Tecnologías de Registro Distribuido y Blockchain
Máster Inter-Universitario en Ciberseguridad (MUNICS)
Universidade da Coruña (UDC) y Universidade de Vigo (UVigo)
Curso 2024-2025

Trabajo tutelado - Propiedad Intelectual

Alexis Bernárdez Hermida, Nuria Codesido Iglesias

Índice

1 Introducción

En este primer capítulo de la memoria se proporciona una breve contextualización de la práctica a presentar. Se incluirá la motivación, la definición del escenario y los objetivos.

1.1 Motivación

La práctica de la que se basa este nuevo proyecto, proporciona una plataforma para la gestión de propiedad intelectual utilizando blockchain, ofreciendo la funcionalidad de registrar archivos y asociarlos a derechos únicos. Sin embargo, presenta ciertas limitaciones significativas en términos de seguridad y privacidad, especialmente en lo relacionado con el cumplimiento del RGPD. Por lo que se pretende presentar una nueva plataforma, que cubra todas estas áreas incorporando medidas avanzadas de protección de datos sensibles.

1.2 Definición del escenario

El escenario contempla una plataforma descentralizada basada en blockchain, diseñada específicamente para la gestión eficiente y segura de derechos de propiedad intelectual. Este sistema integra la seguridad de la blockchain de Ethereum con la capacidad de almacenamiento descentralizado de IPFS, creando una plataforma que garantiza la autenticidad, la trazabilidad y la transparencia en la gestión de archivos digitales.

El sistema permite a los usuarios registrar sus archivos digitales vinculándolas a un archivo almacenado en IPFS, generando un NFT que certifica su autenticidad y propiedad. Además de esta funcionalidad, la plataforma incorpora un sistema de gestión de permisos mediante claims, validados por un oráculo, que asegura un acceso controlado y dinámico a los diferentes archivos registrados en la blockchain.

Este nuevo escenario pretende resolver las diferentes limitaciones detectadas en la práctica anterior, tales como la falta de un sistema robusto de trazabilidad y la necesidad de garantizar el cumplimiento del RGPD.

1.3 Objetivos

El objetivo de desarrollar esta plataforma es mejorar la práctica anterior, asegurando su cumplimiento con las normativas del RGPD. Los objetivos específicos son los siguientes:

- **Cumplimiento normativo (RGPD).** Asegurar que la nueva plataforma cumpla con las normativas del RGPD, incluyendo mecanismos que permitan a los usuarios ejercer el derecho al olvido, permitiendo que los archivos registrados puedan ser eliminados de manera segura. Así como la protección de datos personales.
- **Seguridad mediante oráculos.** Integrar un oráculo que actúe como un mediador externo para validar los permisos de acceso a los archivos registrados. Esto permitirá gestionar dinámicamente quién puede acceder a los archivos, reduciendo el riesgo de accesos no autorizados.
- **Gestión de licencias.** Diseñar y desarrollar funcionalidades que permitan emitir, revocar y renovar claims asociados a los archivos registrados. Esto permite a los

propietarios un control sobre los derechos de uso, facilitando la emisión de licencias temporales o permanentes según sea necesario.

2 Estudio del estado del arte

En este capítulo se procederá a analizar soluciones parecidas con la gestión de propiedad intelectual mediante blockchain, además de comparar estas soluciones con la propuesta en este proyecto.

2.1 Análisis de soluciones parecidas

- **OpenSea.** [1] es una aplicación descentralizada, denominada DApp, que facilita la creación, compra y venta de NFTs, utilizando contratos inteligentes en redes como Ethereum, Polygon, y Solana. Estos contratos gestionan las transacciones entre usuarios y archivan la información en sus respectivas blockchains. Además, OpenSea emplea IPFS, un sistema de almacenamiento descentralizado, para garantizar la disponibilidad y acceso constante de los archivos de los NFTs (como imágenes, videos o audios), distribuidos en una red global de computadoras.

- **Filecoin.** [2] Filecoin es una red peer to peer que permite el almacenamiento de archivos descentralizado y confiable utilizando el protocolo IPFS. A través de un mercado abierto, los usuarios pagan a los proveedores para almacenar archivos de manera segura, mientras que estos proveedores mantienen la integridad de los datos mediante pruebas criptográficas. Además, Filecoin incorpora incentivos económicos que garantizan la disponibilidad continua y la persistencia de los datos a lo largo del tiempo.

- **Snapshot.** [3] es una plataforma de votación de código abierto para proyectos Web3, DAO y comunidades que utiliza IPFS como su capa de almacenamiento principal.

2.2 Comparativa con la solución propuesta

2.2.1 Funcionalidades

Las plataformas analizadas en el anterior capítulo, comparten ciertas funcionalidades relacionadas con la gestión de propiedad intelectual y almacenamiento descentralizado. Sin embargo, cada una tiene enfoques distintos.

En el caso de la primera solución propuesta, se enfoca en la creación, compra, venta e intercambio de NFTs. En cambio, Filecoin está orientado al almacenamiento seguro y persistente de datos. Por otro lado, Snapshot es una plataforma de votación de código abierto que permite a las DAO, los protocolos DeFi o las comunidades NFT votar fácilmente y sin tarifas de gas.

Aunque todas estas soluciones, al igual que la propuesta en este proyecto, utilizan IPFS para el almacenamiento, Filecoin destaca por su sistema de incentivos que asegura la

persistencia a largo plazo. Algo que no está presente en OpenSea ni en Snapshot, que dependen de la red de nodos IPFS para la disponibilidad continua.

2.2.2 Ciberseguridad

En términos de ciberseguridad, al igual que la solución propuesta, OpenSea [4], Snapshot y Filecoin aseguran la integridad de los archivos y las votaciones utilizando blockchain e IPFS, lo que garantiza que todas las transacciones sean transparentes e inmutables. Además de esto, para añadir más seguridad, OpenSea recomienda a los usuarios activar métodos de autenticación de dos factores en las carteras digitales.

Sin embargo, la solución Filecoin [5], ofrece un nivel adicional de seguridad relacionado con el uso de pruebas criptográficas que verifican que los datos estén almacenados correctamente. En lugar de simplemente almacenar los archivos, Filecoin utiliza dos tipos de pruebas criptográficas para asegurar que los datos son almacenados correctamente y de manera persistente. Estas pruebas son, prueba de replicación y prueba de espacio-tiempo, que garantizan que los datos se almacenen de manera correcta y durante el tiempo acordado, lo que proporciona una validación continua de los datos.

3 Metodología

En este tercer capítulo se proporcionará una introducción concisa de lo que es una metodología. A continuación, se detallará más en específico la metodología elegida, justificando su elección.

3.1 Metodología de desarrollo software

La metodología [6] de desarrollo software debe estructurar y especificar el proceso de desarrollo de un proyecto, además de ofrecer recomendaciones sobre las mejores prácticas y técnicas a seguir, con el fin de lograr un desarrollo software exitoso. Estas prácticas están diseñadas para mejorar la eficiencia y la gestión del proyecto, garantizando el cumplimiento de sus objetivos.

La elección de la metodología [7] para abordar el desarrollo de un proyecto es fundamental al inicio, debido a que establece el marco de actuación y define cómo se trabajará en el proyecto. Estas difieren principalmente en dos grupos, las metodologías tradicionales y ágiles, cuyas principales diferencias se centran en la manera de enfocar la planificación, ejecución y adaptabilidad del desarrollo software.

En este formulario [8] se identifican los factores más importantes que intervienen en el desarrollo del proyecto. Su propósito es proporcionar una visión integral sobre las implicaciones específicas que estos factores tienen en la ejecución y planificación del proyecto. La finalidad de esta herramienta es ayudar a determinar que metodología es más adecuada en este proyecto

FACTOR	VALOR			
Prioridad de negocio	Cumplimiento - Valor			
Tiempo del proyecto	Largo - Corto			
Tamaño del equipo	Grande - Pequeño			
Interacción con el cliente	Mínima - Máxima			
Ambiente del desarrollo	Controlado - Incertidumbre			
Dirección	Organizativa - Colaborativa			
Rigidez del producto	Cerrado - Ampliable			
Requisitos	Claros - Ambiguos			
Riesgo de fallos	Bajos - Altos			
Enfoque de desarrollo	Procesos - Personas			
Necesidad de documentación	Alta - Baja			
Probabilidad de cambios en el equipo	Alta - Baja			
Roles intercambiables	No flexible - Flexible			
Frecuencia de entregables	Baja - Alta			
VALORES	1	2	3	4
TOTAL	A	B	C	D
VALORACIÓN	(A+B+C+D) / 14 = VALORACIÓN			

Figure 1: Formulario metodología

Se rellena el formulario marcando solo una opción por cada factor, luego se suman los valores y se divide entre el número de factores. Una vez sumado, se puede orientar qué opción sería la más acertada. Si el resultado está más cerca de 1, significa que la opción óptima sería la tradicional, mientras que, si es más cercano a 4, sería más recomendable utilizar un enfoque ágil.

$$\frac{(1+4+4+1+4+4+1+1+4+1+4+4+4+4)}{14} = 3.21$$

Contemplando el resultado obtenido (3,21), se ha optado por emplear el enfoque ágil, específicamente el marco de trabajo Scrum. Este enfoque se ajusta mejor dentro de las características y necesidades del proyecto.

3.1.1 Adaptación de Scrum al proyecto

Scrum [9] es un marco de trabajo para el desarrollo ágil. Se fundamenta en un modelo de desarrollo que sigue un enfoque iterativo e incremental. Es reconocido por ser ligero, fácil de comprender y extremadamente difícil de llegar a dominar. Scrum se caracteriza por estructurarse en ciclos de desarrollo que suelen durar entre dos y cuatro semanas, conocidos como "sprints", así como breves reuniones denominadas stand-up meeting.

Roles y responsabilidades Debido a las circunstancias de elaboración de este proyecto, el development Team está compuesto por dos personas. Mientras que los roles restantes de SCRUM, el product owner y el SCRUM master son asumidos por el profesor de la asignatura.

Eventos del ciclo de vida de Scrum Durante el desarrollo de este proyecto, se realizaron diferentes reuniones. Las reuniones que se realizaron:

- **Reunión de Planificación del Sprint (Sprint Planning Meeting):** Reunión que se lleva a cabo los miércoles en horario de clase. En este evento se pretende planificar el trabajo a realizar a lo largo de un sprint.
- **Revisión del Sprint (Sprint Review):** Está compuesto por un grupo de profesionales responsables de entregar un incremento de un producto "Terminado" al final de cada sprint.
- **Retrospectiva del Sprint (Sprint Retrospective):** Reunión realizada después del sprint review y previamente antes del sprint planning para contemplar posibles mejoras en el desarrollo. Se realizaron en los mismos días que el sprint review.

Artefactos Durante el desarrollo del proyecto, se han generado diferentes elementos como resultado de la aplicación de la metodología Scrum. Los artefactos utilizados:

- **Lista de producto (product backlog):** Se creó un product backlog formado por historias de usuario, ordenadas prioritariamente.
- **Lista de pendientes del sprint (sprint backlog):** Al inicio de cada sprint, en la reunión de planificación, se genera un sprint backlog que contiene las historias de usuario que se estiman implementar en esa iteración.
- **Incremento:** Es el resultado funcional que se presenta al finalizar la iteración e incluye todas las historias de usuario completadas durante la misma.

4 Análisis

En este capítulo se centra en abordar el análisis del proyecto, donde se describen los requisitos del sistema. Se emplean diversas técnicas para obtener una visión completa de los requisitos funcionales, asegurando así una comprensión exhaustiva.

4.1 Actores

En este sistema, se contemplan varios tipos de usuarios.

1. **Propietarios:** Este tipo de actor es el encargado de registrar archivos en la plataforma, otorgando licencias de acceso.

2. **Usuarios:** Este tipo de actor es el encargado de solicitar licencias de acceso a los archivos registrados y utilizarlos conforme a las condiciones definidas por los propietarios.

4.2 Requisitos

Especificación de requisitos funcionales. En este apartado se procede a detallar los requisitos funcionales de la lógica de esta plataforma. Estos requisitos están diseñados para satisfacer las necesidades de los propietarios y usuarios, garantizando así el correcto funcionamiento del sistema.

Cabe destacar, que esta plataforma se implementa sobre la base de un proyecto anterior, que estableció las funcionalidades esenciales. Sin embargo, esta nueva iteración introduce mejoras significativas, diseñadas para ampliar la funcionalidad y asegurar el cumplimiento normativo, además de integrar nuevas capacidades orientadas a la experiencia del usuario y la seguridad de los datos. A continuación, se va a proceder a detallar los diferentes requisitos funcionales del sistema junto las mejoras añadidas.

RF-01.Registro de archivos. El sistema debe permitir a los usuarios registrar archivos en la plataforma y subirlos a Pinata. Previamente, antes de ser registrados, deben ser cifrados con el fin de asegurar su integridad y privacidad. Durante este proceso, se generará una clave aleatoria, que será usada para cifrar la obra del usuario, asegurando que los datos queden protegidos, y solo puedan ser descifrados por quienes tengan la clave correspondiente. Una vez cifrado, el archivo será subido a un nodo IPFS, obteniendo su identificador único (hash). El contrato Ethereum registra los siguientes metadatos del archivo:

- **Hash CID:** Identificador del archivo en IPFS.
- **Título y descripción:** Información proporcionada por el propietario.
- **Clave cifrado:** La clave utilizada para cifrar el archivo.

Este registro asegura que la información del archivo quede asociada al contrato en la blockchain.

RF-02.Gestión de licencias. El sistema debe permitir a los propietarios del archivo gestionar las licencias de acceso que otorga a otros usuarios. Esto incluye funcionalidades para conceder, revocar y controlar accesos temporales:

1. **Conceder acceso:**

    - El propietario puede conceder acceso a un archivo a un usuario específico. Únicamente tiene que especificar la address del usuario y el tokenId del archivo en cuestión.

2. **Revocar acceso:**
    - El propietario puede revocar el acceso previamente otorgado a un usuario.
3. **Acceso temporal:**
    - El propietario puede otorgar accesos con una duración limitada.
    - La clave generada sólo será válida durante el tiempo especificado.

RF-03.Eliminar archivos. El sistema debe permitir a los propietarios eliminar archivos que ya no desean mantener registrados en la plataforma. Este proceso incluye:

- Eliminar la referencia al archivo en el contrato de Ethereum.
- Asegurar que el CID del archivo sea marcado como eliminado, aunque su contenido pueda persistir en IPFS debido a su naturaleza descentralizada.

RF-04.Listar accesos permitidos. El sistema debe permitir a los propietarios consultar una lista de los usuarios que tienen acceso a un archivo registrado.

RF-05.Auditar archivo. El sistema debe permitir a los propietarios auditar las interacciones asociadas con un archivo para garantizar su integridad. Esto incluye:

- El propietario podrá verificar si el archivo ha sido modificado desde su registro inicial. Para ello, el sistema debe comparar el hash almacenado del archivo con el hash proporcionado por el propietario. Si el archivo ha sido modificado, la plataforma devolverá false.

RF-06.Transferir propiedad. El sistema debe permitir a los propietarios transferir la propiedad de un archivo a otro usuario. Además, debe verificar que la dirección del nuevo propietario no sea una dirección inválida, igual a la del propietario actual y que no haya sido registrada con anterioridad.

RF-07.Consultar certificado. El sistema debe permitir a los propietarios consultar el certificado de registro de un archivo. El usuario deberá únicamente proporcionar el hash del archivo y el sistema le proporcionará la siguiente información: Título, descripción, hash y tiempo.

RF-08.Aceptación de términos. El sistema debe permitir a los usuarios aceptar los términos y condiciones antes de realizar cualquier acción significativa dentro de la plataforma.

RF-09.Registrar disputas. El sistema debe permitir a los usuarios registrar disputas sobre los archivos registrados en caso de desacuerdo sobre la propiedad, acceso, o cualquier otra acción relacionada con un archivo. Para ello, deben especificar el tokenId del archivo y el motivo de la disputa. Al registrar una disputa, se debe generar un hash único que identifique de manera inequívoca esa disputa.

RF-10.Historial de transferencias. El sistema debe permitir a los usuarios visualizar el historial de transferencias relacionadas con un archivo, identificando las direcciones de los propietarios anteriores y actuales, así como las fechas de cada transferencia. El usuario únicamente tiene que proporcionar el tokenId del archivo.

RF-11.Visualizar disputas. El sistema debe permitir a los usuarios visualizar el historial de disputas relacionadas con un archivo. Cada disputa será identificada por su motivo, el usuario que la presentó y la fecha en la que se registró. El usuario únicamente tiene que proporcionar el tokenId del archivo.

RF-12.Visualizar archivo. El sistema debe permitir a los usuarios visualizar el contenido de un archivo registrado en la plataforma, incluyendo su título, descripción, fecha de subida, tokenId, hash y dirección del propietario. En el caso de que el usuario sea el propietario del archivo, el sistema debe ofrecer funcionalidades adicionales, como la gestión de licencias y el acceso a información detallada sobre el estado del archivo y sus interacciones. Estas funcionalidades adicionales se encuentran descritas en los requisitos 4.2, 4.2, 4.2, 4.2, 4.2 y 4.2.

Especificación de requisitos no funcionales. Además de los requisitos funcionales, es fundamental tener en cuenta los requisitos no funcionales. Estos no describen lo que debe hacer el sistema, sino cómo debe hacerlo. Son condiciones que se le impone al sistema relacionados con aspectos de calidad y son cruciales para garantizar la satisfacción del cliente. Los requisitos no funcionales de este proyecto son:

RNF-01.Seguridad: Protección de datos sensibles mediante cifrado. Además, se debe utilizar un sistema de autenticación, en este caso MetaMask, para garantizar que solo los usuarios autorizados puedan realizar acciones críticas. Se debe usar un sistema de hashing para verificar la integridad de los archivos.

RNF-02.Interoperabilidad con IPFS: La plataforma debe estar completamente integrada con IPFS para almacenar los archivos de manera descentralizada.

RNF-03.Usabilidad: Debe de tener una interfaz intuitiva que permita a los usuarios navegar fácilmente.

RNF-04.Cumplimiento con RGPD: El sistema debe cumplir con las normativas del RGPD. Esto incluye:

- Garantizar que los datos personales de los usuarios sean tratados de manera segura
- Los usuarios deben poder visualizar y actualizar la información relacionada con sus archivos registrados en la plataforma, permitiéndoles gestionar el acceso y control sobre sus propios datos.
- El sistema debe permitir a los usuarios ejercer su derecho al olvido, lo que implica poder solicitar la eliminación de los archivos registrados en el sistema.

4.3 Justificación del uso de Ethereum

El uso de Ethereum en este proyecto se justifica debido a varias razones. Una de las principales ventajas [10] es que se trata de una red descentralizada que garantiza la seguridad y privacidad en las transacciones, protegiendo el registro de archivos mediante su mecanismo de consenso distribuido. Esto asegura que los datos estén siempre seguros y accesibles de manera transparente.

Además, la implementación de contratos inteligentes en Ethereum permite automatizar procesos de manera segura y eficiente, eliminando la necesidad de intermediarios y garantizando la transparencia en cada operación. Este proyecto adopta el estándar ERC721 (mediante la biblioteca OpenZeppelin), que es ampliamente reconocido y utilizado en la creación y gestión de Tokens No Fungibles (NFTs) [11]. Lo que permite garantizar la autenticidad y propiedad de los activos digitales, además de ofrecer transparencia en las operaciones realizadas.

El contrato también aprovecha tecnologías complementarias, como Pinata - IPFS [12] para almacenamiento descentralizado de archivos. Esta complementación, puede proporcionar capacidades de almacenamiento seguras, con el objetivo de mejorar el rendimiento general de la red. La arquitectura modular de Ethereum permite esta integración sin ninguna complicación.

Adicionalmente, este contrato incluye funcionalidades como: Términos y condiciones, Gestión de licencias. Esto es gracias a la inmutabilidad de la blockchain de Ethereum, que asegura que todas las transacciones y estados del contrato queden registrados de forma permanente y transparente, lo que facilita su auditoría y verificación en cualquier momento.

En conclusión, Ethereum se consolida como la plataforma ideal para este proyecto al proporcionar una combinación de seguridad, escalabilidad y flexibilidad.

5 Diseño

En este capítulo se detallan los aspectos relevantes de la fase de diseño de este proyecto.

5.1 Casos de uso

En este apartado se presentan dos Products Backlogs junto con las historias de usuario que los componen. Siguiendo la metodología Scrum (como se describe en el capítulo 3 - Metodología), no se emplean casos de uso tradicionales. En su lugar, esta metodología se centra en las historias de usuario, las cuales representan requerimientos específicos desde la perspectiva de los usuarios finales.
La primera lista corresponde con el product backlog de la práctica base, que implementa la funcionalidad esencial del sistema. Este backlog inicial sirvió como el punto de partida para la implementación básica de la plataforma de gestión de propiedad intelectual.
A continuación, se detallan las historias de usuario que integra el Product Backlog del proyecto actual.

ID	Usuario	HU
HU-001	Usuario	Como usuario, quiero registrar mis archivos en la plataforma y en IPFS.
HU-002	Propietario	Como propietario de un archivo, quiero conceder y revocar el acceso a usuarios.
HU-003	Propietario	Como propietario, quiero conceder licencias temporales para permitir acceso limitado.
HU-004	Propietario	Como propietario quiero verificar la integridad de mi archivo, para verificar que no ha sido modificado desde su registro.
HU-005	Propietario	Como propietario quiero transferir la propiedad de un archivo a otro usuario.
HU-006	Propietario	Como propietario quiero consultar el certificado de registro de un archivo.
HU-007	Usuario	Como usuario quiero registrar disputas sobre un archivo.
HU-008	Usuario	Como usuario quiero visualizar el historial de transferencias de un archivo.
HU-009	Usuario	Como usuario quiero consultar el historial de disputas de un archivo.
HU-010	Usuario	Como usuario quiero visualizar los detalles de un archivo registrado en la plataforma.

Table 1: Product backlog - práctica base

ID	Usuario	HU
HU-001	Usuario	Como usuario, quiero registrar mis archivos en la plataforma de manera segura, asegurando que no se visualicen datos sensibles para cumplir con las normativas de privacidad.
HU-002	Propietario	Como propietario de un archivo, quiero proporcionar, revocar y gestionar accesos temporales a mi archivo, mediante un sistema centralizado para asegurar que solo usuarios autorizados puedan acceder.
HU-003	Usuario	Como usuario, quiero cifrar mis archivos al registrarlos en la plataforma para proteger su contenido.
HU-004	Usuario	Como usuario, quiero descifrar los archivos registrados cuando sea necesario, utilizando la clave cifrada.
HU-005	Propietario	Como propietario quiero poder eliminar mis archivos que han sido registrados previamente.
HU-006	Usuario	Como usuario, quiero aceptar los términos y condiciones explícitamente antes de registrar cualquier archivo para garantizar mi consentimiento.
HU-007	Propietario	Como propietario de un archivo, quiero consultar la lista de usuarios que tienen acceso a mi archivo.
HU-008	Propietario	Como propietario, quiero garantizar que las transferencias no se registren más de una vez.
HU-009	Usuario	Como usuario, quiero que la plataforma utilice Pinata.
HU-010	Usuario	Como usuario, quiero renovar el acceso a mi archivo.

Table 2: Product backlog - práctica 3

5.2 Diagramas de secuencias

En la figura se presenta el diagrama de secuencias del proceso que sigue este proyecto. Este diagrama es una representación gráfica del funcionamiento y muestra la interacción de los diferentes objetos que conforman este sistema.

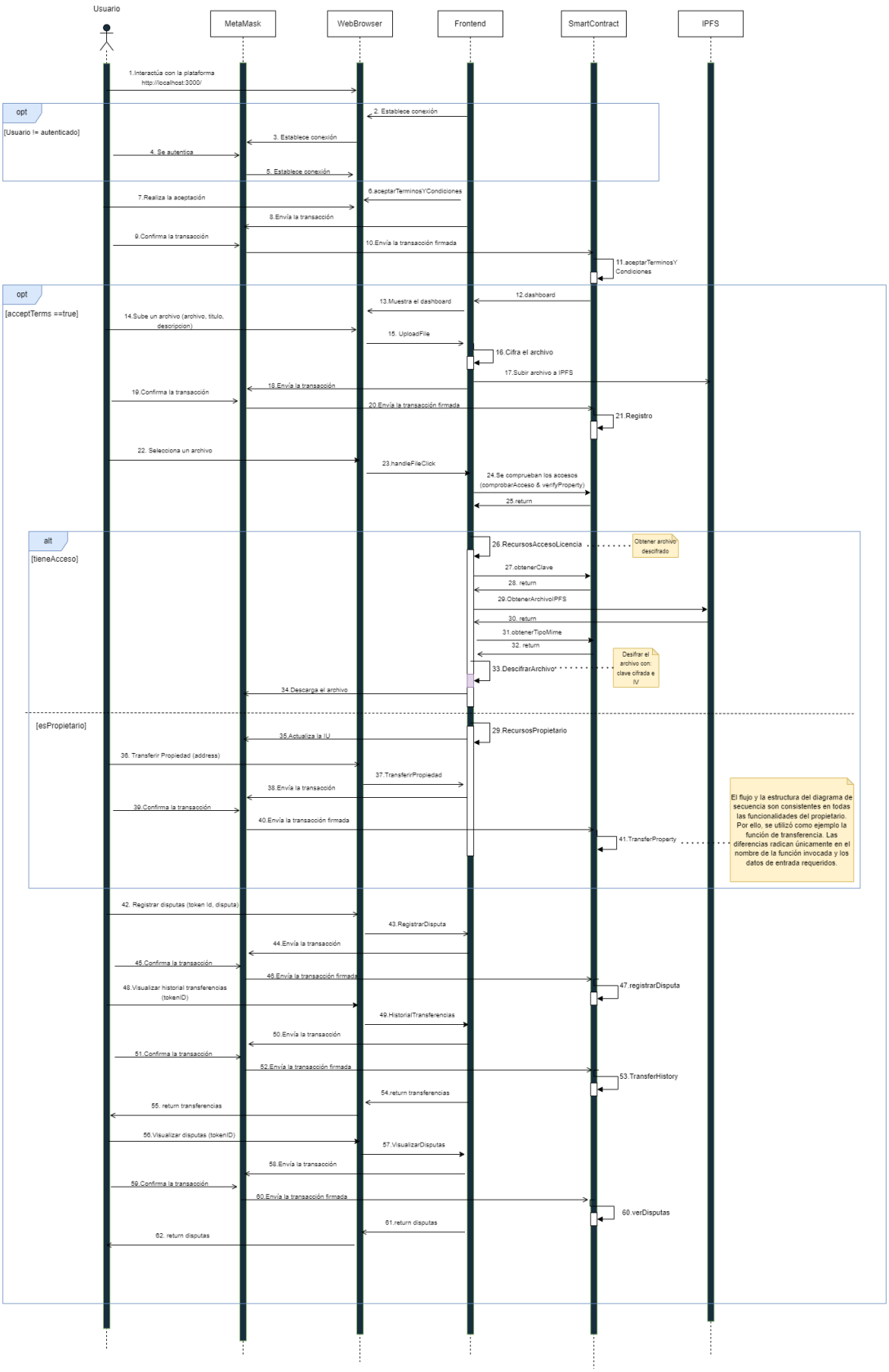

Figure 2: Diagrama de secuencias

5.3 Arquitectura de comunicaciones

En la figura se visualiza el diagrama de la arquitectura de comunicaciones que sigue este proyecto.

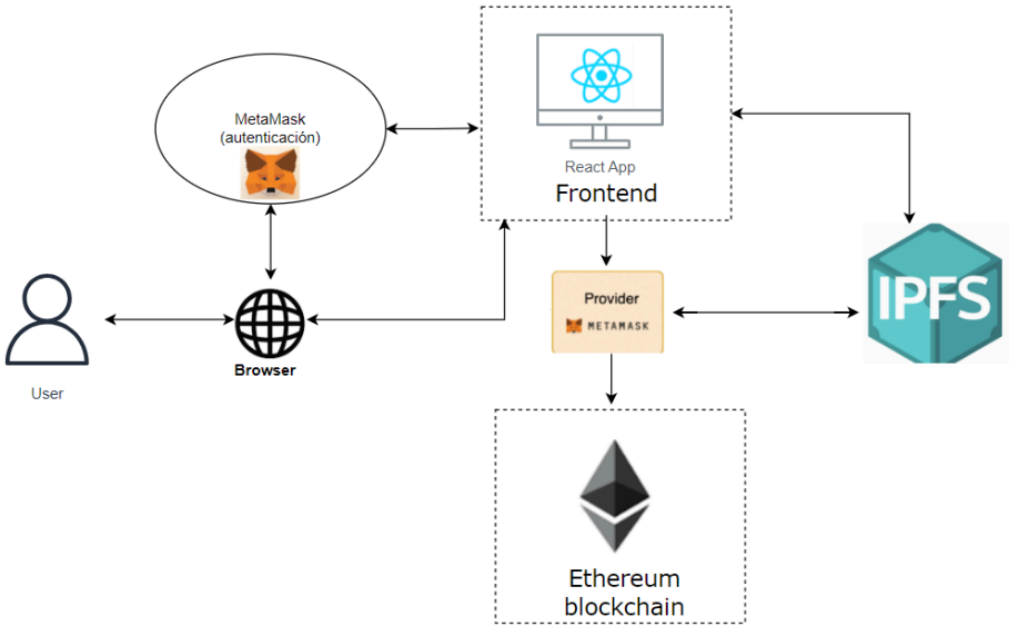

Figure 3: Arquitectura comunicaciones

5.4 Modelo de datos

El modelo conceptual es una herramienta que permite representar de manera simplificada las interrelaciones de las entidades de un sistema de base de datos. En este caso, la estructura de datos en el contrato inteligente utiliza structs para organizar y manejar la información asociada con la gestión de propiedad intelectual. Esto presenta un enfoque distinto en comparación con sistemas tradicionales, ya que en blockchain todo se almacena y gestiona de manera descentralizada.

En este caso, no se relacionan directamente entre sí de la misma manera que en una base de datos relacional. Cada estructura funciona como un contenedor independiente de datos. A pesar de esto, se pueden relacionar de manera lógica a través de los diferentes mapeos y datos compartidos que conectan los conceptos. En este caso, se realiza mediante el uso de identificadores únicos, como tokenId y las direcciones de los usuarios address, que sirven como puntos de referencia entre las estructuras y funcionalidades. A continuación, se presentan las diferentes relaciones lógicas:

- Relación lógica (Archivo - Transferencia): Cada archivo puede tener un historial de transferencias asociado.
  mapping(uint256 => Transferencia[]) private historialTransferencias;
- Relación lógica (Archivo - Disputa): Cada archivo puede estar asociado con múltiples disputas.
  mapping(uint256 => Disputa[]) private historialDisputas;

- Relación lógica (Archivo - Address): Relaciona direcciones de usuarios con los archivos que poseen.
  mapping(address => Archivo[]) private archivos;
- Relación lógica (Archivo - Claim): un Archivo puede estar relacionado con múltiples usuarios y sus respectivos Claim, para gestionar permisos temporales o acceso limitado.
  mapping(uint256 => mapping(address => Claim)) private claims;
- Relación lógica (Usuario - Consentimiento): Relaciona cada dirección de usuario con un estado booleano que indica si ha aceptado los términos y condiciones del sistema.
  mapping(address => bool) private consentimientoDado;

En este contexto, los datos relacionados con la propiedad intelectual se organizan en el contrato inteligente mediante estructuras específicas que representan las entidades principales del sistema. Estas entidades, como Archivos, Transferencias, Disputas y Claim. En las figuras siguientes se muestran los diagramas.

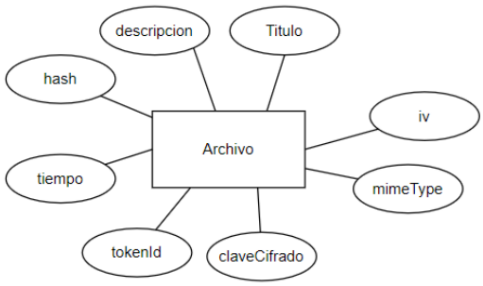

Figure 4: Modelado datos - archivo

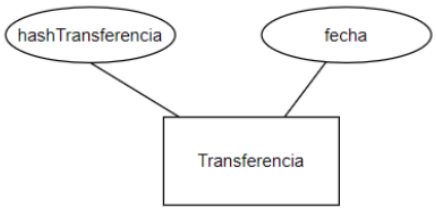

Figure 5: Modelado datos - Transferencia

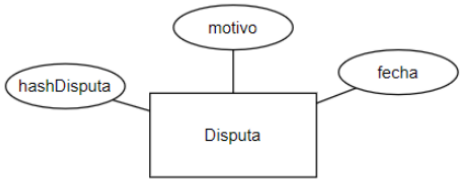

Figure 6: Modelado datos - Disputa

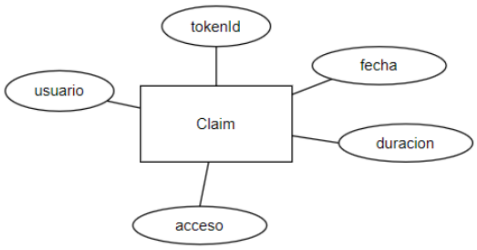

Figure 7: Modelado datos - Claim

5.5 Tecnologías utilizadas

En este apartado se procederá a un análisis exhaustivo de las tecnologías incorporadas en el desarrollo del proyecto. Además, se examinarán también las herramientas complementarias de desarrollo que se utilizarán. A continuación, se detallan las tecnologías incorporadas:

1. **Blockchain - Solidity:** El contrato inteligente está escrito en el lenguaje de programación Solidity en la blockchain de Ethereum. Esta red es una blockchain descentralizada que permite la creación de contratos inteligentes y la ejecución de aplicaciones descentralizadas. En este proyecto, el contrato inteligente administra la propiedad intelectual de manera descentralizada.

2. **OpenZeppelin:** El contrato hace uso del estándar ERC-721, que se utiliza para crear tokens no fungibles (NFTs). Para implementar este estándar, se utiliza la librería OpenZeppelin, que proporciona contratos seguros para interactuar con NFTs.

3. **Pinata:** Los archivos registrados en el sistema se almacenan de manera descentralizada a través de IPFS, utilizando el servicio de Pinata. Este servicio proporciona una plataforma optimizada para gestionar y asegurar la disponibilidad de los archivos en IPFS

4. **MetaMask:** Es una wallet de Ethereum que permite a los usuarios gestionar sus claves privadas y realizar transacciones en la blockchain. En este proyecto, MetaMask facilita la conexión del usuario con el contrato inteligente.

Herramientas utilizadas

1. **Git:** Es un sistema de control de versiones (SCM) distribuido. Estos sistemas, facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Tiene como tarea rastrear y controlar los cambios en el software, algunas de sus características principales:

   - Control de revisiones.
   - Control de configuración.
   - Líneas de base y ramas del proyecto.
   - Seguimiento de defectos.
   - Contabilidad del estado de configuración.

Durante el desarrollo de este proyecto, se ha decidido utilizar el SCM de Git, y en concreto se ha optado por GitHub como plataforma sobre la que alojar el repositorio del proyecto.

2. **Draw.io:** [13] Es un software gratuito de código abierto que permite la creación de diferentes diagramas, flujo, secuenciales, proceso, UML, ER, entre otros. Esta herramienta se utiliza para elaborar los diagramas secuenciales que explican el flujo de los procesos, las interacciones entre los diferentes componentes del sistema y otros aspectos claves del proyecto.

3. **GanttPro:** [14] Es un software de gestión de proyectos online que facilita el proceso de planificación y seguimiento de tareas con la ayuda de diagramas de Gantt. En este proyecto, se utiliza para la creación de los diagramas de Gantt de la planificación y seguimiento.

4. **Latex:** [15] Es un sistema de composición de textos orientado a la creación de documentos técnicos y científicos. Una de sus ventajas clave es su capacidad para gestionar las referencias en un documento. En este proyecto, se utiliza para realizar la memoria de este proyecto.

6 Implementación

En este capítulo se detallan los aspectos más relevantes realizados durante el desarrollo de este sistema.

6.1 Sprint 1

El primer sprint se centró en la comprensión de los nuevos requisitos para la plataforma propuesta en este proyecto. Esto implicó llevar a cabo un análisis detallado de los objetivos del proceso y la integración de medidas para asegurar el cumplimiento de la normativa RGPD.

Un paso fundamental para esta fase fue el diseño del sistema, en el cual se definieron de manera precisa los requisitos y procesos de negocio que se iban a desarrollar. Se definieron las estructuras de datos necesarias para el contrato inteligente, considerando la relación entre las entidades Archivos, Transferencias y Disputas. Se realizó el diseño del modelado de datos y un diagrama de secuencias para representar las interacciones entre los diferentes componentes.

Por otro lado, este proyecto utiliza las tecnologías y la base de la práctica anterior, lo que permitió concentrarse en mejorar la funcionalidad y en el cumplimiento de las normativas. Por lo que no se realizó un análisis exhaustivo de las tecnologías empleadas, dado que ya se había realizado previamente.

Durante este sprint, se realizaron varias modificaciones en el contrato inteligente para cumplir con los nuevos requisitos. Con el fin de cumplir con la normativa RGPD, se modificaron las estructuras ya existentes en la práctica anterior: Disputa y transferencia, para garantizar que no se visualicen directamente datos sensibles de los usuarios address,

y en su lugar se utilizan hashes cifrados para anonimizar esta información. En la figura se visualizan estas estructuras.

```
struct Transferencia {
    bytes32 hashTransferencia
    uint256 fecha;
}

struct Disputa {
    bytes32 hashDisputa;
    string motivo;
    uint256 fecha;
}
```

Figure 8: Structs - Transferencia y disputa

Los hashes se generan dinámicamente en cada función respectiva (*registrarDisputa* y *TransferProperty*). En las dos figuras se visualiza la creación de estos hashes respectivamente.

```
// Crear un hash para la disputa
  bytes32 hashDisputa = keccak256(abi.encodePacked(msg.sender, motivoDenuncia, block.timestamp));
```

Figure 9: Generar Hash disputa - Función (registrarDisputa)

```
// Crear el hash identificador de la transferencia
        bytes32 identificadorHash = keccak256(
            abi.encodePacked(antiguoPropietario, nuevoPropietario, tokenId, block.timestamp)
        );
```

Figure 10: Generar Hash Transferencia - Función (TransferProperty)

Control de acceso. Se desarrolló un contrato adicional *OraculoVerificacionAcceso*, diseñado para gestionar y verificar el acceso de los usuarios a archivos protegidos. Este contrato implementa un sistema que permite conceder, revocar y validar accesos de manera segura, asegurando que solo los usuarios autorizados puedan interactuar con los archivos registrados. En la figura se visualiza el siguiente contrato.

```
contract OraculoVerificacionAcceso {
    mapping(uint256 => mapping (address => bool)) private accesoUsuarios;

    function actualizarAcceso(address usuario, uint256 tokenId, bool acceso) external {
        accesoUsuarios[tokenId][usuario] = acceso;
    }

    function validarAcceso(address usuario, uint256 tokenId) external view returns (bool) {
        return accesoUsuarios[tokenId][usuario];
    }
}
```

Figure 11: OraculoVerificacionAcceso

Por otro lado, para garantizar la interoperabilidad con el contrato principal *PropiedadIntelectual* se implementó a mayores una interfaz *IOracle*, que actúa como un "puente" entre ambos contratos. En la figura se visualiza la interfaz.

```
interface IOracle {
    function actualizarAcceso(address usuario, uint256 tokenId, bool acceso) external;
    function validarAcceso(address usuario, uint256 tokenId) external view returns (bool);
}
```

Figure 12: Interfaz IOracle

Funcionalidades de Gestión de Claims. Este nuevo contrato permite gestionar los accesos mediante la emisión de claims. En este contexto, los claims son registros verificables que permiten registrar los derechos de un usuario sobre un archivo específico. A continuación se detallan, las funcionalidades que se implementaron/modificaron en el contrato principal para aprovechar las funcionalidades del *OraculoVerificacionAccess*:

emisionClaims. Permite emitir un claim de acceso a un archivo para un usuario, utilizando la nueva estructura Claim diseñada específicamente para esta gestión. En las figuras se visualiza la nueva estructura y la funcionalidad de emitir claim.

```
struct Claim {
      address usuario;
      uint256 tokenId;
      uint256 fecha;
      uint256 duracion; // 0 para claims permanentes, valor positivo para temporales
      bool acceso;
  }
```

Figure 13: Struct - Claim

```
function emisionClaims(address usuario, uint256 tokenId, uint256 duracionClaim) public soloPropietario(tokenId) {
    // Solo se agrega el claim si el usuario no está en la lista ya
    require(!usuarioTieneAcceso(usuario, tokenId), "El usuario ya tiene acceso");

    claims[tokenId][usuario] = Claim({
        usuario: usuario,
        tokenId: tokenId,
        fecha: block.timestamp,
        duracion: duracionClaim == 0 ? 0 : block.timestamp + duracionClaim,
        acceso: true
    });

    IOracle oracle = IOracle(oracleAddress);
    oracle.actualizarAcceso(usuario, tokenId, true);

    usuariosConAcceso[tokenId].push(usuario); // Añadir el usuario a la lista de accesos
    emit ClaimEmitido(usuario, tokenId, duracionClaim);
}
```

Figure 14: emisionClaims

verificacionClaims. Permite validar los permisos actuales a un archivo en específico. En la figura se visualiza la implementación.

```
function verificacionClaims(address usuario, uint256 tokenId) public view returns (bool) {
        Claim storage claim = claims[tokenId][usuario];

        if (!claim.acceso || (claim.duracion > 0 && block.timestamp > claim.duracion)) {
            return false;
        }

        // Validar acceso con el oráculo
        IOracle oracle = IOracle(oracleAddress);
        return oracle.validarAcceso(usuario, tokenId);
    }
```

Figure 15: verificacionClaims

revocacionClaims. Facilita la eliminación o actualización de permisos de manera segura. En la figura se visualiza la implementación.

```
function revocacionClaims(address usuario, uint256 tokenId) public soloPropietario(tokenId) {
     Claim storage claim = claims[tokenId][usuario];
     require(claim.acceso == true, "El claim no esta activo.");

     // Revocar el acceso en el oráculo
     IOracle oracle = IOracle(oracleAddress);
     oracle.actualizarAcceso(usuario, tokenId, false);

     claim.acceso = false;
     eliminarUsuarioDeAccesos(tokenId, usuario); // Eliminar al usuario de la lista de acceso
     emit ClaimRevocado(usuario, tokenId);
 }
```

Figure 16: revocacionClaims

Cambios en el contrato principal. A partir de esta nueva implementación, se modificó en el proyecto base la manera de gestionar los accesos a los archivos registrados. En primer lugar, se modificó la función de revocar acceso por *revocacionClaims*. Su implementación se visualiza en la figura anterior.

concederAcceso. También se modificó la función *concederAcceso*, fue modificada para integrar la emisión de claims a través de *emisionClaims*. En la figura se visualiza la implementación.

```
function concederAcceso(address usuario, uint256 tokenId) external soloPropietario(tokenId) {
    emisionClaims(usuario, tokenId, 0);
}
```

Figure 17: concederAcceso

comprobarAcceso. Por otro lado, la función *comprobarAcceso* fue actualizada para utilizar *verificacionClaims*. En la figura se visualiza la implementación.

```
function comprobarAcceso(uint256 tokenId, address usuario) external view returns (bool) {
        return verificacionClaims(usuario, tokenId);
    }
```

Figure 18: comprobarAcceso

darLicenciaTemporal. Se modificó también la manera de gestionar los accesos en la licencia temporal, usando la función *emisionClaims*. En la figura se visualiza la implementación.

```
function darLicenciaTemporal(uint256 tokenId, address usuario, uint256 duracionLicencia) external soloPropietario(tokenId) {
      emisionClaims(usuario, tokenId, duracionLicencia);
  }
```

Figure 19: darLicenciaTemporal

registro. Se agregó la funcionalidad de *emisionClaims* en la función registro de archivos, garantizando que los accesos estén documentados desde el momento de la creación. En la figura se visualiza la implementación.

```
function registro(string calldata hash_ipfs, string calldata titulo, string calldata descripcion, bytes32 claveCifrado, bytes32 iv, string calldata mimeType) external {
    require(_consentimientoDado[msg.sender], "Debes aceptar los terminos y condiciones antes de registrar");
    require(bytes(hash_ipfs).length > 0, "Hash invalido");
    require(bytes(titulo).length > 0, "Titulo invalido");
    require(bytes(descripcion).length > 0, "Descripcion invalida");

    uint256 tokenId = ++_tokenIdCounter;
    _safeMint(msg.sender, tokenId);
    _setTokenURI(tokenId, string(abi.encodePacked("ipfs://", hash_ipfs)));

    _archivos[msg.sender].push(Archivo(titulo, descripcion, hash_ipfs, block.timestamp, tokenId, claveCifrado, mimeType,iv));

    if (!direccionesRegistradas[msg.sender]) {
        direccionesRegistradas[msg.sender] = true;
        propietarios.push(msg.sender);
    }

    // Almacenar la clave de cifrado y IV para este archivo
    clavesCifrado[tokenId] = claveCifrado;
    ivs[tokenId] = iv;  // Almacenar el IV asociado al archivo


    emisionClaims(msg.sender, tokenId, 0);
    emit RegistroRealizado(msg.sender, hash_ipfs, titulo, block.timestamp, tokenId);
}
```

Figure 20: registro

revocarLicenciasVencidas. Se implementó la función *revocarLicenciasVencidas*, para revocar las licencias vencidas de un archivo registrado. En la figura se visualiza la implementación.

```
function revocarLicenciasVencidas(uint256 tokenId) public soloPropietario(tokenId) {
    // Obtener los usuarios con acceso
    address[] storage usuarios = usuariosConAcceso[tokenId];

    for (uint256 i = 0; i < usuarios.length; ) {
        address usuario = usuarios[i];
        Claim storage claim = claims[tokenId][usuario];

        // Si es temporal y expiró
        if (claim.duracion > 0 && block.timestamp > claim.duracion) {
            // Revocar acceso
            claim.acceso = false;

            // Remover usuario de la lista de acceso
            eliminarUsuarioDeAccesos(tokenId, usuario);

            // Actualizar el oráculo
            IOracle oracle = IOracle(oracleAddress);
            oracle.actualizarAcceso(usuario, tokenId, false);

            // Emitir el evento de revocación
            emit ClaimRevocado(usuario, tokenId);

            // No incrementa directamente i porque eliminamos usuarios
            continue;
        }
        i++;
    }
}
```

Figure 21: revocarLicenciasVencidas

Cifrado y descifrado de archivos registrados. Se implementaron nuevas capacidades para fortalecer la protección y el manejo de archivos registrados.
En el caso del cifrado, se añadió la funcionalidad de cifrado de archivos al momento de registrarlos en la plataforma, asegurando que su contenido esté protegido contra accesos no autorizados. Esta implementación se realizó en el frontend antes de enviar los archivos al sistema. Para el cifrado, se utiliza el algoritmo AES-CBC con claves de 256 bits y un IV de 128 bits, ambos generados de manera aleatoria para cada archivo.

En cuanto al proceso de descifrado, se implementan las funciones *descifrarArchivo*, *obtenerTipoMime* y *obtenerClave*. En el frontend, se realiza la mayor parte del proceso de descifrado, pasando como parámetros el archivo cifrado, la clave en formato hexadecimal y el IV correspondiente. Luego, se descifra el archivo utilizando el mismo algoritmo AES-CBC que se utilizó en el cifrado, asegurando que el archivo se descifre correctamente. En las figuras se visualizan la implementación de *descifrarArchivo*, *obtenerTipoMime* y *obtenerClave*.

```
function descifrarArchivo(uint256 tokenId) external view returns (bytes32) {
    // Verificar que el usuario tiene un claim válido
    require(verificacionClaims(msg.sender, tokenId), "No tienes acceso");

    // Retornar la clave derivada con el nonce actual
    return obtenerClave(tokenId, msg.sender);
}
```

Figure 22: descifrarArchivo

```
function obtenerTipoMime(address propietario, uint256 tokenId) public view returns (string memory) {
    // Recorrer los archivos del propietario
    Archivo[] storage archivosPropietario = _archivos[propietario];

    for (uint256 i = 0; i < archivosPropietario.length; i++) {
        if (archivosPropietario[i].tokenId == tokenId) {
            return archivosPropietario[i].mimeType;  // Retorna el mimeType del archivo correspondiente al tokenId
        }
    }

    revert("Archivo no encontrado");  // Si no se encuentra el archivo con ese tokenId
}
```

Figure 23: obtenerTipoMime

```
function obtenerClave(uint256 tokenId, address usuario) public view returns (bytes32) {
    // Verificar que el usuario tiene acceso
    require(verificacionClaims(usuario, tokenId), "No tienes acceso");

    // Verificar que la clave cifrada esté definida para el tokenId
    require(clavesCifrado[tokenId] != bytes32(0), "Clave no encontrada");

    // Devolver la clave derivada
    return clavesCifrado[tokenId];
}
```

Figure 24: obtenerClave

Finalmente, se logró completar la integración de estas funcionalidades en el frontend del sistema. No obstante, debido a complicaciones durante la implementación y problemas técnicos en el funcionamiento, las funcionalidades de cifrado y descifrado no pudieron ser finalizadas dentro de este sprint.

6.2 Sprint 2

En esta iteración, se completaron las funcionalidades de cifrado y descifrado, junto con aquellas relacionadas con el cumplimiento del RGPD.

Eliminación de archivos. Además, se implementó la funcionalidad de eliminación de archivos como parte del "derecho al olvido", permitiendo a los usuarios borrar su información registrada. En la figura se visualiza la implementación.

```
function eliminarArchivo(uint256 tokenId) external soloPropietario(tokenId) {
    // Obtener el hash del archivo
    string memory hash_ipfs;
    Archivo[] storage archivos = _archivos[msg.sender];
    for (uint256 i = 0; i < archivos.length; i++) {
        if (archivos[i].tokenId == tokenId) {
            hash_ipfs = archivos[i].hash;
            archivos[i] = archivos[archivos.length - 1];
            archivos.pop();
            break;
        }
    }

    _burn(tokenId);
    emit ArchivoEliminado(hash_ipfs, tokenId);
}
```

Figure 25: EliminarArchivos

Aceptación de términos. Se incluyó la funcionalidad para que los usuarios acepten los términos y condiciones antes de registrar cualquier archivo en la plataforma, asegurando un consentimiento explícito. En la figura se visualiza la implementación.

```
function aceptarTerminosYCondiciones() external {
    require(!_consentimientoDado[msg.sender], "Ya aceptaste los terminos y condiciones");
    _consentimientoDado[msg.sender] = true;
    emit ConsentimientoAceptado(msg.sender);
}

function verificarConsentimiento(address usuario) external view returns (bool) {
    return _consentimientoDado[usuario];
}
```

Figure 26: Aceptación de términos

listarClaimsArchivo. Se implementó la función *listarClaimsArchivo*, que permite visualizar las licencias activas y accesos permitidos asociados a un archivo. En la figura se visualiza la implementación.

```
function listarClaimsArchivo(uint256 tokenId) public soloPropietario(tokenId) returns (Claim[] memory) {
        // Revocar licencias vencidas antes de listar
        revocarLicenciasVencidas(tokenId);

        uint256 claimsTotales = 0;

        // Recorrer todas las direcciones de usuarios con acceso
        for (uint256 i = 0; i < usuariosConAcceso[tokenId].length; i++) {
            address usuario = usuariosConAcceso[tokenId][i];
            Claim storage claim = claims[tokenId][usuario];

            // Verificar que el claim esté activo y no haya expirado
            if (claim.acceso && (claim.duracion == 0 || block.timestamp <= claim.duracion)) {
                claimsTotales++;
            }
        }
```

Figure 27: listarClaimsArchivo

verificarTransferencia. Se implementó la función *verificarTransferencia*, integrada en *TransferProperty*, para verificar que la transferencia no haya sido registrada previamente. En la figura se visualiza la implementación.

```
function verificarTransferencia(uint256 tokenId, address antiguoPropietario, address nuevoPropietario, uint256 fecha) internal view returns (bool) {
    bytes32 hashPropuesto = keccak256(abi.encodePacked(antiguoPropietario, nuevoPropietario, tokenId, fecha));

    // Verificar si el hash coincide con alguno en el historial
    Transferencia[] storage transferencias = _historialTransferencias[tokenId];
    for (uint256 i = 0; i < transferencias.length; i++) {
        if (transferencias[i].hashTransferencia == hashPropuesto) {
            return true; // La transferencia ya está registrada
        }
    }
    return false; // No se ha encontrado la transferencia
}
```

Figure 28: verificarTransferencia

renovarClaim. Se implementó la función renovarClaim. Sin embargo, debido a contratiempos no se pudo completar su funcionalidad en el frontend.

```
    function renovarClaim(address usuario, uint256 tokenId, uint256 nuevaDuracion) public soloPropietario(tokenId){
        Claim storage claim = claims[tokenId][usuario];
        require(claim.acceso == true, "El claim no esta activo.");

        claim.duracion = block.timestamp + nuevaDuracion;
        emit ClaimRenovado(usuario, tokenId, nuevaDuracion);
    }
```

Figure 29: renovarClaim

6.3 Sprint 3

Durante el último sprint del proyecto se enfocó en la realización de esta memoria.

7 Demostración del funcionamiento del sistema

En este capítulo se centra en abordar la demostración del sistema, para ello realizamos un vídeo "promocional" de la dApp. A continuación, se explicará el proceso del despliegue de la plataforma.

7.1 Despliegue

Para el despliegue, se hizo uso de Remix. Una vez completado el código del contrato, se compila utilizando el compilador integrado en Remix. Esto convierte el código en bytecode que puede ser ejecutado en la blockchain. Cabe destacar, que debido a que es un contrato bastante extenso, para poder realizar el despliegue sin problemas, se tiene que habilitar la opción "Enable optimization" en las configuraciones advanzadas, para optimizar el contrato. Después de compilar el contrato, se selecciona el entorno de conexión como Wallet Connect, y se procede al despliegue del contrato *OraculoVerificacionAcceso*. La transacción se firma y se paga (sin gasto de gas) a través de MetaMask, lo que permite que el contrato sea registrado en la red Ethereum. Finalmente, se despliega el contrato *PropiedadIntelectual* siguiendo, y para ello se tiene que hacer uso de la contract address del otro contrato.

8 Análisis de los riesgos de seguridad (e.g., SRM)

En este capítulo se centra en abordar el análisis de los riesgos de seguridad (SRM), siguiendo un enfoque estructurado para identificar, evaluar y mitigar los riesgos potenciales que podrían afectar la seguridad y la integridad de la plataforma.

En primer lugar, se identificaron los posibles riesgos y vulnerabilidades que podrían llegar a afectar a la plataforma. Uno de los riesgos más significativos fue la posible exposición de los archivos almacenados en Pinata, debido a la falta de medidas de cifrado adecuadas. Además, se detectó el riesgo de accesos no autorizados a los archivos registrados.

Una vez identificados estos riesgos, se procedió a evaluarlos considerando su probabilidad de ocurrencia y su impacto potencial, priorizando las tareas necesarias para mitigarlos, como se detalla en el product backlog del producto. Los posibles impactos identificados incluyeron la exposición de datos sensibles, el riesgo de fallo en el descifrado debido a una gestión incorrecta de claves o cifrado, y los errores de validación.

Finalmente, se desarrollaron medidas para minimar su impacto. En el caso de la exposición de los archivos, se implementó un cifrado AES-CBC. Mientras, que en relación al riesgo de que no se realice la verificación de manera adecuada, se realizó una doble validación mediante un oráculo descentralizado y un registro (claim) de permisos específicos asociados a los usuarios.

9 Plan de pruebas

Para validar el correcto funcionamiento de la plataforma, se diseñó y ejecutó un plan de pruebas exhaustivo utilizando Remix. Este proceso se llevó a cabo después de desplegar los contratos *OraculoVerificacionAcceso* y *PropiedadIntelectual*, como se explica en el apartado del despliegue. Una vez desplegados, se pueden visualizar en la sección Deployed Contracts.

En un primer instante, para poder probar el funcionamiento completo de esta plataforma, se llevó a cabo un protocolo de pruebas que contenía todos los posibles escenarios de las diferentes funcionalidades, manteniendo un registro detallado de cada prueba realizada. A continuación, se simuló cada funcionalidad y una vez simulado, se registró su estado como aprobado, no aprobado, pendiente o error. Esto garantizó un seguimiento preciso de los cambios que se tenían que realizar a posteriori.

10 Lean canvas

Lean Canvas [16] se utiliza al inicio para definir claramente los objetivos y el modelo de negocio. Esto permite al equipo Scrum entender el "por qué" detrás del proyecto. En este proyecto, se ha integrado esta herramienta con el proceso ágil de Scrum.

Al inicio del proyecto, se creó el Lean Canvas para nuestro proyecto. El cual supuso un lienzo para establecer la estrategia inicial y capturar las hipótesis y suposiciones sobre el negocio y el producto, incluyendo los problemas, soluciones, métricas clave, entre otros.

Dentro de la metodología aplicada, el Lean Canvas actúa como una visión inicial que guiará el trabajo realizado por los integrantes de nuestro equipo SCRUM (en este caso, compuesto por dos personas). Gracias al uso de esta herramienta, permitió ayudar a priorizar el backlog de producto, asegurando que el trabajo del equipo se enfoque en lo que tiene más valor y es más crítico para la viabilidad del producto. En la figura se muestra el lienzo de Lean Canva que se generó en este proyecto.

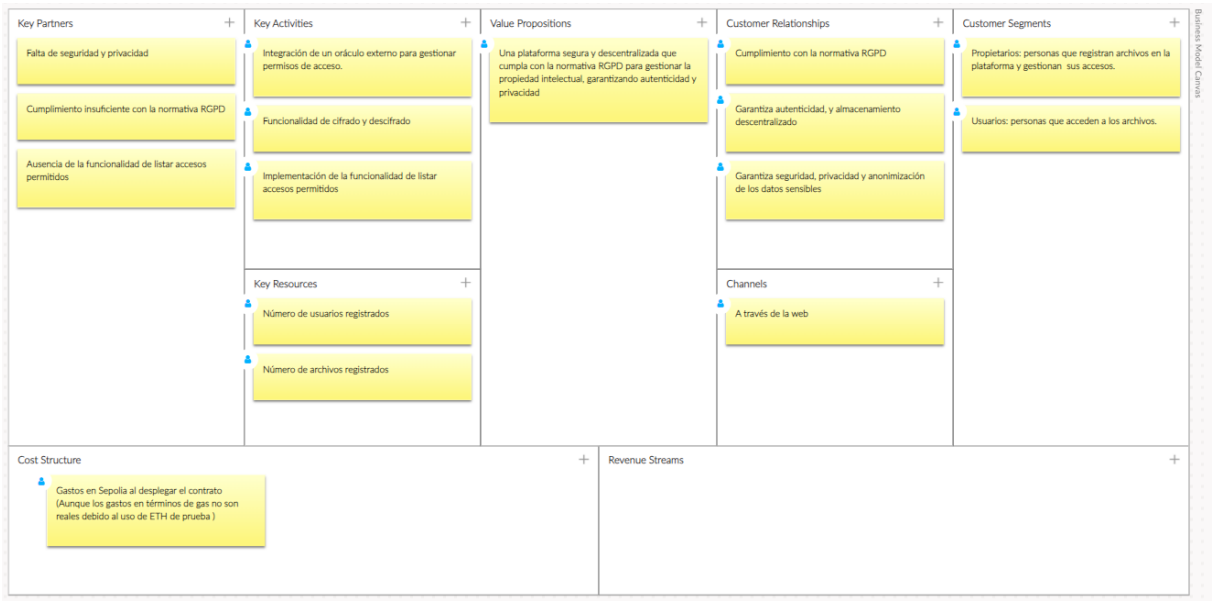

Figure 30: Lean Canva

El siguiente paso consistió en definir experimentos para probar las hipótesis planteadas en el Lean Canva. Los experimentos se diseñaron para generar información procesable de manera rápida, con cada uno contando con una hipótesis, una métrica, un método y un criterio de éxito claro. En el contexto de Scrum, estos experimentos se planificaron como historias de usuario (HU), las cuales se desglosaron en tareas dentro de los sprints. Esto asegura que cada sprint tenga un propósito claro.

Posteriormente, se llevó a cabo la planificación del proyecto (explicada en el capítulo de Planificación). En esta fase, se planificaron los sprints según los experimentos y objetivos previamente establecidos. Se crearon historias de usuario, tareas y objetivos de sprint, todos alineados con las hipótesis y experimentos del Lean Canvas.

Una vez llevada a cabo la planificación, tuvo lugar la fase de Implementación, en la cual se ejecutaron los sprints siguiendo los principios de Scrum: reuniones diarias, trabajo en equipo, revisiones de código, pruebas y demostraciones. Basándose en los resultados de aprendizaje obtenidos durante cada sprint, se actualizó el Lean Canvas para reflejar los nuevos conocimientos adquiridos.

11 Aspectos legales

En este apartado, se procede a describir los aspectos legales que deben considerarse para asegurar el cumplimiento y la protección de los usuarios involucrados.

- **Cumplimiento RGPD.** La plataforma cumple con la normativa RGPD, asegurando la privacidad de los datos personales, cumpliendo con requisitos como el derecho al olvido (eliminación de archivos), la protección de datos sensibles mediante el cifrado y la transparencia en la gestión de permisos de acceso (hashes).
- **Aceptación de Términos y Condiciones.** Los usuarios deben aceptar los términos y condiciones antes de realizar cualquier acción en el sistema, garantizando que estén informados y de acuerdo con las políticas de uso y privacidad de la plataforma.
- **Ciberseguridad.** La plataforma implementa medidas criptográficas y mecanismos como un oráculo y validaciones que aseguran que solo las partes autorizadas accedan a los archivos.

12 Manual de usuario (opcional)

Con el objetivo de facilitar el uso de esta plataforma a los usuarios, a continuación se explican detalladamente las diferentes pantallas y funcionalidades disponibles. Es importante destacar que, para garantizar la autenticación de los usuarios y asegurar la integridad y trazabilidad de las transacciones, se utiliza MetaMask en todas las interacciones con la plataforma.

12.1 Estructura del proyecto

Aceptar términos y condiciones. En la figura se visualiza un pop up que solicita al usuario que acepte términos y condiciones. Este paso es obligatorio para acceder al resto de las funcionalidades.

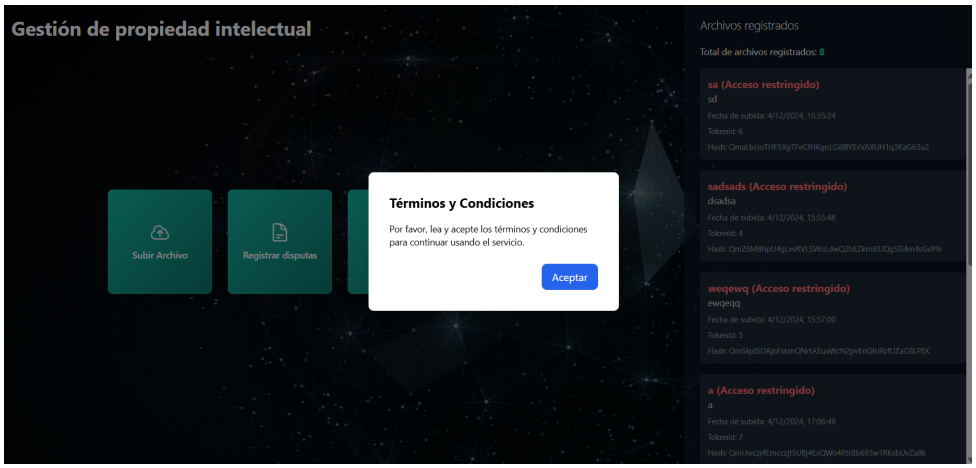

Figure 31: Aceptar terminos y condiciones

A continuación, una vez que el usuario haya aceptado los términos, se le redirige a la página principal. A la derecha, podrá observar los archivos registrados en el sistema junto con información relevante. Cabe destacar, solo el propietario o usuarios autorizados podrán acceder al contenido completo. En la figura, se visualiza la página principal.

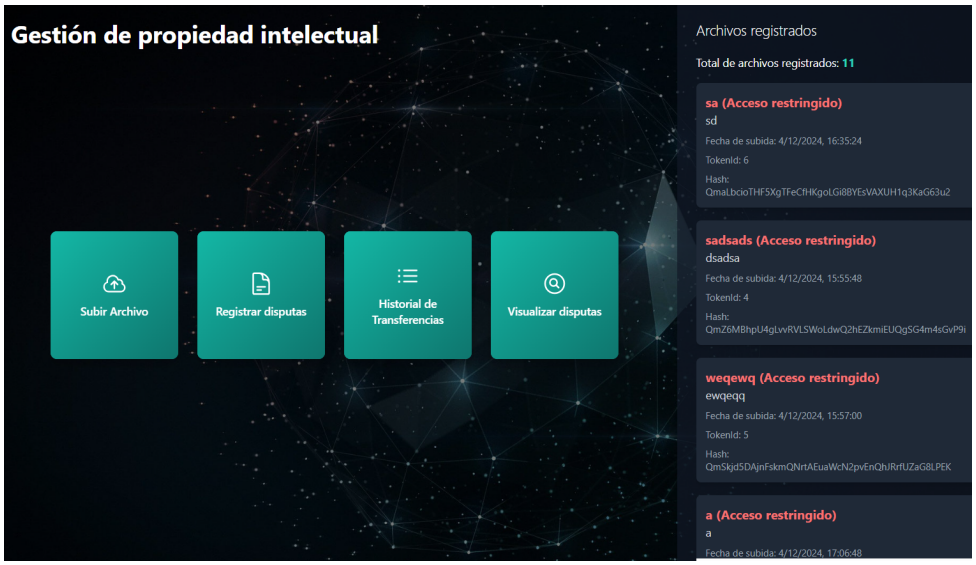

Figure 32: Página principal

Subir archivo El usuario, podrá subir archivos a Pinata para registrarlos en la plataforma. Para ello deberá especificar el archivo que quiere subir, junto un título y descripción.

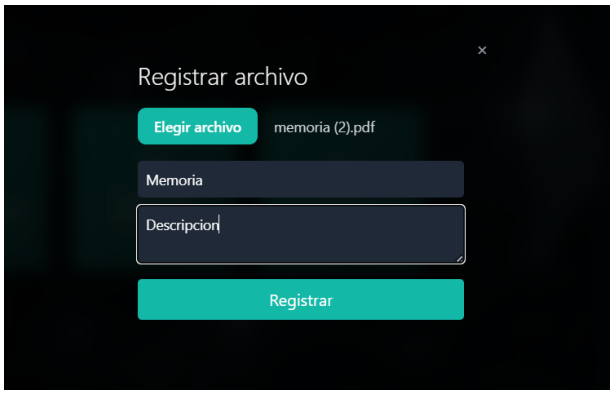

Figure 33: Subir archivo

En la figura, se visualiza el archivo subido a Pinata.

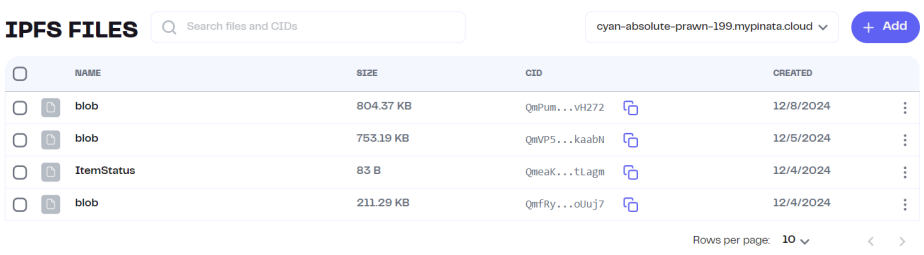

Figure 34: Pinata - subir archivo

En la figura, se visualiza el archivo subido a la plataforma.

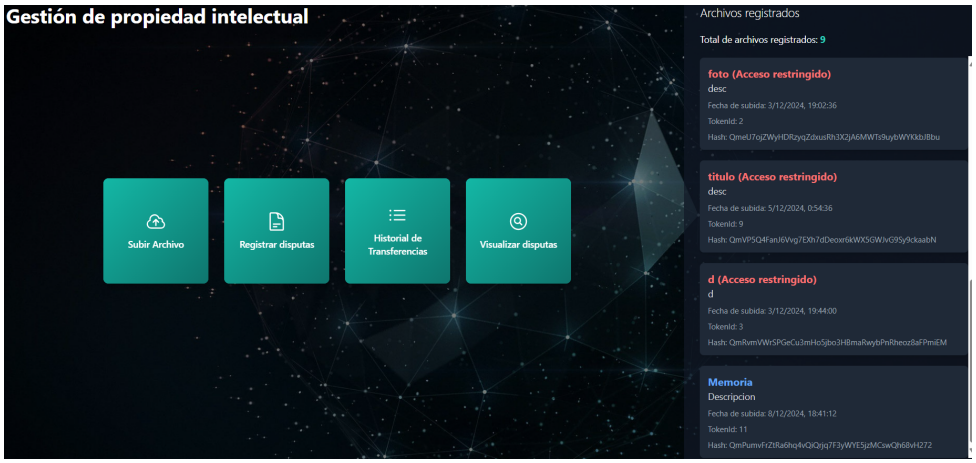

Figure 35: Archivo subido

Opciones propietario. El propietario tiene acceso a múltiples funcionalidades para gestionar sus archivos. En la figura se visualizan estas funcionalidades.

Figure 36: Opciones propietario

Una de las primeras opciones es **Transferir propietario**, en esta funcionalidad, el propietario puede transferir la propiedad de este archivo. Para ello, únicamente debe proporcionar el hash del usuario al que quiere transferir la propiedad.

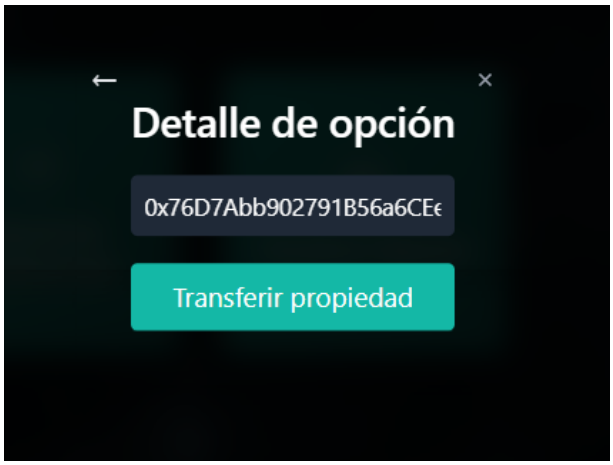

Figure 37: Transferencia de propiedad

Las demás funcionalidades siguen un proceso similar a este. El usuario solo necesita proporcionar los requisitos indicados por la plataforma para completar la acción que desea realizar.

Opciones no propietario. Los usuarios con acceso autorizado al archivo, pero que no son propietarios, podrán visualizar su contenido. En la figura se visualiza la funcionalidad de acceder archivo.

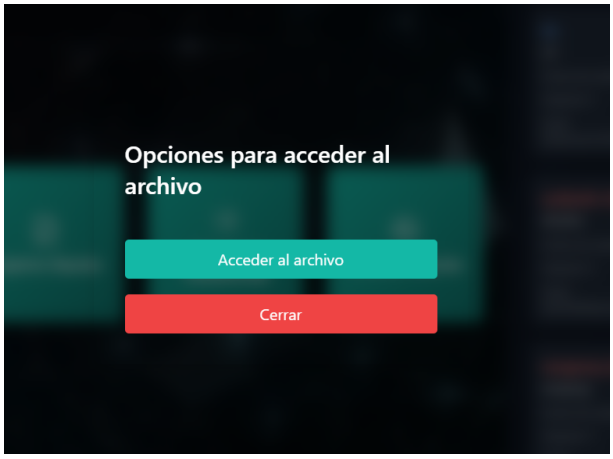

Figure 38: Opciones no propietario

Una vez que el usuario le de al botón de acceder archivo, podrá visualziar un visor con la vista previa del archivo en cuestión.

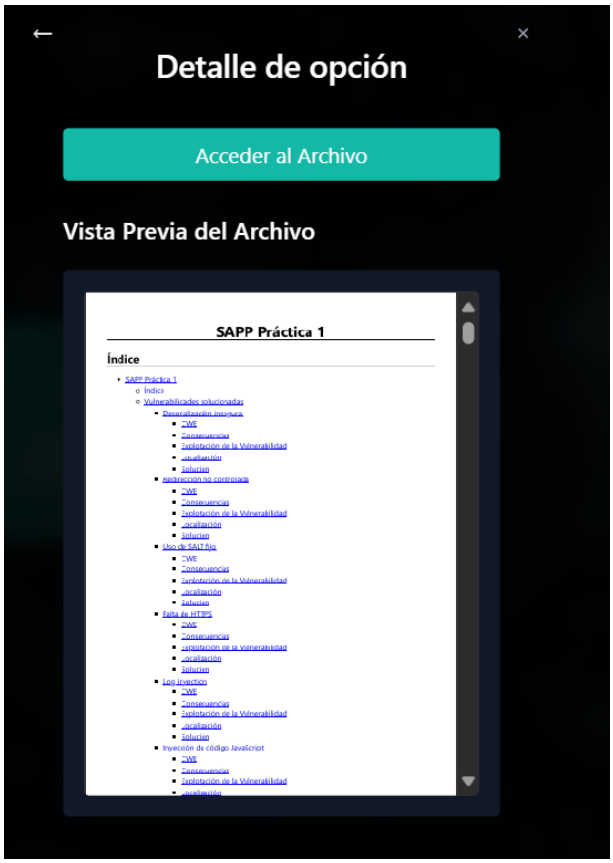

Figure 39: Acceder archivo

Registrar disputa. El usuario podrá registrar una disputa relacionada con un archivo. Para ello, deberá proporcionar el tokenId del archivo en cuestión, así como el motivo que justifica la disputa. Esta funcionalidad no requiere permisos.

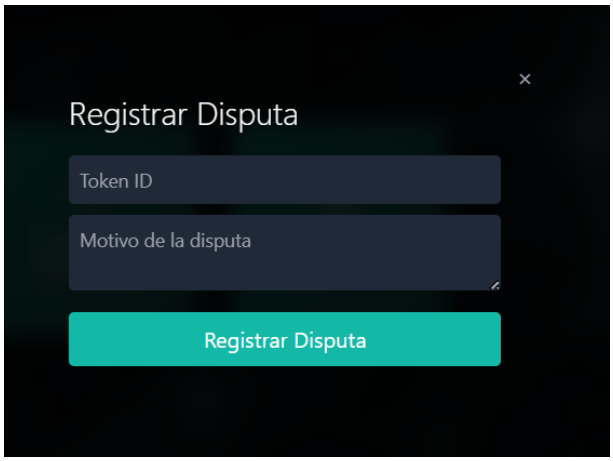

Figure 40: Registrar disputa

Visualizar disputa. El usuario podrá consultar las disputas asociadas a un archivo. Para hacerlo, deberá ingresar el tokenId del archivo correspondiente. Esta funcionalidad no requiere permisos.

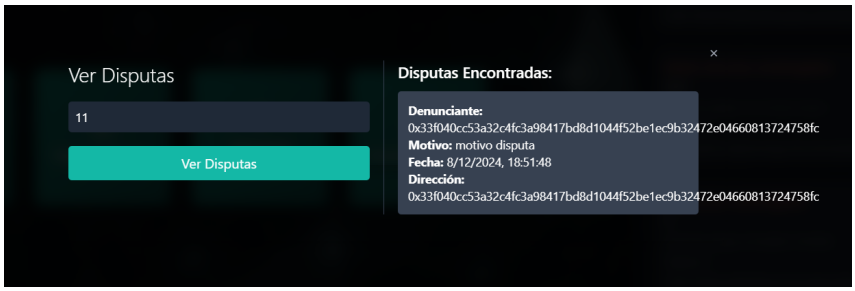

Figure 41: Visualizar disputa

Historial transferencias. El usuario podrá consultar el historial de transferencias de un archivo. Para acceder a esta funcionalidad, no se requiere ningún permiso.

Figure 42: Historial transferencias

13 Conclusiones

Tras alcanzar el final del proceso de desarrollo de este proyecto, se contempló si se habían cumplido los diferentes objetivos especificados. Esta evaluación permitió determinar si el proyecto desarrollado había sido exitoso. Los logros alcanzados incluyen:

- Implementación de diferentes medidas para garantizar el cumplimiento de la normativa RGPD, como la anonimización de datos sensibles mediante el uso de hashes. Así como también, el "derecho al olvido", establecido por esta normativa, para permitir la eliminación de archivos registrados.
- Implementación de funcionalidades de cifrado y descifrado de archivos, asegurando que los datos sensibles se mantuvieran protegidos.
- Se incorporaron nuevos mecanismos más seguros para asegurar que los usuarios solo pudieran acceder a los archivos para los cuales tenían permisos.

En conclusión, este proyecto alcanzó los objetivos propuestos en el inicio, que consistían en mejorar las funcionalidades del sistema anterior, integrar nuevas capacidades de seguridad y privacidad.

14 Líneas futuras

A pesar de alcanzar todos los objetivos propuestos, se podrían añadir algunas funcionalidades adicionales que permitirían mejorar este sistema.

Una de las futuras mejoras que se podría realizar, sería la incorporación de una funcionalidad para la resolución de disputas utilizando un oráculo como mediador. Este oráculo, al no tener interés directo en el resultado del conflicto, actuaría como un intermediario neutral. Proporcionaría datos objetivos y verificables para resolver desacuerdos entre los usuarios sobre cuestiones relacionadas con la propiedad o el acceso a los archivos registrados. Al integrar estos datos externos en el sistema, el oráculo aseguraría que la resolución del conflicto se realice de manera justa

Otra mejora, sería agregar la funcionalidad de renovar el acceso a un archivo. A pesar de haber realizado el análisis y diseño del caso de uso para la renovación de claims, su integración en el sistema no se completó por completo debido a contratiempos y a la falta de tiempo en el desarrollo.

15 Lecciones aprendidas

15.1 Incidencias

Durante el desarrollo del proyecto, se han identificado varios desafíos técnicos y se han implementado soluciones que han permitido mejorar la eficiencia y seguridad de la plataforma. A continuación se detallan las principales incidencias:

- **Cifrado y descifrado.** Uno de los principales retos fue implementar un sistema de cifrado que garantizara la seguridad de los archivos. Inicialmente, se utilizó una combinación de nonces, IVs y clave cifrada para el cifrado de los archivos. Sin embargo, este enfoque presentó problemas, ya que no se podía descifrar correctamente el archivo. Posteriormente, se descubrió que no era necesario el uso del nonce, ya que el IV por sí solo era suficiente para garantizar la seguridad del cifrado, lo que simplificó el proceso y resolvió los inconvenientes.

- **Gestión de accesos.** Por otra, parte, la gestión de accesos fue otro reto en este proyecto. Ya que se quería buscar un mecanismo que asegurase la validación de manera descentralizada y transparente. Para ello se implementó un oráculo que gestionase los permisos de acceso a los archivos.

16 Tiempo dedicado/esfuerzo

- Actualizar Contrato inteligente:
- Integración frontend-backend:
- Validaciones:
- Documentación de la memoria:

Las horas totales en realizar esta tarea entre los dos compañeros, fue de ? horas aproximadamente.

17 Planificación

En este capítulo, se examinará la planificación del desarrollo de este proyecto. El desarrollo de este sistema se llevará a cabo utilizando un enfoque de fases en donde cada fase del desarrollo se divide en múltiples iteraciones o Sprints.
Para llevar a cabo la planificación de este proyecto, se empleó una de las herramientas de gestión de proyectos, conocida como el diagrama de Gantt. Esta técnica representa en una escala temporal de tiempos, cada una de las iteraciones del proyecto mediante barras, que representan su duración en fechas del calendario. La longitud se establece de manera proporcional. Las tablas que se muestran detallan las fechas estimadas del progreso de cada sprint. Se estimó como fecha de finalización del proyecto el 4 de diciembre, dejando una holgura de 4 días para la fecha de entrega. Esta holgura proporciona cierta flexibilidad para que en caso de que se produjese un retraso en el desarrollo de alguno de los Sprint, permita tener tiempo para poder abordar y resolver los problemas sin comprometer la fecha de entrega, el 8 de diciembre. La estimación es uno de los aspectos más importantes para llevar a cabo una buena planificación, supone el primer paso para poder efectuar un plan de proyectos. Es la actividad que permite para un proyecto, obtener respuestas a preguntas, como: ¿cuánto tiempo llevará hacerlo?

Iteración	**Duración (días)**	**Fecha inicio estimada**	**Fecha fin estimada**
Sprint 1	7	20/11/2024	27/11/2024
Sprint 2	7	28/11/2024	4/12/2024

Table 3: Planificación

El gráfico que se presenta corresponde a una estimación del proyecto que permitirá estimar el tiempo de desarrollo de cada tarea.

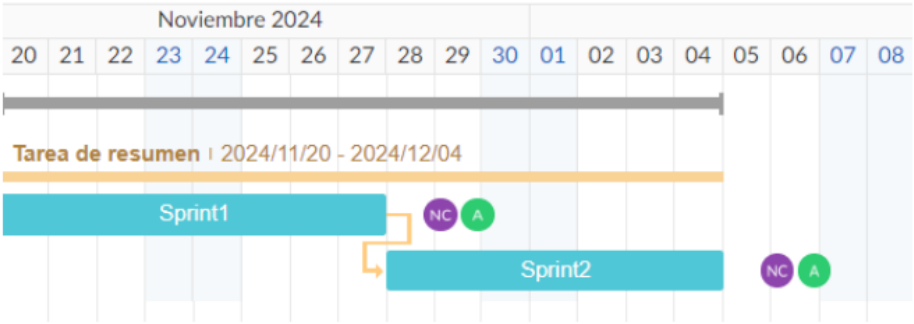

Figure 43: Diagrama de gantt

Sprint 1. En este primer sprint consistió en ver que mejoras se podrían agregar al proyecto base de la práctica anterior. Adicionalmente, en este sprint también se realizó la fase de diseño, que involucró la creación de diagramas. También se inicia con el desarrollo de las mejoras del proyecto base. En la reunión de planificación, revisamos las hipótesis clave definidas en el Lean Canvas, como la necesidad de seguridad para los archivos y la gestión de accesos, por lo que decidimos priorizar historias como 'HU-001' y 'HU-002' que permitirán probar estas hipótesis en el primer sprint. En consecuencia, la lista del sprint backlog es la siguiente:

ID	HU	Tareas	Estado
HU-001	Como usuario, quiero registrar mis archivos en la plataforma de manera segura, asegurando que no se visualicen datos sensibles para cumplir con las normativas de privacidad.	Análisis y diseño de esta historia.	Completada
		Implementar la funcionalidad de registro seguro de archivos.	Completada
		Agregar validaciones de datos sensibles.	Completada
		Implementar el cumplimiento de normativas de privacidad.	Completada
HU-002	Como propietario de un archivo, quiero proporcionar, revocar y gestionar accesos temporales a mi archivo, mediante un sistema centralizado para asegurar que solo usuarios autorizados puedan acceder.	Análisis y diseño de esta historia.	Completada
		Implementar la gestión de accesos.	Completada
		Agregar/modificar funcionalidades en el frontend.	Completada
HU-003	Como usuario, quiero cifrar mis archivos al registrarlos en la plataforma para proteger su contenido.	Análisis y diseño de esta historia.	Completada
		Implementar la funcionalidad de cifrado.	Completada
		Agregar funcionalidad en el frontend.	Iniciada
HU-004	Como usuario, quiero descifrar los archivos registrados cuando sea necesario, utilizando la clave cifrada.	Análisis y diseño de esta historia.	Completada
		Implementar la funcionalidad de descifrado.	Completada
		Agregar funcionalidad en el frontend.	Iniciada

Table 4: Product backlog - Sprint 1

ID	HU	Tareas	Estado
HU-005	Como propietario quiero poder eliminar mis archivos que han sido registrados previamente.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de derecho al olvido.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada
HU-006	Como usuario, quiero aceptar los términos y condiciones explícitamente antes de registrar cualquier archivo para garantizar mi consentimiento.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de consentimiento explícito.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada
HU-007	Como propietario de un archivo, quiero consultar la lista de usuarios que tienen acceso a mi archivo.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de consulta de accesos.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada
HU-008	Como propietario, quiero garantizar que las transferencias no se registren más de una vez.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de verificación de transferencia.	No iniciada
HU-009	Como usuario, quiero que la plataforma utilice Pinata .	Análisis y diseño de esta historia.	No iniciada
		Configuración.	No iniciada
		Implementación.	No iniciada
HU-010	Como usuario, quiero renovar el acceso a mi archivo.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de renovar acceso.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada

Table 5: Product backlog - Sprint 1

Una vez finalizado este sprint, en la reunión de la revisión se evaluó el incremento producido y se analizaron los cambios que se obtuvieron en el sprint backlog. Como se

observa en la tabla, hubo 17 tareas que no se pudieron iniciar y dos tareas iniciadas pero pero incompletas. Este retraso se atribuyó a la complejidad inesperada en el desarrollo, especialmente en las funcionalidades relacionadas con el cifrado y descifrado, las cuales resultaron más desafiantes de lo previsto.

Sprint 2. En este sprint se centró en completar las tareas que no se pudieron completar en el anterior sprint. En la reunión de planificación, se propuso terminar HU-003 y HU-004, y realizar las siguientes historias de usuario. En consecuencia, la lista del sprint backlog es la siguiente:

ID	HU	Tareas	Estado
HU-003	Como usuario, quiero cifrar mis archivos al registrarlos en la plataforma para proteger su contenido.	Agregar funcionalidad en el frontend.	Completada
HU-004	Como usuario, quiero descifrar los archivos registrados cuando sea necesario, utilizando la clave cifrada.	Agregar funcionalidad en el frontend.	Completada
HU-005	Como propietario quiero poder eliminar mis archivos que han sido registrados previamente.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de derecho al olvido.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada
HU-006	Como usuario, quiero aceptar los términos y condiciones explícitamente antes de registrar cualquier archivo para garantizar mi consentimiento.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de consentimiento explícito.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada

Table 6: Product backlog - Sprint 1

ID	HU	Tareas	Estado
HU-007	Como propietario de un archivo, quiero consultar la lista de usuarios que tienen acceso a mi archivo.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de consulta de accesos.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada
HU-008	Como propietario, quiero garantizar que las transferencias no se registren más de una vez.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de verificación de transferencia.	No iniciada
HU-009	Como usuario, quiero que la plataforma utilice Pinata .	Análisis y diseño de esta historia.	No iniciada
		Configuración.	No iniciada
		Implementación.	No iniciada
HU-010	Como usuario, quiero renovar el acceso a mi archivo.	Análisis y diseño de esta historia.	No iniciada
		Implementar la funcionalidad de renovar acceso.	No iniciada
		Agregar funcionalidad en el frontend.	No iniciada

Table 7: Product backlog - Sprint 1

Una vez finalizada esta segunda iteración, durante la reunión de la revisión se evaluó el progreso alcanzado y se analizaron los cambios que se obtuvieron en el sprint backlog. En la tabla se observan dos tareas que no se pudieron iniciar. Dado que la fecha de entrega se aproximaba y la redacción de la memoria del proyecto aún no había comenzado, se decidió posponer la implementación de esta funcionalidad para futuros trabajos. Este enfoque permitió priorizar las tareas más importantes y garantizar la entrega del proyecto dentro del plazo establecido

Sprint 3. Una vez finalizado el proceso de desarrollo, se procedió a realizar la redacción de la memoria del proyecto.

Como se observa, el desarrollo supuso más tiempo del estimado debido a contratiempos en ciertas funcionalidades, especialmente en el cifrado. Por lo que, el hecho de haber dejado una holgura de 4 días, permitió mayor flexibilidad a la hora de gestionar los retrasos y mantener el plazo de entrega del proyecto el 8 de diciembre.Aunque fue necesario posponer una funcionalidad para futuros desarrollos, este margen adicional garantizó la finalización de las tareas críticas dentro del tiempo establecido.

17.1 Referencias

[1] *Opensea*. [Online]. Available: https://www.criptonoticias.com/criptopedia/opensea-nft-mercado-ethereum-polygon-solana-seaport-subastas-comprar-vender/.

[2] *Filecoin*. [Online]. Available: https://docs.filecoin.io/basics/what-is-filecoin/.

[3] *Snapshot*. [Online]. Available: https://docs.ipfs.tech/case-studies/snapshot/.

[4] *Openseaseg*. [Online]. Available: https://www.learningheroes.com/aprende-crypto/opensea-que-es-y-como-funciona.

[5] *Fileseg*. [Online]. Available: https://kriptomat.io/es/cotizacion-criptomonedas/filecoin-fil-precio/que-es/.

[6] *Metodologia*. [Online]. Available: https://biblus.us.es/bibing/proyectos/abreproy/70201/fichero/02+-+Ingenieria+del+Software.pdf.

[7] *Eleccion metodologia*. [Online]. Available: https://www.eseibusinessschool.com/es/agile-vs-traditional-project-management/.

[8] *Formulario metodologia*. [Online]. Available: https://openaccess.uoc.edu/bitstream/10609/73151/6/jgarcianavarroTFG0118memoria.pdf.

[9] *Scrum*. [Online]. Available: https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf.

[10] *Ethereum*. [Online]. Available: https://zeyo.io/diccionario/que-es-ethereum-cuales-son-sus-beneficios-y-desventajas/#:~:text=Beneficios%20de%20Ethereum&text=Estos%20beneficios%20incluyen%20seguridad%20mejorada,eficientes%20con%20la%20mejor%20calidad..

[11] *Nfts*. [Online]. Available: https://coleccionnft.es/ethereum-y-nfts-guia-para-principiantes-y-expertos/#:~:text=Una%20de%20las%20principales%20ventajas,transparencia%20en%20las%20operaciones%20realizadas..

[12] *Ipfs*. [Online]. Available: https://www.coinbase.com/es-es/learn/crypto-glossary/what-is-the-interplanetary-file-system-ipfs-in-crypto.

[13] *Drawio*. [Online]. Available: https://www.getapp.es/software/91063/draw-io.

[14] *Ganttpro*. [Online]. Available: https://help.ganttpro.com/hc/es/articles/360019585457--Qu%C3%A9-es-GanttPRO.

[15] *Latex*. [Online]. Available: https://guiasbib.upo.es/latex.

[16] *Lean*. [Online]. Available: https://www.linkedin.com/advice/3/how-do-you-integrate-lean-canvas-your-agile-scrum?lang=es&originalSubdomain=es.