.4.2. Historia del producto

En la parte inferior de la página de visualización de un producto aparece una línea cronológica con los eventos sufridos por el respectivo producto, donde se observa en cada uno el tipo de evento y la fecha en la que fue efectuado.

Es posible moverse hacia los eventos futuros y pasados a través de las flechas laterales. De la misma forma se pueden filtrar por tipo de evento, pulsando sobre uno o varios de los botones, de la parte superior, con el nombre del tipo a filtrar. Además, existe la opción de realizar una filtración por fecha mediante el uso de dos calendarios. El calendario situado a la izquierda permite filtrar las fechas a partir de una fecha específica, incluyendo dicha fecha. Por otro lado, el calendario ubicado a la derecha permite filtrar las fechas hasta una fecha determinada, también incluyendo dicha fecha. Finalmente, es posible borrar cualquiera de los filtros seleccionados a partir del botón “Limpiar filtros”.

Figura B.7: Historia de un producto

Es posible hacer clic sobre cualquiera de los eventos para ver sus características en detalle. De esta forma, aparecerá una ventana emergente con los detalles del respectivo tipo evento (Ver Figuras B.7 – B.9).

Figura B.8: Detalles evento de Alta de producto

Figura B.9: Detalles evento de Transferencia

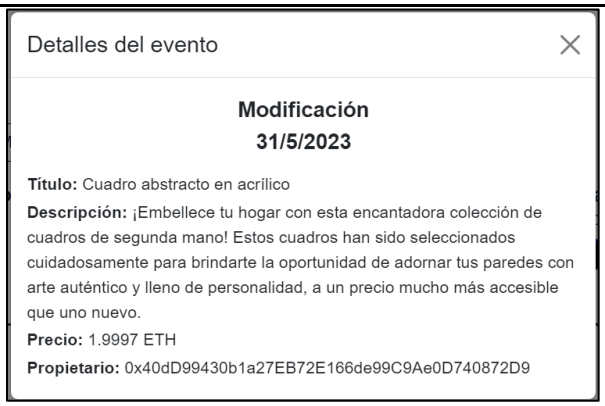


Figura B.10: Detalles evento de Modificación

B.5. Sobre nosotros

En la página “Sobre nosotros” aparece una breve introducción al objetivo de la aplicación e información relevante de la misma, haciendo referencia a ciertos apartados presentes en este documento de memoria.

B.6. Cambio de cuenta Metamask

Dentro de cualquiera de las páginas de la aplicación es posible cambiar de cuenta Metamask, lo cual resultará en una actualización inmediata de la información de la página en la que se encuentre el usuario.

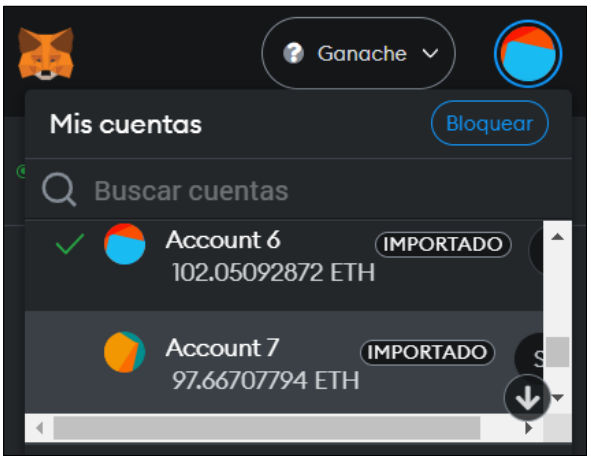


Figura B.11: Cambio de cuenta en Metamask