GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE VERIFICACIÓN DE CERTIFICADOS DE AUTENTICIDAD BASADO EN BLOCKCHAIN

Autor: Juan Antonio Gil Carrera

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid – Julio 2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Sistema de Verificación de Certificados de Autenticidad basado en Blockchain en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2021/22 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Firmado por GIL CARRERA JUAN ANTONIO - ***2106** el día 05/07/2022 con un certificado emitido por AC FNMT Usuarios

Fdo.: Juan Antonio Gil Carrera Fecha: 05/07/2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

71216314B ATILANO FERNÁNDEZ-PACHECO

Firmado digitalmente por 71216314B ATILANO FERNÁNDEZ-PACHECO
Fecha: 2022.07.05 16:53:01 +02'00'

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: 05/07/2022

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE VERIFICACIÓN DE CERTIFICADOS DE AUTENTICIDAD BASADO EN BLOCKCHAIN

Autor: Juan Antonio Gil Carrera

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid – Julio 2022

Agradecimientos

Quiero agradecer a mi familia, en especial a mis padres, por su paciencia, su ayuda y haberme dado la oportunidad de llegar a este punto. Han estado siempre apoyándome y apoyándome a lo largo de los altibajos de mi vida y les estoy agradecido.

A mi tía Beatriz, por su interés en mi Trabajo Fin de Grado, que me ha llevado a realizar un mejor trabajo y con meno fallos.

A mis amigos, que siempre están apoyándome y dispuestos a ayudarme de manera desinteresada, sea cual sea la situación.

Por último, agradecer a mi director del proyecto, Atilano, por permitirme realizar este trabajo, animándome en todo momento y por la paciencia que ha tenido ante mi escaso tiempo por estar realizando prácticas.

SISTEMA DE VERIFICACIÓN DE CERTIFICADOS DE AUTENTICIDAD BASADO EN BLOCKCHAIN

Autor: Gil Carrera, Juan Antonio
Director: Fernández-Pacheco Sánchez-Migallón, Atilano
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas)

RESUMEN DEL PROYECTO

En este Trabajo Fin de Grado se ha diseñado y desarrollado un sistema, comprendido por una web, una REST API y un contrato inteligente. Este desarrollo facilita los tramites de digitalización de certificados de autenticidad almacenado información relevante del certificado en la cadena de bloques de la red Ethereum. Con el fin de crear un sistema que pueda ser usado por todo el mundo, tanto desde la web como la API, y hacerlo de manera descentralizada para poder alcanzar una validez global de los documentos subidos al sistema.

Palabras clave: Blockchain, contratos inteligentes, Certificados de Autenticidad

1. Introducción

En 2008 Satoshi Nakamoto, publicó un *paper* en el que explicaba *Bitcoin: un sistema de dinero electrónico peer-to-peer*[1]. En dicho documento explica por primera vez el concepto de criptomoneda. Su objetivo era crear una forma de pago electrónica, instantánea, segura y sin depender de una institución financiera. Propuso un sistema en el que dos usuarios se transfieren dinero de manera directa, utilizando una red *peer-to-peer*. Esta red sella la transacción y la almacena en una cadena continua de *proof-of-work*, quedando por tanto registrada y por tanto, no pudiéndose modificar sin rehacer el *proof-of-work*. Explicando de tal manera el concepto de Blockchain [1].

En 2014 se extrapolo el uso de Blockchain de las criptomonedas, a los contratos inteligentes. Estos fueron introducidos por Vitalik Buterin, cuando publicó un *paper* titulado: *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform* [2]. En este *paper* introdujo el concepto de *Smart Contracts*. Este nuevo uso de la red Blockchain, se conoce como Blockchain 2.0.

Los Smart Contracts son pequeños programas que corren en la red Blockchain. Estos, se regulan de manera automáticamente, ejecutándose cuando se cumplen algunas condiciones en concreto. La ventaja que ofrece con respecto a la forma tradicional de ejecutar código, es que al estar en la red Blockchain, nadie puede modificar dicho código, previniendo fraude y hackeos [3].

Este uso de la tecnología Blockchain es el que se implementará en los procesos de digitalización de certificados de autenticidad, mejorando los sistemas actuales y buscar una validez a nivel mundial.

2. Definición del proyecto

En este Trabajo Fin de Grado se busca diseñar y crear un sistema descentralizado, para mejorar tanto en seguridad, transparencia y aceptación global, los certificados de autenticidad digitales.

1 Red entre pares, red de igual a igual, P2P

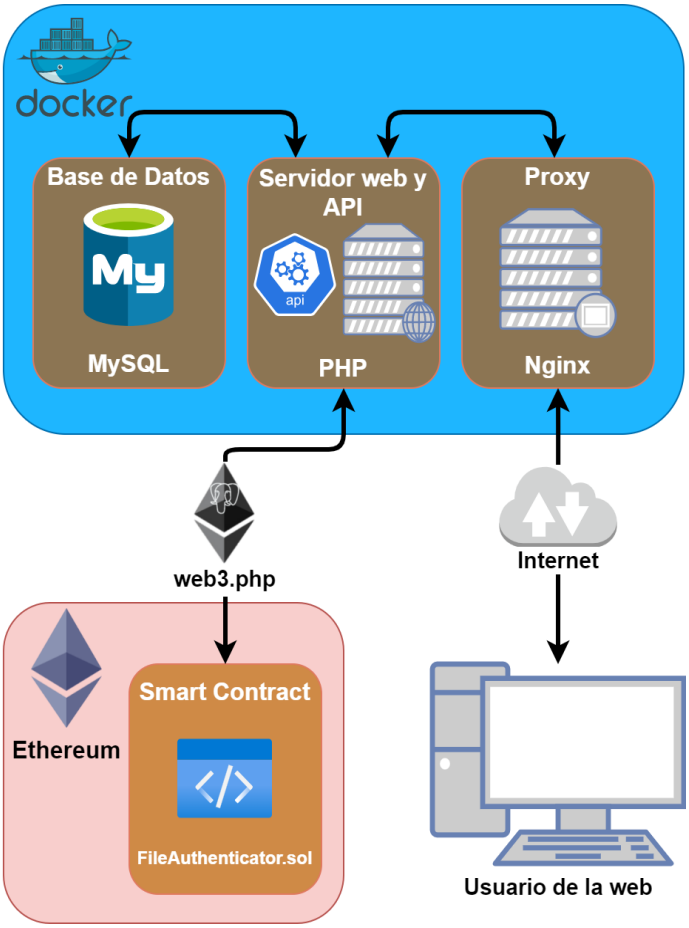
Para ello se desarrollará un contrato inteligente, el cual será desplegado en Ethereum. Este contrato inteligente, permitirá a los usuarios almacenar informacion para determinar si el archivo a sido modificado e informacion extra, para facilitar la verificación de la autenticidad de la obra. También se desarrollará un servidor REST API y página web, para poder así facilitar el uso al mayor número de personas.

Para este desarrollo se analizarán los requisitos estudiando las historias de usuario, en función de estos se determinará la funcionalidad a desarrollar, tanto en el contrato inteligente, como en el servidor y la REST API. Posteriormente, tras haber realizado la etapa de diseño, se procederá al desarrollo de *software*. Durante el proceso del desarrollo del software se realizaron pruebas de la web por diversas personas y se valoró su *feedback* para mejorar el desarrollo o corregir posibles fallos que hubieran aparecido.

Se ha determinado que la funcionalidad que se desea desarrollar en este trabajado, permita al usuario subir archivos a la blockchain con la informacion necesaria, que pueda añadir un co-dueño, que pueda cambiar el dueño por una venta y por último que pueda en función de un archivo obtener toda su información que se encuentra almacenada en el blockchain.

3. Descripción del sistema

El sistema desarrollado se compone por dos grandes bloques, como se puede ver en la *Figura 1*, la red de Blockchain, en este caso Ethereum, y Docker, donde se encuentran tres contenedores, un servidor proxy, el servidor web y API y por último la base de datos.



Fuente: Elaboración propia
Figura 1: Diseño de la arquitectura del sistema

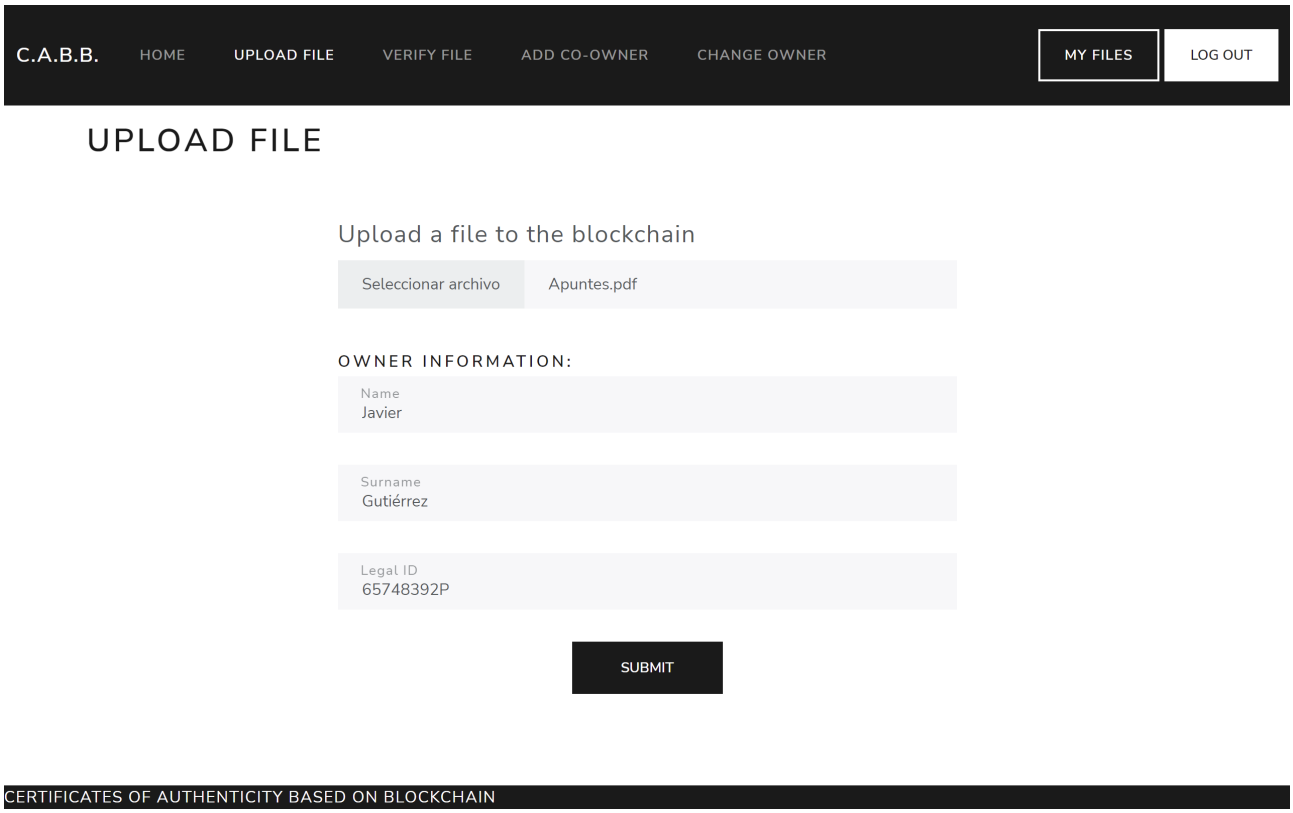
Empezando por la red de blockchain, se ha elegido Ethereum, debido a su robustez, amplia documentación y soporte en cuanto al desarrollo de los contratos inteligentes. Es en esta misma donde se almacenará la información critica debido a las ventajas que aporta respecto a la integridad de la información. Cada vez que el sistema quiera realizar una interacción con la blockchain, este llamará al contrato inteligente para poder obtener o añadir la información deseada.

El bloque principal del sistema desarrollado en este Trabajo Fin de Grado es Docker. Se ha elegido Docker por las ventajas y facilidades que ofrece en todas las etapas de vida del código. Permite tener un entorno de desarrollo idéntico en distintas maquinas independiente del dispositivo en el que se este y el software que tenga previamente instalado, porque Docker cuando crea los contenedores, instalará lo necesario en cada uno de ellos. Por otro lado de cara al despliegue, permite tener los servicios separados lo que los hace, por diseño, independientes entre ellos. Esto permite la creación y destrucción de contenedores en función de la demanda del trafico en cada instante realizando de manera muy sencilla el escalado horizontal.

4. Resultados

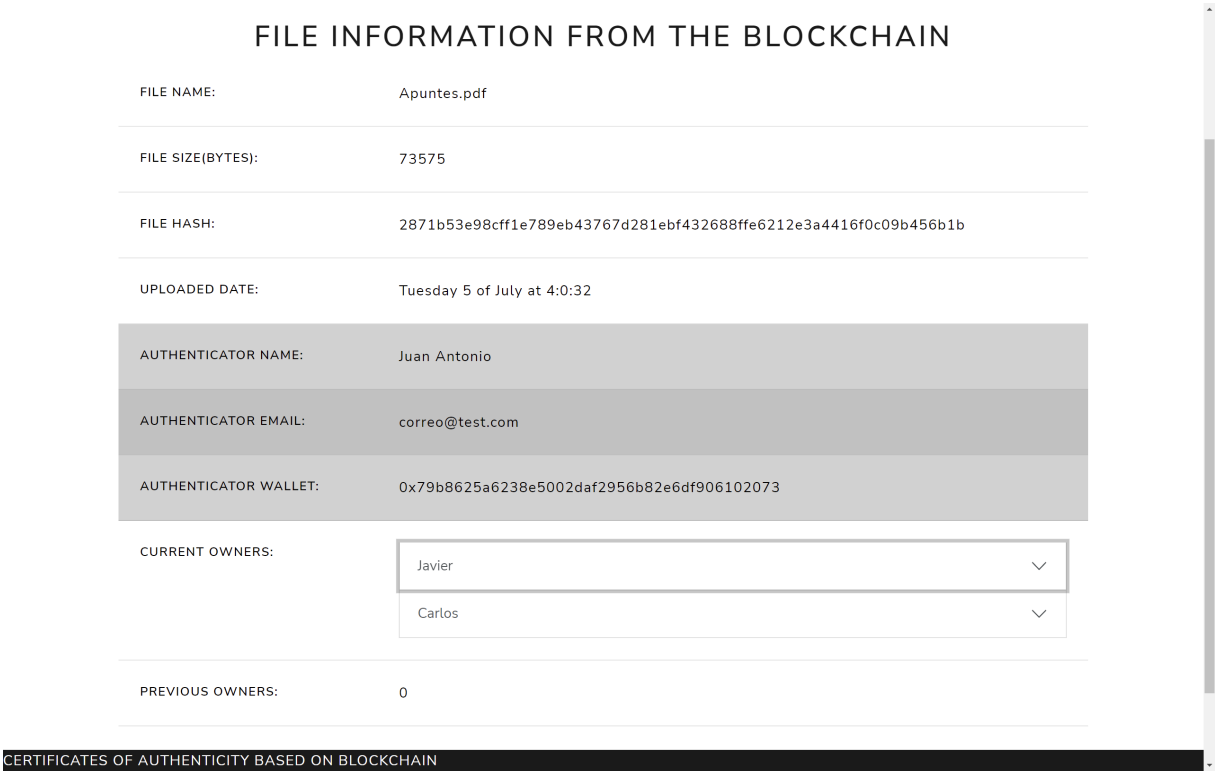
Los resultados de este Trabajo Fin de Grado han sido satisfactorios. Se ha conseguido desarrollar e implementar un sistema que interactué con la cadena de bloques mediante un contrato inteligente. Para facilitar el uso del contrato inteligente, se desarrollo una web y un API. La web es completamente funcional y toda las acciones que se pueden realizar o bien llamando al contrato inteligente o a la API, se pueden realizar desde la web.

Esta web posee varias vistas desde donde se pueden realizar distintas acciones. En la vista que se muestra en la *Figura 2* se puede ver los pasos seguidos para subir un nuevo archivo a la blockchain. Por otro lado en la *Figura 3* se aprecia como seria el proceso de verificación del mismo archivo que se ha subido en el ejemplo anterior.



Fuente: Elaboración propia

Figura 2: Vista para subir un fichero a la cadena de bloques



Fuente: Elaboración propia

Figura 3: Vista de verificación del archivo Apuntes.pdf

5. Conclusiones

En este Trabajo Fin de Grado se han alcanzado los objetivos marcados al desarrollar un contrato inteligente que permita almacenar informacion de un certificado de autenticidad para que así, se pueda garantizar que el pdf es completamente idéntico al subido a la cadena de bloques además de almacenar informacion extra para complementar el certificado y así en caso de querer volver certificar la obra, se tenga información de los dueños anteriores facilitando el estudio del experto.

El sistema desarrollado, tiene rutas para seguir desarrollándose. Habría que revisar bien el flujo de autenticidad, para garantizar que cumple con los estándares actuales. También se tendria que investigar la integración con carteras de criptomonedas, de esta forma el usuario tiene más confianza al realizar las transacciones. Aún así el producto final entregado, es completamente funcional.

6. Referencias

[1] Satoshi Nakamoto (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

[2] Vitalik Buterin (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*. https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

[3] Robert Sheldon (9 de agosto de 2021). *A timeline and history of blockchain technology*. https://whatis.techtarget.com/feature/A-timeline-and-history-of-blockchain-technology

VERIFICATION SYSTEM OF CERTIFICATES OF AUTHENTICITY BASED ON BLOCKCHAIN

Author: Gil Carrera, Juan Antonio
Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano
Collaborating Entity: ICAI – Universidad Pontificia Comillas)

ABSTRACT

On this Final Degree Project a system has been designed and develop, which consists of a smart contract, a web server and a REST API. It aims to make the current digitalization processes much more simple, by storing on the blockchain relevant information about the certificate. The goal is to create a system which could be used globally, either by calling the smart contract directly, by using the web or using the REST API. Making it on a decentralized network, makes it easier getting a globally used and accepted system of digitalizing printed certificates.

Keywords: Blockchain, smart contract, certificates of authenticity

1. Introduction

In 2008 Satoshi Nakamoto publish a paper explaining *Bitcoin: A Peer-to-Peer Electronic Cash System*. On this paper the concept of cryptocurrency was explained for the first time. Its objective was to create an electronic, instant, secure payment method without depending on a financial institution. A system was proposed in which two users transfer money directly, using a peer-to-peer network. This network seals the transaction and stores it in a continuous proof-of-work chain, thus remaining registered and therefore not being able to be modified without redoing the proof-of-work. Explaining the concept of Blockchain [1].

In 2014, a new use case of Blockchain was created, smart contracts. They where introduced by Vitalik Buterin when he publish a paper called *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform* [2]. With this paper and the introduction of the concept of smart contracts, Blockchain 2.0 was created.

Smart Contracts are small programs that run on the Blockchain network. These are regulated automatically, running when certain specific conditions are met. The advantage it offers over the traditional way of executing code is that by being in the Blockchain network, no one can modify the code, preventing fraud and hacking [3].

This use of Blockchain technology is what will be implemented in the digitization processes of certificates of authenticity, improving current systems and seeking worldwide validity.

2. Project definition

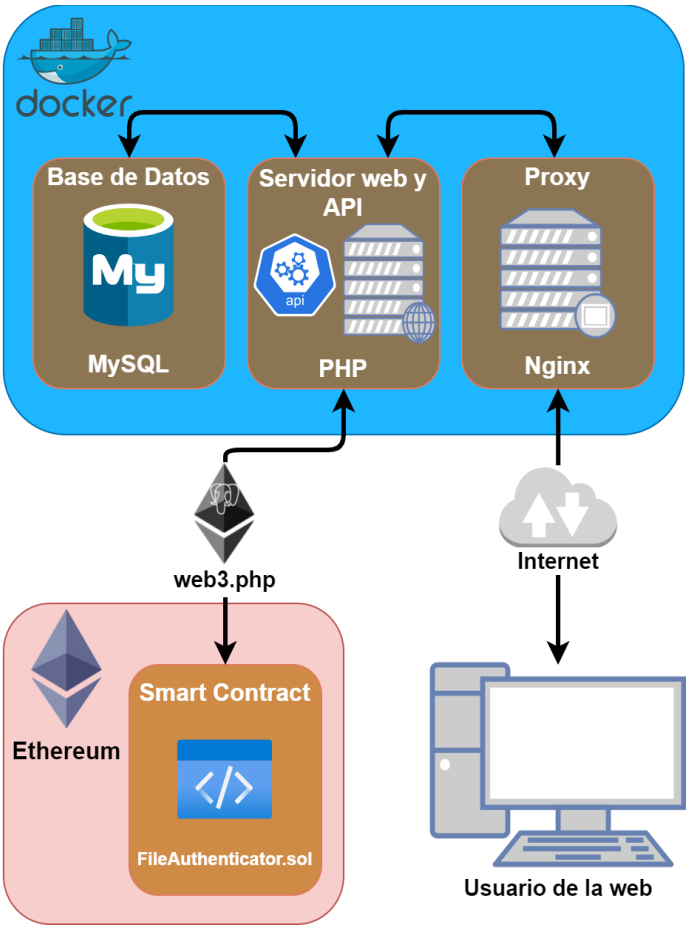
This Final Degree Project seeks to design and create a decentralized system to improve both security, transparency and global acceptance of digital certificates of authenticity. For this, a smart contract will be developed, which will be deployed in Ethereum. This smart contract will allow users to store information to determine if the file has been modified and extra information to facilitate verification of the authenticity of the art. A REST API server and web page will also be developed, in order to facilitate its use to the greatest number of users.

For this development, the requirements will be analyzed by studying the user stories, based on these the functionality to be developed will be determined, both in the smart contract, as well as in the server and the REST API. Subsequently, after having carried out the design stage, the software development will proceed. During the software development process, tests of the website were carried out by various users and their feedback was valued to improve the development or correct possible errors that had appeared.

It has been determined that the functionality to be developed in this Final Degree Project will allow the user to upload files to the blockchain with the necessary information, they will be able to add a co-owner, they will be able to change the owner due to a sale and finally, they will be able to, based on a file get all your information that is stored in the blockchain.

3. System description

The developed system is composed of two large blocks, as can be seen in *Figura 4*, the Blockchain network, in this case Ethereum, and Docker, where there are three containers, a proxy server, the web server and API and the database.



Fuente: Elaboración propia
Figura 4: Diseño de la arquitectura del sistema

Starting with the blockchain network, Ethereum has been chosen, due to its robustness, extensive documentation and support in terms of the development of smart contracts. The Ethereum network is where the critical information will be stored due to the advantages it brings regarding the integrity of the information. Every time the system wants to perform an

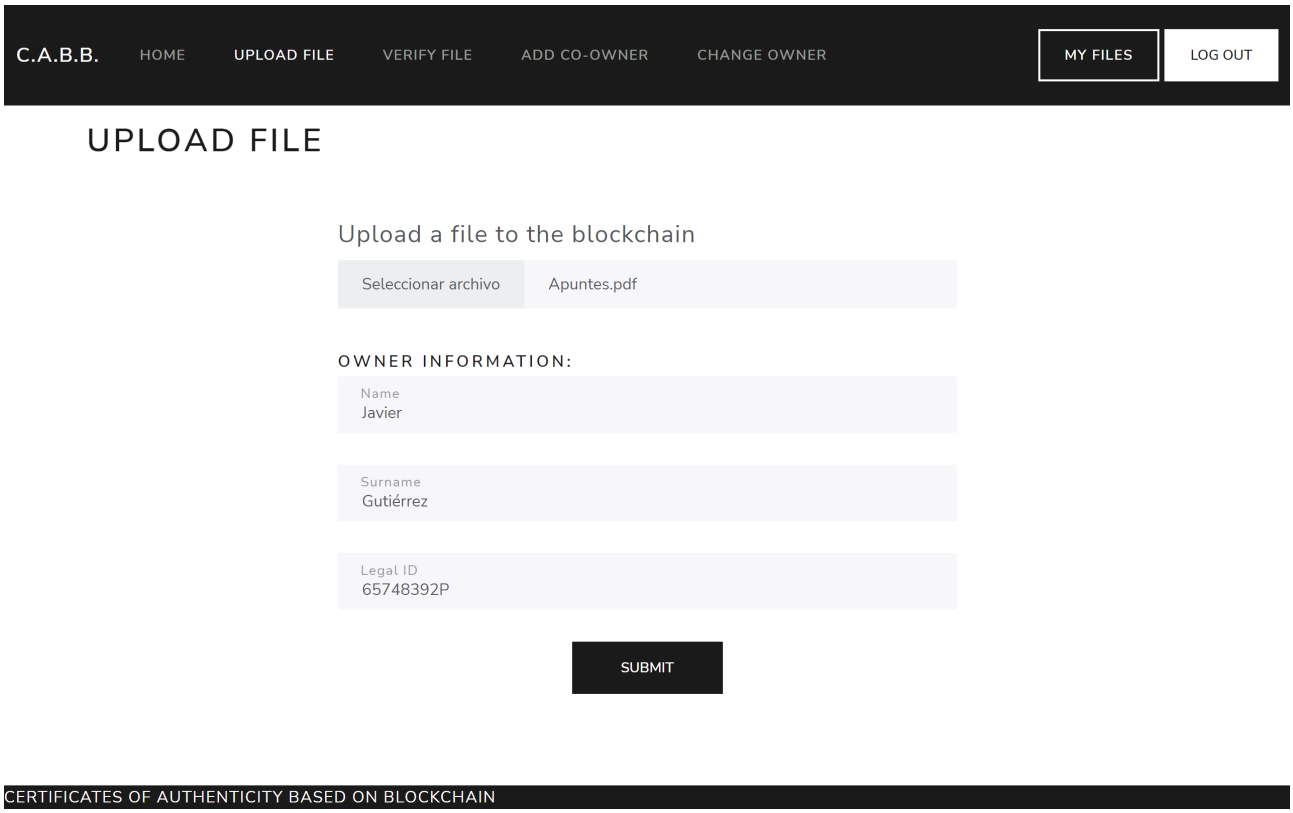
interaction with the blockchain, it will call the smart contract to obtain or add the desired information.

The main block of the system developed in this Final Degree Project is Docker. Docker has been chosen for the advantages and facilities it offers in all stages of the life of the code. It allows you to have an identical development environment on different machines regardless of the device you are on and the software you have previously installed, because when Docker creates the containers, it will install the necessary software in each one of them. On the other hand, with regard to deployment, it allows having separate services, which makes them, by design, independent of each other. This allows the creation and destruction of containers based on traffic demand at any given moment, making horizontal scaling very simple.

4. Results

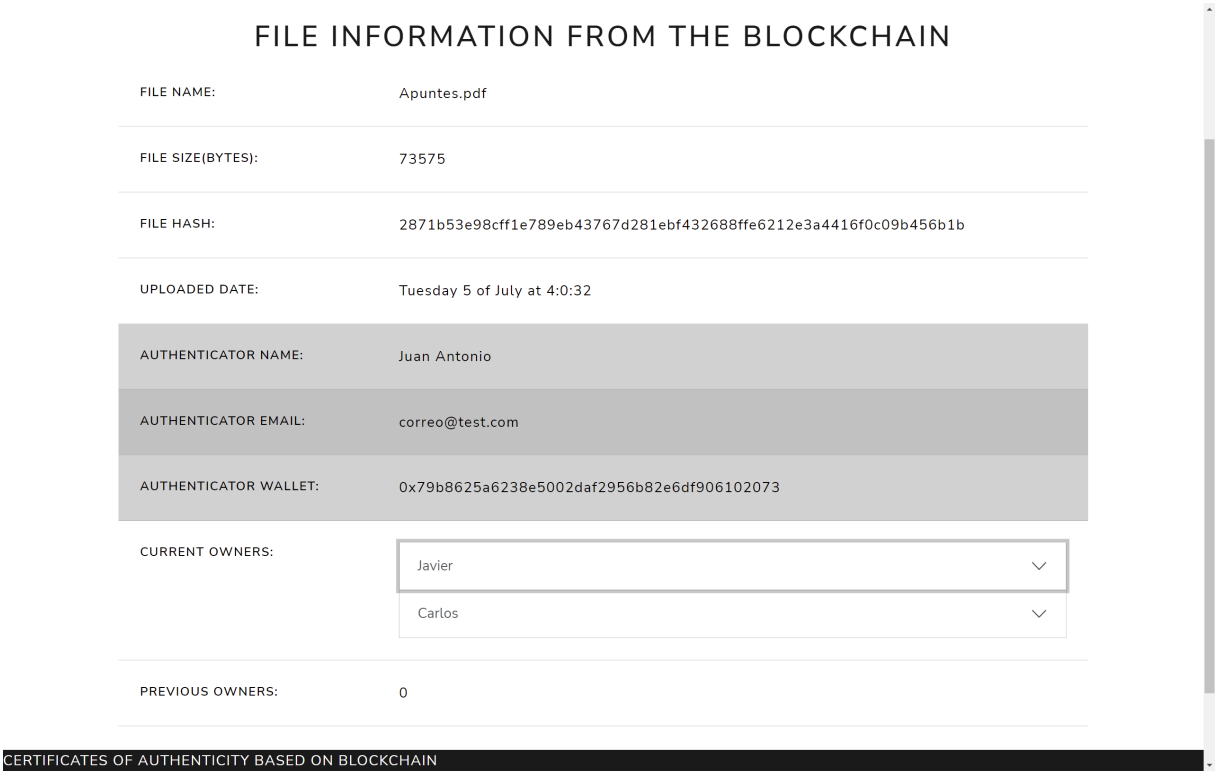
The results of this Final Degree Project have been satisfactory. It has been possible to develop and implement a system that interacts with the block chain through a smart contract. To facilitate the use of the smart contract, a website and an API were developed. The website is fully functional and all the actions that can be performed either by calling the smart contract or the API can be performed from the web.

This website has several views from where you can perform different actions. In the view shown in the *Figura 5* you can see the steps followed to upload a new file to the blockchain. On the other hand, in the *Figura 6* it can be seen how the verification process is for the same file that has been uploaded in the previous example.



Fuente: Elaboración propia

Figura 5: Vista para subir un fichero a la cadena de bloques



Fuente: Elaboración propia

Figura 6: Vista de verificación del archivo Apuntes.pdf

5. Conclusion

In this Final Degree Project, the objectives set have been achieved by developing a smart contract that allows the storing of information from a certificate of authenticity so that it can be guaranteed that the pdf is completely identical to the one uploaded to the chain of blocks also additional information will be stored to complement the certificate and thus in case of wanting to certify the work again, information on the previous owners is available, facilitating the study of the expert.

The developed system has room for growth and continue its develop. The authentication flow should be thoroughly reviewed to ensure that it meets current standards. Integration with cryptocurrency wallets should also be investigated, this way the user has more confidence when carrying out transactions. Even so, the final product delivered is fully functional.

6. References

[1] Satoshi Nakamoto (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

[2] Vitalik Buterin (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*. https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

[3] Robert Sheldon (2021). *A timeline and history of blockchain technology*. https://whatis.techtarget.com/feature/A-timeline-and-history-of-blockchain-technology

Indice

Capítulo 1 Introducción 1

Capítulo 2 Descripción de las Tecnologías 4

2.1 Blockchain 4
2.2 Ethereum 4
2.2.1 Smart contract 5
2.3 Aplicación 6
2.3.1 Back-end 6
2.3.2 Base de Datos 8
2.3.3 Front-end 9
2.4 Otras Herramientas 10
2.4.1 Docker 10
2.4.2 Composer 10
2.4.3 Visual Studio Code 10
2.4.4 Ethereum Remix 11
2.4.5 Ganache 11

Capítulo 3 Estado de la Cuestión 12

3.1 Blockchain 12
3.1.1 Función hash criptográfica 12
3.1.2 Transacciones 13
3.1.3 Criptografía de clave asimétrica 14
3.1.4 Direcciones 15
3.1.5 Registro contable 15
3.1.6 Bloques 16
3.1.7 Cadena de bloques 17
3.2 Protocolo de consenso 18
3.2.1 Prueba de trabajo (PoW) 19
3.2.2 Prueba de participación (PoS) 19
3.2.3 Prueba de capacidad (PoC) 20
3.2.4 Prueba de tiempo transcurrido (PoET) 20
3.2.5 Tolerancia práctica de fallos bizantinos (PBFT) 20
3.3 Tipos de redes blockchain 20
3.3.1 Sin permiso o pública 20
3.3.2 Con permiso o privada 21
3.4 Procesos actuales de certificación 21
3.5 Proyectos de certificación basados en blockchain 22

Capítulo 4 Definición del trabajo 24

4.1 Motivación 24

4.2 Objetivos 24
4.3 Metodología 25
4.4 Estimación económica 26
4.4.1 Recursos materiales 27
4.4.2 Recursos humanos 29
4.4.3 Presupuesto total 30
Capítulo 5 Sistema Desarrollado 33
5.1 Análisis del modelo 33
5.1.1 Historias de usuario 34
5.1.2 Casos de uso 36
5.2 Diseño 37
5.2.1 Diseño de la arquitectura 37
5.2.2 Estructura de datos 39
5.2.3 Diagrama de secuencia 44
5.3 Implementación 46
5.3.1 Contrato inteligente 47
5.3.2 API 50
5.3.3 Página web 52
Capítulo 6 Análisis de resultados 54
Capítulo 7 Conclusiones y trabajos Futuros 62
ANEXO I: Alineación del proyecto con los ODS 66
ANEXO II: Manual de instalación 68
A. II 1 Ganache 68
A. II 2 Remix IDE 69
A. II 3 Docker 72
ANEXO III: Arboles Merkle 75

Índice de figuras

Figura 1: Diseño de la arquitectura del sistema....10
Figura 2: Vista para subir un fichero a la cadena de bloques....12
Figura 3: Vista de verificación del archivo Apuntes.pdf....12
Figura 4: Diseño de la arquitectura del sistema....15
Figura 5: Vista para subir un fichero a la cadena de bloques....16
Figura 6: Vista de verificación del archivo Apuntes.pdf....17
Figura 7: Lenguajes de programación de páginas web en función de su uso y cantidad de tráfico....7
Figura 8: Ejemplo de una transacción....14
Figura 9: Cadena de bloques genérica....18
Figura 10: Ejemplo de cadena de bloques alterada....18
Figura 11: Flujo de PoW....19
Figura 12: Panel Kanban del Trabajo Fin de Grado....26
Figura 13: Ventas de obras de arte a nivel mundial 2009-2021....30
Figura 14: Casos de uso del sistema desarrollado....36
Figura 15: Diseño de la arquitectura del sistema....37
Figura 16: Diagrama Entidad Relación de la base de datos....41
Figura 17: Diagrama de la clase del contrato inteligente....43
Figura 18: Diagrama de secuencia para la subida de un certificado a la blockchain....45
Figura 19: Diagrama de secuencia para la verificación y de un certificado....46
Figura 20: Diagrama de navegabilidad de la página web desarrollada....52
Figura 21: Pantalla de inicio de la web....54
Figura 22: Pantalla de verificación de archivos de la web....55
Figura 23: Resultados de una verificación de un archivo en la web....56
Figura 24: Resultados de una verificación con detalle en el dueño actual....56
Figura 25: Pantalla de registro en el sistema desde la web....57
Figura 26: Vista de subida de un nuevo archivo a la cadena de bloques....58
Figura 27: Modal de confirmación de la subida de un fichero....58
Figura 28: Vista de archivo subido a la cadena de bloques satisfactoriamente....59
Figura 29: Vista de añadir un co-dueño a un archivo....60
Figura 30: Vista de verificación del archivo Apuntes.pdf....60
Figura 31: Vista de los archivos de un usuario....61
Figura 32: Objetivos de desarrollo sotenible de las Naciones Unidas....66
Figura 33: Paso 1 de la configuración de Ganache....68
Figura 34: Paso 2 de la configuración de Ganache....69

Figura 35: Abrir la extensión de Ganache....70
Figura 36: Conectar Remix a la red de Ethereum....70
Figura 37: Compilar el contrato inteligente....71
Figura 38: Desplegar en contrato inteligente en la red de Ethereum....71
Figura 39: Donde obtener la dirección y el ABI del contrato desplegado en la red de blockchain....72
Figura 40: Varibles a modificar de MySQL....73
Figura 41: Árbol Merkle....75

Índice de tablas

Tabla 1: Ejemplo de texto de entrada y su correspondiente valor aplicando SHA-256.....12
Tabla 2: Diagrama de GANT del cronograma del proyecto.....26
Tabla 3: Especificaciones de ordenador portátil.....27
Tabla 4: Especificaciones de monitor.....27
Tabla 5: Desglose de las horas de dedicación de este Trabajo Fin de Grado.....28
Tabla 6: Resumen de los costes materiales.....29
Tabla 7: Coste humano del desarrollo.....29
Tabla 8: Coste anual de mantenimiento del desarrollo.....30
Tabla 9: Ingresos anuales en el escenario uno.....31
Tabla 10: Calculo de los beneficios brutos anuales en el escenario uno.....31
Tabla 11: Ingresos anuales en el escenario dos.....32
Tabla 12: Calculo de los beneficios brutos anuales en el escenario dos.....32

Capítulo 1 Introducción

La tecnología ha ido evolucionando y creciendo a lo largo de los años. Los dispositivos se han ido haciendo más pequeños, más potentes y más eficientes. Estos avances en la tecnología nos han ayudado en las tareas que realizamos de manera diaria, tanto en el hogar como en el trabajo y aportándonos un ahorro de tiempo y/o económico. Estos avances han sido acogidos de manera positiva por la sociedad y las empresas. Entre estos avances destaca internet. Internet nos ha simplificado y facilitado la vida de una manera inimaginable, desde poder buscar información en todo tipo de fuentes, hasta poder estar en una reunión con personas de Japón y Canadá al mismo tiempo, sin que ninguno abandone el confort de su propia casa.

Frente a los avances de las nuevas tecnologías, en muchas situaciones, las leyes de cada país se quedan obsoletas. Esto podría generar situaciones en las que un individuo o empresa utiliza una nueva tecnología de una determinada manera durante años, hasta que la legislación de dicho país regula esta tecnología. En este momento, el individuo o la empresa, podría tener que cambiar por completo su forma de utilizar la tecnología, porque podría incluso ser ilegal. A este problema, se añade que las leyes entre diversos países varían, por diferencias sociales, culturales, políticas, etc.

Esta lenta evolución no solo sucede en la legislación de los países, sino también en el mundo empresarial. Así por ejemplo, ¿quien no ha tenido que ir presencialmente al banco para firmar un documento? Viviendo en la era de información y del Internet de las Cosas (IoT), esa acción de firmar un documento o presentar el DNI, se podría realizar de manera sencilla a través de internet. Esta lenta evolución y otros motivos llevaron a la aparición del Blockchain y las criptomonedas, en especial el Bitcoin.

En 2008, Satoshi Nakamoto publicó un *paper* en el que explicaba *Bitcoin: un sistema de dinero electrónico peer-to-peer*[2]. En dicho documento explica por primera vez el concepto de criptomoneda [1]. Su objetivo era crear una forma de pago electrónica, instantánea, segura y sin depender de una institución financiera. Propuso un sistema en el que dos usuarios se transfieren dinero de manera directa, utilizando una red *peer-to-peer*. Esta red sella la transacción y la almacena en una cadena continua de *proof-of-work.* Para así poder solucionar el problema del doble gasto, el cual nunca se ha podido corregir a la perfección en los sistemas actuales de banca. Con dicho *paper*, explicó las bases del concepto que se conoce como Blockchain.

Blockchain como cualquier tecnología, siguió avanzando y aportando nuevas posibilidades de uso según se desarrollaba dicha tecnología. Esta separación de la tecnología Blockchain de las criptomonedas, no sucedió hasta 2014, cuando Vitalik Buterin publicó un *paper* titulado: *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* En este *paper*

2 Red entre pares, red de igual a igual, P2P

introdujo el concepto de *Smart Contracts* [2]. Este nuevo uso de la red Blockchain, se conoce como Blockchain 2.0.

Blockchain 3.0 fue la siguiente versión, y en ésta se introdujo el concepto de aplicaciones descentralizadas (DApps)[3]. A diferencia de las aplicaciones tradicionales, que cuentan con un backend centralizado, las DApps poseen su backend en una red descentralizada, pudiendo también alojar de manera descentralizada el frontend [3].

La última versión de Blockchain, a día de hoy es Blockchain 4.0. Esta versión de Blockchain esta orientada en hacer esta tecnología útil para las industrias y empresas, Su objetivo consiste en adaptar DApps a las necesidades diarias de un negocio [4].

En este Trabajo Fin de Grado se ha realizado una implementación de los contratos inteligentes, pertenecientes a la tecnología Blockchain 2.0, al proceso de expedición de certificados de autenticidad y la verificación de su validez. Este desarrollo será presentado mediante una aplicación web y un servicio de REST API[4] [5].

Un certificado de autenticidad es un documento oficial que demuestra que el producto es auténtico y que cumple con unas características en concreto [5]. Por ejemplo en el caso de obras de arte, un certificado de autenticidad suele incluir el autor de la obra, una fotografía, datos específicos sobre la obra, los materiales utilizados, el estado del mercado y en algunas ocasiones su valor estimado [6].

Para obtener un certificado de autenticidad de una obra de arte, se debe acudir a un perito. Dicho perito realiza un estudio de la obra de arte y determina su autenticidad. Una vez éste ha determinado su autenticidad, se entregan los documentos impresos que acreditan dicha obra, poniendo en algunos casos una etiqueta de autenticidad en el dorso de la obra [7]. El perito que acredita la obra, debe haber estudiado un Grado y Máster que se encuentren relacionados con el objeto que se desea certificar, o haber realizado un curso oficial que le certifique como perito en ese ámbito. Dicho perito podría unirse a una asociación, como por ejemplo *ASPEJURE*[6], y así poder estar respaldado por la asociación, puesto que antes de entrar al cliente el certificado, manda a la asociación su estudio, para así poder recibir el visto bueno y avalar su certificación.

En el caso de querer realizar una copia del certificado, se debe realizar una compulsación. Una compulsación consiste en dar validez que la copia realizada es idéntica a la original. Este trámite de compulsación lo puede realizar la propia entidad que expidió el documento original o algún funcionario o entidad pública (por ejemplo un Ayuntamiento) [8]. Desde julio de 2006 el gobierno

3 https://ethereum.org/en/developers/docs/dapps/
4 Representational State Transfer Application Programming Interface (REST API)
5 https://www.ibm.com/es-es/cloud/learn/rest-apis
6 Asociación Profesional Colegial de Peritos Judiciales del Reino de España (ASPEJURE)

avala la compulsación electrónica de documentos originalmente a papel, mediante el uso de la *firma electrónica reconocida de un funcionario o empleado del Ministerio* [9]. Dicho documento digital posee la misma validez que el documento original.

Como se puede ver, tanto el proceso de compulsación digital como el de obtener el certificado de manera digital, son procesos largos y con muchas dependencias de terceros y de entidades gubernamentales. Por eso, con este Trabajo Fin de Grado se busca simplificar los procesos, haciendo que sean independientes de cualquier gobierno y empresa, para así poder llegar a una validez global.

Capítulo 2 Descripción de las Tecnologías

En este capitulo se mencionarán y explicarán las tecnologías y herramientas usadas para el desarrollo de este trabajo: Blockchain, Ethereum, Aplicación y otras tecnologías

2.1 BLOCKCHAIN

Blockchain, como se ha comentado en la Introducción, remonta sus orígenes al *paper* publicado por Satoshi Nakamoto en 2008 [1]. Dicho *paper* tiene como objetivo la creación de un sistema de pagos electrónicos, descentralizado, independiente, seguro y que no posea el problema del doble gasto. Al describir lo que llamó Bitcoin, describió lo que un futuro seria conocido como Blockchain.

Como explica IBM en su web, *Blockchain es un libro mayor compartido e inmutable que facilita el proceso de registro de transacciones y de seguimiento de activos en una red de negocios* [10]. El libro mayor al que se refiere IBM es la cadena de bloques que compone una red Blockchain, esa cadena es muy difícil de modificar. Para poder modificarla se debería poseer el control de por lo menos el 51% de los nodos pertenecientes a la red, para así llegar al consenso de que la información ilícita es correcta. Esto se debe al protocolo de consenso, el cual se explicará más adelante en el Estado de la Cuestión. Debido a esa dificultad de modificación se considera a la cadena de bloques inmutable. Aun siendo inmutable la información almacenada en la cadena de bloques, el proceso de registrar transacciones en la misma es sencillo, puesto que la operación se añadirá al próximo bloque, con la condición de que ésta sea verificada y aceptada por el resto de los nodos, mediante el protocolo de consenso.

Cabe destacar que la red Blockchain es compartida y descentralizada, puesto que todos los nodos que forman parte de ella deben poder acceder a ésta, tanto para poder verificarla como para añadir bloques adicionales.

Esta tecnología se ha usado en este Trabajo Fin de Grado, para almacenar los metadatos necesarios de los archivos que se desean subir a la red Blockchain al ser una red distribuida, segura, inmutable e independiente.

2.2 ETHEREUM

Ethereum fue propuesto por Vitalik Buterin en un *paper* que publicó en 2014 [2], como ya se ha mencionado en la Introducción. En dicho *paper* se explica que los objetivos de Ethereum son la mejora de los conceptos de *scripting*, *altcoins* y el permitir a los desarrolladores crear aplicaciones basadas en el protocolo de consenso de la red de Blockchain. Estas aplicaciones contarían con escalabilidad, estandarización, interoperabilidad y serán fáciles de programar. El objetivo principal

una plataforma que sirva para más que criptomonedas, y para ello se introdujo el concepto de *smart contract.*

2.2.1 SMART CONTRACT

Los *smart contracts* o contratos inteligentes, son contratos digitales que se almacenan en la cadena de bloques y que se ejecutan automáticamente cuando se cumplen las condiciones necesarias. Se puede ejecutar desde una transacción entre dos personas hasta registrar que un producto ha llegado a un almacén.

Estos contratos inteligentes son ejecutados en lo que se conoce como EVM[7]. Esta máquina virtual es una única entidad, la cual es ejecuta en múltiples nodos. Estos nodos sólo deben estar ejecutando un cliente de Ethereum para así poder formar parte de las red de ordenadores que ejecutan las EVM. De esta forma, el funcionamiento de la máquina es continuo, ininterrumpido e inmutable y debe cumplir estas características puesto que es donde todos los contratos inteligentes y las cuentas de Ethereum están almacenadas. Como bien se puede ver esta EVM es una parte troncal de Ethereum, puesto que sin esta la red no funcionaria, además esta máquina es la que define las reglas que determinar la validez de los bloques de la cadena [11]. Una forma de ver la EVM es como un súper ordenador descentralizado que mantiene la red de Ethereum y puede ser usado para ejecutar código.

Si se desea desarrollar un contrato inteligente y subirlo a la red de Ethereum, se puede hacer de manera sencilla. Los contratos inteligentes en Ethereum pueden ser programados en diversos lenguajes, cada uno con sus ventajas y desventajas.

- **Solidity**: es el primer lenguaje de programación para contratos inteligentes. Este fue muy influenciado por C++. Solidity es de alto nivel, orientado a objetos y de tipo estático[8] [12]. Un ejemplo de Solidity[9] seria:

```
pragma solidity ^0.8.0;
contract MyContract {
    function helloWorld() public pure returns (string memory) {
        return "Hello, World!";
    }
}
```

- **Vyper**: es un lenguaje de programación que se asemeja a Python y sigue sus mismos principios de simplifidad y seguridad [12]. Un ejemplo de Vyper seria[10]:

7 Ethereum Virtual Machine
8 Los tipos de las variables se conocen en tiempo de compilación
9 https://soliditylang.org/#selectedContract
10 https://gist.github.com/coding-ai/533e2be9ccdbd3a0d6dc70683257e6a9

```
greeting: public(String[100])

@external
def __init__():
    self.greeting = "Hello World!"

@external
def greet() -> String[100]:
    return self.greeting
```

- **Yul/Yul+**: Yul era llamado antiguamente como Julia, y está orientado a una optimización a alto nivel de los estados. Yul+ es una extensión a bajo nivel y eficiente de Yul [12].

- **Fe**: es el más moderno de los cuatro, su objetivo es que sea fácil de aprender. Esta inspirado en Python y Rust y es de tipo estático [12]. Un ejemplo de Fe seria[11]:

```
# The `contract` keyword defines a new contract type
contract HelloWorld:
    message: String<100>

    pub fn __init__(self, _message: String<100>):
        self.message = _message
    pub fn update(self, newMessage: String<100>):
        self.message = newMessage
    pub fn getMessage(self) -> String<100>:
        return self.message.to_mem()
```

En este trabajo se ha decido elegir Solidity como lenguaje de programación, debido a su funcionalidad y abundancia de herramientas para su desarrollo.

2.3 APLICACIÓN

En este apartado, detallamos las tecnologías y lenguajes de programación usados para el desarrollo de la aplicación que hace uso del contrato inteligente que se ha desarrollado para este trabajo. Se va a dividir en tres partes, la parte del servidor (Back-end), la parte del almacenamiento de los datos (Base de Datos) y por ultimo la parte de la página web (Front-end).

2.3.1 BACK-END

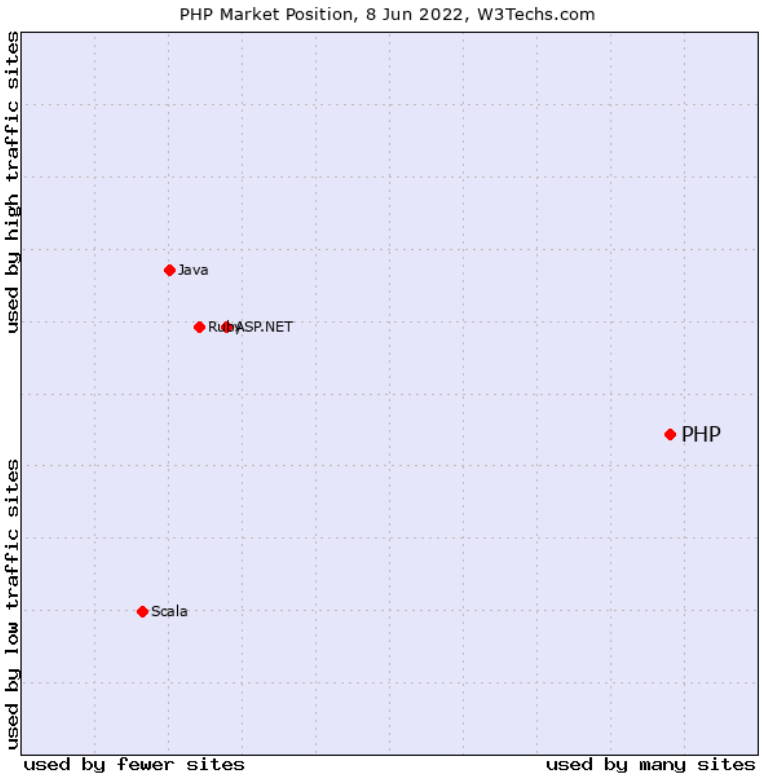
El servidor o *Back-end* hace referencia a la parte de la aplicación tradicional que se ha desarrollado para complementar y hacer más *user friendly* el uso del contrato inteligente. En éste se ha usado PHP como lenguaje de programación, Laravel como *framework*, Nginx como proxy y por

11 https://gist.github.com/adityak74/722b9a7de697962c3d3c25fa94e5463c

último la librería web3.php para conectar el Back-end a la red de Ethereum y así poder hacer uso del contrato inteligente que se ha desarrollado.

2.3.1.1 PHP

PHP es un lenguaje de programación para cualquier tipo de uso, pero está especialmente pensado para el desarrollo web. PHP (*Hypertext Preprocessor*[12]) [14] se caracteriza por ejecutarse en el lado del servidor, además de por ser rápido, flexible y pragmático [15]. Como se puede ver en la *Figura 7* [13], PHP es un lenguaje de programación bastante usado y en cuanto a la cantidad de tráfico que reciben se encuentra de entre las 5 primeras. Por todos estos motivos se decidió usar como lenguaje del *Back-end*.



Fuente: Cortesía de W3Techs [13]

Figura 7: Lenguajes de programación de páginas web en función de su uso y cantidad de tráfico

12 Lenguaje de Programación Interpretado

13 https://w3techs.com/diagram/market_technology/pl-php

2.3.1.2 Laravel

Laravel es un *framework* de código abierto para PHP, orientado a aplicaciones web. Las características de este framework es su arquitectura MVC (Modelo Vista Controlador)[14], *Eloquent ORM* (Mapeado Objeto-Relación)[15] que facilita la consulta de la base de datos, al tener cada tabla un modelo asociado en el código. Laravel también se caracteriza por sus migraciones de base de datos, puesto que permite realizar modificaciones en la base de datos, sin necesidad de tener que borrar las tablas. Por último también se caracteriza por las librerías y la modularidad que posee el *framework* porque aparte de contar con sus propias librerías, también se pueden importar librerías de la comunidad [16]. Se ha decidido usar este *framework* por las facilidades y herramientas que aporta al desarrollo del software en un proyecto con PHP

2.3.1.3 Nginx

Se trata de un servidor web, que se puede utilizar como un servidor proxy. El software es gratuito y de código abierto [17]. Se ha utilizado este software por la alta eficiencia y baja latencia que tiene. Al usarlo como proxy, se está usando para redirigir al usuario si intenta acceder a una URL que no existe, y también para limitar el tamaño máximo de los archivos que se van a subir al servidor, puesto que todas las peticiones pasan por este software.

2.3.1.4 Web3.php

Web3.php[16] es una adaptación a PHP de la librería Web3.js[17], que esta hecha para *Java Script*. Estas librerías permiten interactuar con la Blockchain de Ethereum. Tanto Web3,php como Web3.js son de código abierto y permiten por ejemplo llamar a métodos de contratos inteligentes en tiempo de ejecución, Se ha utiliza Web3.php porque se deseaba que fuera el propio servidor el que se conectara a la blockchain, y porque como se ha explicado antes, se ha elegido PHP como lenguaje de programación.

2.3.2 BASE DE DATOS

Como se ha mencionado antes, blockchain posee la capacidad de almacenar información, actuando como una base de datos, pero con las ventajas de blockchain, las cuales se han explicado anteriormente. Pero la gran desventaja es que cada escritura en la cadena de bloques, ya sea para realizar una inserción de datos o una actualización va a suponer un coste. Por estos motivos, para los datos no esenciales a la funcionalidad *core* que se busca, se ha decidido utilizar una forma de almacenamiento centralizada, usando MySQL.

14 *Model View Controller*
15 Object-Relational Mapping
16 https://github.com/web3p/web3.php
17 https://web3js.readthedocs.io/en/v1.7.3/

2.3.2.1 MySQL

Es un sistema de gestión de bases de datos relacionales. Este se caracteriza por ser de código abierto, gratuito, rápido, fiable, escalable y fácil de usar[18]. Está desarrollado y mantenido por *Oracle*[19]. Debido a todas estas ventajas y a la abundancia de recursos que hay para consultar, se ha decido hacer uso de este sistema.

2.3.3 FRONT-END

El *front-end*, corresponde a la parte que el usuario final utiliza para interactuar con el servidor del sistema. En este trabajo se ha desarrollado una página web como *front-end*. Ello permite a los usuarios acceder desde cualquier navegador, independientemente de su sistema operativo, siendo la cantidad de potenciales usuarios, todos lo que posean una conexión a internet.

Normalmente las páginas web se componen de tres partes: el contenido, el estilo y la funcionalidad. Para este trabajo se ha usado *Blade* para representar el contenido, *CSS* para darle el estilo deseado a dicho contenido y por último *Java Script* para aportarle funcionalidad al contenido mostrado.

2.3.3.1 Blade

Blade es un motor de plantillas que viene incluido en Laravel[20]. Estas plantillas son programadas en PHP, pero con anotaciones de Blade, lo que facilita el desarrollo del código. Estas vistas son compiladas a PHP puro y almacenadas en cache hasta que cambie algún dato de alguna de ellas. Esto añade una latencia mínima, tanto en tiempo de compilación como de uso. Debido a su integración con Laravel y la baja carga que añade a la aplicación, se ha decidido utilizar este lenguaje frente a otros.

2.3.3.2 CSS

Hojas de Estilo en Cascada (CSS)[21] es un lenguaje usado para dar estilo a los documentos escritos en HTML o XML. No es un lenguaje de programación como tal, sino más un lenguaje de marcado, que permite aplicar a elementos en concreto, estilos en concreto. Existen herramientas y bibliotecas que facilitan su desarrollo, pero en el desarrollo de este trabajo fin de grado, no se ha utilizado ninguna herramienta para generar los archivos CSS. Pero sí que se ha hecho uso de *Bootstrap*[22] y de una plantilla de *Bootswatch*[23] para facilitar el desarrollo del estilo de la web

18 https://www.mysql.com/about/
19 https://www.oracle.com/index.html
20 https://laravel.com/docs/9.x/blade
21 Cascading Style Sheets
22 https://getbootstrap.com/docs/5.2/getting-started/introduction/
23 https://bootswatch.com/

2.3.3.3 Java Script

Es un lenguaje de programación de alto nivel, está orientado a objetos y está débilmente tipado. Este es compatible con todos los navegadores web de manera nativa. Al igual que Java, éste se ejecuta en una máquina virtual, siendo mucho más rápida en comparativa. Esta es la única semejanza entre Java y Java Script, aparte de copmpartir la palabra Java en el nombre. Todo esto lo hace el lenguaje de programación ideal para añadir funcionalidad a la web.

2.4 OTRAS HERRAMIENTAS

En el transcurso del desarrollo de software de este trabajo, se han usado otras herramientas para poder realizar la aplicación.

2.4.1 DOCKER

Docker es una plataforma de código abierto, para todas las fases de vida de una aplicación. Docker permite empaquetar el software en lo que se llama contenedores[24]. Cada contenedor posee lo necesario para ejecutar el software. En la etapa de desarrollo, permite de manera sencilla tener todo lo necesario para poder trabajar en el software, puesto que sólo requiere tener un gestor de contenedores instalado para funcionar. En el nivel de despliegue la principal ventaja es que se puede configurar que en función de la carga de la aplicación, levante o apague más contenedores para así poder tratar todas las peticiones entrantes. Por estas ventajas que aporta en todas las etapas del desarrollo, se ha decidido utilizarlo.

2.4.2 COMPOSER

Es una herramienta de código abierto que gestiona las dependencias de PHP[25]. Esta herramienta permite de manera muy sencilla gestionar, instalar y actualizar todas las dependencias y librerías de un proyecto.

2.4.3 VISUAL STUDIO CODE

Visual Studio Code o VS Code es un editor de código fuente desarrollado por *Microsoft*[26]. VS Code se caracteriza por su capacidad de instalar extensiones. Estas extensiones permiten al editor ser usado para cualquier lenguaje. Además, al ser de código abierto, ha generado que estén disponibles bastantes extensiones realizadas por la comunidad. En este desarrollo, hay una extensión que se requiere, Ethereum Remix.

24 https://www.docker.com/
25 https://getcomposer.org/
26 https://code.visualstudio.com/

2.4.4 ETHEREUM REMIX

Ethereum Remix es una herramienta para el desarrollo de contratos inteligentes. Esta herramienta facilita la compilación, despliegue y ejecución de pruebas de los contratos inteligentes sobre la red de blockchain. En este desarrollo, se utilizado la extensión que hay disponible para VS Code[27], pero también existen versiones web[28].

2.4.5 GANACHE

Ganache permite crear y desplegar una red Ethereum en local. Esta herramienta de código abierto, facilita el desarrollo de los contratos inteligentes y aplicaciones que los usen, puesto que al ser una red de Blockchain privada, y tener el control absoluto sobre ésta, los despliegue de los contratos y las modificaciones de la cadena de bloques, no conllevan un cargo real. Esta herramienta permite simular una red real de Ethereum, de tal manera que el coste económico que conlleva es cero.

27 https://marketplace.visualstudio.com/items?itemName=RemixProject.ethereum-remix
28 https://remix.ethereum.org/

Capítulo 3 Estado de la Cuestión

En este capítulo se detallarán los trabajos y soluciones que existen en el ámbito de este trabajo fin de grado, haciendo mayor hincapié en la tecnología principal de este desarrollo, el blockchain.

3.1 BLOCKCHAIN

Como se ha comentado anteriormente, Blockchain surgió con el Bitcoin, en 2008. Desde entones esta tecnología ha ido evolucionando y según evolucionaba se le ha ido dando nuevos usos fuera del espacio de las criptomonedas, como por ejemplo, cuando se introdujeron los contratos inteligentes, surgieron nuevos usos como por ejemplo el uso de Blockchain en el sistema sanitario de Estonia[29]. Hicieron que los documentos médicos de los ciudadanos estuvieran en la blockchain, de esta forma si se modificaban quedaba un registro de ello, si se accedían a ellos, también quedaba registrado. Esta es solo una de las miles utilidades que se le esta dando y que se le puede dar a blockchain.

Blockchain es una tecnología que puede parecer compleja, por eso para que se pueda entender de una manera más sencilla se va a explicar componente a componente.

3.1.1 FUNCIÓN HASH CRIPTOGRÁFICA

Un componente importante de blockchain es el uso de funciones hash criptográficas para muchas operaciones. Estas funciones encriptan la entrada de una manera similar a las funciones hash. Lo que significa que para cualquier tipo de dato de entrada que se le aplique estas funciones, el resultado será el mismo para el mismo valor de entrada y la misma función, independiente del resto de variables. El mínimo cambio en la entrada, aunque sea por un bit, conlleva un resultado completamente distinto, como se puede ver en la *Tabla 1* los tres valores de entrada poseen una salida distinta, pero de la misma longitud, independiente da la entrada.

Texto entrada	Salida aplicando SHA-256
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
Hola mundo	0xca8f60b2cc7f05837d98b208b57fb6481553fc5f1219d59618fd025002a66f5c

Tabla 1: Ejemplo de texto de entrada y su correspondiente valor aplicando SHA-256

Las funciones hash criptográficas deben cumplir los siguientes puntos [18]:

1. Resistencia a preimágenes: dada una salida será difícil encontrar su entrada asociada.

2. Segunda resistencia a preimágenes: será difícil encontrar una entrada que posea la misma salida que otra entrada distinta.

29 https://nortal.com/blog/blockchain-healthcare-estonia/

3. Resistente a colisiones: será difícil encontrar dos entradas distintas con la misma salida.

Una función hash criptográfica usada por algunas redes de Blockchain es SHA-256[30], este tiene una salida de 256 bits, pero se suele representar como 64 caracteres hexadecimal, como se puede ver en la *Tabla 1*. Esto significa que hay $2^{256} \approx 10^{77}$ posibles valores. Este algoritmo y algunos otros esta definidos por el NIST[31] [32].

Las funciones hash criptográficas en una red Blockchain son usadas de distintas maneras:

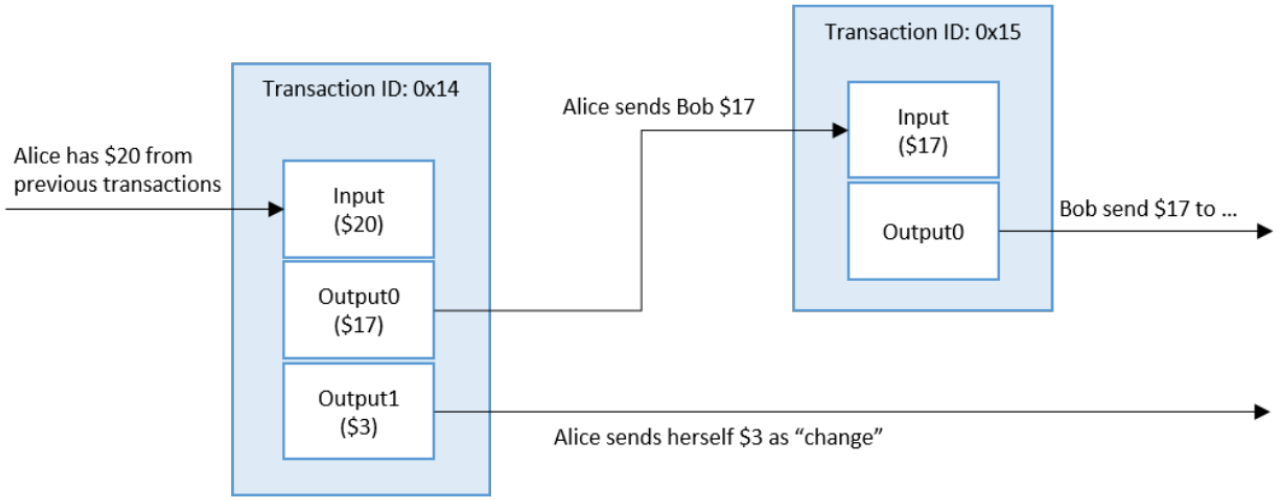
- Direcciones: se detallará más adelante en *Direcciones*
- Crear identificadores únicos
- Securizar los datos de un bloque: cuando se crea un bloque, se calcula su hash y se añade a la cabecera de este mismo, así se puede verificar que el contenido no ha sido modificado desde su creación.
- Securizar la cabecera del bloque: se calcula el hash de la cabecera del bloque actual y el resultado será incluido en la cabecera del próximo bloque de la cadena.

3.1.2 TRANSACCIONES

Una transacción en Blockchain representa una interacción entre dos cuentas. La transacción más simple es transferir criptomonedas de una cuenta a otra. En la *Figura 8* se puede ver un ejemplo de como Alice manda a Bob 17$.

Los datos de cada transacción varían dependiendo de la implementación de Blockchain. Pero los mecanismos para realizar una transacción suelen ser los mismos. El usuario que realiza la transacción manda la información a la blockchain. Esta información suele incluir la dirección del ordenante, su clave pública, una firma digital, unos parámetros de entrada y unos parámetros de salida [19].

30 Secure Hash Algorithm (Algoritmo de Hash Seguro)
31 National Institute of Standards and Technology
32 https://csrc.nist.gov/projects/hash-functions



Fuente: Cortesía de NIST [20]
Figura 8: Ejemplo de una transacción

Una transacción de criptomonedas suele tener como mínimo la siguiente información, aunque puede tener más:

- Argumentos de entrada (*Inputs*): suelen ser una lista de los activos que se van a transferir. Una transacción normalmente indica el origen de los activos que se van a transferir, ya sea la transacción en los que se recibió o en el caso de ser nuevos, el evento de origen. El ordenante también deberá demostrar que posee acceso a los activos que se desea transferir, esto se hace normalmente con la firma digital.

- Argumentos de salida (*Outputs*): suele ser la dirección de la cuenta de destino de los activos que se van a transferir, y un valor indicando cuanto del activo va a recibir.

3.1.3 CRIPTOGRAFÍA DE CLAVE ASIMÉTRICA

Blockchain utiliza la tecnología de clave simétrica, también conocida por criptografía de clave publica. Esta tecnología usa un par de claves, una privada y otra pública.

- Clave privada: es almacena por el usuario y solo el conocerá.

- Clave pública: será pública y conocida por todos, esto no reduce la seguridad del proceso. Esta clave es generada a partir de la privada, pero es computacionalmente muy difícil conseguir la clave privada partiendo de la pública.

Esta tecnología permite a dos usuarios confiar entre ellos, puesto que lo cifrado con clave privada puede ser descifrado con la pública y viceversa, lo cifrado con la clave publica, solo podrá ser descifrado por la privada. Este es el mecanismo usado para verificar la autenticidad e integridad

de las transacciones. Una transacción firmada por el ordenante con su clave privada, podrá ser solo descifrado con su clave pública, la cual esta disponible por todos, lo que les permite verificar que el dueño de la transacción es el ordenante. Este proceso se conoce como firmar digitalmente. Por otro lado, uno podría cifrar datos con la clave pública de un usuario, permitiendo solo a ese usuario descifrarlo y obtener los datos [21].

La criptografía de clave asimétrica es implementada de manera distinta dependiendo de la red, pero los usos habituales suelen ser los siguientes:

- Clave privada: usada para firmar digitalmente transacciones.
- Clave pública: usada para calcular la dirección y para que otros usuarios verifiquen una firma digital.
- Criptografía de clave asimétrica: permite verificar que el usuario ordenante de la transacción posee la clave privada capaz de firmar la transacción.

3.1.4 DIRECCIONES

Muchas redes Blockchain hacen uso de una dirección. Esta es una cadena de caracteres alfanuméricos que se calcula en función de la clave pública del usuario, usando una función de hash criptografía y algún otro dato, dependiendo de la implementación de Blockchain la forma de generar la dirección varia. Un ejemplo de una dirección sería:

0xedfca068ed063a856f20bb629e7d03de3149f92b

Las redes Blockchain utilizan estas direcciones como identificadores del ordenante y el destinatario de las transacciones, en estas situaciones se le suele llamar dirección de cartera [22].

3.1.5 REGISTRO CONTABLE

El registro contable es el ingreso de información de los movimientos de recursos en los libros de contabilidad, de tal forma de llevar una bitácora de cada operación realizada. Cada movimiento se registra en dos cuentas, mostrando el uso y el origen de los recursos [23]. Este registro puede almacenarse de manera centralizada o descentralizada. Tradicionalmente a estado centralizado, situándose en un único libro o en un único servidor, pero Blockchain lo que propone es descentralizar los registros y la propiedad de dichos registros.

El interés en registros contables descentralizados ha ido creciendo por las ventajas y mejoras que suponen con respecto al centralizado. Las ventajas de tener el registro contable descentralizado son las siguientes:

- El registro contable esta almacenado es muchos nodos, lo que supone una gran redundancia de los datos, puesto que se podrían perder varios nodos y seguir teniendo toda la información disponible. En cambio si esta centralizado, si se borra o destruye se pierde dicha información.

- En una red blockchain los nodos suelen poseer distinto *hardware* y estar en distintas redes, por lo que si un nodo se ve visto vulnerado, no se afectado el registro contable, a diferencia si este está centralizado, la red y el *hardware* suelen ser los mismos, por lo que si una maquina se ve vulnerada, fácilmente se puede propagar y afectar al registro contable.

- Una red blockchain esta distribuida entre varias zonas geográficas, por lo que si la conexión a internet de toda una zona geográfica se cayera, se podría seguir accediendo al registro contable, puesto que sigue estando disponible en otras zonas geográficas.

- Todas las transacciones son transparentes para toda la red de blockchain, puesto que estas deben ser verificadas por la red. Esto protege frente a inserciones de datos no autorizados, en cambio en un despliegue centralizado, las transacciones no son transparentes.

- Los datos del registro contable no pueden ser modificados en una red blockchain, lo que permite garantizar su fiabilidad, frente a un despliegue centralizado que el dueño del registro lo podría alterar, por lo que no posee la misma fiabilidad.

3.1.6 BLOQUES

Los bloques en una red de blockchain es lo que se almacena en dicha red, dependiendo de la implementación de la tecnología de blockchain estos bloques almacenarán un tipo de información u otra. Normalmente en las criptomonedas la información que se almacena en los bloques son las transacciones validas que han tenido lugar en la red [24].

El proceso de creación de bloques si en estos se almacenan transacciones es el siguiente:

1. Un usuario publica a un nodo una transacción.

2. El nodo al que ha sido publicada la transacción, la propaga al resto de nodos. Destacar, que aún no ha sido añadida a la red blockchain.

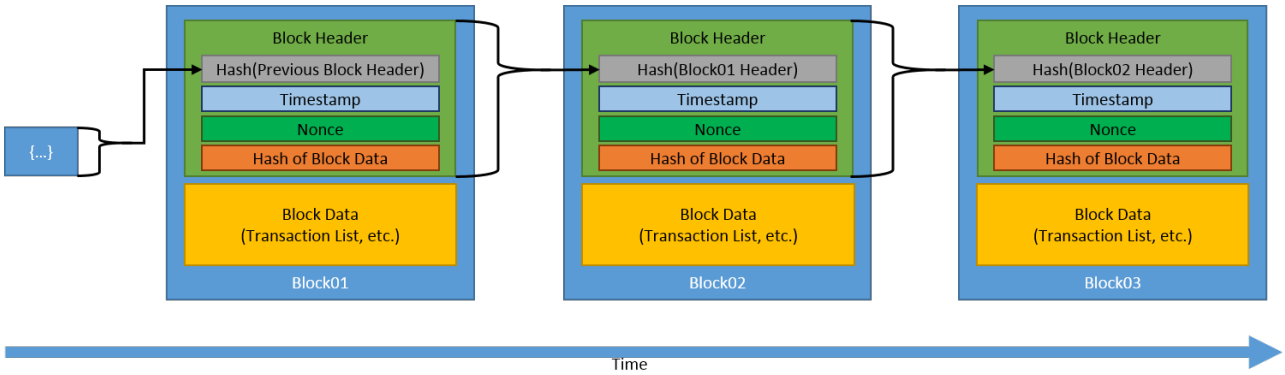
3. Una vez la transacción se ha distribuido entre varios nodos, esta se queda en estado pendiente, hasta que un nodo publicador la añade a un bloque que será añadido a la red blockchain.

Los bloques de la red Blockchain se componen por una cabecera y su contenido. Cada implementación define el contenido del bloque, pero a continuación se mencionará el contenido más común en los despliegues actuales [19].

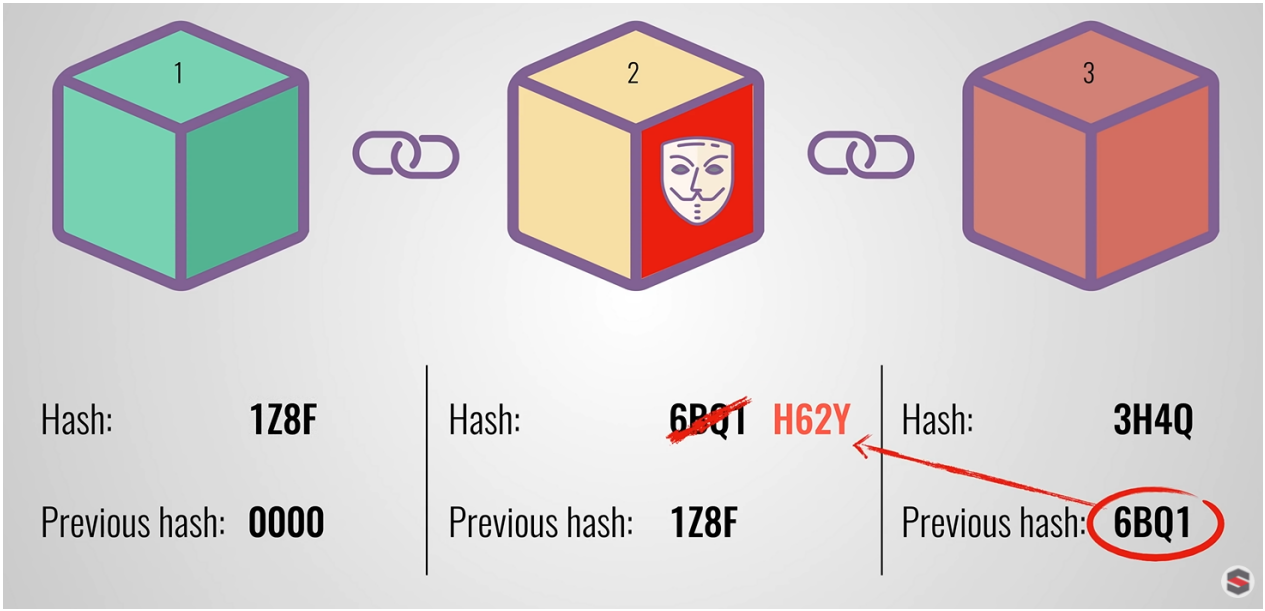
- Cabecera del bloque: contiene los metadatos del propio bloque, algunos de los más comunes son:
  - El hash de la cabecera del bloque anterior
  - Un hash representando los datos del bloque, suele ser el hash raíz del árbol de Merkle (En el *ANEXO III: Arboles Merkle* se explica los árboles de Merkle)
  - Marca de fecha (*timestamp*) de la creación del bloque, se representa en segundos desde la época Unix (1 de enero de 1970)
  - *Nonce*, es un contador determinado por el nodo publicador para cumplir el protocolo de consenso
  - El tamaño del bloque en bytes
- Datos del bloque: contiene una lista de las transacciones autenticadas y validadas que han sido publicadas para añadir a la red blockchain

3.1.7 CADENA DE BLOQUES

Como se ha explicado antes, en la cabecera de los bloques sen encuentra el hash de la cabecera del bloque anterior, por lo que un bloque siempre va ligado al anterior, creando así la cadena de bloques (*Figura 9*). Esta forma de unir los bloques hace que se pueda determinar fácilmente si un bloque a sido alterado, en la *Figura 10* se puede ver como cuando un bloque a sido alterado la cadena ya no es correcta.



Fuente: Cortesía de NIST [20]
Figura 9: Cadena de bloques genérica



Fuente: Cortesía de Simply Explained [25]
Figura 10: Ejemplo de cadena de bloques alterada

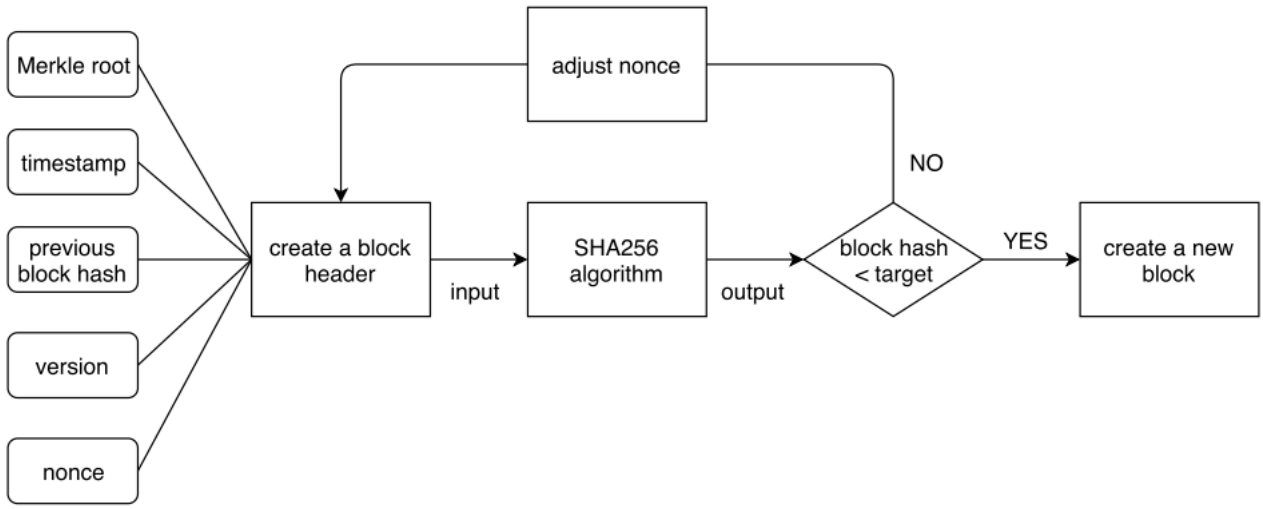
3.2 PROTOCOLO DE CONSENSO

Consenso en la red hace referencia al proceso por el cual se llega a un acuerdo entre los participantes de la red del estado correcto de los datos en el sistema. Este proceso lleva a que todos los nodos de la red compartan exactamente los mismos datos y previene modificaciones maliciosas o no permitidas [26].

El objetivo del protocolo de consenso es garantizar que todos los nodos participantes están de acuerdo en el mismo registro contable [27]. Para llegar a este acuerdo, se utilizan diversos algoritmos de consenso, a continuación se explicarán algunos de ellos.

3.2.1 PRUEBA DE TRABAJO (POW[33])

En este algoritmo de consenso los nodos que están intentando validar el bloque deben encontrar un valor de hash del bloque que cumpla unos criterios de dificultad fijados por la red. Para ello, debe ir cambiando del valor *nonce* de la cabecera del bloque. En la *Figura 11* se puede ver el flujo que sigue el algoritmo PoW. El nodo que consiga encontrar dicho hash antes que los demás, validará el bloque y por tanto recibirá una recompensa por ello [28].



Fuente: Cortesía de ICT Express [28]
Figura 11: Flujo de PoW

3.2.2 PRUEBA DE PARTICIPACIÓN (POS[34])

En PoS el nodo que valida la transacción se elije en función de cuanto tenga el nodo invertido, en vez de la potencia computacional, como se hace en PoW. Una vez el nodo a sido seleccionado, este debe seguir resolviendo un puzzle con una dificultad fijada por la red. Pero en vez de modificar el *nonce* depende de la cantidad de monedas que el nodo tenga invertidas. Cuando el nodo resuelve el puzzle, el resto de nodos validarán el bloque, y si este es valido, se añadirá a la cadena y se le devolverá al nodo las monedas que había usado para validar el bloque más una bonificación extra.

33 Proof of Work
34 Proof of Stake

Pero si el bloque resulta no ser valido, las monedas que el nodo había invertido en el proceso de validación del bloque no son devueltas [28].

3.2.3 PRUEBA DE CAPACIDAD (POC[35])

PoC en vez de utilizar recursos de calculo, utiliza el almacenamiento. Cuanto más almacenamiento un nodo aloque para este algoritmo, mayor son las probabilidades de validar el bloque y llevarse la recompensa. El espacio de almacenamiento alocado, se utiliza para almacenar soluciones aleatorias antes de recibir el bloque, este proceso puede tardar días, depende de la cantidad de espacio alocada. Cuando el nodo esta listo, compara sus soluciones con el próximo bloque y el primer nodo en resolverlo, recibe la recompensa por validar el bloque [29].

3.2.4 PRUEBA DE TIEMPO TRANSCURRIDO (POET[36])

Este algoritmo de consenso es muy parecido a PoW, pero en vez de recibir la recompensa el primer nodo en resolver el puzzle, la recompensa se asigna de manera aleatoria en función de un tiempo aleatorio. El ganador es el nodo que se le acabe antes el temporizador [30].

3.2.5 TOLERANCIA PRÁCTICA DE FALLOS BIZANTINOS (PBFT[37])

El objetivo de este algoritmo de consenso es determinar si la información presentada al registro contable es valida o no. Cada nodo de la red mantiene su propio estado, cuando recibe un nuevo mensaje, determina si es valido o no. Las respuestas individuales de los nodos son mandadas al registro contable de los nodos y en función de las decisiones de todos los nodos se añade o no la nueva información [29].

3.3 TIPOS DE REDES BLOCKCHAIN

Hay dos principales tipos de redes blockchain, sin permiso y con permiso.

3.3.1 SIN PERMISO O PÚBLICA

Las redes blockchain públicas se caracterizan porque la publicación de transacciones esta permito a todo el mundo, no se requiere ningún tipo de permisos, solo poseer una forma valida de pagar las tasas de la transacción. También cualquier persona puede incorporarse al conjunto de nodos validadores de la red blockchain sin ningún tipo de restricciones [31]. Estas redes suelen ser de código abierto.

35 Proof of Capacity
36 Proof of Elapsed Time
37 Practical Byzantine Fault Tolerance

3.3.2 CON PERMISO O PRIVADA

Las redes blockchain privadas no permiten la libre publicación de transacciones ni la libre incorporación como nodo validador. No se garantiza a los usuarios que las transacciones que hayan publicado sean aceptadas y añadidas a la red, aun teniendo forma de pagar las tasas e incluso teniendo acceso a la red.

3.4 PROCESOS ACTUALES DE CERTIFICACIÓN

Las obras de arte pertenecen a un mercado muy especializado, y más hoy en día donde según mejoran las tecnologías, también lo hacen las obras falsificadas, por eso el proceso de certificación de una obra debe ser muy exhaustivo para poder cerciorar sin lugar a fallo que la obra es original. Este proceso es muy importante, puesto que hay obras de arte que son muy antiguas y su valor económico es muy alto, pero solo si es la original, si esta obra resulta ser una copia, su valor es completamente distinto.

La importancia de poseer un certificado de autenticidad de una obra que deseas vender es primordial a día de hoy. Por eso esta tarea es realiza por expertos en el ámbito de la obra en concreto a autenticar.

Cada experto tiene sus métodos para realizar el correcto juicio de si la obra es original o una mera copia, pero hay tres puntos que son claves en el proceso [32].

- Examinar la obra de arte: el experto debe poder analizar la obra para poder realizar un análisis visual en persona. En este análisis visual, buscan detalles de sus partes, las imperfecciones, el formato, el estado de conservación de la propia obra y las técnicas con las que fue creada. Puesto que cada artista suele tener sus métodos y manías y un experto es conocedor de dichos detalles.

- Analizar la documentación: esta fase consiste en analizar toda la documentación que rodee la obra, escritos que la describan o mencionen, información de los dueños anteriores. En resumen, toda información relevante a la procedencia de la obra en cuestión.

- Pruebas científicas: esta fase es la más objetiva de las tres, pero no puede demostrar la autenticidad de la obra, solamente puede demostrar lo contrario, que no es original. Por eso es una etapa de gran importancia. En esta se realizan diversas pruebas en función del tipo de obra, las más típicas suelen ser la del carbono catorce, para determinar la antigüedad de la obra.

Una vez realizado todo el proceso de autentificación y el veredicto es que la obra es original se genera un certificado que autentica que la obra es original y autentica. Este certificado suele incluir información de la propia obra, como el autor, la calidad de la obra, la procedencia, etc.

Al ser un proceso largo y extensivo, es importante no perder el certificado, puesto que no tenerlo, significaría volver a tener que realizar el proceso entero. Por eso, es una gran ventaja poder tenerlo de manera digital, pero los sistemas actuales de compulsación digital (comentado en la *Introducción*) son procesos tediosos y no hay estándares a nivel global.

Cabe destacar que la legislación española permite el uso de la firma electrónica o sello informático, para firmar documentos digitales y que dichos documentos posean el mismo valor que su semejante en papel [33]. Pero ese documento digital no tiene la misma validez legal en otro país, pero el original a papel si que lo tiene. Este es uno de los problemas que se intentan solucionar con la implementación de la tecnología de blockchain a estos procesos.

3.5 *PROYECTOS DE CERTIFICACIÓN BASADOS EN BLOCKCHAIN*

Actualmente se están desarrollando nuevas aplicaciones de la tecnológica Blockchain. Un nuevo uso que se le está dando es el de los NFT[38].

Los NFT, son muy parecidos a las criptomonedas, pero en vez de representar una moneda, representa un recurso digital. Un NFT es solo la representación de ese recurso, no indica que el dueño del NFT sea dueño del recurso. A parte de usarse para intercambio, por dinero, se están utilizando por artistas, para certificar la autenticidad de sus creaciones, tanto físicas como digitales [34]. Esto se consigue generando un identificador único en función de la obra y subiendo dicho número entre otro tipo de informaciones a la blockchain. Haciendo esto, se puede demostrar que esa obra es la misma que se subió a la blockchain, puesto que se puede volver a calcular el identificador y así localizarla en la red y al ser la primera aparición de la obra en la red se puede así demostrar que es la original.

Por otro lado, hay empresas que están modernizando el proceso tradicional de expedición de certificados de autenticidad. Por ejemplo la web de Blockchain Art Collective[39] mediante el uso de etiquetas NFC que se ponen en la obra de arte, y con información que se introduce en su web, se sube al Blockchain, quedando por tanto registrada toda la información. Al estar en la Blockchain, se puede acceder a su trazabilidad, por lo que Blockchain Art Collective anuncia, que puedes ver el historial de dueños de tus obras.

38 Token no fungible (Non-funglible token)
39 https://blockchainartcollective.com/

Pukkart[40] es una empresa que certifica tanto obras digitales como físicas utilizando la tecnología blockchain. Esta empresa también permite crear varias ediciones de una misma obra. Por otro lado al igual que Blockchain Art Collective, también permite ver el historial de dueños de las obras de arte que uno posee. Esta empresa ofrece distintos niveles de certificación, el gratis, incluye un certificado que se puede imprimir, pero no se sube a la red blockchain, para ello habría que contratar el nivel básico o uno superior. En el nivel Oro, la obra se sube a la red Blockchain, se envia por correo postal el certificado en un papel especial y unas pegatinas de seguridad que se rompen al intentar despegarlas.

Si nos centramos más en certificados digitales, podemos encontrar por ejemplo la empresa española BlockTac[41]. Esta empresa se dedica a y cito de su página web *BlockTac se apoya en la tecnología Blockchain para aportar las características de inviolabilidad, inmutabilidad y verificación abierta para todos sus certificados digitales* [35]. Como se puede ver, se centran más en garantizar la inmutación de los certificados digitales, mediante el uso de la red Blockchain.

Como se puede apreciar hay empresas que están realizando tareas parecidas a las propuestas en este Trabajo Fin de Grado, pero ninguna solución proporciona las mismas garantías y seguridad que aporta la herramienta que se ha desarrollado en este trabajo. En el caso de Pukkart, no hay una valoración de un experto que determina si la obra que dice es ser, es verdaderamente la que es. Por otro lado BlockTac está más centrado en certificados digitales, los cuales no permiten validar la autenticidad de un documento con mucha facilidad y de manera independiente de si la empresa sigue operativa o no.

40 https://www.pukkart.com/
41 https://www.blocktac.com/

Capítulo 4 Definición del trabajo

4.1 MOTIVACIÓN

Como se ha podido ver en la *Introducción* y en el *Estado de la Cuestión*, los procesos de digitalización de certificados, o compulsación, es un procedimiento largo con dependencias de terceros y de entidades gubernamentales. Si se deseara digitalizar un certificado de autenticidad, se debe, según estipula el BOE, acudir a un ayuntamiento o a la entidad que expidió dicho certificado, para poder así compulsar la copia digital de dicho certificado [9]. Existe la posibilidad de que cuando se obtiene el certificado, se realice de manera digital, pero no hay estándares sobre como se debe realizar. Además la entidad sigue dependiendo de terceros, para obtener la firma electrónica, con el cual puede firmar o sellar el documento de manera digital.

Por otro lado si analizamos algunas de las herramientas que hay actualmente en el mercado que intentan agilizar estos procesos utilizando blockchain, nos encontramos algunas empresas como por ejemplo Blockchain Art Collective o Pukkart. Como se ha explicado en el *Estado de la Cuestión* estas empresas se orientan más en obras nuevas, puesto que ningún experto participa en el proceso de certificación, observando y analizando la obra para determinar su autenticidad de cara a la correcta expedición del certificado. Se orientan más en que el propio autor de la obra autentique la veracidad de su trabajo y obra.

A la vista de la situación en la que nos encontramos a día de hoy y con el fin de agilizar los procesos de digitalización de certificados, facilitar los procesos para que cualquier usuario independientemente de sus conocimientos informáticos pueda obtener y verificar la autenticidad de certificados y dotar de plena transparencia se ha decidido implementar la tecnología blockchain a estos procesos, generando un contrato inteligente, el cual ira acompañado de una web y un servicio de REST API[42]. Es decir se creará un sistema donde se podrá verificar si un archivo esta en la blockchain y cuando y quien lo subió. Este sistema podrá ser utilizado desde una web, desde un cliente REST API o realizando directamente las llamadas sobre el contrato inteligente desplegado en al red blockchain.

Con este trabajo se busca descentralizar un proceso centralizado, pero hacerlo de una manera que para el usuario final sea más fácil de usar y tenga las mismas o incluso mejores garantías que los métodos tradicionales, de que los archivos no han sido modificados.

4.2 OBJETIVOS

Como se ha ido indicado a lo largo de este documento, el objetivo de este Trabajo Fin de Grado es el desarrollo de todo un sistema transparente que permita hacer accesible al mayor número de

42 Representational State Transfer Application Programming Interface (REST API)

personas la verificación de certificados de autenticidad. Para conseguir esto se marcaron los siguientes objetivos:

- Desarrollar un contrato inteligente que permita a los usuarios subir el hash de un archivo a la blockchain para que este se quede registrado en la cadena de bloques. También que un usuario pueda obtener la información que se encuentra en la cadenas de bloques de un archivo.

- Implementar el contrato inteligente en un servidor REST API, tradicional para que los usuarios que no deseen interactuar directamente con el contrato inteligente lo puedan realizar a través de esta API. A parte de ser utiliza esta API por usuarios, también podría ser utilizada por terceros para ser implementada en sus servicios o aplicaciones.

- Hacer accesible a todos los usuarios este sistema, mediante el desarrollo de una web que interactué con la API, la cual a su vez interactuará con el contrato inteligente. Esta web será lo más sencilla posible para que todos los usuarios lo puedan usar sin problema, independientemente de sus conocimientos en estas áreas.

- Realizar pruebas en una red simulada y con varios usuarios, para poder determinar el correcto funcionamiento y su viabilidad en el mercado.

4.3 METODOLOGÍA

Debido a la existencia de una fecha limite para realizar el desarrollo de este proyecto, se ha seguido una metodología en cascada[43], intentando implementar *agile* en la medida de lo posible, en cada una de las etapas de la cascada.

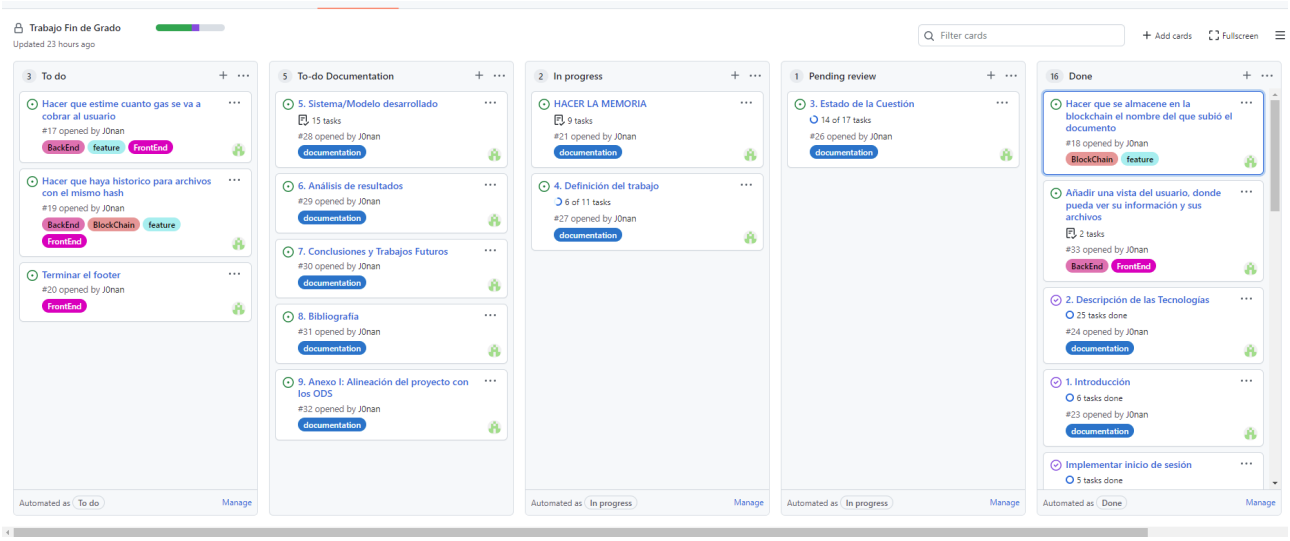
Se realizó la planificación de la *Tabla 2* la cual se a seguido a lo largo del proyecto. Dentro de cada una de las etapas del proyecto, se dividió en sud-tareas más pequeñas. Estas tareas se crearon como *Issues* dentro de GitHub, y a su vez esos *Issues* se añadieron al *Kanban* del proyecto de GitHub.

43 *Waterfall model*

	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio
Análisis									
Diseño									
Desarrollo									
Prueba									
Documentación									

Tabla 2: Diagrama de GANT del cronograma del proyecto

El proyecto de GitHub se ha usado para llevar un registro de las tareas en todos sus estados, en la *Figura 12* se puede ver como se ha divido el estado de las tareas en cuatro tareas fundamentales, por hacer, en progreso, pendientes de verificar y realizada. Cada estado es auto explicativo, pero cabe destacar que se han separado en dos columnas las tareas por hacer, una para las de código (*To-Do*) y la otra para la documentación (*To-Do Documentation*).



Fuente: Elaboración propia
Figura 12: Panel Kanban del Trabajo Fin de Grado

4.4 ESTIMACIÓN ECONÓMICA

En este apartado, se realizará un análisis económico del desarrollo de este trabajo. Se analizará el coste de desarrollo y el coste de desarrollo. Estos análisis se realizarán en dos partes, la parte material y la parte humana. Por último se analizaran distintos tipos de escenarios para determinar los beneficios y/o perdidas que tendría este proyecto si se comercializara.

4.4.1 RECURSOS MATERIALES

En cuanto a los recursos materiales que han sido utilizados en el desarrollo de este Trabajo Fin de Grado, han sido un ordenador portátil y un monitor, las prestaciones de dichos dispositivos se encuentran en la *Tabla 3* y *Tabla 4* respectivamente.

Ordenador Portatil Lenovo Legion 5i Pro Gen 6	
Procesador	Intel® Core™ i7-11800H
Sistema operativo	Windows 10 Enterprise LTSC
Tarjeta Gráfica	NVIDIA® GeForce RTX™ 3060
Pantalla	16” WQXGA (2560 x 1600)
Memoria RAM	16 GB DDR4 a 3200 MHz
Almacenamiento	1 TB SSD PCIe
Precio	1.703’20 €

Tabla 3: Especificaciones de ordenador portátil

BenQ EX2510 MOBIUZ	
Tamaño de la pantalla	24’5 pulgadas
Resolución	FHD (1920 x 1080 a 144 Hz)
Tipo de panel	IPS con retro-iluminación luz de fondo LED
Audio	Altavoces incorporados 2x2’5W
Precio	229’00 €

Tabla 4: Especificaciones de monitor

Actualmente tanto el ordenador portátil como el monitor tienen una vida útil fiscal de 4 años (35.040 horas), según estipula la Agencia Tributaria[44]. Como ambos dispositivos no se han usado exclusivamente para la realización de este trabajo, se realizará una estimación del coste de los dispositivos en función del uso en el trabajo. Ambos dispositivos han sido usados en el desarrollo de este Trabajo Fin de Grado unas 845 horas (en la *Tabla 5* se puede ver con mayor detalle la distribución de las horas). En función de estas horas se realizaron cálculos para poder determinar el valor amortizado de los materiales en el desarrollo del software. En la *Ecuación 1* y *Ecuación 2* se puede ver los cálculos realizados para determinar que en el desarrollo de este Trabajo Fin de Grado, el coste material es 46’56 €.

44 https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html

Actividad	Horas
Organización, planificación, reuniones y consultas	15
Análisis, estudio y pruebas de tecnologías	150
Diseño	350
Desarrollo	
Pruebas	30
Documentación	300
TOTAL	**845**

Tabla 5: Desglose de las horas de dedicación de este Trabajo Fin de Grado

$$Coste\ por\ hora(Portátil) = \frac{1703'20\,€}{35040\,horas} \approx 0'0486\,€/hora$$
$$Coste\ total(Portátil) = 0'0486\,€/hora \times 845\,horas \approx 41'07\,€$$

Ecuación 1: Cálculo del coste total del ordenador portátil en el desarrollo de este trabajo

$$Coste\ por\ hora(Monitor) = \frac{229\,€}{35040\,horas} \approx 0'0065\,€/hora$$
$$Coste\ total(Monitor) = 0'0065\,€/hora \times 845\,horas \approx 5'49\,€$$

Ecuación 2: Cálculo del coste total del monitor en el desarrollo de este trabajo

En cuanto al software utilizado, todos los programas que han sido necesarios para el desarrollo de este Trabajo Fin de Grado, son gratuitos y de código abierto, salvo el sistema operativo del ordenador portátil, pero venia instalado con el propio ordenador, por lo que su coste esta ya incluido en el de la *Tabla 3*.

Pasando al coste material del mantenimiento del proyecto, si hostearamos la web, la API y la base de datos en Google Cloud[45] nos supondría un coste mensual aproximado de unos 85 €, lo que supone unos 1020 € anuales. Con respecto al despliegue del contrato inteligente, solo habría que pagar una vez, el valor aproximado de gas es de unos 2.536.168, lo que equivale a 112 € [46] en el momento de escritura.

En la *Tabla 6* se puede apreciar todos los gastos comentados en este aparto. Hay un coste único de desarrollo de unos 75'56 € y un coste anual de 1.020 €.

45 https://cloud.google.com/
46 https://www.cryptoneur.xyz/gas-fees-calculator

Tipo de gasto	Concepto	Costo
Desarrollo	Ordenador	41’07 €
	Monitor	5’49 €
	Contrato inteligente	112 €
	Total	**158’56 €**
Mantenimiento	Web y API	493’56 €/año
	Base de datos	526’44 €/año
	Total	**1.020 €/año**

Tabla 6: Resumen de los costes materiales

4.4.2 RECURSOS HUMANOS

Si analizamos los recursos humanos, para llevar a cabo el desarrollo de este Trabajo Fin de Grado, se ha requerido un desarrollador de *software* y la consulta de un perito, el cual estaba certificado para autenticar obras de arte.

Actualmente un desarrollador de *software* en España cobra de media 2.121 € al mes según Indeed[47]. Por otro lado un perito cobra unos 50 € la hora por realizar un servicio de consulta[48]. Con estos valores y teniendo en cuenta las horas de la *Tabla 5* se puede realizar una estimación del coste inicial del desarrollo realizado en este trabajo.

Concepto	Coste	Tiempo trabajado	Coste total
Desarrollador	2.121 €/mes	5’3 meses	11.241’30 €
Perito judicial	50 €/hora	2 horas	100 €
TOTAL		**11.341’30 €**	

Tabla 7: Coste humano del desarrollo

Como se puede ver en la *Tabla 7* el coste total humano del desarrollo de este Trabajo Fin de Grado es de unos 11.341’30 €.

Pasando al coste de mantenimiento, surgen varias posibilidades, puesto que el proyecto va a ser de código abierto. Se podría contratar a un desarrollador para que mantenga la herramienta y revise los cambios y mejoras propuestos por la comunidad. La otra opción es que directamente, el código sea mantenido por voluntarios y la comunidad.

47 https://es.indeed.com/career/desarrollador-de-software/salaries
48 https://certificadoautenticidad.com/cuanto-cuesta-un-informe-pericial-por-un-perito-judicial/

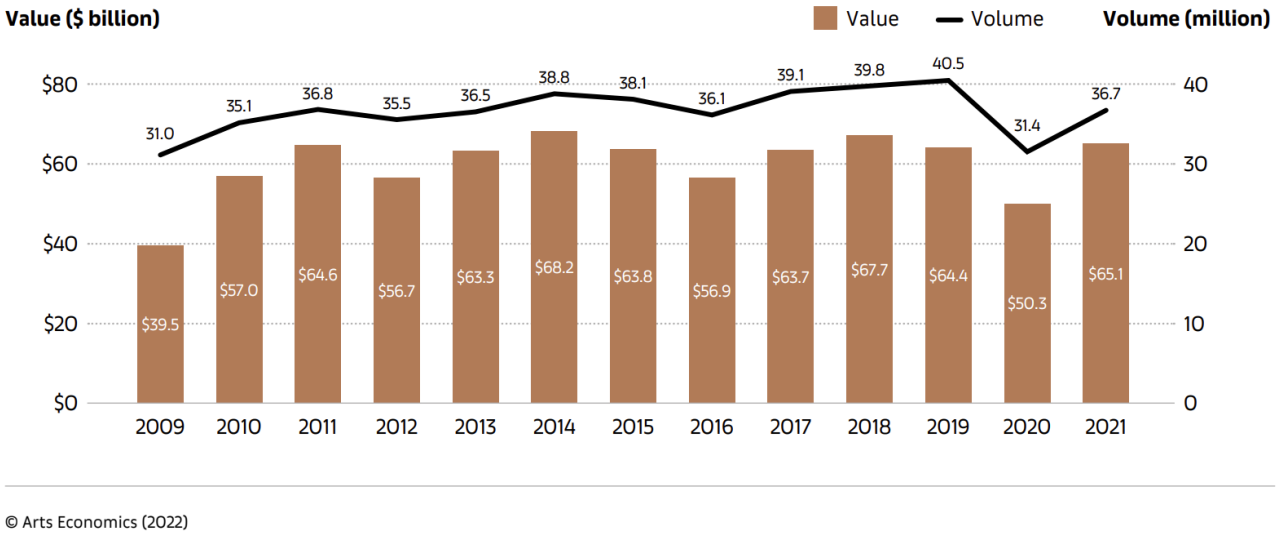
Para este análisis se va a optar por la opción más realista, es decir la opción de contratar a un desarrollador para que mantenga y mejora los sistema, a su vez esta persona hará de soporte técnico, en caso de ser necesario. Por ello a parte de tener que pagar el sueldo del desarrollador, se contratará una linea móvil para que el desarrollador posea un teléfono orientado a la empresa y el soporte técnico. En la *Tabla 8* se puede ver el coste anual del mantenimiento del desarrollo.

Concepto	Coste anual
Desarrollador	29.694 €
Móvil de soporte técnico	95'40 € [49]
TOTAL	**29.789'40 €**

Tabla 8: Coste anual de mantenimiento del desarrollo

4.4.3 PRESUPUESTO TOTAL

Habiendo realizado un análisis de los costes iniciales y de mantenimiento, podemos proceder a realizar un análisis de los beneficios en distintas situaciones. Para este análisis se analizarán dos escenarios, en los que variará la cantidad media anual de obras subidas a la web. En 2021 se realizaron 36'7 millones de transacciones de obras de arte a nivel mundial[50], pero, como se puede apreciar en la *Figura 13* el volumen de ventas de obras de arte a lo largo de los años fluctuá entorno a los 36 millones, por lo que se considerará dicho valor como constante a lo largo de los años venideros.



Fuente: Cortesía de Art Basel y UBS[50]
Figura 13: Ventas de obras de arte a nivel mundial 2009-2021

49 https://www.lowi.es/
50 https://d2u3kfwd92fzu7.cloudfront.net/The_Art_Market_2022.pdf

Antes de realizar el análisis hay que determinar la media de las tasas que se debe pagar el usuario a la blockchain por subir un archivo al contrato inteligente. Tras realizar varias pruebas, el gas medio que se ha pagado por subir ficheros es de unos 341.500 lo que en el momento de escritura equivale a unos 15 €[51]. Para realizar los siguientes análisis se utilizará este valor como tasas que pagan los clientes a la blockchain por subir sus ficheros y se determina que la web cobra un 10% de lo que el usuario tenga que pagar como tasas a la blockchain, por lo que de media, por subida de archivo, recibiremos un ingreso de 1'5€ €.

4.4.3.1 Escenario uno

En este escenario se asumirá una captación del mercado del 1 % de las transacciones globales de arte. En la *Tabla 9* se puede ver los ingresos anuales en este escenario.

Porcentaje de cuota de mercado	Numero de transacciones	Ingresos anules por comisiones
1 %	360 000	540.000 €

Tabla 9: Ingresos anuales en el escenario uno

Contabilizando los gastos por mantenimiento determinado en los apartados anteriores, podemos determinar que el beneficio bruto anual seria de

Concepto	Precio anual
Gastos de mantenimiento	30 809'40 €
Estimación de ingresos	540 000 €
BENEFICIOS	**509 190'60 €**

Tabla 10: Calculo de los beneficios brutos anuales en el escenario uno

Teniendo en cuenta el coste inicial de 11.499'86 € calculados en la *Tabla 6* y *Tabla 7*, el primer año se recuperaría el coste de la inversión inicial, quedando 497.690'74 € de beneficios el primer año.

4.4.3.2 Escenario dos

Este escenario es más pesimista que el *Escenario uno*, se asumirá que solo se capta un 0'06 % de las transacciones a nivel global. Como se puede ver en la *Tabla 11* los ingresos anuales serian unos 32.400 €

51 https://www.cryptoneur.xyz/gas-fees-calculator

Porcentaje de cuota de mercado	Numero de transacciones	Ingresos anules por comisiones
0’06 %	21 600	32 400 €

Tabla 11: Ingresos anuales en el escenario dos

Teniendo en cuenta estos valores y los calculados en los apartados anteriores podemos determinar los beneficios brutos anuales, *Tabla 12*.

Concepto	Precio anual
Gastos de mantenimiento	30 809’40 €
Estimación de ingresos	32 400 €
BENEFICIOS	**1 590’60 €**

Tabla 12: Calculo de los beneficios brutos anuales en el escenario dos

Como se ha podido ver, anualmente tendríamos unos beneficios brutos de 1.590’60 €, por lo asumiendo que se mantuviera el 0’06 % de cuota de mercado y teniendo en cuenta el coste inicial total de 11.499’86 € calculados en la *Tabla 6* y *Tabla 7*, tras 8 años se recuperaría la inversión inicial.

Capítulo 5 Sistema Desarrollado

En este capitulo se explicará el sistema desarrollado en este Trabajo Fin de Grado. Este sistema consiste en un servicio REST API, el cual utiliza un contrato inteligente para almacenar y leer información de los archivos en la blockchain, además utiliza una base de datos centralizada, para mayor rapidez en las consultas generales. A su vez, también se ha desarrollado una web que interactuá con la API, así de esta manera, cualquier usuario puede hacer uso del sistema desarrollado. Al realizar una API y una web que consuma la API, se consigue desacoplar de manera funcional ambos, lo que conlleva que si se desea realizar un cambio en la web, este no afecta a los diversos servicios o aplicaciones que consuman la API.

Con el fin de explicar el sistema desarrollado, se dividirá en tres partes, el análisis del modelo, donde se explicara la funcionalidad que se desea mejorar implementando la tecnología del blockchain, para poder así determinar que se desea desarrollar. El siguiente punto que se explicará es el diseño, aquí se explicará el diseño de la arquitectura utilizado, que estructura tienen los datos, tanto en la base de datos centralizada, como en la blockchain y se realizarán un par de diagramas de secuencia. Por ultimo se explicará la implementación que se ha realizado en los tres componentes fundamentales, el contrato inteligente, el servicio REST API y la web.

5.1 ANÁLISIS DEL MODELO

Como se ha explicado en apartados anteriores, con este desarrollo se desea simplificar el proceso de digitalización de certificados de autenticidad. Para ello, vamos a repasar cual es el proceso actualmente legal y vigente en España. Primero debemos distinguir varias situaciones.

1. En esta primera situación la obra es nueva, y es el propio autor el que desea certificar su autenticidad para poder realizar una venta. En este caso el sujeto, debe acudir a un perito judicial el cual este cualificado para certificar el tipo de arte que haya realizado el artista. Una vez realizado el proceso de certificación, el perito, puede generar un pdf del certificado de autenticidad y firmarlo digitalmente con una firma electrónica expedida por una entidad cualificada por el gobierno del país donde se este expendiendo el certificado[52].

2. Una segunda situación es que el dueño actual de una obra que no esta certificada su autenticidad, este debe al igual que en la situación 1, acudir a un perito judicial cualificado y que este tras realizar su análisis genere el certificado digitalmente y lo firme con su firma electrónica.

3. La siguiente situación es en la que el dueño de una obra de arte, la cual posee ya un certificado, pero en papel, desea digitalizarlo. Esta persona puede hacer dos cosas, puede ir

52 https://sedeaplicaciones.minetur.gob.es/Prestadores/Inicio.aspx

al perito que expidió el certificado original para que le genere uno digital el cual debe firmar de manera electrónica o bien puede realizar una compulsación digital. Para realizar esta compulsación, debe acudir a una ayuntamiento para que así un funcionario compulse la digitalización del certificado en cuestión.

Estas son las posibles situaciones en las que se puede encontrar un vendedor o un dueño de una obra de arte y lo que a día de hoy tendría que hacer. Pero, si pasamos a analizamos el otro lado de las transacciones, el comprador, nos encontramos que este lo que desea es tener la certeza que la obra que desea comprar es autentica. Para asegurarse que dicha obra es autentica, el comprador suele requerir documentos o pruebas que vinculen la obra con su autor. Estas pueden ser por ejemplo un certificado de autenticidad. En el caso de que este certificado de autenticidad sea digital, lo que debe hacer es acceder a VALIDe[53] y subir el documento en cuestión para verificar la firma electrónica incrustada en este. De esta forma verifica que el documento firmado por el perito no ha sido modificado. Pero esto es valido solo para firmas electrónicas realizadas con certificados expedidos por entidades cualificadas por el gobierno de España. Cuando la transacción es internacional, el comprador no suele utilizar estas herramientas, bien porque desconoce su existencia o porque no le inspira confianza, por lo que dependiendo de la obra, podría pedir a un experto de su confianza que examine la obra y la analice.

Por último el perito que esta realizando la autenticación y que va a expedir una copia digital de su certificación, como se ha explicado antes, debe firmar electrónicamente el documento con un certificado expedido por una entidad cualificada. Estos certificados suelen tener un periodo de validez de un año, por lo que cada año debe renovarlo.

5.1.1 HISTORIAS DE USUARIO

Habiendo explicado los procesos actuales, definiremos los distintos tipos de usuarios de forma que no haya confusión en las historias de usuario. Se han identificado cuatro tipos de actores:

- Dueño: es la persona o grupo de personas que poseen una obra de arte y su pertinente certificado de autenticidad.
- Comprador: es la persona o grupo de estas, que desean comprar una obra de arte.
- Autentificador: es la persona con los credenciales y cualificaciones necesarios para poder certificar obras de arte.
- Artista: es la persona creadora de una o más obras de arte.

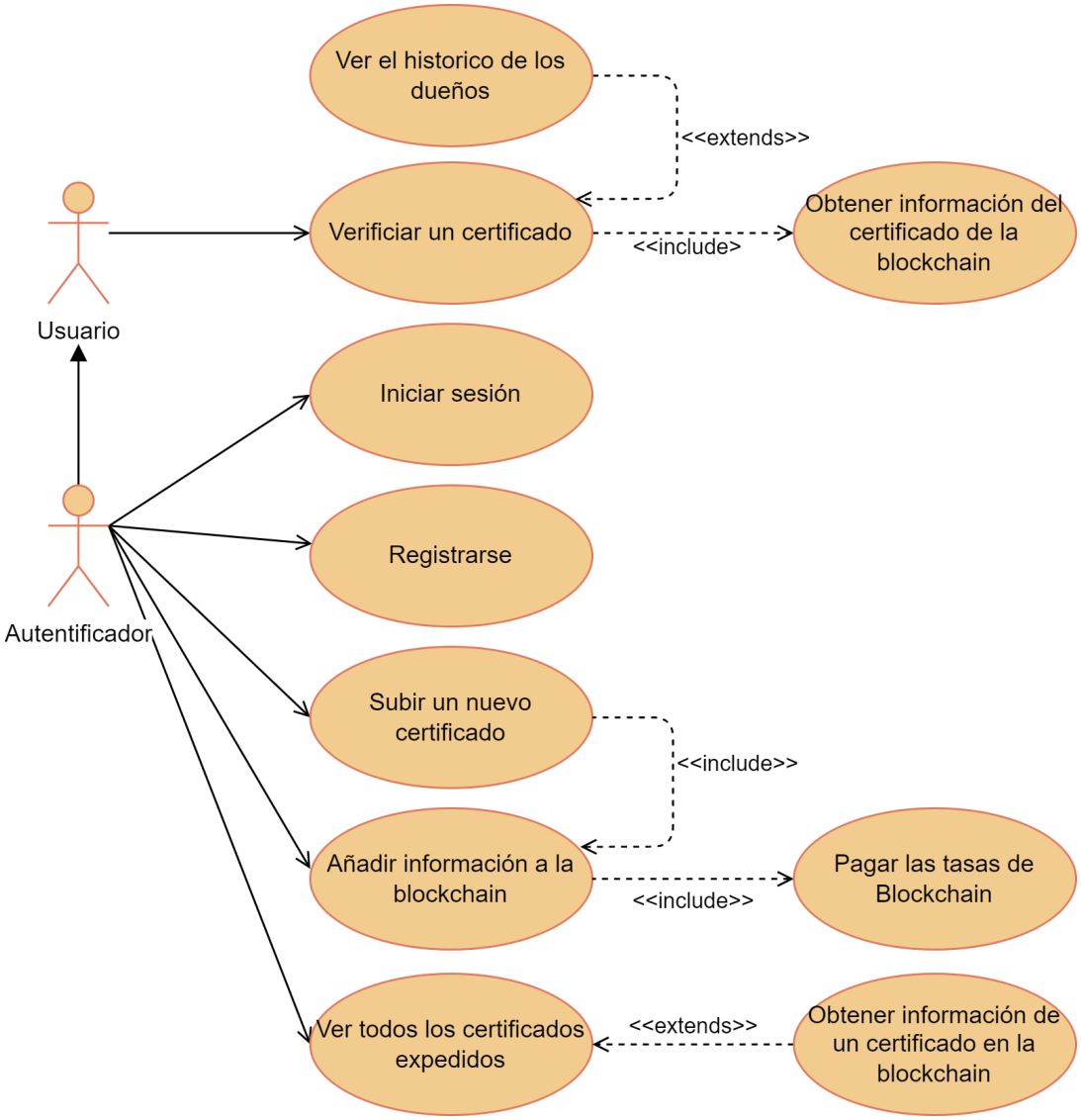
53 https://valide.redsara.es/valide/validarFirma/ejecutar.html

Una vez definido los distintos roles, se puede proceder a realizar las historias de usuario teniendo en cuenta los objetivos expuestos en el apartado *4.2*.

- Como dueño quiero obtener digitalmente mi certificado de autenticidad para no perderlo con tanta facilidad como la versión a papel.
- Como dueño quiero verificar que mi certificado de autenticidad digital no ha sido modificado para así poder demostrar la autenticidad de la obra al venderla.
- Como dueño quiero ver los dueños anteriores de mi obra para así poder poseer un histórico de la obra.
- Como comprador quiero ver quien fue el autentificador para poder así verificar sus credenciales y cualificaciones.
- Como comprador quiero verificar que el certificado de autenticidad que me presentan no a sido alterado desde su expedición para así comprar una obra original.
- Como comprador quiero acceder al histórico de los dueños de la obra para poder acreditar su autenticidad en función de sus origines.
- Como comprador quiero que quede registrado que le compré la obra a su dueño anterior para así mantener una trazabilidad.
- Como comprador quiero un sistema global para no tener que buscar la página de validación de firmas electrónicas del país donde se expidió el certificado de autenticidad.
- Como autentificador quiero ver todos los certificados que he expedido para poder llevar un registro de ellos.
- Como autentificador quiero darme de alta en el sistema para poder digitalizar certificados de autenticidad que expida.
- Como autentificador quiero ver los dueños anteriores de la obra para confirmar la autenticidad de esta.
- Como autentificador quiero añadir quien es el nuevo dueño de una obra para que exista un histórico de los dueños.
- Como autentificador quiero no depender de gobiernos para no tener que preocuparme por obtener un certificado valido en los distintos países que trabaje.
- Como artista quiero poder ver los dueños de mis obras para conocer el recorrido de mis obras.

5.1.2 CASOS DE USO

Habiendo determinado las historias de usuario, se procedió a realizar los casos de uso, los cuales se pueden apreciar en la *Figura 14*.



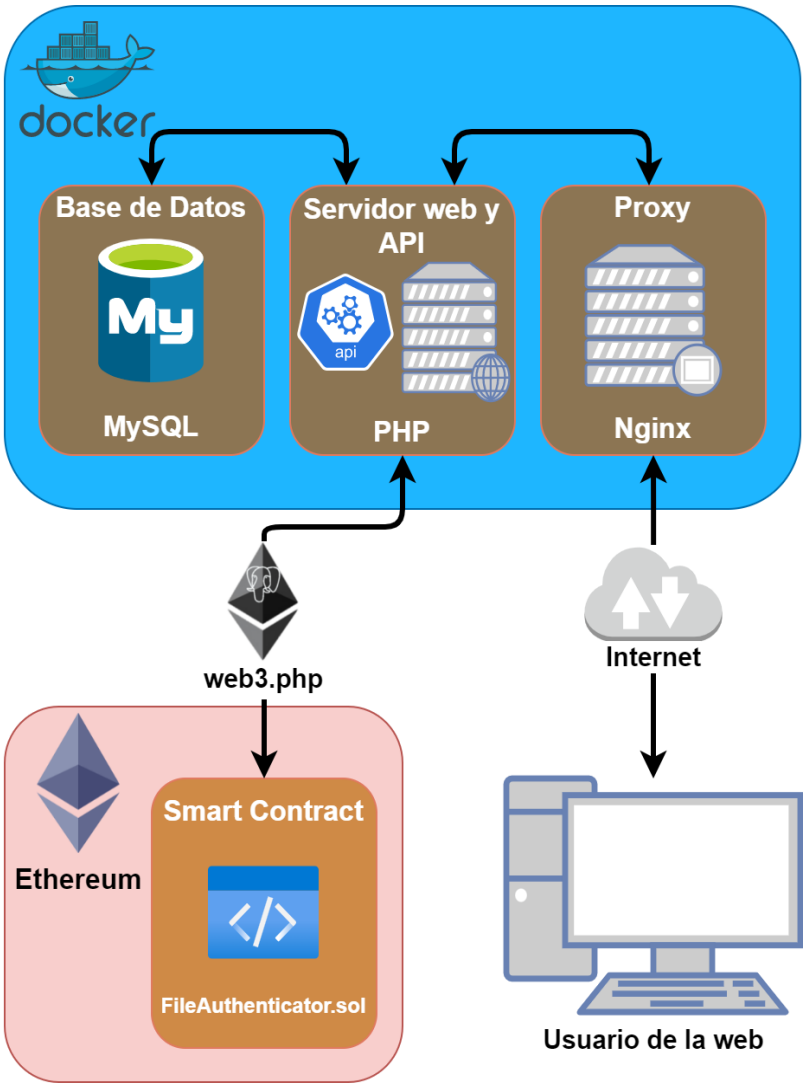
Fuente: Elaboración propia
Figura 14: Casos de uso del sistema desarrollado

Como se puede ver, en los casos de uso solo se han puesto dos actores, usuario y autentificador. Esto se debe a que se ha agrupado al comprador, vendedor y artista bajo el actor de usuario puesto que todos ellos poseen los mismos casos de uso.

5.2 DISEÑO

Tras haber definido los requisitos de usuario procedemos a realizar el diseño del sistema que se desarrollará en este Trabajo Fin de Grado, para así poder tener un sistema estructurado. Para comprender mejor el diseño se hará uso de diagramas de los componentes. Para entender la arquitectura se usará, diagramas de entidad relación para mostrar y explicar el diseño de la base de datos y por último un par de diagramas de secuencia para así poder ver el flujo y como interactúan las distintas clases entre sí. En cuanto a la interfaz web, se realizará un diagrama de navegación de lo que se desea desarrollar.

5.2.1 DISEÑO DE LA ARQUITECTURA



Fuente: Elaboración propia
Figura 15: Diseño de la arquitectura del sistema

Como se puede ver en la *Figura 15* el sistema se compone por dos grandes bloques, la red de Blockchain, en este caso Ethereum, y Docker, donde se encuentran tres contenedores, un servidor proxy, el servidor web y API y por último la base de datos.

Empezando por la red de blockchain, se ha elegido Ethereum, debido a su robustez, amplia documentación y soporte en cuanto al desarrollo de los contratos inteligentes. Es en esta misma donde se almacenará la información critica debido a las ventajas que aporta respecto a la integridad de la información. Cada vez que el sistema quiera realizar una interacción con la blockchain, este llamará al contrato inteligente para poder obtener o añadir la información deseada.

El bloque principal del sistema desarrollado en este Trabajo Fin de Grado es Docker. Se ha elegido Docker por las ventajas y facilidades que ofrece en todas las etapas de vida del código. Permite tener un entorno de desarrollo idéntico en distintas maquinas independiente del dispositivo en el que se este y el software que tenga previamente instalado, porque Docker cuando cree los contenedores, instalará lo necesario en cada uno de ellos. Por otro lado de cara al despliegue, permite tener los servicios separados lo que los hace, por diseño, independientes entre ellos. Esto permite la creación y destrucción de contenedores en función de la demanda del trafico en cada instante.

Como se ha mencionado en Docker se han creado tres contenedores distintos, MySQL, PHP y Nginx.

- **MySQL**: este contenedor contiene una imagen de MySQL en su versión 8.0.28. Esta base de datos contiene la información del servidor. Se utilizara para almacenar información de los usuarios y de los archivos que suben. Esto se hace porque si esta información estuviera en la Blockchain, la aplicación sería muy lenta, y los tiempos de respuestas serían muy grandes. Por eso se ha decidido crear este contenedor, pero se deja claro al usuario cuando este viendo información de archivos almacenada en la base de datos y cuando en la blockchain. Para que así exista total transparencia de cara a que el usuario, sepa cuando lea información, si existe una posibilidad que esta pudiera haber sido alterada, por estar almacenada en una base de datos tradicional, o cuando esta es inmutable, por haber sido almacenada en la cadena de bloques.

- **Servidor web y API**: este contenedor contiene una imagen de Alpine Linux[54] en su versión 3.15. En este linux se instala PHP 7.4 y todas las herramientas necesarias para la ejecución del código. En este contenedor contiene dos partes muy importantes del sistema, el servidor web y la API. Esta separación se ha realizado, para que así si un desarrollador o grupo de estos, desean implementar este servicio en su web o app, lo pueden hacer de manera sencilla. Cuando un usuario accede a la web, se genera la vista y esta es mandada al usuario.

54 https://www.alpinelinux.org/

Pero si el usuario realiza una acción distinta a obtener una vista de la web llama a distas URIs de la API, para realizar una acción u obtener información del servidor. El servidor en función de las acciones que este realizando puede realizar consultas con la base de datos mencionada, o utilizando la librería web3.php realizar una consulta a la blockchain. Si la consulta a la blockchain, requiere una modificación, el usuario deberá pagar una tasa a la red de blockchain, en cambio si la consulta no realiza ninguna modificación de la cadena de bloques, esta no conlleva costo alguno.

- **Nginx**: este contenedor contiene una imagen Alpine Linux en la que ha sido instalado Nginx. Este contenedor se utiliza como proxy de todas las peticiones que se realizan al servidor. Este limita el tamaño de los archivos, la cantidad de peticiones que puede hacer un usuario y en el caso que se intente acceder a una dirección no valida, muestra una pantalla de error. Este contenedor es al que acceder todos los usuarios, y si la petición es correcta, se redirige al contenedor del servidor web y la API, para que sea procesada.

Al utilizar una arquitectura basada en contenedores, si se quisiera desplegar, se podría levantar cada contenedor en maquinas distintas e incluso añadir balanceadores de carga y varios instancias de un mismo contenedor. De esta forma el sistema puede crecer o decrecer en tiempo real, dependiendo de la cantidad de tráfico que este recibiendo la web y la API. Esta capacidad de realizar un escalado horizontal en caliente viene dada por haber realizado una arquitectura basada en contenedores.

5.2.2 ESTRUCTURA DE DATOS

Habiendo definido la arquitectura del sistema, debemos definir la estructura de los datos. Se va a dividir en dos partes, la base de datos y blockchain.

5.2.2.1 Base de datos

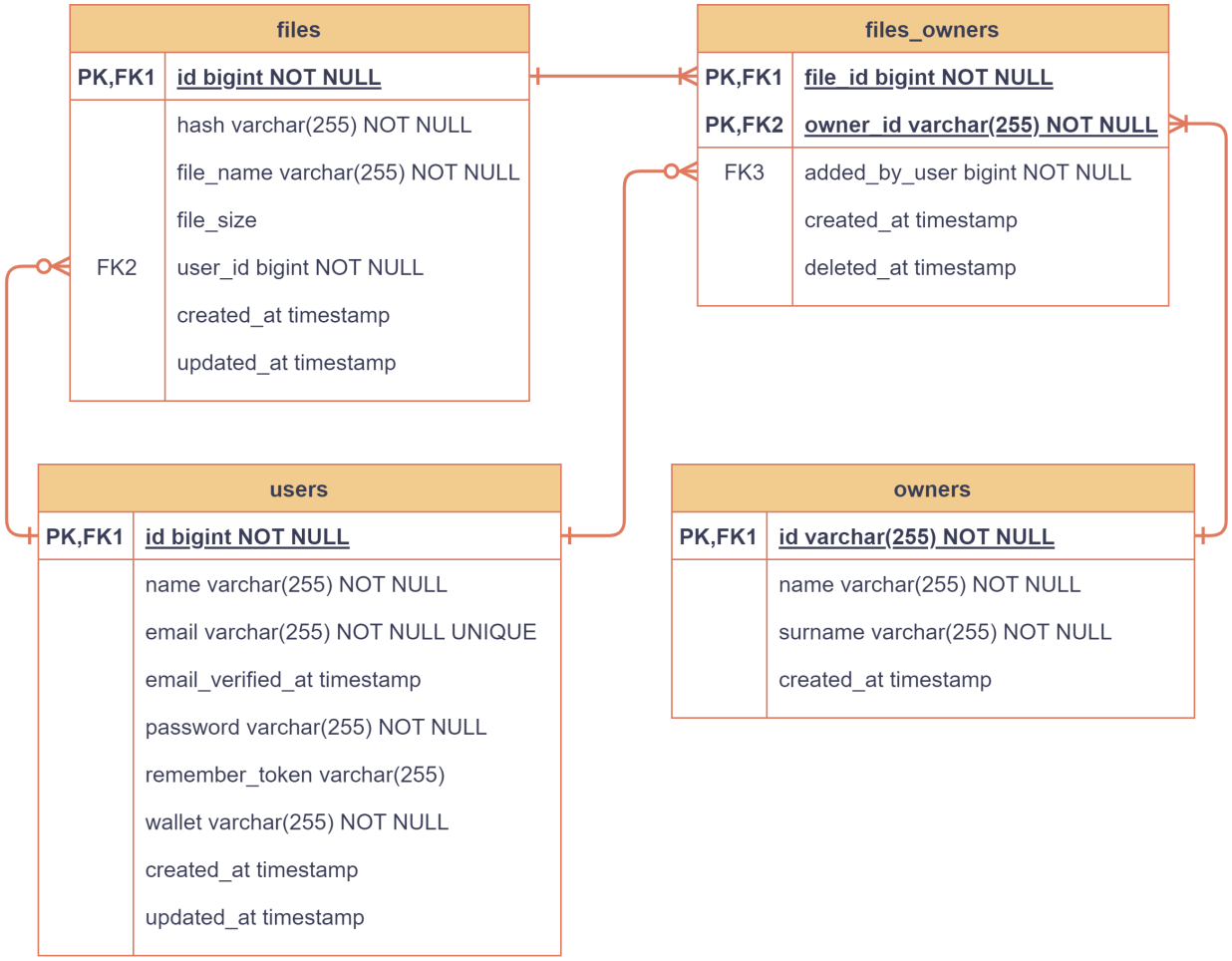
Como se ha comentado antes, en la base de datos, se almacenará la información necesaria para el registro y gestión de usuarios, al igual que información de los archivos. Esta información de los archivos, se almacena para mayor rapidez de las respuestas. A su vez se usa para poder acceder a información de un archivo en la blockchain, sin necesidad de poseer el archivo.

En la *Figura 16* podemos ver el diagrama de entidad relación de la base de datos, correspondiente a los archivos. Para entender mejor cada tabla y su finalidad se explicarán por separado, para luego detallar las relaciones entre ellas.

- ***users***: esta tabla almacena la información de lo que se había definido como autentificador en el apartado 5.1.1. Las columnas más relevantes de esta tabla son:

- *name*: es el nombre completo y legal del autentificador.
- *email*: es el correo de contacto del autentificador, y también sirve para poder iniciar sesión.
- *wallet*: es la dirección de la cartera del autentificador, esta cartera es donde se cobrará las tasas de blockchain, y es la que se usará para relacionar unívocamente el autentificador al archivo en la blockchain.
- *password*: este campo contiene la contraseña para poder iniciar sesión. Este campo no se almacena como texto plano, si no que es salteado de manera aleatoria y luego se le aplica una función de hash

- ***owners***: esta tabla almacena la información de los dueños de obras de arte. Se almacena su nombre, apellidos y la columna del id, es algún tipo de identificador legal de la persona, como por ejemplo el DNI o el número de pasaporte.
- ***files***: esta tabla almacena parte de la información que se ha subido a blockchain de un archivo.
- ***files_owners***: esta tabla almacena la relación de uno o varios dueños con un archivo, en un rango de fechas.



Fuente: Elaboración propia
Figura 16: Diagrama Entidad Relación de la base de datos

Habiendo explicado las tablas, podemos explicar como se relacionan entre ellas. Como se puede ver en la *Figura 16* un usuario (autentificador) puede tener varios o ningún archivo. Cada uno de esos archivos poseen uno o varios dueños. A su vez un autentificador puede haber añadido varias o ninguna asociación entre el dueño y el archivo. Por último un dueño puede estar asociado a una o varias obras.

5.2.2.2 Blockchain

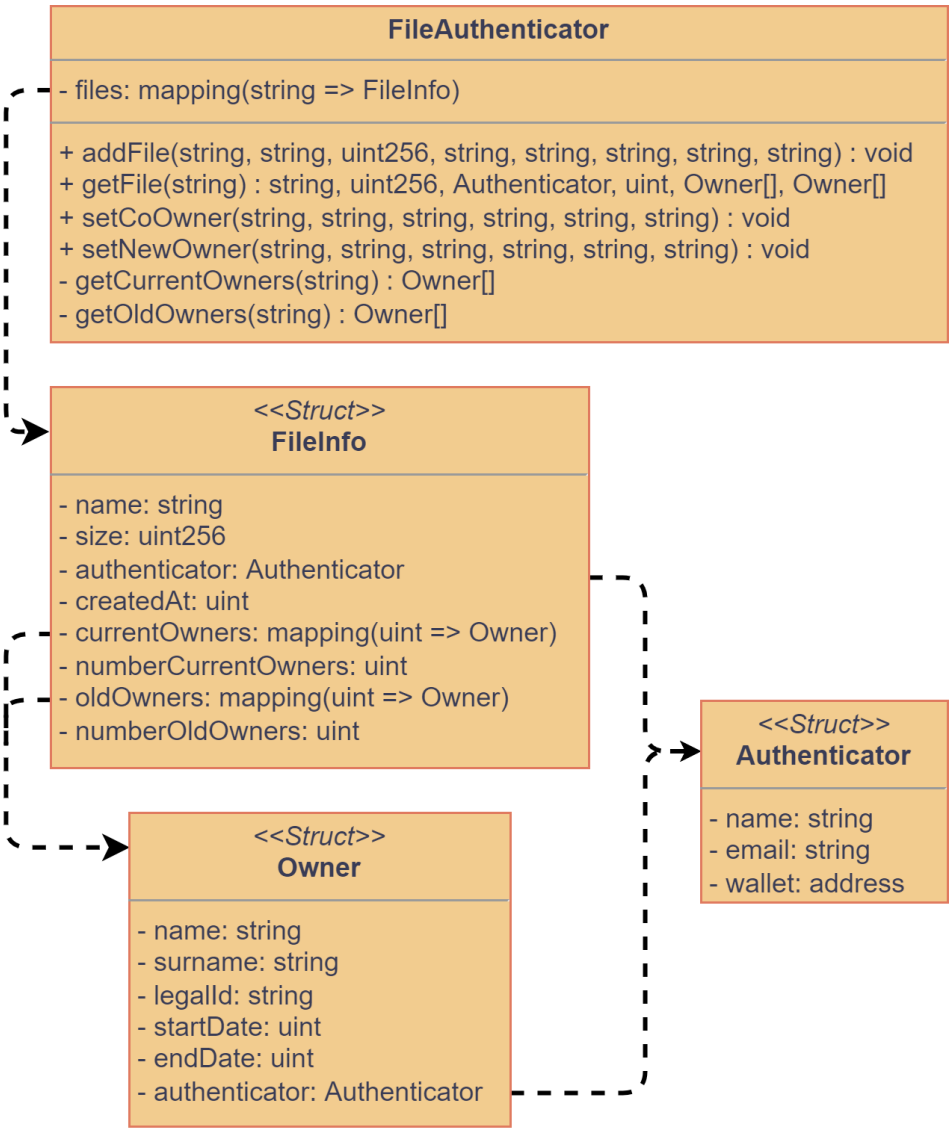
Para la funcionalidad requerida, solo se necesita programar una clase en blockchain. Esta clase tendrá un único atributo, cuatro métodos públicos y dos privados. En la *Figura 17* se puede ver la clase `FileAuthenticator`, con sus métodos, atributo y las tres estructuras que están relacionadas.

Empezaremos explicando el atributo que posee la clase, `files`, es una variable mapeada, se asemeja a los diccionarios de Python, donde los valores se almacenan en parejas de clave y valor. En este caso se esta tomando como clave el hash del fichero, y como valor asociado la estructura de datos `FileInfo`.

La estructura `FileInfo`, posee toda la información del archivo. El nombre del archivo, su tamaño, la fecha de cuando se subió a la cadena de bloques, el número de dueños actuales, el número de dueños antiguos. Estos campos son de tipos ya definidos por Solidity, pero la esta estructura posee un campo, `authenticator` que hace referencia a la estructura `Authenticator`. También posee dos mapeos de la estructura `Owner`, uno de estos es para almacenar la información de los dueños actuales, y el otro es para almacenar la información de los dueños anteriores. Ambas variables utilizan un uint como clave, el cual por diseño se ha decidió que sera un valor que empiece en cero y vaya aumentado en incrementos de una unidad.

La estructura `Owner`, posee la información necesaria para identificar a un dueño de una obra de arte. Para ello se almacena en la cadena de bloques la siguiente información del dueño: su nombre, apellidos, algún tipo de identificador legar, como por ejemplo el DNI, la fecha en la que comenzó a ser dueño de la obra a la que este asociado, la fecha en la que dejó de ser el dueño de esta misma y por último, el autentificador que modificó la información del dueño por última vez. Cabe destacar que durante el tiempo en el que el dueño haya poseído la obra, el autentificador asociado al dueño, es en todo momento la persona que le puso como dueño, certificando el traspaso del dueño anterior al nuevo. Cuando el dueño actual, pasa a pertenecer a la categoría de dueños anteriores, el autentificador asociado, será el que certificó el traspaso de este a otro dueño (el nuevo).

Para terminar las estructuras, tenemos la estructura Authenticator, esta contiene la información del autentificador que certifica la modificación a adición a la cadena de bloques. Esta estructura contiene el nombre completo del autentificador, su correo electrónico y su dirección de la cartera.



Fuente: Elaboración propia

Figura 17: Diagrama de la clase del contrato inteligente

Habiendo explicado los atributos, se pueden explicar los métodos. Se explicarán en el mismo en el que aparecen en al *Figura 17*.

- **addFile**: este método recibe toda la información de la estructura `FileInfo`, salvo algunos campos que los generá de manera automática el propio método. Cuando se añade un fichero, se empieza con un único dueño, si luego se quiera añadir co-dueños se debe utilizar el método **setCoOwner**. Y para realizar un traspaso de dueños se utiliza el método **setNewOwner**. En el caso que se intente añadir un fichero que ya estuviera en la cadena de bloques, este se descarta la transacción y vuelve al estado anterior a la recepción de la petición.

- **getFile**: este método en función del hash recibido, debe devolver toda la información que haya de ese archivo en la cadena de bloques Si no se encuentra un archivo con dicho hash, se devolverá un error.

- **setCoOwner**: este método añade un dueño, fijando como fecha de inicio, como estado de dueño, el día y hora en el que el autentificador llama a este método.

- **setNewOwner**: este método fija un nuevo dueño, remplazando los anteriores. Pero antes de remplazarlos, estos son añadidos al histórico de dueños, fijando la fecha de fin de la posesión de la obra.

- **getCurrentOwners**: este método es privado y sirve para obtener un vector de los dueños actuales del archivo asociado al hash pasado. La funcionalidad principal de este método es la conversión de los dueños, puesto que para devolver la información fuera de la blockchain, debe estar como vector, y no como un mapeo.

- **getOldOwners**: este método es similar al anterior, pero en vez de devolver los dueños actuales, devuelve un vector del histórico de los dueños

5.2.3 DIAGRAMA DE SECUENCIA

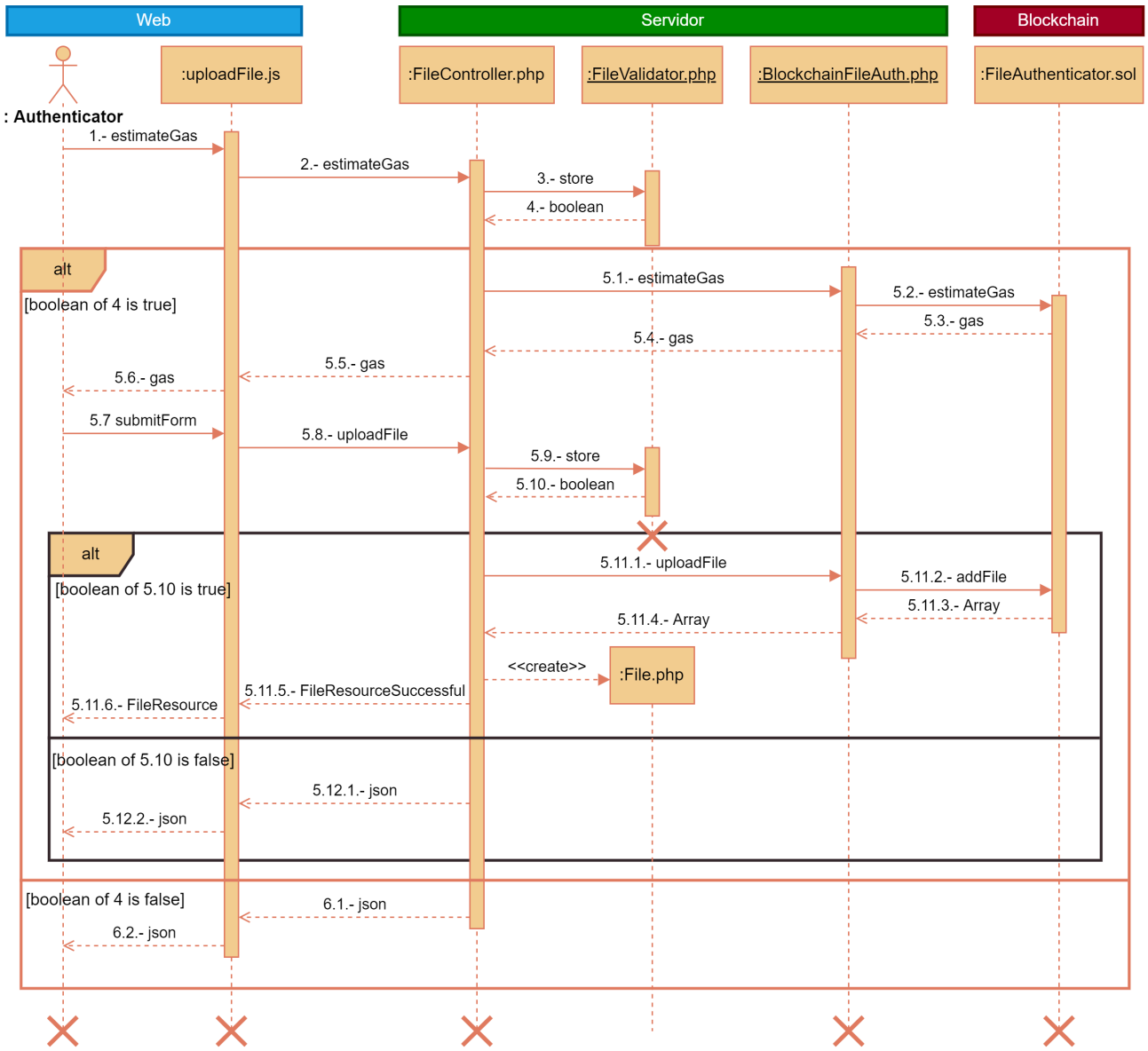
Para completar el diseño del sistema a desarrollar en este Trabajo Fin de Grado, se han realizado dos diagramas de secuencia en los que se podrá ver el flujo que realizaran los usuarios al realizar una acción.

En estos diagramas se omitirá todo lo relacionado con la autenticación de los usuarios, pues no es la parte primordial del flujo.

El primer diagrama (*Figura 18*) muestra el flujo que un usuario sigue cuando va a subir un archivo a la blockchain. El flujo consta de dos partes, una la estimación del gas y las segunda la subida del archivo como tal.

Empezando por la estimación de gas, se manda al servidor toda la información que se desea añadir a la blockchain, para que esta estime cuanto va a tener que pagar de tasas. Antes de mandar la información al contrato inteligente para que realice una estimación se valida que la información este completa y correcta, para evitar errores al ejecutar el contrato inteligente. Esta validación se realiza en el paso 3, por eso a partir de la contestación, hay dos posibles situaciones. En la primera de ellas la información es correcta, por lo que se procede a realizar el calculo de la estimación del gas. En la segunda situación, la información es incorrecta o incompleta, por lo que se notifica al usuario del fallo.

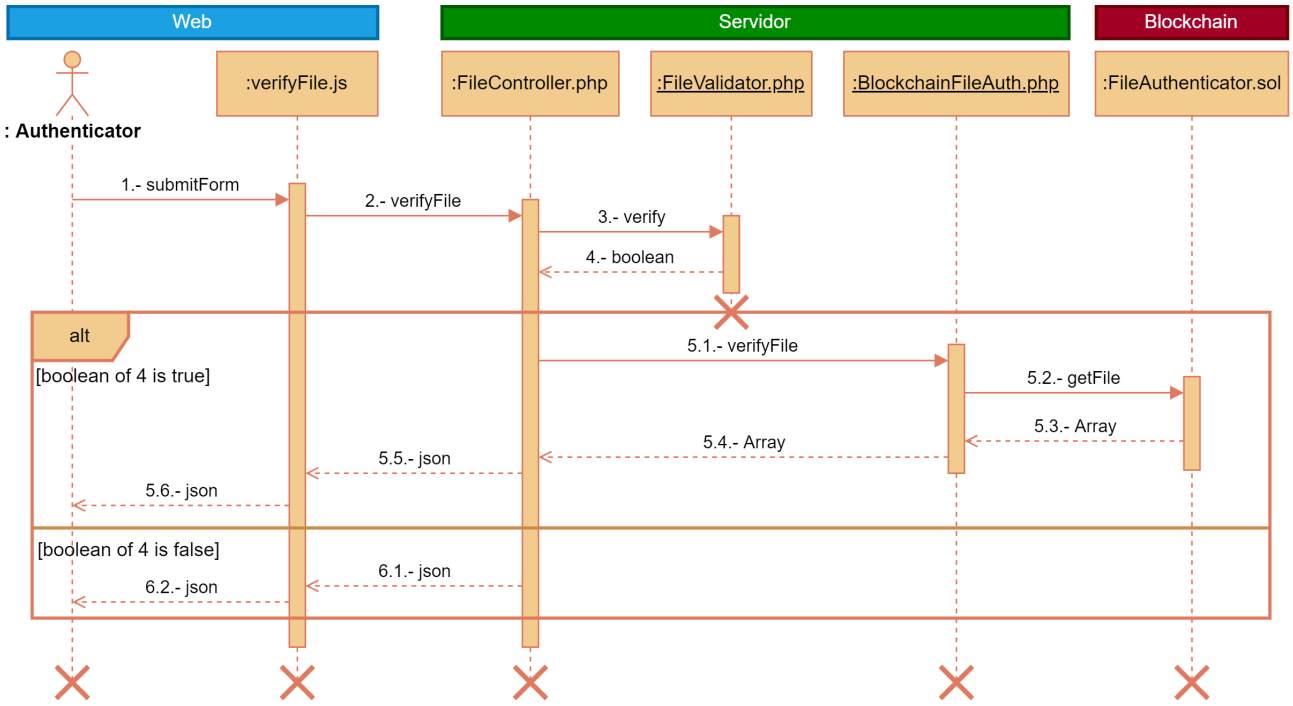
Volviendo al primer escenario, donde se ha llegado a mostrar al usuario cuanto serian las tasas por la subida del archivo a la blockchain, si este decide proseguir, se continuaría la secuencia por el paso 5.7. Al igual que con la estimación de gas, en esta parte del flujo se vuelve a verificar que la información subida este completa y correcta. Si la información es correcta se añade la información a la cadena de bloques, el caso que no lo sea, se notifica al usuario del problema en los datos suministrados



Fuente: Elaboración propia

Figura 18: Diagrama de secuencia para la subida de un certificado a la blockchain

El otro diagrama de secuencia representa el flujo que sigue un usuario cuando desea comprobar la información asociada a un certificado de autenticidad (*Figura 19*). Este diagrama es muy parecido a la segunda parte del diagrama anterior. Pero los procesados que realizan internamente los métodos son completamente distintos. El proceso que más se diferencia es el del propio contrato inteligente, puesto que en esta situación al solo leer de la cadena de bloques no hay que pagar tasas a la blockchain.



Fuente: Elaboración propia
Figura 19: Diagrama de secuencia para la verificación y de un certificado

5.3 IMPLEMENTACIÓN

Habiendo explicado el diseño del sistema, se procederá a detallarár de manera más especifica las funcionalidades que se han implementado en el desarrollo de software de este Trabajo Fin de Grado. Se detallará la funcionalidad implementada de los tres componentes más importantes en este desarrollo, el contrato inteligente, la API y la web. Para el caso del contrato inteligente, se incluirá el código en este propio documento, al no ser muy grande, pero el resto del código sera subido a un repositorio publico de GitHub, y se suministrara, el pertinente enlace para poder acceder.

5.3.1 CONTRATO INTELIGENTE

Como se detalla en la *Figura 17*, el contrato inteligente posee un único atributo, el cual es un mapeo de la estructura *file*. Esta estructura a su vez tiene otras dos anidadas. A continuación se puede ver el código que define este atributo y las estructuras.

```
struct Authenticator {
	string name;
	string email;
	address wallet;
}
struct Owner {
	string name;
	string surname;
	string legalId;
	uint startDate;
	uint endDate;
	Authenticator authenticator;
}
struct FileInfo {
	string name;
	uint256 size;
	Authenticator authenticator;
	uint createdAt;
	mapping(uint => Owner) currentOwners;
	uint numberCurrentOwners;
	mapping(uint => Owner) oldOwners;
	uint numberOldOwners;
}
```

Pasando a los métodos que posee el contrato, empezamos por los métodos *getCurrentOwners* y *getOldOwners*. Ambos métodos son muy parecidos, como se puede ver abajo. Estos devuelven un vector de dueños, que dependiendo del método, serán o dueños actuales o dueños anteriores. Por diseño se ha decidido que estos métodos sean privados y para que un usuario pueda obtener la información de los dueños, lo haga a través de *getFile*.

```
function getCurrentOwners(string memory _fileHash) private view returns (Owner[] memory){
	uint numberOwners = files[_fileHash].numberCurrentOwners;
	Owner[] memory owners = new Owner[](numberOwners);
	for(uint i = 0; i < numberOwners; i++) {
		owners[i] = files[_fileHash].currentOwners[i];
	}
	return (owners);
}
function getOldOwners(string memory _fileHash) private view returns (Owner[] memory){
	uint numberOwners = files[_fileHash].numberOldOwners;
	Owner[] memory owners = new Owner[](numberOwners);
	for(uint i = 0; i < numberOwners; i++) {
		owners[i] = files[_fileHash].oldOwners[i];
	}
	return (owners);
}
```

El tercer método es *getFile*, este método te devuelve toda la información almacenada en la cadena de bloques para un hash determinado. Este método al ser público debe comprobar si el hash recibido existe ya en la cadena de bloques. Esto como se puede ver en el código, se realiza accediendo a la fecha de creación del mapeo del hash recibido. Esto se hace, porque ese campo nunca será cero siempre que el archivo exista, de ahí que se utilice como campo para realizar la comprobación. En el caso de no encontraste en la cadena de bloques, descarta todos los cambios realizados en la llamada. Pero si el archivo si que se haya en la cadena, se devuelve toda la información, haciendo uso de los dos métodos explicados antes.

```
function getFile(string memory _fileHash) public view returns (
                    string memory fileName, uint256 fileSize,
                    Authenticator memory authenticator, uint fileCreatedAt,
                    Owner[] memory currentOwners, Owner[] memory oldOwners) {
        if (files[_fileHash].createdAt > 0) {
              fileName = files[_fileHash].name;
              fileSize = files[_fileHash].size;
              authenticator = files[_fileHash].authenticator;
              fileCreatedAt = files[_fileHash].createdAt;
              currentOwners = getCurrentOwners(_fileHash);
              oldOwners = getOldOwners(_fileHash);
        } else {
              revert("File not found");
        }
  }
```

El siguiente método añade para un nuevo archivo su información a la cadena de bloques. El método es *addFile*. Este al contrario que *getFile*, requiere que no haya ningún archivo en la cadena de bloques con el mismo hash, puesto que no se desea sobrescribir la información de las variables. Para ello se utiliza el mismo procedimiento que antes. Si *createdAt* es mayor que cero indica que ya existe el archivo, por lo que ejecuta la función de *revert*, volviendo al estado previo de la llamada y no cobrando las tasas al ordenante. Cuando se esta añadiendo un archivo, la fecha de *endDate*, que indica la fecha de final de posesión de la obra, se deja fijado a 0.

```
  function addFile(string memory _fileHash, string memory _fileName, uint256 _fileSize,
                          string memory _authName, string memory _authEmail,
                          string memory _ownerName, string memory _ownerSurname, string memory _ownerLegalId) public {
        if (files[_fileHash].createdAt > 0) {
              revert("File already in blockchain");
        } else {
              uint currentTimestamp = block.timestamp;
              Authenticator memory authenticator = Authenticator(_authName, _authEmail, msg.sender);
              files[_fileHash].name = _fileName;
              files[_fileHash].size = _fileSize;
              files[_fileHash].authenticator = authenticator;
              files[_fileHash].createdAt = currentTimestamp;
              files[_fileHash].currentOwners[0] = Owner(_ownerName, _ownerSurname, _ownerLegalId, currentTimestamp, 0,
                                                        authenticator);
              files[_fileHash].numberCurrentOwners = 1;
              files[_fileHash].numberOldOwners = 0;
        }
  }
```

Para terminar el contrato inteligente, faltan los métodos *setCoOwner* y *setNewOwner*. Estos métodos modifican la posesión de los archivos, ya sea porque añade co-dueños o sustituye todos los dueños actuales por uno nuevo. El método *setCoOwner*, como se puede ver en código, si el archivo al que se quiere añadir un co-dueño existe, este es añadido al mapeo de dueños actuales del archivo pasado. En cambio el método *setNewOwner*, copia todos los dueños actuales al mapeo de dueños anteriores y añade el nuevo dueño. Cuando se está realizando el proceso de copia de los dueños, se fija la fecha del momento de ejecución como *endDate*, indicando que ese dueño, ahora es un dueño anterior, también se fija como autentificador del dueño anterior, la persona que esta certificando el traspaso de la obra de arte.

```
    function setCoOwner(string memory _fileHash,
                                string memory _ownerName, string memory _ownerSurname, string memory _ownerLegalId,
                                string memory _authName, string memory _authEmail) public {
          if (files[_fileHash].createdAt > 0) {
                uint currentTimestamp = block.timestamp;
                Authenticator memory authenticator = Authenticator(_authName, _authEmail, msg.sender);
                uint numberCurrentOwners = files[_fileHash].numberCurrentOwners;
                files[_fileHash].currentOwners[numberCurrentOwners] = Owner(_ownerName, _ownerSurname, _ownerLegalId,
                                                                            currentTimestamp, 0, authenticator);
                files[_fileHash].numberCurrentOwners = numberCurrentOwners + 1;
          } else {
                revert("File not found");
          }
    }

    function setNewOwner(string memory _fileHash,
                                string memory _ownerName, string memory _ownerSurname, string memory _ownerLegalId,
                                string memory _authName, string memory _authEmail) public {
          if (files[_fileHash].createdAt > 0) {
                uint currentTimestamp = block.timestamp;
                Authenticator memory authenticator = Authenticator(_authName, _authEmail, msg.sender);
                uint numberCurrentOwners = files[_fileHash].numberCurrentOwners;
                uint numberOldOwners = files[_fileHash].numberOldOwners;
                for (uint i = 0; i < numberCurrentOwners; i++) {
                      uint j = numberOldOwners+i;
                      files[_fileHash].oldOwners[j] = files[_fileHash].currentOwners[i];
                      files[_fileHash].oldOwners[j].endDate = currentTimestamp;
                      files[_fileHash].oldOwners[j].authenticator = authenticator;
                }
                files[_fileHash].currentOwners[0] = Owner(_ownerName, _ownerSurname, _ownerLegalId, currentTimestamp, 0,
                                                          authenticator);
                files[_fileHash].numberCurrentOwners = 1;
                files[_fileHash].numberOldOwners = numberOldOwners + numberCurrentOwners;
          } else {
                revert("File not found");
          }
    }
```

5.3.2 API

Habiendo explicado el contrato inteligente, procedemos a explicar los distintos *endpoints* de la API. Estos *endpoints*, interactúan tanto como con la blockchain, como con el servidor y base de datos.

Los *enpoints* que se han implementado en el sistema desarrollado, son los siguientes:

- POST → /api/register → crea una cuenta de usuario
- POST → /api/login → inicia sesión
- GET → /api/logout → cierra sesión del usuario con la sesión activa
- GET → /api/user → obtener la información del usuario con la sesión activa
- GET → /api/user/files → obtiene los archivos subidos por el usuario con la sesión activa
- POST → /api/file → subir el archivo y su información a la blockchain
- POST → /api/file/estimateGas → estima cuanto gas se va a consumir por subir el archivo a la blockchain
- POST → /api/file/verify → verifica y devuelve la información en la blockchain de un archivo
- GET → /api/file/{fileId} → obtiene información del archivo
- POST → /api/file/coowner → añade a otra persona en la blockchain como co-dueño de un archivo
- POST → /api/file/owner → cambia a un nuevo dueño en la blockchain de un archivo

A continuación se explicará con mayor detalle los endpoints más importantes de este desarrollo, los que interactúan con la blockchain.

El *endpoint* /api/file, como se ha mencionado, es el encargado de procesar los datos y mandarlos al contrato inteligente para que se añada a la cadena de bloques. Este *endpoint* recibe en cuerpo del POST un json con el siguiente contenido:

- Token de autenticación: este campo es obligatorio, puesto que identifica que autentificador es el que esta realizando la petición.

- Archivo: este campo es obligatorio y es el archivo como tal que se desea subir a la blockchain. No puede ser superior a 10 MB.

- Dueño: este campo es obligatorio y contiene en formato json la información del dueño. Esta información de incluir, el nombre, sus apellidos y algún tipo de identificador legal, como el número de DNI, el número de pasaporte, etc.

- Gas: este campo es opcional y contiene el máximo gas que esta dispuesto a pagar el usuario por realizar la transacción. Habitualmente será el devuelto por el *endpoint* /api/file/estimateGas

Cuando el servidor recibe una petición lo primero que hace es verificar que la petición posee el token y que este sea valido. Una vez validado el token se verifican el resto de campos, tanto que estén presentes, como que sean del tipo correcto. Si todo lo anterior se ha cumplido, se procede a obtener la información de la base de datos del autentificador, para así tener toda la información para llamar al método *addFile* del contrato inteligente. Cuando la contestación del contrato inteligente es satisfactoria, se almacena la información recibida inicialmente, más la recibida de la blockchain en la base de datos, como se ha explicado anteriormente, esto se hace para que los tiempos de respuesta de la aplicación sean más rápidos. Habiendo guardado la información en la base de datos, se genera un json, con información de la transacción realizada en la blockchain y se devuelve al usuario.

Otro *endpoint* fundamental para el funcionamiento del sistema es /api/file/verify. Este como se ha explicado anteriormente, es el encargado de verificar que un archivo esta en la blockchain y en ese caso, obtener su información. A diferencia de la petición de subir un archivo nuevo, este no requiere estar dado de alta en el sistema, por lo que el servidor solo espera un único parámetro, un archivo. Este archivo no debe superar los 10 MB.

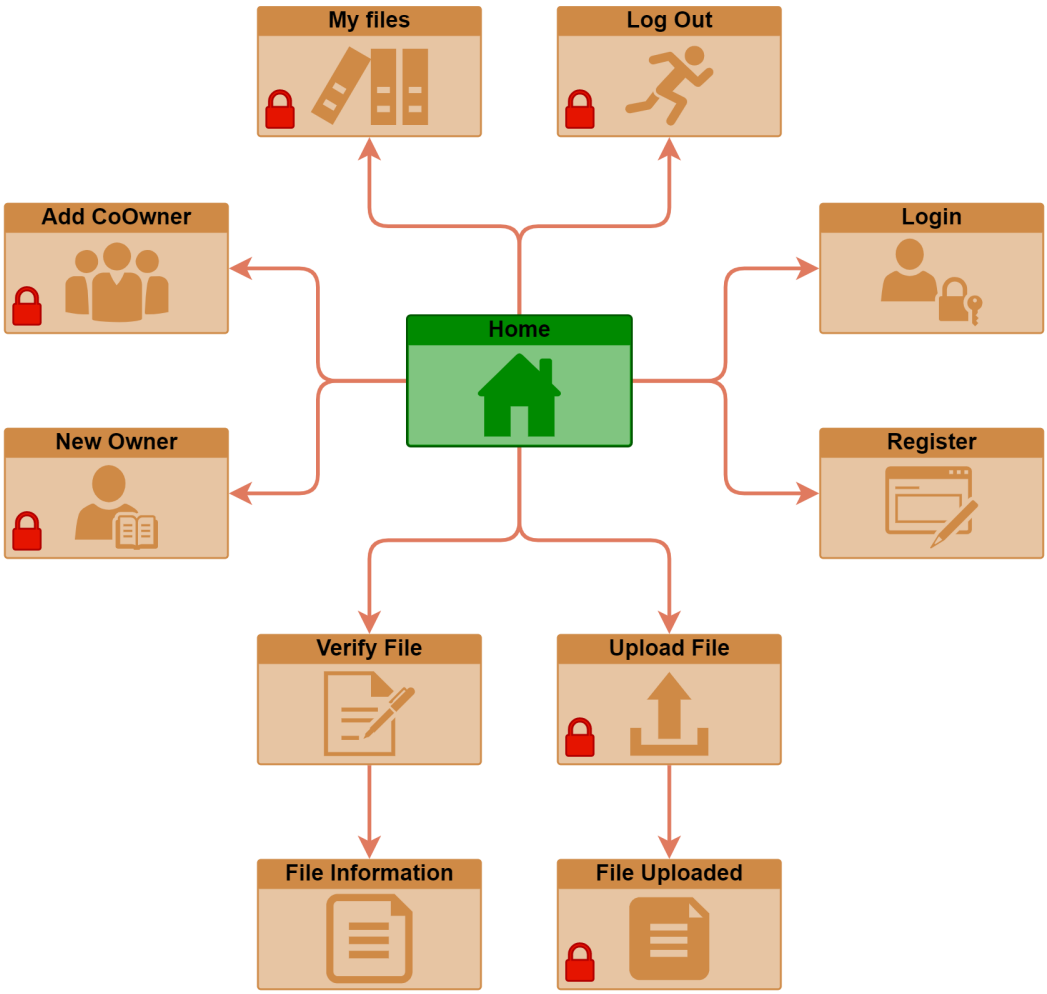
Una vez recibido el servidor el archivo, este calcula su hash, para así poder llamar al método *getFile* del contrato inteligente. Cuando el contrato conteste de vuelta, puede haber dos tipos de respuesta, una de ellas en la que se ha encontrado el fichero y nos devuelve toda su información. La otra situación es en la que no se ha encontrado el fichero y por tanto devuelve un error. En el caso de haber obtenido de manera correcta la información del archivo, esta se manda de vuelta para que el usuario la pueda ver. Pero en el caso de no encontrase el archivo, se comunica al usuario que el archivo no se encuentra en la cadena de bloques. Esto puede ser porque nunca se ha subido, o a sido modificado con respecto al original.

5.3.3 PÁGINA WEB

Por último, esta la página web. Su objetivo es ser sencilla y fácil de usar, siendo intuitiva, para no poner impedimentos en usar el sistema. La web puede hacer todo lo que permite la API, pero una manera intuitiva y sin conocimientos de programación.

Para conseguir abarcar toda la funcionalidad deseada de la web se han desarrollado distintas vistas, cada una con una única funcionalidad.

En la *Figura 20* se puede ver el diagrama de navegabilidad de todas las vistas de la web. Todas las vistas que posean un candado, requieren estar autenticado, por lo que si se intenta acceder a alguna de ellas, estas automáticamente redirigen al usuario a la vista *LogIn* para que pueda iniciar sesión.



Fuente: Elaboración propia
Figura 20: Diagrama de navegabilidad de la página web desarrollada

Como se ha explicado en la fase de diseño, toda acción que conlleve una modificación de la cadena de bloques, requiere que el usuario este autenticado, para poder ligar los cambios a el. Pero la parte de lectura de la cadena de bloques, no requiere iniciar sesión, ni pagar ninguna tasa. Solo se requiere el archivo del que se desea verificar la informacion en la blockchain.

Todas las vistas poseen una barra de navegación en la parte superior, desde donde se puede acceder a otras vistas. A su vez, es desde ahí desde donde se accede a las opciones de inicio de sesión, registrarse, y cerrar sesión. Solo las vistas de *File Information* y *File Uploaded* no pueden ser accedidas desde la barra de navegación, puesto que se son vistas que se muestran despues de haber subido un archivo al servidor, para poder obtener su información o subirla a la cadena de bloques, respectivamente.

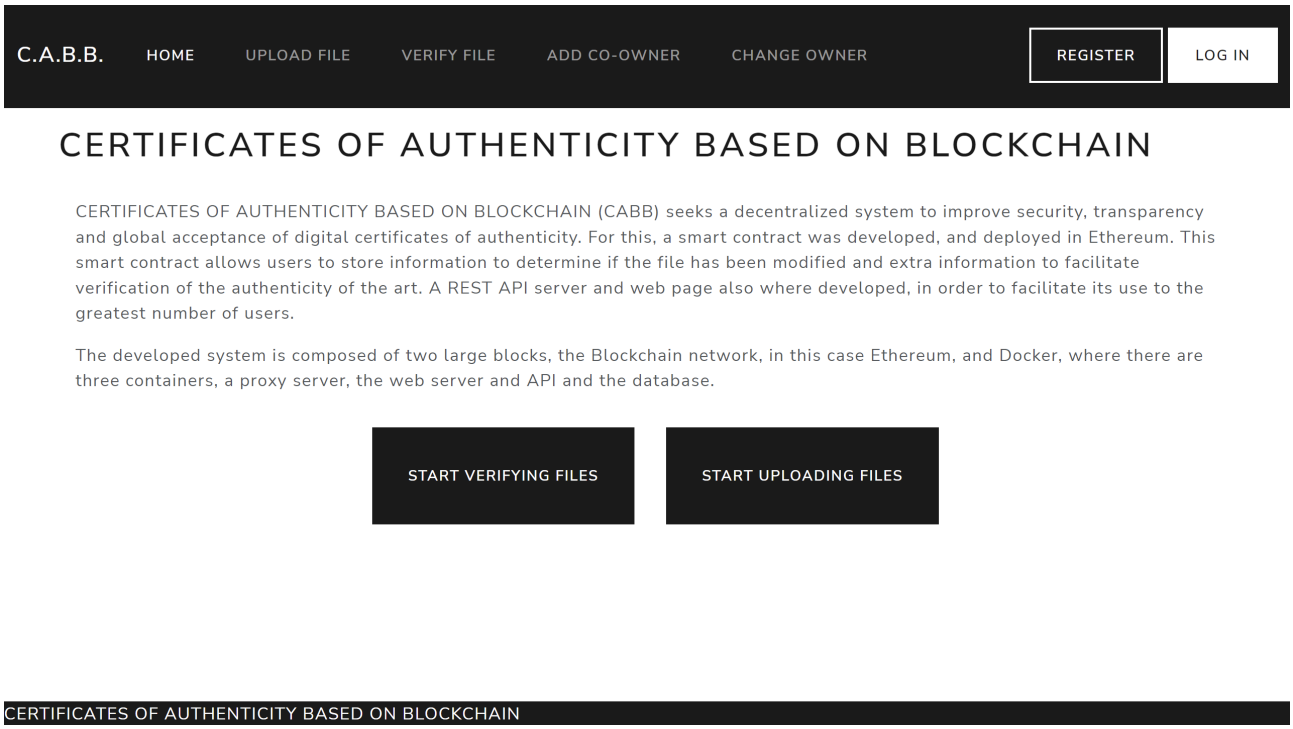
En la medida de lo posible se ha diseñado una web responsiva, por lo que esta se adapta al tamaño de la pantalla del dispositivo desde donde se esta accediendo. Por último se ha decidido realizar la página web en inglés puesto que es el idioma más hablado en todo el mundo.

Capítulo 6 Análisis de resultados

Habiendo desarrollado el sistema según lo especificado en el *Capítulo 5*, se pudieron a su vez alcanzar todos los objetivos.

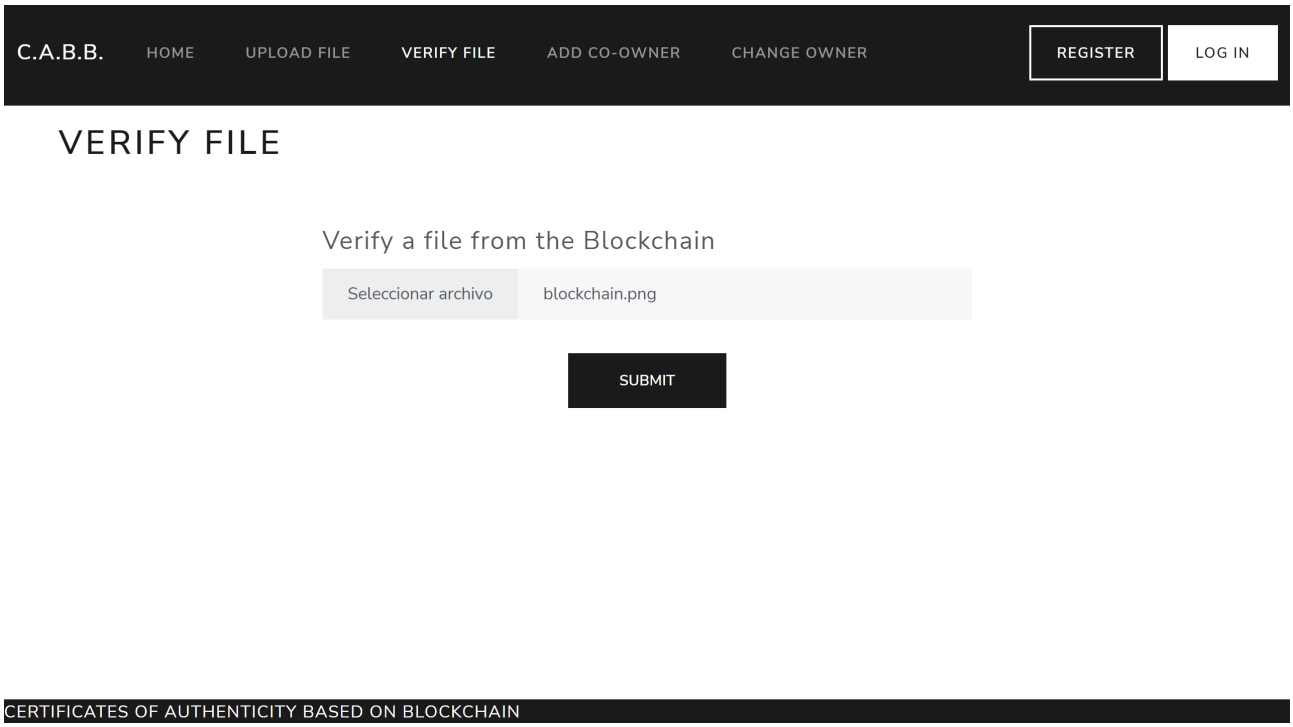
Puesto que el proyecto se centra en conseguir que la mayoría pueda utilizar el sistema, ya sea desde la web, desde la API o directamente desde el contrato inteligente, en este apartado se explicará y analizará el producto final de la página web, puesto que es la que posee un mayor potencial de uso. Al no requerir conocimientos específicos para poder usar una página web. Se explicará primero las posibles vistas que tiene un usuario sin que este registrado, luego se explicará la pantalla de registro y por último todas las posibilidades que tiene un usuario registrado.

Empezando por un usuario sin cuenta, según accede a la web se encuentra con la vista de la *Figura 21*. En esta puede leer un pequeña descripción de la web y también acceder al resto de vistas de todo el sistema.



Fuente: Elaboración propia
Figura 21: Pantalla de inicio de la web

Como usuario de invitado, solo puede acceder a la vista de verificación de un archivo. Desde esta vista, según esta definido en el diseño, el usuario ha de ser capaz de obtener toda la información almacenada en la blockchain de un archivo. Para ello el usuario ha de acceder a la vista de verificación y seleccionar el archivo sobre el que desea realizar la consulta (Figura 22).

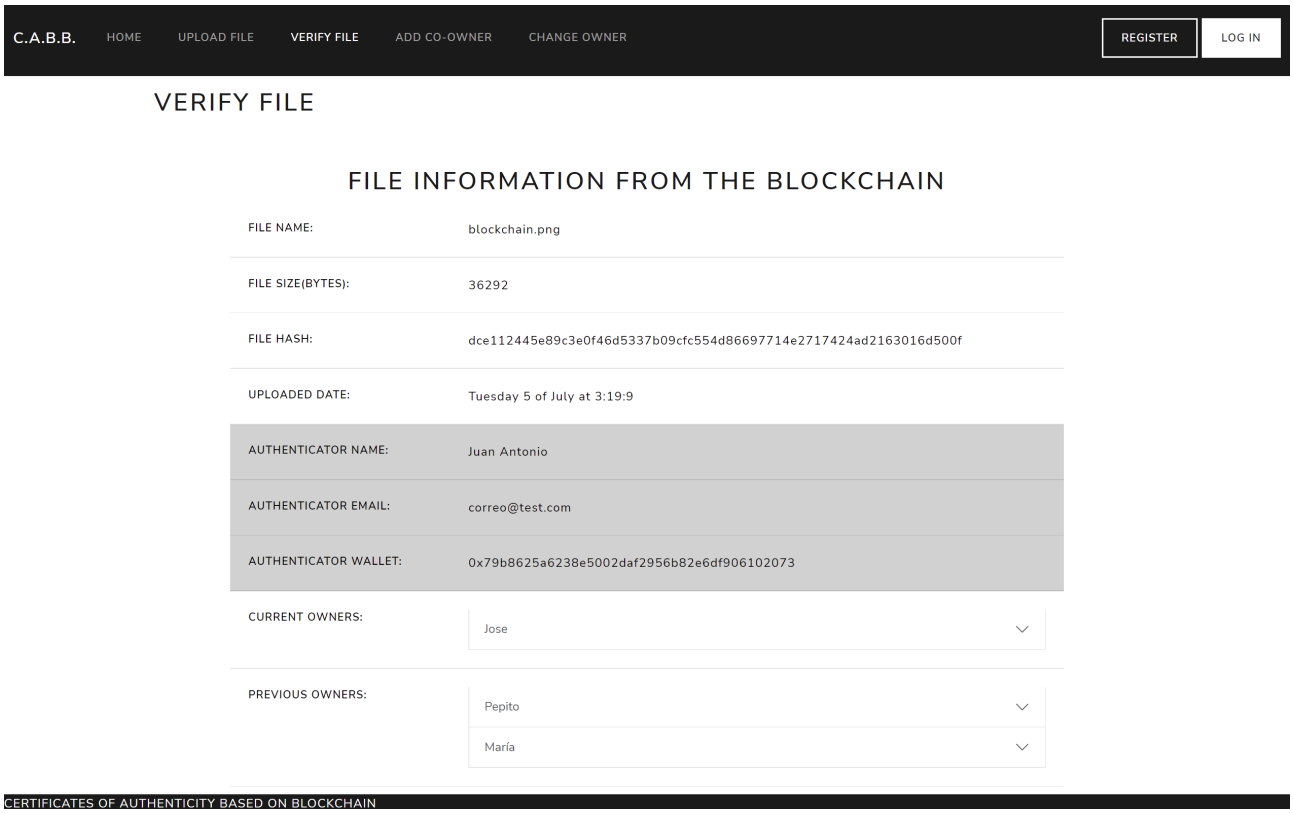


Fuente: Elaboración propia
Figura 22: Pantalla de verificación de archivos de la web

Una vez subido el fichero, el servidor calcula el hash del archivo y lo mando a la cadena de bloques para obtener toda la informacion sobre este. De la informacion obtenida, sin modificarse va directo al usuario, donde es entonces donde se modifica por primera vez. Se modifica la informacion para hacerla más legible. En la *Figura 23* se puede ver como se muestra toda la informacion al usuario. Cabe destacar, que por una limitación de la librería, se tuvieron que modificar los contratos inteligentes. La librería aún no soporta estructuras, por lo que se crearon métodos para obtener los dueños actuales y anteriores. El mismo navegador es el que determina cuantos debería mostrar, tras hacer una solicitud para obtenerlos.

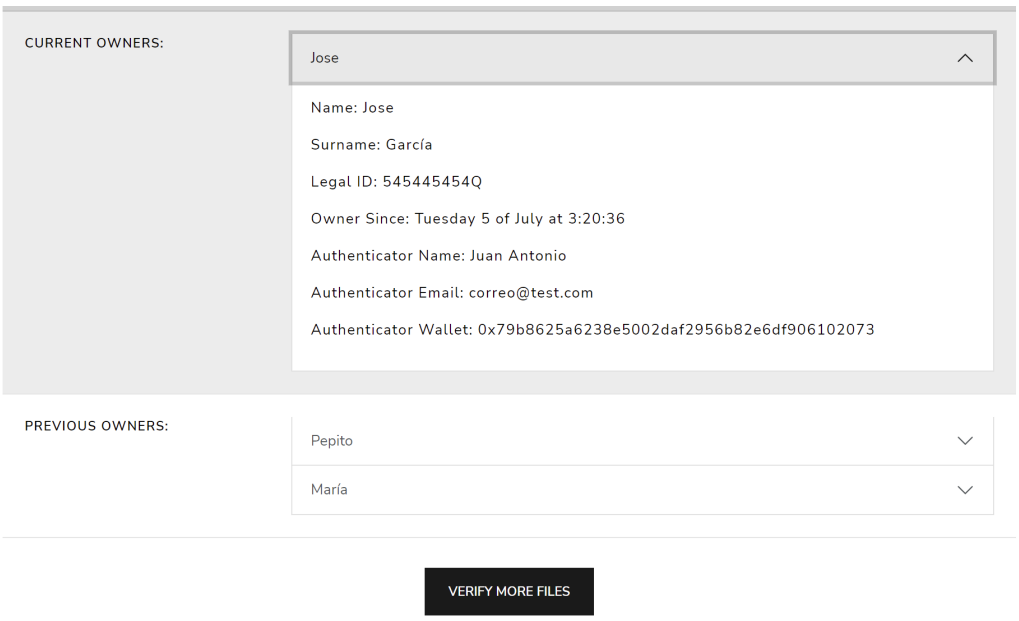
En la misma vista de verificación, se puede desplegar información de cada dueño, como se aprecia en la *Figura 24* en la vista se puede obtener la misma informacion que si se llamará al contrato inteligente. Pero desde la web es menos complejo y más amigable de cara al usuario final.

En caso de que el archivo no se encuentre en la cadena de bloques se comunica al usuario, que no se ha podido encontrar. Esto como se ha explicado previamente, puede deberse o bien porque el archivo nunca a estado en la cadena de bloques o bien a sido modificado. Cabe destacar, que cambiar el nombre de un archivo no cambia su contenido, por lo que el hash seguirá siendo el mismo, por ello es por lo que se decidió almacenar en la cadena de bloques el nombre del fichero.



Fuente: Elaboración propia

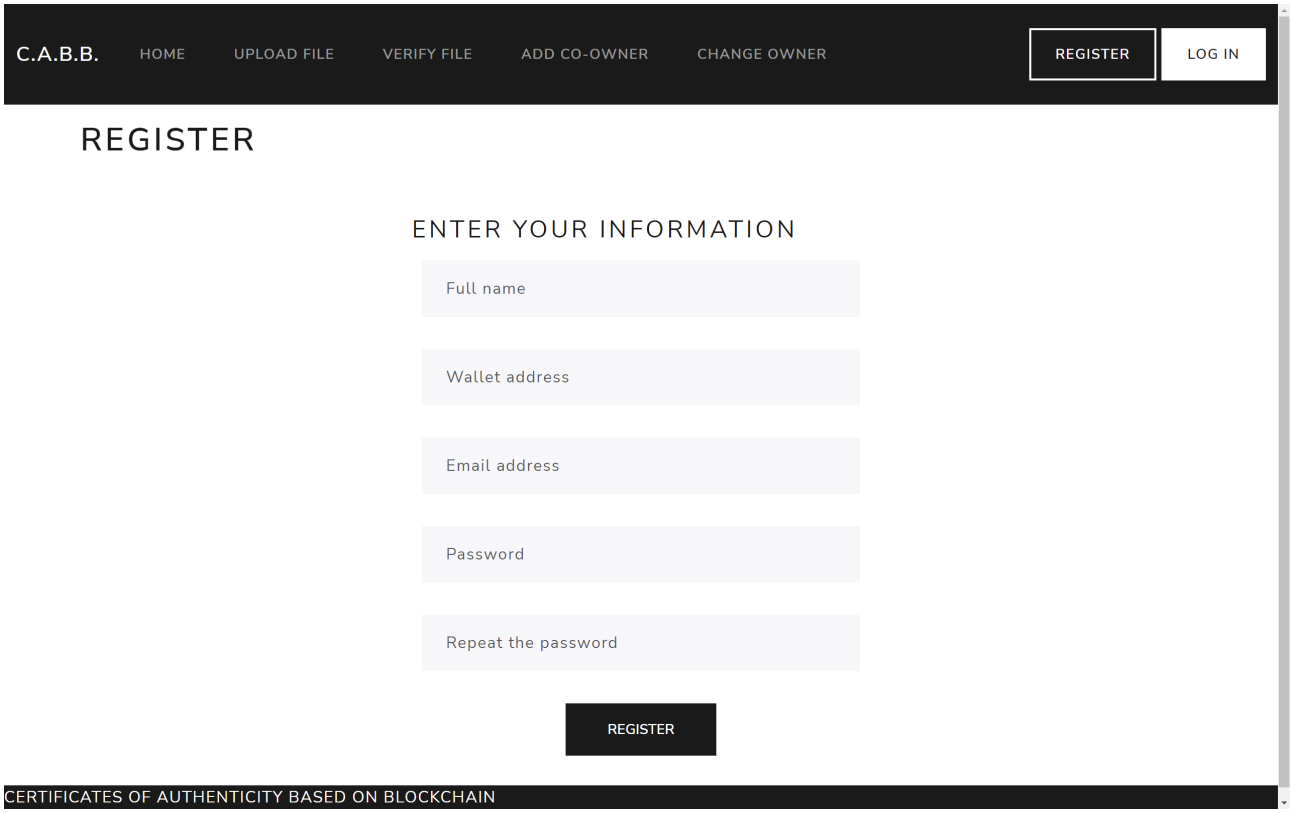
Figura 23: Resultados de una verificación de un archivo en la web



Fuente: Elaboración propia

Figura 24: Resultados de una verificación con detalle en el dueño actual

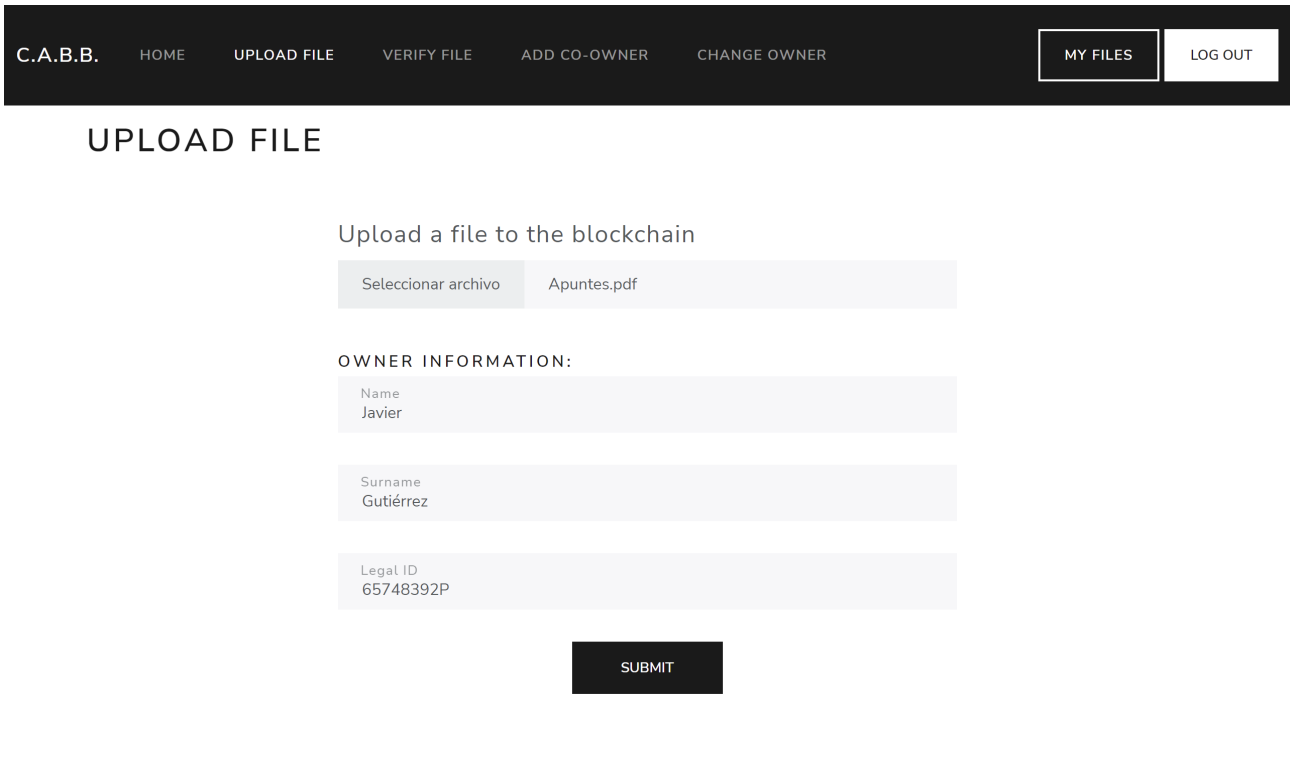
Por último, un usuario invitado, podrá registrase como autentificador y así poder hacer uso del resto de la funcionalidades del sistema.



Fuente: Elaboración propia
Figura 25: Pantalla de registro en el sistema desde la web

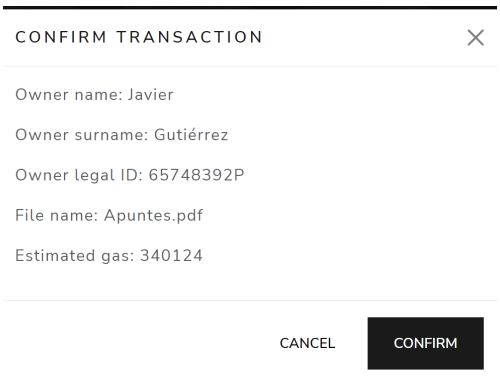
Cuando el usuario introduce todos los datos que se le piden (*Figura 25*) este podrá iniciar sesión. Al iniciar sesión los botones de Registrarse y de iniciar sesión, se cambiarán automáticamente por un botón para cerrar sesión y otro botón de ver todos los archivos almacenados en la base de datos, puesto que si se desea verificar con al cadena de bloques se debe usar la vista explicada antes de verificación de archivos.

Cuando el usuario ha iniciado sesión este puede acceder a la vista para añadir un nuevo archivo a la cadena de bloques. Para ello debe seleccionar el archivo e introducir la informacion del dueño (*Figura 26*). Inicialmente solo se puede poner un dueño, pero según sea subido, se le podría añadir la información de un co-dueño.



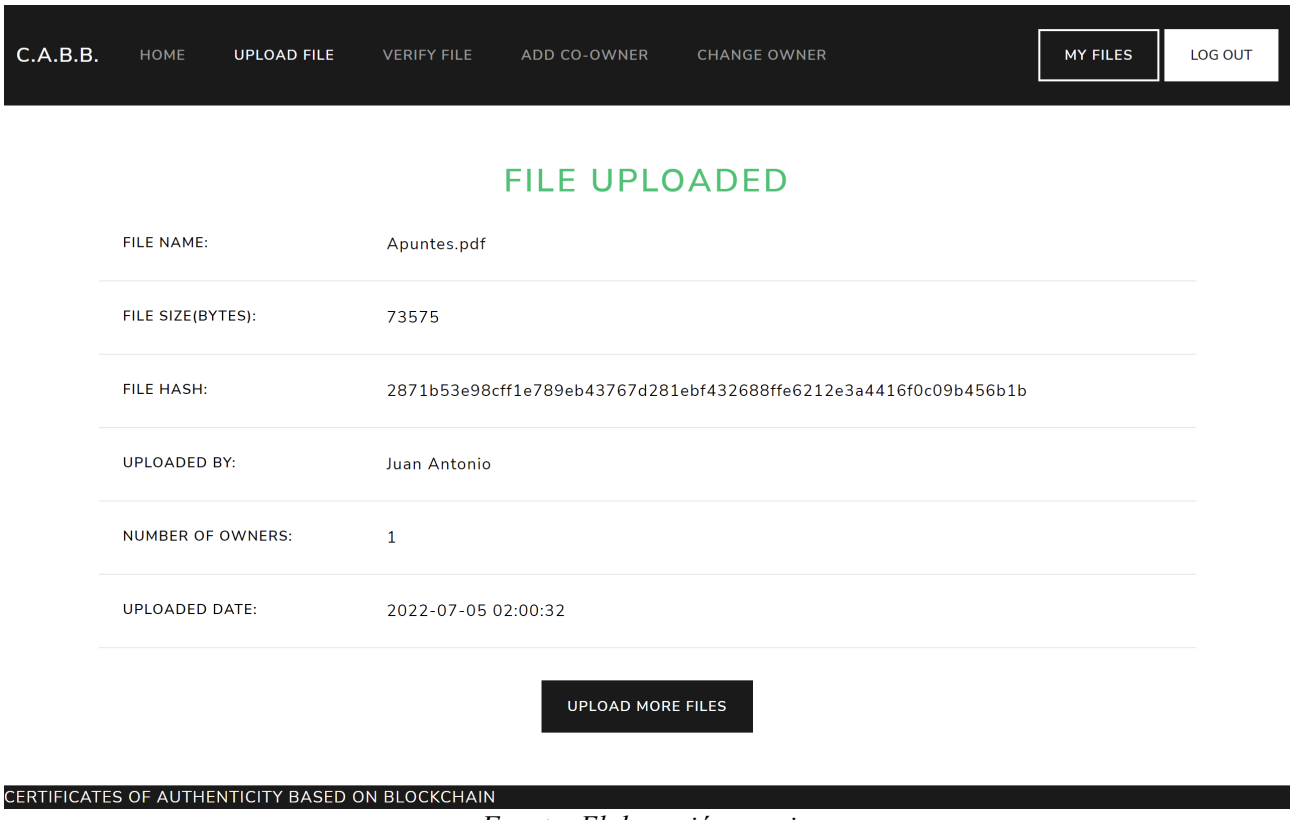
Fuente: Elaboración propia
Figura 26: Vista de subida de un nuevo archivo a la cadena de bloques

Trás haber introducido de manera correcta la informacion del nuevo archivo, se realiza una petición al servidor y este al contrato inteligente, para que estime cuanto se va a tener que pagar de tasas (gas). Esta información, con la introducida por el usuario, es mostrada (*Figura 27*), para que verifique si esta de acuerdo con todo, puesto que hasta ahora no se ha realizado ningún cobro a la cartera del usuario.



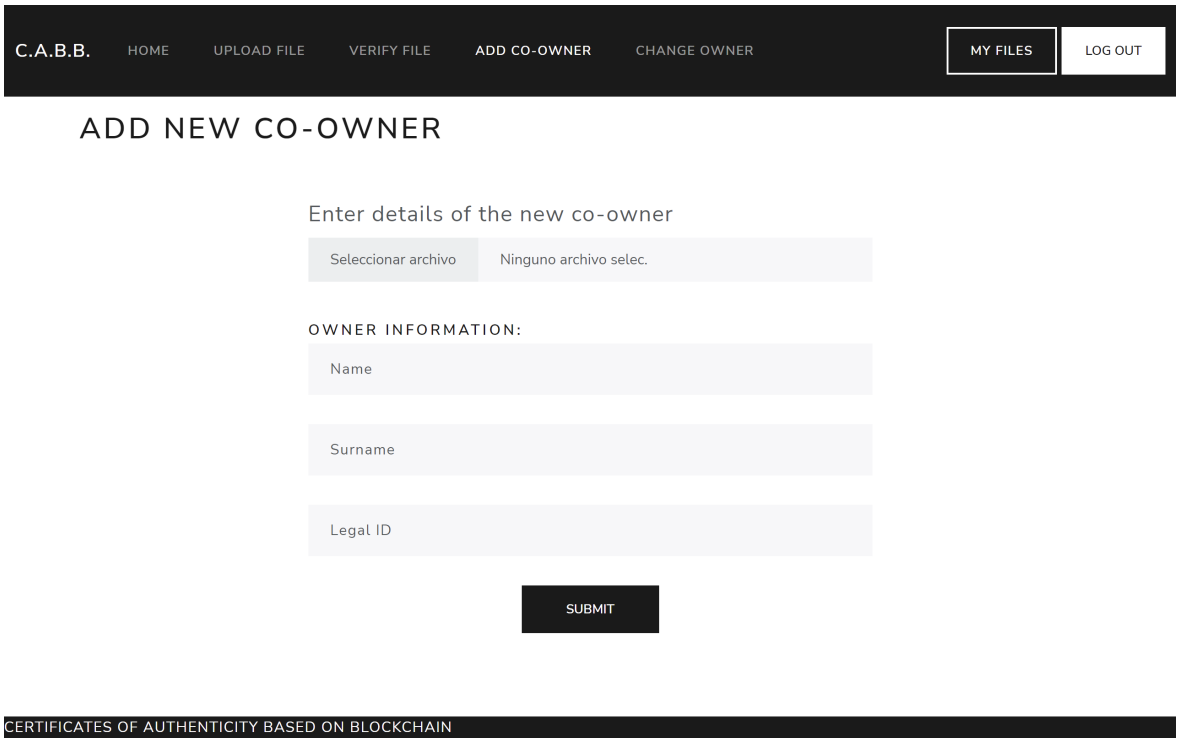
Fuente: Elaboración propia
Figura 27: Modal de confirmación de la subida de un fichero

Cuando el usuario acepta la ejecución de la transacción, se manda la información al servidor y luego al contrato inteligente. Cuando el contrato inteligente termina la ejecución, se verifica el estado de la transacción, si esta a sido satisfactoria, se redirige al usuario a la una vista resumen del archivo subido (*Figura 28*). En esta vista, a diferencia de en la de verificación, las fechas se muestran en UTC, en la vista de verificación se muestran las horas de España.



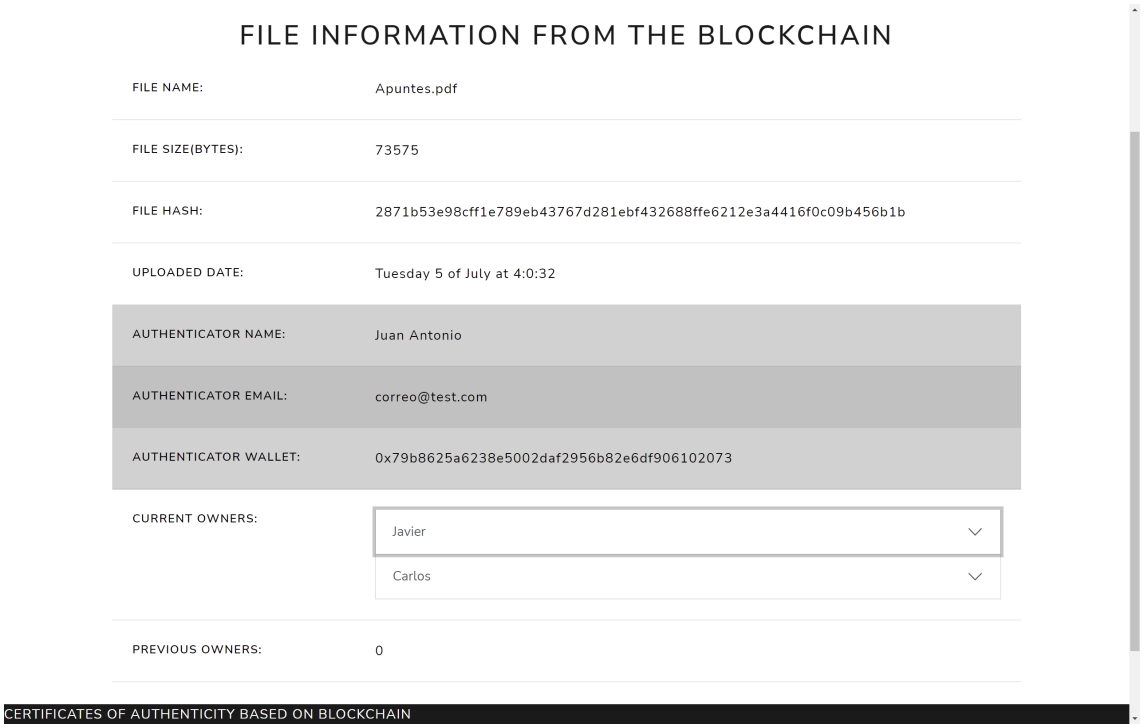
Fuente: Elaboración propia
Figura 28: Vista de archivo subido a la cadena de bloques satisfactoriamente

Tanto la vista de añadir un co-dueño, como la de cambiar de dueño por uno nuevo, visualmente son idénticas, salvo los títulos, el resto es igual (Figura 29). También el comportamiento es prácticamente idéntico, solo se diferencian por el *endpoint* al que llaman. Si añadimos un nuevo co-dueño al archivo de *Apuntes.pdf*. Podemos ver ambos dueños en la *Figura 30*. Si ahora hacemos un cambio de dicho dueño, podemos ver (), como Carlos y Javier ahora están como dueños anteriores, y Jaime está como dueño único de dicho archivo. Esta situación se daría cuando se realiza una venta de una obra, quedando así un registro viable de propietarios con fechas. Esto facilita al autentificador en caso de tener que ser re-autenticada.



Fuente: Elaboración propia

Figura 29: Vista de añadir un co-dueño a un archivo



Fuente: Elaboración propia

Figura 30: Vista de verificación del archivo Apuntes.pdf

Por último, está la vista desde un usuario puede ver los archivos que ha subido (*Figura 31*). Esta vista obtiene su informacion de la base de datos. Porque como se explicó anteriormente, en vistas más informativas que funcionales, se decidió obtener la información de la base de datos para mayor velocidad en las respuestas.

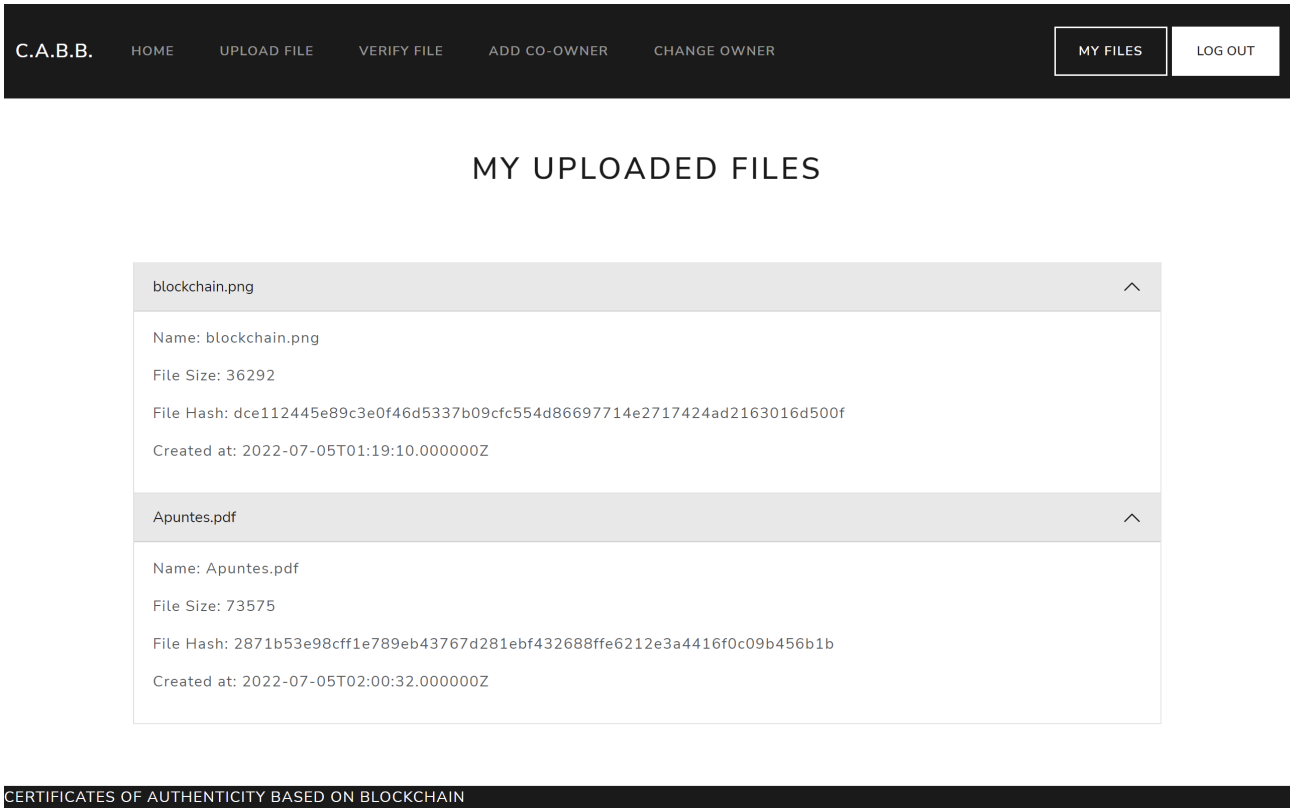


Figura 31: Vista de los archivos de un usuario

Como se ha podido ver, todas las funcionales diseñadas, se han podido implementar de manera completamente funcional, aunque como se ha explicado, se tuvieron que realizar una pequeñas variaciones con respecto al diseño. Pero este cambio a seguido bajo los requisitos fijados por las historias de usuario.

Capítulo 7 Conclusiones y trabajos Futuros

En la realización de este Trabajo Fin de Grado, se han cumplido con los objetivos fijados en *4.2 Objetivos*.

- El primer objetivo consistía en desarrollar un contrato inteligente que permitiera subir y verificar información de un archivo a la blockchain. Este objetivo se ha conseguido, como se ha podido ver en el apartado *5.3.1* de la *Implementación*. Como se extra se añadió la funcionalidad de almacenar el registro de los dueños actuales y pasados.

- Como siguiente objetivo que se fijó determinaba que se debía desarrollar un servidor REST API, que facilitara el acceso al contrato inteligente al mayor número de usuarios posibles. Este objetivo se ha cumplido al poder a todos los métodos públicos del contrato inteligente desde la misma API. Cabe destacar que hay más *endpoints* que enriquecen la API y con unos tiempos de respuesta menores, al sacar la información de una base de datos en vez de blockchain.

- El tercer objetivo consistía en el desarrollo de una web que interactuara con el servidor REST API. Esto como se ha podido ver se ha conseguido, puesto que no hay funcionalidad que la web no te permita realizar y que la API si. Además se ha diseñado una web simple e intuitiva.

- Por último, se fijo un objetivo para realizar pruebas en una red simulada para determinar su correcto funcionamiento y su viabilidad. Esto se ha ido realizando en las distintas etapas del desarrollo. Cuando se implementaba una gran nueva funcionalidad, se hacían diversas pruebas y distintas personas, escuchando su *feedback*. Analizando el desarrollo realizado y hablando con las personas que lo han probado se ha determinado que este sistema seria viable en el mercado, pero harían falta algunos cambios, que se salían del ámbito de este Trabajo Fin de Grado.

Como se puede apreciar, la tecnología de blockchain abre un gran abanico al desarrollo de nuevas tecnologías y sistemas. Y cada año que pasa, se implementa esta tecnología en nuevos campos, incluso llevando a cambios de paradigma en algunas industria.

En el ámbito de las certificaciones de autenticidad, aún no ha llegado a expandirse el uso de blockchain, como se explicó en el *Estado de la Cuestión* no hay muchas empresas que este persiguiendo esta industria tan grande y con un movimiento de capital enorme, en 2021 se vendieron 65 mil millones de dólares de obras. de arte.

Pero como se ha mencionado en el cuarto objetivo, si se quisiera sacar al mercado el sistema desarrollado en este Trabajo Fin de Grado, habría que hacer algunos cambios, adaptaciones y

mejoras al sistema, a parte de tener que formar a la gente. El mayor problema del blockchain es la desinformación de la población, al solo asociar blockchain a las criptomonedas y no conocer los beneficios y ventajas que aportan blockchain 2.0, 3.0 y 4.0.

Con respecto al sistema desarrollado, si se quisiera seguir desarrollando lo para mejorarlo o para introducirlo al mercado, habría que realizar los siguientes cambios, mejoras o adiciones:

- Integrar la plataforma con carteras conocidas de criptomonedas para poder tener un mayor control sobre el pago de las tasas. De esta forma se daría mayor confianza y seguridad a los usuarios del sistema.

- Habría que estudiar el despliegue y utilización de otras redes blockchain, puesto que Ethereum, es una de las más caras en cuanto a tasas y no es de las más rápidas. Por eso se cree que analizar la migración a otras redes blockchain sería una cuestión a analizar en el futuro desarrollo del sistema.

- Habría que pulir un poco los flujos de inicio de sesión, para que se alineen mejor con los estándares de OAuth 2.0[55].

- Se tendría añadir el sistema de comisiones para que así el sistema se pueda llevar una comisión al ser usado y de esta forma, poder recibir ingresos para mantener dicho sistema, y poder ofrecer soporte técnico cuando se requiera.

Blockchain como tecnología, tiene muchísimas posibilidades y ofrece algunas ventajas nunca antes vistas. El problema que tiene, es que aún no es lo suficientemente maduro para que las grandes lo implementes, de ahí blockchain 4.0. Cuando blockchain para la industria madure, podría suponer un cambió de paradigma en muchas industrias.

55 https://oauth.net/2/

Bibliografía

1: Satoshi Nakamoto (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf

2: Vitalik Buterin (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*. https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf

3: Unibrigth (7 de diciembre de 2017). *Blockchain evolution: from 1.0 to 4.0*. https://unibrightio.medium.com/blockchain-evolution-from-1-0-to-4-0-3fbdbccfc666

4: iFour team (11 de abril de 2019). *Blockchain history and evolution*. https://www.ifourtechnolab.com/blog/blockchain-history-and-evolution

5: Fundación Wikimedia (23 de febrero de 2021). *Certificado de autenticidad*. https://es.wikipedia.org/wiki/Certificado_de_autenticidad

6: Ashlyn Gentile (5 de diciembre de 2017). *Cómo hacer un certificado de autenticidad*. https://www.agora-gallery.com/advice/blog/2017/12/05/como-hacer-un-certificado-de-autenticidad/?lang=es

7: Certificado de Autenticidad (s.f.). *Certificados de Autenticidad de Obras de Arte y cuadros de artistas*. https://certificadoautenticidad.com/certificado-de-autenticidad-de-obras-de-arte

8: Lo entiendo (13 de octubre de 2021). *Cómo y dónde se pueden compulsar documentos*. https://loentiendo.com/compulsar-documentos-como-y-donde-hacerlo/

9: Ministerio de la Presidencia (16 de mayo de 2006). *Orden ITC/1475/2006, de 11 de mayo, sobre utilización del procedimiento electrónico para la compulsa de documentos en el ámbito del Ministerio de Industria, Turismo y Comercio*. https://www.boe.es/eli/es/o/2006/05/11/itc1475

10: IBM (s.f.). *¿Qué es la tecnología de blockchain?*. https://www.ibm.com/es-es/topics/what-is-blockchain

11: Griffin Ichiba Hotchkiss y Sam Richards (19 de septiembre de 2020). *ETHEREUM VIRTUAL MACHINE (EVM)*. https://ethereum.org/en/developers/docs/evm/

12: Varios autores (15 febrero 2022). *SMART CONTRACT LANGUAGES*. https://ethereum.org/en/developers/docs/smart-contracts/languages

13: W3Techs (2022). *Usage statistics of PHP for websites*. https://w3techs.com/technologies/details/pl-php

14: W3Schools (s.f.). *PHP Introduction*. https://www.w3schools.com/PHP/php_intro.asp

15: Varios autores (26 de noviembre de 2021). *PHP*. https://www.php.net/

16: Laravel LLC. (s.f.). *Laravel Documentation*. https://laravel.com/docs/9.x

17: Igor Sysoev (s.f.). *Nginx*. https://nginx.org/en/

18: Jorge Alberto Medina Rosas (2009). *Funciones Hash criptográficas*. Facultad de Ciencias Universidad Nacional Autónoma de México

19: Instituto Nacional de Tecnologías de la Comunicación (2014). *BITCOIN: Una moneda criptográfica*. INCIBE-CERT

20: Dylan Yaga, Peter Mell, Nik Roby y Karen Scarfone (2018). *Blockchain Technology Overview*. National Institute of Standars and Technology

21: Fábrica Nacional de Moneda y Timbre (s.f.). *Criptografía de clave asimétrica*. https://www.cert.fnmt.es/curso-de-criptografia/criptografia-de-clave-asimetrica

22: Norbert Bodziony, Paweł Jemioło, Krzysztof Kluza y Marek R. Ogiela (2021). *Blockchain-Based Address Alias System*. Journal of Theoretical and Applied Electronic Commerce Research

23: BBVA (s.f.). *Registro contable*. https://www.bbva.mx/educacion-financiera/r/registro_contable.html

24: Mitchell Clark (9 de septiembre de 2021). *BLOCKCHAIN, EXPLAINED*. https://www.theverge.com/22654785/blockchain-explained-cryptocurrency-what-is-stake-nft

25: Simply Explained (2017). *How does a blockchain work*. https://www.youtube.com/watch?v=SSo_EIwHSd4

26: Marianna Belotti, Nikola Božić, Guy Pujolle y Stefano Secci (2019). *A Vademecum on Blockchain Technologies: When, Which, and How*. IEEE Communications Surveys & Tutorials

27: Yang Xiao, Ning Zhang, Wenjing Lou y Y. Thomas Hou (2020). *A Survey of Distributed Consensus Protocols for Blockchain Networks*. IEEE Communications Surveys & Tutorials

28: Shijie Zhang y Jong-Hyouk Lee (2020). *Analysis of the main consensus protocols of blockchain*. ICT Express

29: Abdul Wahab y Waqas Mehmood (2018). *Survey of Consensus Protocols*. arXiv

30: Mehrdad Salimitari y Mainak Chatterjee (2018). *An Overview of Blockchain and Consensus Protocols for IoT Networks*. arXiv preprint arXiv:1809.05613

31: Siamak Solat, Philippe Calvez y Farid Naït-Abdesselam (2020). *Permissioned vs. Permissionless Blockchain: How and Why There Is Only One Right Choice*. Journal of Software

32: Perito Judicial (s.f.). *Expertización de obras de Arte. La importancia del certificado de autenticidad*. https://peritojudicial.com/expertizacion-obras-arte/

33: Jefatura del Estado (12 de noviembre de 2021). *Ley 6/2020, de 11 de noviembre, reguladora de determinados aspectos de los servicios electrónicos de confianza*. https://www.boe.es/eli/es/l/2020/11/11/6/con

34: Mitchell Clark (18 de agosto de 2018). *NFTs, explained*. https://www.theverge.com/22310188/nft-explainer-what-is-blockchain-crypto-art-faq

35: BlockTac (s.f). *Acerca de BlockTac*. https://www.blocktac.com/acerca-de/

ANEXO I: Alineación del proyecto con los ODS

Los Objetivos de Desarrollo Sostenible (ODS), también conocidos como los objetivos mundiales, fueron aprobados por las Naciones Unidas en 2015. Son un llamamiento para terminar con la pobreza, proteger el planeta y garantizar la paz y la prosperidad en 2030. Los ODS son 17 los cuales se pueden ver en la *Figura 32*.

Fuente: Cortesía de Naciones Unidas[56]
Figura 32: Objetivos de desarrollo sotenible de las Naciones Unidas

A lo largo del desarrollo del este Trabajo Fin de Grado, se han seguido el Objetivo 9.

- **Objetivo 9: Industria, Innovación e Infraestructura**: Este proyecto esta implementado nuevas tecnologías como lo es blockchain a una industria con pocos avances en los últimos años. Este desarrollo a buscado facilitar y hacer accesible una plataforma que permita ser usada sin poseer conocimientos específicos de blockchain. A su vez al ser un contrato inteligente que se ejecuta en una red descentralizada, no requiere una única gran infraestructura, si no pequeñas y repartidas geográficamente. Esta infraestructura, cabe destacar que es de uso publico, por lo que es compartida por lo que cuando no este siendo

56 https://www.un.org/sustainabledevelopment/es/news/communications-material/

usada por este sistema, podría estar siendo usada por otro, optimizando por tanto los recursos.

Pero debido a las tecnologías usadas, se han perseguido algunos objetivos de manera indirecta, tales como el 1 y el 8. Estos objetivos debido al funcionamiento de blockchain, se están persiguiendo. Como se ha explicado en este Trabajo Fin de Grado, blockchain se basa en protocolos de consensos, estos necesitan de nodos que validen las transacciones y por hacerlo, se llevan una recompensa económica. Haciendo que cualquier persona que tenga un dispositivo electrónico con conexión a internet, pueda ingresarlo a la red blockchain cuando no lo esta usando y conseguir algunos ingresos de forma pasiva. Dando le un uso extra a esos dispositivos que ya posee la gente y que no usan en todo momento.

Aunque de primeras al ser un desarrollo de software, puede parecer que no se alinea con ningún Objetivo de Desarrollo Sostenible, si se analizan detalladamente es posible ver como ese desarrollo puede ser beneficioso para la sociedad y como se alinea con los ODS fijados por Naciones Unidas.

ANEXO II: Manual de instalación

En este anexo se detallarán las aplicaciones necesarias como instalar y como configurar el sistema tanto como para el entorno de desarrollo como en el entorno de producción.

Para poder ejecutar el código y levantar la aplicación se requiere tener instalados los siguientes programas:

- Ganache[57] (Solo necesario para el entorno de producción)
- Remix IDE (Puede ser instalado en Visual Studio Code[58], usado desde la web[59] o instalado como aplicación[60])
- Docker[61]

A. II 1 GANACHE

En este apartado se explicará la configuración necesaria de Ganache para que funcione de manera correcta con el software desarrollo.

1. Abrir Ganache y crear un nuevo entorno de trabajo con la configuración de la *Figura 33*.

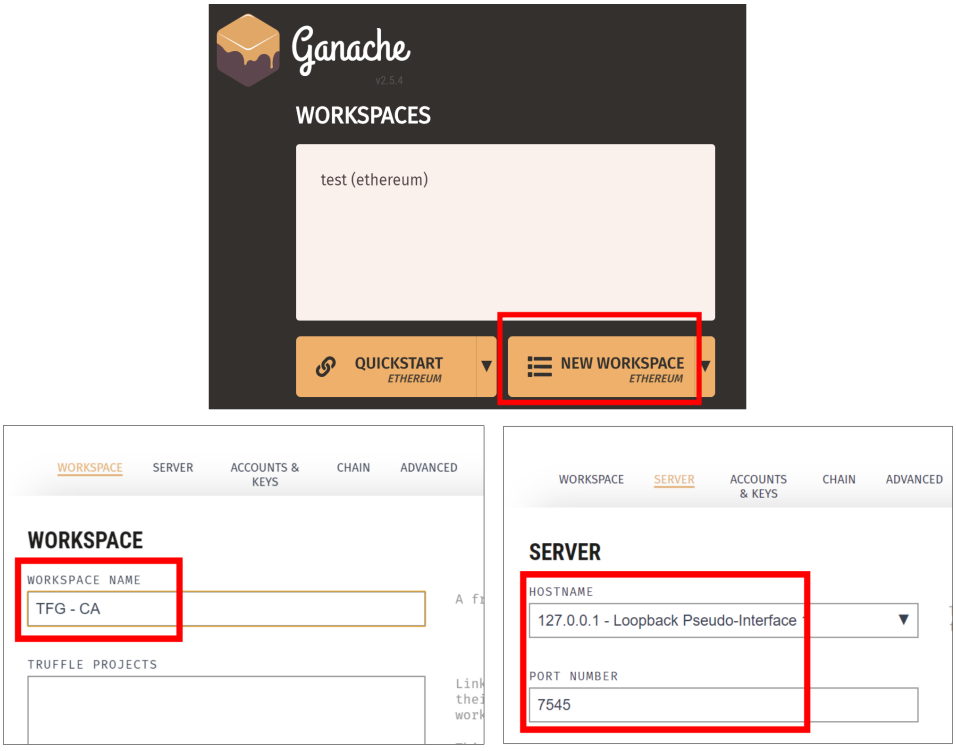


Figura 33: Paso 1 de la configuración de Ganache

57 https://trufflesuite.com/ganache/
58 https://marketplace.visualstudio.com/items?itemName=RemixProject.ethereum-remix
59 https://remix.ethereum.org/
60 https://remix-project.org/
61 https://www.docker.com/products/docker-desktop/

2. Guardamos el entorno de trabajo y se nos abrirá automáticamente. No aparecerá la vista de la *Figura 34*, donde podremos ver todas las cuentas y su balance.

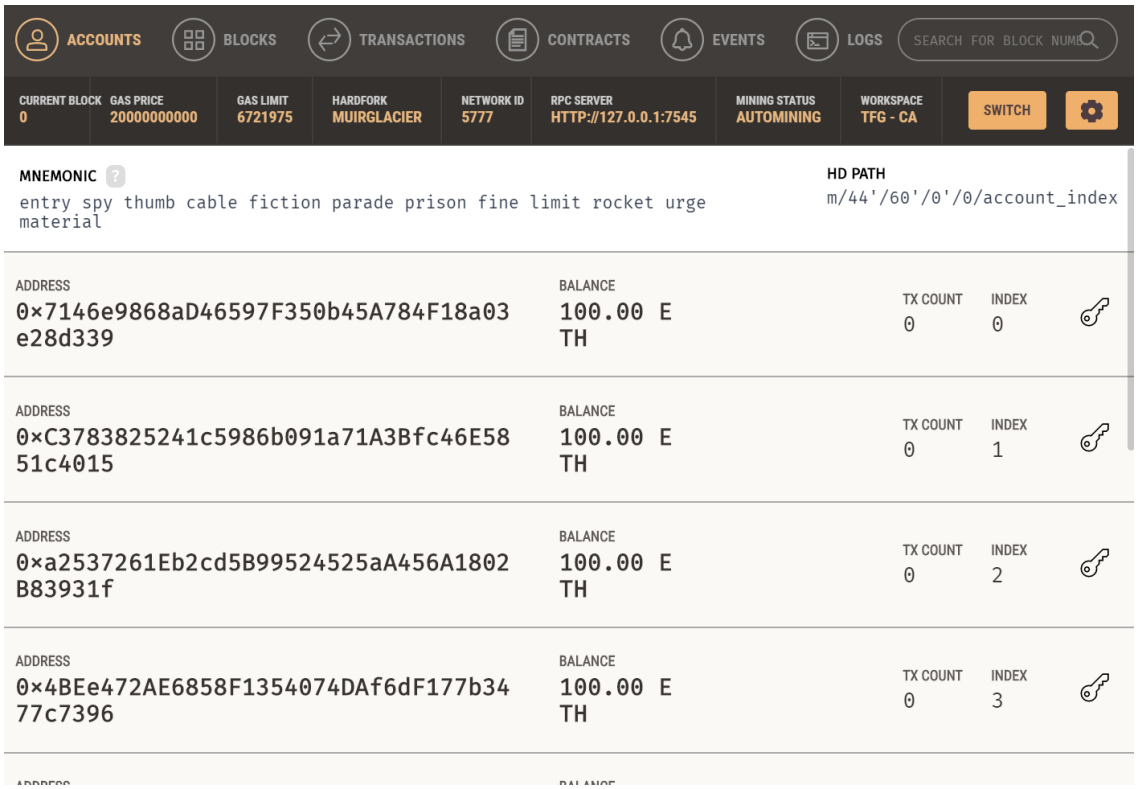


Figura 34: Paso 2 de la configuración de Ganache

Habiendo seguido estos pasos, Ganache ya estaría delegado en nuestra maquina y por tanto una red Ethereum para poder realizar pruebas.

A. II 2 REMIX IDE

En este apartado se explicará como desplegar el contrato inteligente en al red de Ethereum que se ha creado con Ganache. Se usará Remix en Visual Studio Code (VSC).

1. Abrimos la extensión de Remix en VSC, siguiendo los pasos mostrados en la *Figura 35*

2. Se nos abrirá una pestaña para que nos conectemos a la red de blockchain, debemos poner la dirección y el puerto que pusimos en Ganache. Si se ha seguido esta guía, se debe poner como se muestra en la *Figura 36*

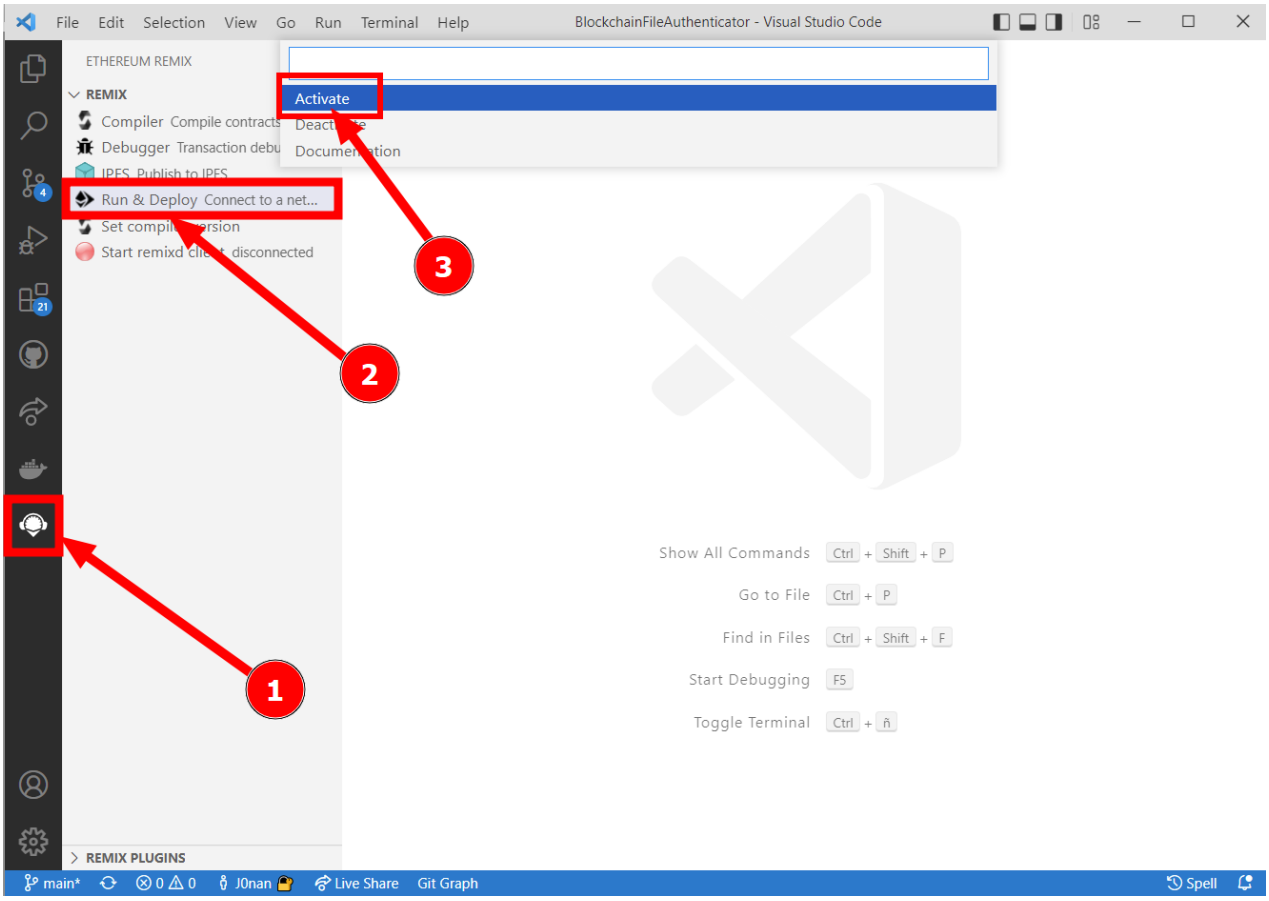


Figura 35: Abrir la extensión de Ganache

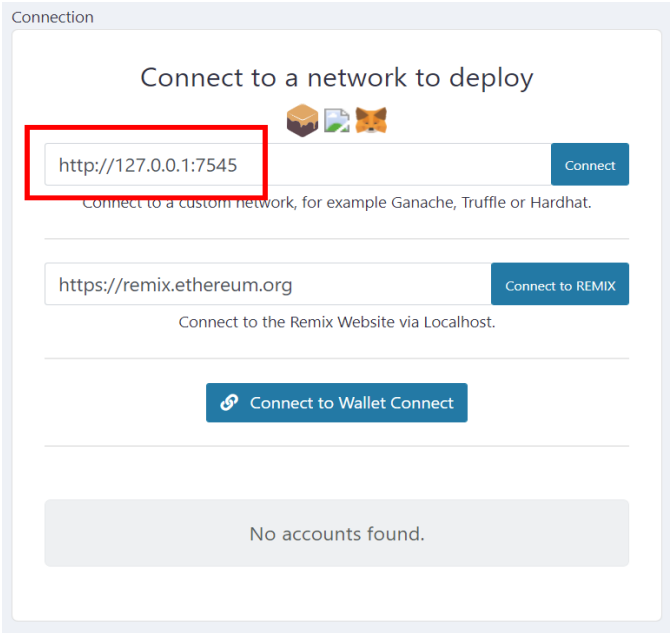


Figura 36: Conectar Remix a la red de Ethereum

3. Para compilar el contrato, debemos seleccionar lo como se aprecia en la *Figura 37*, y luego darle a *Compile*. En algunas situaciones el desplegable aparece vacío, si abrimos en contrato en una nueva pestaña, Remix ya lo reconocerá y nos aparecerá.

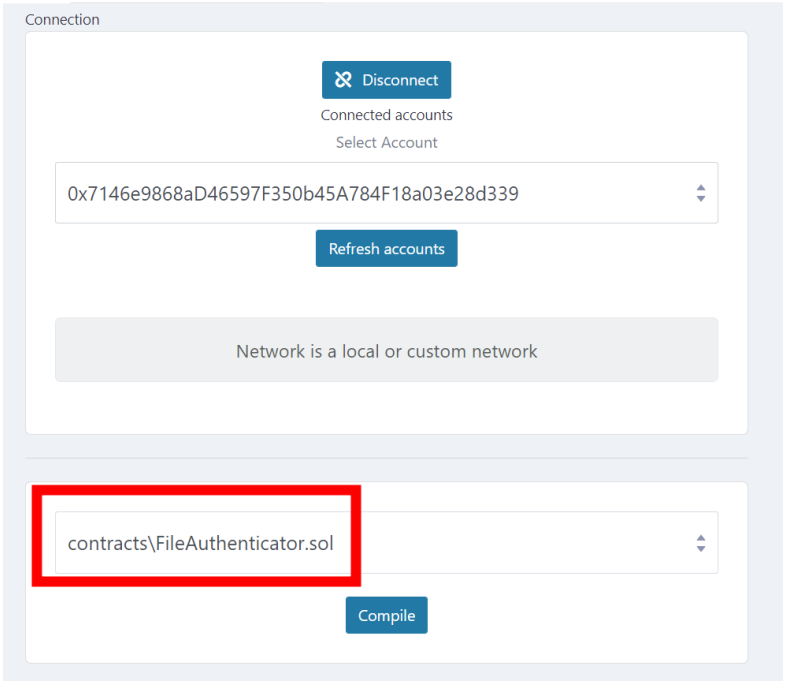


Figura 37: Compilar el contrato inteligente

4. Ahora vamos a desplegar el contrato, para ello, seleccionamos el contrato y le damos al botón de *Deploy*, como se muestra en la *Figura 38*

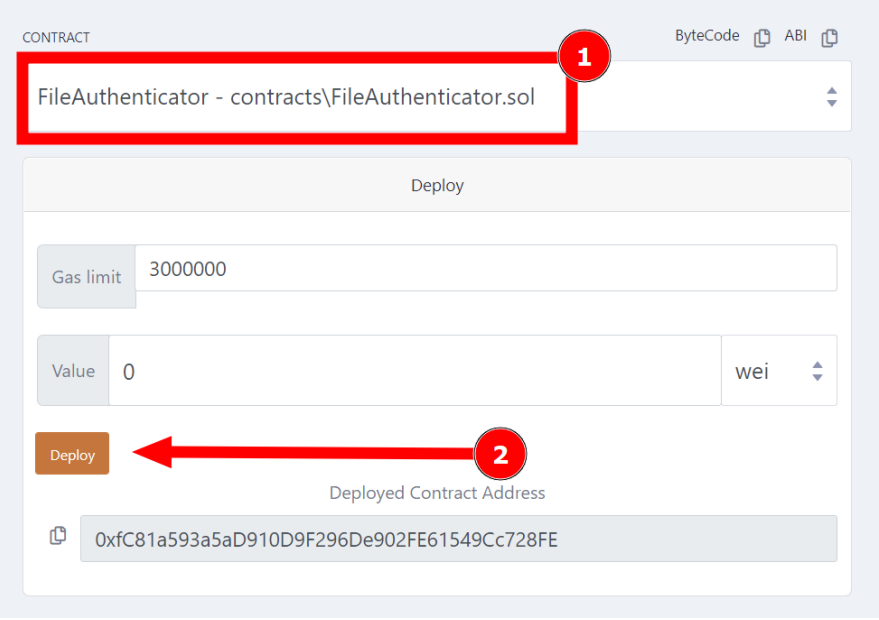


Figura 38: Desplegar en contrato inteligente en la red de Ethereum

Si se han seguido estos cuatro pasos sin que el contrato tuviera ningún error, el contrato se debería haber desplegado de manera satisfactoria en la red que hemos creado con Ganache.

Antes de pasar a explicar Docker, debemos guardarnos la dirección donde se ha desplegado el contrato y el ABI, en la *Figura 39* se muestra con un 1 la dirección del contrato en la blockchain y con un 2 el botón para poder copiar el código ABI.

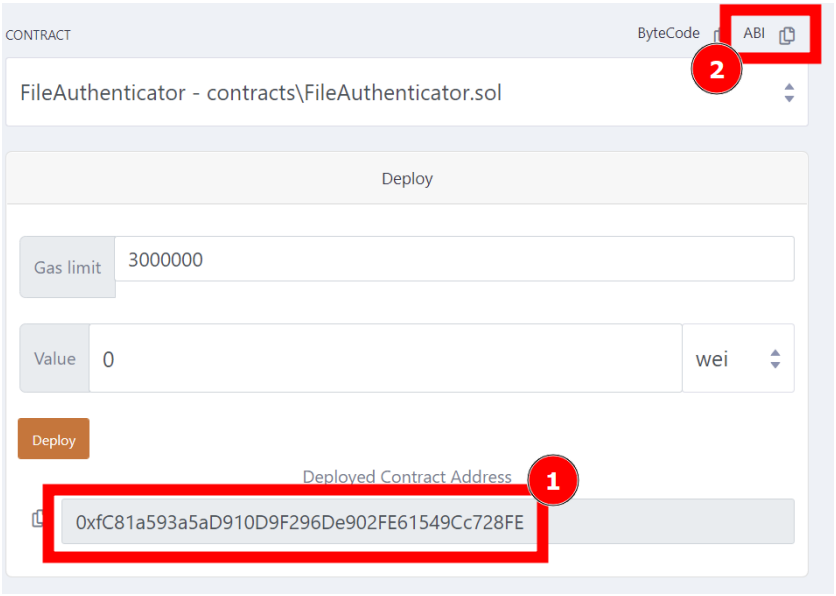


Figura 39: Donde obtener la dirección y el ABI del contrato desplegado en la red de blockchain

A. II 3 DOCKER

En este apartado se explicará como configurar y desplegar la aplicación con Docker.

1. Para fijar los credenciales de la base de datos, debemos acceder al archivo `docker-compose.yml` que se haya en la raíz del proyecto. En el debemos modificar al gusto las variables marcadas en la *Figura 40*.

```
21  mysql:
22    image: mysql:8.0.28
23    container_name: mysql
24    # restart: unless-stopped
25    ports:
26      - "3306:3306"
27    volumes:
28      - ./mysql:/var/lib/mysql
29    environment:
30      MYSQL_DATABASE: tfg
31      MYSQL_USER: mysql
32      MYSQL_PASSWORD: secret
33      MYSQL_ROOT_PASSWORD: secret
34    networks:
35      - laravel
```

Figura 40: Varibles a modificar de MySQL

2. Creamos una copia del archivo `.env.example` y renombramos el nombre del archivo a `.env` el cual se haya en la raíz del proyecto.

3. Rellenamos las siguientes variables de la base de datos, en caso de haber puesto la misma configuración que en las figuras, se debe poner lo mismo:

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=tfg
DB_USERNAME=mysql
DB_PASSWORD=secret
```

4. A continuación se debe rellenar las variables del contrato, para ello en la variable con nombre `CONTRACT_ADDRESS` se debe poner la dirección del contrato, en la variable `ABI`, el valor ABI en una única linea y rodeado de comillas simples.

5. Poner la dirección de la red blockchain, en el caso de tener el despliegue de Docker y Ganache en la misma maquina, habría que definir la variable así:

```
ETH_NETWORK="http://host.docker.internal:7545"
```

6. Acceder desde el terminal a la ruta del proyecto y crear y levantar los contenedores con el comando: `docker-compose up -build`

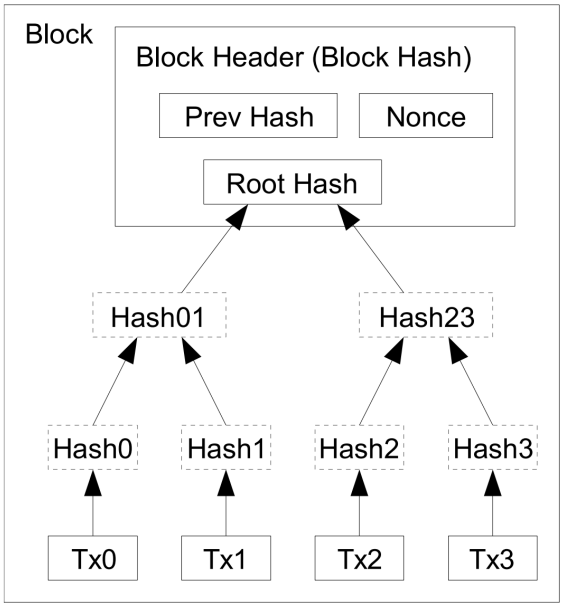
7. Una vez levantados los contenedores, acceder a la tabla `oath_clients` y obtener la información para rellenar las variables del archivo `.env`:

- `PASSPORT_PERSONAL_ACCESS_CLIENT_ID`
- `PASSPORT_PERSONAL_ACCESS_CLIENT_SECRET`
- `PASSPORT_PASSWORD_GRANT_CLIENT_ID`
- `PASSPORT_PASSWORD_GRANT_CLIENT_SECRET`

Habiendo ejecutado todos estos pasos, ya podríamos acceder a la web a través de la dirección: http://localhost:8080/

ANEXO III: Arboles Merkle

Los arboles Merkle son arboles binarios de hashes. Un árbol binario es aquel que un nodo padre como máximo posee dos hijos.



Transactions Hashed in a Merkle Tree
Fuente: Cortesía de Satoshi Nakamoto[1]
Figura 41: Árbol Merkle

En la *Figura 41* se puede apreciar su funcionamiento. Las distintas transacciones dentro de un bloque son hasheadas, luego se calcula el hash poniendo como entrada la combinación de los hashes de dos hojas. Y este procedimiento se repite hasta tener un único hash. Como se puede ver en la *Figura 41*, *Hash0* y *Hash1* son los hashes de *Tx0* y *Tx1* respectivamente. Luego se calcula *Hash01* que usa como entrada *Hash0* y *Hash1*, luego *RootHash* es calculado realizando el mismo concepto, poniendo como entrada *Hash01* y *Hash23*.