UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

Curso 2021/2022

TRABAJO FIN DE GRADO

"Descentralización de almacenaje de archivos en la nube mediante el uso de la tecnología Blockchain"

DIRECTOR: Vicente García Díaz

AUTOR: Sergio Vega Pineda

Agradecimientos

En primer lugar, me gustaría agradecer a mi tutor que me ha estado ayudando y dando consejos hasta el final sobre la elaboración de este proyecto.

Por otro lado, agradecer a mi amiga Patricia y a mis amigos de la universidad Laura, Miguel y Pablo por todo el apoyo mostrado durante este curso 2021-2022, sobre todo en la etapa final de exámenes y elaboración del Trabajo Fin de Grado.

Por último, tengo que mencionar a mi familia y sobre todo mi perro Pancho, que me estuvo acompañando día sí y día también durante la elaboración de todo el proyecto.

Resumen

Este proyecto tiene dos objetivos principales. Por un lado, consiste en demostrar una forma de almacenaje en la nube diferente, segura y privada respecto a las que se pueden encontrar actualmente en el mercado. Se trata de un sistema que usa tecnologías innovadoras y novedosas con mucho recorrido por delante, pero con una barrera de entrada elevada. Por otro lado, tiene el propósito de implementar e integrar la tecnología Blockchain dentro de la plataforma Neodoc, un gestor de documentos elaborado por la empresa Neosystems donde el autor de este documento está realizando las prácticas. Dentro de esta empresa, se elaborará un proyecto de I+D para añadir una capa de seguridad y privacidad a los contenidos custodiados y gestionados por la plataforma.

Concretamente, se hará uso de la tecnología Blockchain combinada con una red distribuida de almacenaje con el fin de conseguir un sistema descentralizado. Para llevar a cabo este proyecto, será necesaria una larga investigación sobre dichas tecnologías para tener una mayor compresión sobre el tema y llevar a cabo el desarrollo de la idea.

El sistema por desarrollar estará integrado dentro de una aplicación web para que cualquier usuario pueda hacer uso de este de manera individual. Aun así, este sistema también ofrecerá las herramientas necesarias a cualquier tercero, como puede ser el ya mencionado Neodoc, para integrar la plataforma dentro de su producto o servicio.

Como se puede observar, se van a hacer uso de varios sistemas que deben trabajar conjuntamente para llevar a cabo el objetivo del proyecto. Por lo tanto, cada uno es independiente del resto siguiendo una arquitecta denominada microservicios, entre los que se encuentran:

- La red distribuida de almacenaje en la nube.
- La red Blockchain.
- La aplicación web para la interacción con los usuarios.

En definitiva, se construirá un proyecto compuesto por diferentes servicios que se comunicarán entre sí, llevando a cabo cada uno una tarea concreta. En consecuencia, los usuarios podrán encontrar una nueva vía por la que descentralizar el almacenaje de sus archivos en la nube, consiguiendo así más seguridad y privacidad.

Palabras Clave

Blockchain, Ethereum, IPFS, servicio web, descentralización, almacenamiento en la nube, archivos.

Abstract

This project has two main objectives. On the one hand, it consists of demonstrating a different way of cloud storage more secure and private than the currently ones on the market. It is a system that uses innovative and novel technologies with a long way to go, but with a high entry barrier. On the other hand, it has the purpose of implementing and integrating Blockchain technology within the Neodoc platform, a document manager created by the Neosystems company where the author of this document will carry out his internship. Within this company, an R&D project will be developed to add a layer of security and privacy to the content guarded and managed by the platform.

Specifically, Blockchain technology combined with a distributed storage network will be used in order to achieve a decentralized system. To complete this project, long research on these technologies will be necessary to have a greater understanding of the subject and carry out the development of the idea.

The system to be developed will be integrated into a web application so that any user can use it individually. Even so, this system will also offer the necessary tools to any third party, such as the aforementioned Neodoc, to integrate the platform into their product or service.

As can be seen, several systems will be used that must work together to carry out the objective of the project. Therefore, each one is independent from the rest following an architecture called microservices, among which are:

- The distributed cloud storage network.
- The Blockchain network.
- The web application for interaction with users.

To conclude, the project will be built made up of different services that will communicate with each other, each one carrying out a specific task. Consequently, users will be able to find a new way to decentralize the storage of their files in the cloud, thus achieving more security and privacy.

Keywords

Blockchain, Ethereum, IPFS, web service, decentralization, cloud storage, files.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1. Memoria del Proyecto

En términos generales, este proyecto constituye el estudio e implementación de nuevas tecnologías, como Blockchain y redes distribuidas de almacenamiento, dentro de la empresa Neosystems [1] de Gijón. En este capítulo se introducirán los aspectos más importantes del proyecto.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Neosystems es una empresa localizada en Gijón especializada en la realización de software a medida y prestar los servicios de su producto principal Neodoc [2]. Este producto se trata de un gestor de contenido empresarial creado y comercializado por la misma compañía. El desarrollo de este sistema ha sido financiado por la propia empresa como un proyecto de I+D con la finalidad de integrar la tecnología Blockchain dentro de este producto.

El uso de plataformas de almacenaje en la nube está cada vez extendido, ya sea en el entorno empresarial, estudiantil o personal. Forman parte del día a día y la privacidad y seguridad de nuestros documentos son una preocupación común de los usuarios.

En cuanto a la seguridad, nadie quiere que sus documentos sean accedidos por terceros desconocidos a los que no se les han concedido los permisos necesarios para ello. Además, los sistemas de almacenaje habituales, como podrían ser los gestores documentales, están fuertemente centralizados, permitiendo así ataques a gran escala a un solo punto y derivando en caídas en el servicio y filtraciones de información.

Por otro lado, las normas de privacidad de cada plataforma pueden afectar al almacenaje de los documentos, llegando incluso a permitir a la propia plataforma a acceder a tus documentos para restringir ciertos tipos de archivos que no permiten dentro de esta.

Este proyecto consistirá en construir un sistema en el que los usuarios son realmente los dueños de su contenido y no la propia plataforma donde son subidos. De esta manera, los usuarios obtendrán una aplicación web con el que podrán subir archivos e interactuar con sus documentos, los cuales estarán almacenados de una forma descentralizada, segura y privada.

Los motivos principales por los que se decidió realizar este proyecto son ofrecer una aplicación web que asegure la privacidad y seguridad de los archivos subidos, actuando como un gestor documental, y la posibilidad de adentrarse dentro de la tecnología Blockchain, una tecnología muy novedosa que está en pleno desarrollo y muy en auge durante los últimos años. Otro punto a favor es que dicha tecnología no está presente dentro del marco educativo de la Escuela por el momento, por lo que es una buena oportunidad para estudiar sobre ella, ya que es una tecnología puntera y muy útil para un futuro cercano.

1.2 Resumen de Todos los Aspectos

En esta sección, se resumirá el contenido de cada apartado que forma el documento de forma breve.

- **Capítulo 1. Memoria del Proyecto:** Resumen de la motivación, objetivos y alcance del proyecto.

- **Capítulo 2. Introducción:** Justificación y objetivos del proyecto. Estudio de la situación actual.

- **Capítulo 3. Aspectos Teóricos:** Aspectos teóricos de las tecnologías utilizadas.

- **Capítulo 4. Planificación inicial del Proyecto y Presupuestos:** Planificación y presupuesto inicial del proyecto.

- **Capítulo 5. Análisis:** Estudio de los requisitos, elaboración de casos de uso y escenarios, diseño del diagrama preliminar del análisis, diseño inicial de interfaces y especificación del plan de pruebas.

- **Capítulo 6. Diseño:** Diseño de diagramas del sistema y arquitectura del sistema. Especificación del plan de pruebas.

- **Capítulo 7. Implementación del Sistema:** Herramientas, lenguajes, estándares y normas utilizados. Problemas encontrados durante el desarrollo.

- **Capítulo 8. Desarrollo de las Pruebas:** Desarrollo y resultado de las pruebas diseñadas.

- **Capítulo 9. Manuales del Sistema:** Manuales de instalación, de ejecución, de usuario y del programador.

- **Capítulo 10. Conclusiones y Ampliaciones:** Conclusiones del proyecto y ampliaciones futuras.

- **Capítulo 11. Planificación final del Proyecto y Presupuestos:** Planificación y presupuesto final del proyecto.

- **Capítulo 12. Referencias Bibliográficas.**

- **Capítulo 13. Apéndices:** Glosario y diccionario de datos, índice alfabético y contenido del archivo adjuntado.

Capítulo 2. Introducción

En la presente sección, se expondrán los motivos principales de la creación de este proyecto, sus objetivos y se realizará un estudio de la situación actual de las tecnologías usadas en el proyecto, así como de otras ya existentes.

2.1 Justificación del Proyecto

Durante los últimos años, la tecnología Blockchain ha ocupado portadas de periódicos digitales, reportajes en telediarios y videos por internet. La causa de principal de su fama es su relación directa con las criptomonedas, ya que es la tecnología principal detrás de su funcionamiento. Dentro del mercado financiero, las criptomonedas han dado un paso adelante y se han extendido hasta formar parte de los portafolios de la mayoría de los inversionistas.

Aun así, la tecnología Blockchain va mucho más allá que solamente dedicarse al mundo financiero, y es que, a través de su desarrollo y mejor entendimiento en el mundo de la informática, se están resolviendo y cubriendo nuevos problemas y casos de uso como en el entorno de cadena de suministros o la seguridad de certificados y activos.

De tal manera que, a pesar de no haberse cursado en ningún momento dentro del marco académico de la universidad, realizar esta labor de investigación y desarrollo sobre esta reciente tecnología es una muy buena oportunidad para comprender y encontrar nuevas formas de resolver problemas del día a día y especializando así el perfil profesional.

La idea principal de este proyecto es la de integrar dentro del gestor de contenido empresarial Neodoc, de la empresa Neosystems, un sistema de descentralización de archivos. Esta descentralización permitirá salvaguardar de forma más segura los archivos de los usuarios, asegurando la privacidad e integridad de estos y ofreciendo una mayor disponibilidad respecto a un modelo centralizado. Además de esta integración dentro de Neodoc, se implementará además una aplicación web para demostrar el funcionamiento del sistema.

El sistema estará compuesto por diversas partes bien diferenciadas: por un lado, estaría la interfaz de usuario, con la que el usuario podrá interactuar fácilmente con el sistema; por otro lado, hará uso de la plataforma de descentralización de archivos InterPlanetary File System [3] o Sistema de Archivos Interplanetario, la cual se combinará con una red Blockchain famosa ya existente denominada Ethereum. Además, se ofrecerá la posibilidad de que cualquier gestor documental integre este sistema dentro de su modelo, como Neodoc, dando pie a posibles ampliaciones.

2.2 Objetivos del Proyecto

El objetivo principal claramente es conseguir que una serie de sistemas se comuniquen y trabajen conjuntamente para llevar a cabo un sistema de descentralización de archivos en la nube.

Para llevar a cabo el desarrollo, será necesario completar los siguientes subobjetivos:

1. Desarrollo de la integración del sistema con la red Blockchain.
2. Desarrollo de la integración del sistema con la red distribuida IPFS.
3. Desarrollo de la aplicación web para los usuarios.
4. Desarrollo de las herramientas que permitan la integración del sistema en productos de terceros.
5. Integrar el sistema dentro del producto Neodoc de la empresa Neosystems.

2.3 Estudio de la Situación Actual

Esta sección tiene como objetivo dar una visión general del estado actual de la tecnología Blockchain y de las diversas formas de almacenamiento en la nube.

2.3.1 Tecnología Blockchain

Hoy en día, la tecnología de cadena de bloques está en pleno auge y desarrollo, pisando fuerte sobre todo en el mercado financiero. En cuanto al ámbito software, es todavía un mundo lleno de posibilidades con mucho por explorar y descubrir, claramente visible por la demanda de desarrolladores enfocados a este campo.

Brevemente, se profundizará en los siguientes apartados sobre las principales redes Blockchain existentes: Bitcoin y Ethereum.

2.3.1.1 Bitcoin

Bitcoin es la red Blockchain más famosa y grande existente hasta la fecha. Tiene su origen en 2009, fundando así los orígenes de la tecnología Blockchain. El autor, por ahora desconocido bajo el seudónimo de Satoshi Nakamoto, tenía como objetivo emular el pago en efectivo que se realiza en el día a día entre personas, pero de manera digital. Para ello, creó la criptomoneda Bitcoin, la cual no requiere de ningún ente central, ya sea un banco o una institución mediadora, para el intercambio de dinero [4].

A lo largo de los años, esta criptomoneda ha aumentado su fama debido a su cotización en el mercado, llegando a alcanzar los 61 mil dólares por bitcoin en octubre de 2021. Un dato que impacta bastante si nos remontamos a sus inicios, en 2010, donde la moneda valía unos céntimos. En el año 2013 presenció un repunte hasta los más de mil dólares, manteniéndose por debajo de los mil dólares durante los próximos años. En 2017, la moneda empezó su escalada hasta cifras de 17 mil dólares, fluctuando posteriormente entre los 4 mil y 10 mil

dólares [5]. No sería hasta el año 2021 cuando el Bitcoin se volviera un activo en la boca de todos junto a la tecnología que lo respalda, surgiendo una infinidad de nuevos proyectos relacionados, noticias online y una multitud de personas adentrándose en este mercado.

Este aumento en su cotización ha sido empujado principalmente por noticias positivas en cuanto a la tecnología Blockchain y las posibilidades de esta. En cuanto a las más destacadas, tenemos que el Estado de El Salvador, país de centro América, ha adoptado en 2021 el Bitcoin como moneda de curso legal, permitiendo así usar Bitcoin como moneda para realizar cualquier compra dentro del país [6]. Por otro lado, el interés de las grandes compañías como Facebook con su creación del metaverso denominado Meta [7] y el uso de criptomonedas dentro de este, ha conseguido que gente nueva se interese y apueste por Bitcoin.

A pesar de la vuelta de la criptomoneda a valores entorno a los 20 mil dólares [5], este activo sigue considerándose como un refugio de valor contra estado inflacionistas, una forma válida de intercambio de valor o una forma de inversión de dinero, sea cual sea el capital aportado.

Muchos son los economistas, estados, empresas e instituciones que apuestan por esta nueva forma de dinero digital, así como su posible legalización dentro de los países [8]. Aun así, no son pocos sus retractores, que la tachan de estafa defendiendo que vale 0 y es una pérdida de dinero [9].

De todas maneras, guste o no, es un activo que ya se ha adentrado en nuestro día pero que todavía tiene mucho por recorrer. Y justo, en estos momentos, es cuando reluce si realmente será algo de futuro o se estancará terminando en el olvido e ignorado.

2.3.1.2 Ethereum

Ethereum es la red Blockchain más famosa después de la tratada anteriormente, Bitcoin. Fue desarrollado por el ruso Vitálik Buterin durante el 2014, saliendo a la luz en el 2015 por primera vez. Junto al lanzamiento de la red pública Ethereum, se lanzó con una criptomoneda asociada, denominada Ether, que sigue las mismas bases que Bitcoin [10].

En su nacimiento, la criptomoneda cotizaba a unos céntimos en el mercado, con un repunte en 2017 por encima de los mil dólares y con una caída posterior fluctuando entorno a los 100-300 dólares hasta el año 2021, que, junto a Bitcoin, presenció un aumento exponencial hacia su pico más alto de 4867 dólares. Hoy en día, ha vuelto a cotizar a valores máximos que tuvo en 2017, entorno a los mil dólares, produciéndose así una gran corrección en su precio desde su precio más alto de 2021 [11].

La principal ventaja de esta red Blockchain respecto a Bitcoin, es que introduce la posibilidad de ejecutar código dentro de la misma cadena de bloques. Este código ejecutable dentro de la red es denominado como *SmartContract* o contrato inteligente, que como su propio nombre indica, se trata de un contrato con la particularidad de que no necesite de un ente central o intermediario para ejecutar una acción. De esta manera, los desarrolladores pueden crear aplicación descentralizadas utilizando esta tecnología.

Así como Bitcoin, Ethereum ha presenciado un gran crecimiento durante los dos últimos años. El empuje principal a la fama de esta criptomoneda ha sido la irrupción de los *NFT* o *Non Fungible Tokens*, que permiten crear activos digitales de forma única. Miles de celebridades,

equipos de fútbol y empresas han lanzados sus propios proyectos *NFT* para comercializar obras de arte, piezas únicas coleccionables o contenido exclusivo [12]. Incluso, han surgido un nuevo tipo de videojuego, denominados, *Play-to-Earn*, donde se puede conseguir criptomonedas a través de avanzar y ganar dentro del juego.

Aun así, estos nuevos términos como *NFT* o los juegos *Play-to-Earn* tienen sus retractores, que defienden que no tienen valor alguno, son una estafa y que no resuelven ningún problema real [13].

2.3.2 Almacenamiento en la nube

El almacenamiento en la nube es un sistema claramente adentrado en el día a día de la población mundial. A pesar de estar muy extendido, estos mecanismos siempre son dependiente de grandes o no tan grandes empresas que custodian la información subida por los usuarios, llegando en algunos casos a restringir según qué tipo de contenido o violar la privacidad de los usuarios.

Por lo tanto, se profundizará también sobre los mecanismos de almacenamiento en la nube actuales: Dropbox y Google Drive.

2.3.2.1 Dropbox

Dropbox es un servicio de almacenamiento en la nube multiplataforma que permite almacenar archivos de varios tipos y compartirlos con otros usuarios. Fue lanzado oficialmente en septiembre de 2008 y cuenta con más de 500 millones de usuarios registrados [14].

Actualmente, Dropbox tiene diferentes planes de suscripción para utilizar sus servicios. Presenta una modalidad gratis con un espacio libre limitado de 2GB, una cuenta *plus* por 9.99€ al mes con 2TB de espacio y una cuenta *pro* por 25.99€ al mes con 3TB de espacio [15].

En cuanto a la seguridad, Dropbox mantiene que todo el almacenamiento está cifrado en la nube y defiende que los mismos trabajadores no pueden acceder a ningún contenido. Nada más lejos de la realidad, ya se han dado casos demostrados en el pasado en los que los empleados de la propia plataforma tienen acceso a los documentos almacenados [16].

En cuanto a las políticas de privacidad, estas son diferentes en cada país, sobre todo de aquellos fuera de Unión Europea, por lo que los servidores albergados en estos países pueden no necesariamente respetar la privacidad del contenido. Por otro lado, Dropbox depende de infraestructuras y proveedores *cloud* de terceros, como pueden ser AWS (Amazon Web Services) y Salesforce, por lo que lo que gran parte de los datos dependen de organizaciones externas. Por último, Dropbox designa una serie de “empresa de confianza”, como Google y Amazon, con las que comparte los datos “para dar un servicio más óptimo” [17], [18].

Como se puede observar, Dropbox es una plataforma interesante y en principio segura pero que depende de muchos terceros para ofrecer el servicio, los datos están en constante movimiento y la privacidad de los mismo es cuestionable.

2.3.2.2 Google Drive

Google Drive es un servicio de alojamiento de archivos multiplataforma en la nube creado por la compañía Google. Fue lanzada en abril de 2012 y ya cuenta con más de 800 millones de usuarios en el mundo entero [19].

Actualmente, presenta diversos planes de suscripción más uno gratuito con 15GB de almacenamiento. En cuanto a los de pago, está el plan *Business starter* de 4,68€ al mes con 30GB de almacenamiento, el plan *Business Standar* con 2TB de espacio y el plan *Business plus* con 5TB de espacio. Cabe decir que con la suscripción también se incluyen otros servicios como el de email, videollamadas y otras aplicaciones de ofimática [20].

Al igual que Dropbox, Google Drive encripta todos los archivos y datos del usuario y protege todas las conexiones para evitar intercepciones. Aun así, es la misma Google quien alberga los algoritmos de encriptación y desencriptación, por lo que podría desencriptar cualquier archivo subido.

En cuanto a la privacidad, Google Drive analiza todos los archivos subidos dentro de una organización para detectar software malicioso o phishing. Pero esto va mucho más allá cuando Google puede bloquear el contenido que subes si cree que infringe los "términos de servicio" que impone la misma empresa. Por lo tanto, esto demuestra que la empresa analiza y accede al contenido de los archivos subidos. En la mayoría de los casos eliminaría los documentos, pero podría llegar a banear la cuenta y restringir su uso totalmente [21].

Como se puede observar, a pesar de ser una plataforma muy útil y segura con una infinidad de servicios a parte de los de almacenamiento, sus políticas de privacidad son bastantes cuestionables, llegando incluso a eliminar o restringir el contenido que se sube a la plataforma.

2.4 Evaluación de alternativas

A continuación, se mostrarán y discutirán las principales tecnologías consideradas para el desarrollo del proyecto, sus ventajas e inconvenientes. Concretamente, las tecnologías seleccionadas para realizar el trabajo son JavaScript como lenguaje de programación con los framework Nodejs, Express y Web3js, IPFS como red distribuida de almacenaje, Ethereum como red Blockchain junto con su lenguaje de programación asociado Solidity y MongoDB como base de datos.

2.4.1 Lenguaje de programación

A continuación, se discuten los posibles lenguajes de programación para desarrollar el proyecto.

2.4.1.1 Java

Se trata de un lenguaje de programación orientado a objetos de propósito general lanzado en 1995 por Sun Microsystems. Sigue siendo hoy en día uno de los lenguajes de programación más demandados a pesar del surgimiento de otros competidores como Python o JavaScript.

2.4.1.1.1 Ventajas

La principal ventaja de Java es que cualquier programa escrito en este lenguaje de programación podrá ser ejecutado en cualquier máquina que disponga de la máquina virtual de Java (JVM). Es decir, se programa una vez y puede ejecutarse en múltiples plataformas.

En cuanto al marco educativo del Grado de la Escuela, el lenguaje de programación Java es un pilar básico, estando presente en las asignaturas más importantes como Algoritmia o Estructuras de datos. Además, presenta un número amplio de librerías y frameworks como Spring Boot para desarrollo web o Hibernate para relacionarse con bases de datos.

2.4.1.1.2 Inconvenientes

La principal desventaja de este lenguaje es que no se adapta bien a la tecnología Blockchain. Esto se debe principalmente a la escasez de documentación y frameworks relativos a la cadena de bloques existentes. Por otro lado, lenguajes como JavaScript presentan una inmensa cantidad de herramientas con el propósito buscado, además de una gran documentación asociada.

2.4.1.2 JavaScript

Se trata de un lenguaje de programación interpretado, de alto nivel, orientado a objetos, débilmente tipado y dinámico de propósito general lanzado en 1995 por Netscape Communications. Su principal virtud es que es utilizado en la mayoría de los navegadores web.

2.4.1.2.1 Ventajas

La principal ventaja de este lenguaje son los frameworks existentes para realizar desarrollo web como Nodejs y Expressjs, que agilizan esta parte del sistema. Por otro lado, durante los últimos años se ha extendido el término web 3.0 como una idea de un nuevo tipo de web más evolucionado, albergando así todas las tecnologías Blockchain, tanto para desarrolladores como usuarios, dentro de este término. Por consiguiente, se desarrolló un framework explícitamente para interaccionar con la tecnología Blockchain utilizando JavaScript, denominado Web3js.

Además, otra tecnología principal del proyecto, IPFS, solamente ofrece sus herramientas de desarrollo para dos lenguajes de programación: Go y JavaScript.

2.4.1.2.2 Inconvenientes

Entre sus desventajas, se puede encontrar que el autor de este proyecto no está muy familiarizado con el lenguaje de programación. En cuanto al lenguaje de programación en sí, se

encuentra que el programa puede ser interpretado de forma distinta según el navegador en el que se ejecute.

2.4.1.3 Python

Se trata de un lenguaje de programación de alto nivel, interpretado, dinámico y multiplataforma que soporta tanto programación funcional como orientada a objetos. Fue lanzado en 1991 por Guido van Rossum.

2.4.1.3.1 Ventajas

Una característica muy destacable de este lenguaje de programación es su facilidad de aprendizaje al ser de alto nivel. Presenta una gran variedad de frameworks muy potentes, sobre todo para Inteligencia Artificial y desarrollo web, lo que agiliza y facilita el desarrollo de programas.

2.4.1.3.2 Inconvenientes

La principal desventaja de Python es el rendimiento. Esto se debe a la misma naturaleza del lenguaje, al ser tan dinámico y versátil hace que su velocidad de ejecución sea menor que en otros lenguajes.

2.4.2 Sistema distribuido de almacenaje

A continuación, se discuten los posibles sistemas distribuidos de almacenaje para desarrollar el proyecto.

2.4.2.1 IPFS

InterPlanetary File System o Sistema de Archivos Interplanetario se trata de un sistema de archivos descentralizado diseñado con el protocol P2P (*Peer-to-Peer*). Creado por Juan Benet en 2015 con la finalidad de crear una red de computadores global que permita almacenar archivos.

2.4.2.1.1 Ventajas

Principalmente garantiza la seguridad y privacidad de los archivos almacenados en el sistema. Además, permite una alta escalabilidad.

2.4.2.1.2 Inconvenientes

Es todavía una tecnología joven en pleno desarrollo. Consume mucha banda ancha de red además de que su uso no está muy extendido.

2.4.2.2 Apache Cassandra

Se trata de un software de gestión de bases de datos NoSQL caracterizado por su naturaleza distribuida lanzado en 2008 por la Apache Software Foundation.

2.4.2.2.1 Ventajas

Permite un gran volumen de datos de una manera distribuida, consiguiendo así cierta descentralización. Además, permite una buena escalabilidad.

2.4.2.2.2 Inconvenientes

Los archivos deben limitarse a no más de 10MB para evitar fallos o errores en el almacenaje, lo que limita mucho su uso.

2.4.3 Red Blockchain

A continuación, se discuten las posibles redes Blockchain para desarrollar el proyecto.

2.4.3.1 Ethereum

Es una red Blockchain de código abierto que permite la ejecución de programas dentro de la misma red a través de contratos inteligentes. Fue lanzada en 2015 por el ruso Vitálik Buterin.

2.4.3.1.1 Ventajas

Es la red Blockchain, quitando Bitcoin, más famosa que existe, por lo que existe una gran cantidad de frameworks y documentación asociada para el desarrollo software. Presenta su propio lenguaje de programación denominado Solidity, para la creación de contratos inteligentes. Tiene una criptomoneda asociada denominada Ether para promover el uso de la red.

2.4.3.1.2 Inconvenientes

La red puede congestionarse ya que permite alrededor de 30 transacciones por segundo. Además, las comisiones por cada transacción son realmente altas comparadas con otras redes Blockchain.

2.4.3.2 Solana

Es una red Blockchain que permite la ejecución de programas dentro de la misma red a través de contratos inteligentes. Fue creado en 2017 por el ruso llamado Anatoly Yakovenko.

2.4.3.2.1 Ventajas

Permite realizar 50.000 transacciones por segundo y las comisiones de estas son muy bajas respectos a otras como puede ser Ethereum. Tiene una criptomoneda asociada denominada Solana para promover el uso de la red. Presenta una documentación excelente para comenzar a desarrollar.

2.4.3.2.2 Inconvenientes

No es una red Blockchain tan famosa como puede ser Ethereum y aunque la documentación asociada es elevada, no existe una gran comunidad de desarrolladores como en Ethereum.

2.4.4 Base de datos

A continuación, se discuten las posibles bases de datos para desarrollar el proyecto.

2.4.4.1 MongoDB

Es un sistema de base de datos NoSQL, no relacional, lanzada en 2009 por la empresa MongoDB inc.

2.4.4.1.1 Ventajas

Permite gran escalabilidad y es más rápida que una base de datos relacional.

2.4.4.1.2 Inconvenientes

Falta de estándares definidos y poca portabilidad.

2.4.4.2 MySQL

Es un sistema de gestión de bases de datos SQL, relacional, lanzado en 1994 por la empresa Oracle Corporation.

2.4.4.2.1 Ventajas

Presenta una gran trayectoria y es ampliamente usado en el mundo del desarrollo software. Tiene una gran portabilidad y unos estándares bien definidos.

2.4.4.2.2 Inconvenientes

El problema principal es su escasa escalabilidad, debido a que el crecimiento en volumen de los datos en almacenamiento implica un alto coste de mantenimiento.

Capítulo 3. Aspectos Teóricos

En esta sección se profundizará en la definición de las tecnologías que serán utilizadas, sus conceptos básicos y el uso que se le dará dentro del proyecto en cuestión.

3.1 Red Blockchain

La tecnología Blockchain es una parte fundamental en este proyecto ya que constituye uno de los servicios que conforman el proyecto. Por lo tanto, es necesario el correcto entendimiento de esta tecnología de una forma general y su integración de esta dentro del sistema para los lectores.

En líneas generales, la Blockchain no es más que una estructura de datos, y como su nombre indica, se trata de una cadena (*chain*) de bloques (*block*). Pero, antes de nada, hay que conocer su nacimiento y cómo surge esta nueva tecnología. En el año 2008 se registra la marca Bitcoin.org y se publica un artículo científico titulado "Bitcoin: A Peer-to-Peer Electronic Cash System" [22] bajo el seudónimo de Satoshi Nakamoto, surgiendo así uno de los grandes misterios de esta época, ¿quién o quiénes han sido los creados del Bitcoin? Y es que hasta la fecha nadie ha reclamado ser el autor y tampoco se ha hallado alguno, aunque sí que ha habido intentos de identificar al creador, pero todos han fracasado.

En el año 2009, poco tiempo después de la publicación antes mencionada, se implementa como código libre y la red comienza con su funcionamiento. El principal objetivo del creador o creadores es el de simular un sistema de pago en efectivo de forma digital, es decir, conseguir eliminar cualquier intermediario entre dos partes donde cada una es anónima, solo necesita ser conocida por la otra parte de la transacción y por nadie más.

Por lo tanto, desde un primer momento, la red Bitcoin está destinada al intercambio de valor entre personas de manera digital y anónima a través de la criptomoneda desarrollada por Satoshi. El desarrollo de este sistema supuso los cimientos del nacimiento de la tecnología Blockchain.

Una vez introducido de donde viene esta tecnología, debe definirse qué es. Como se ha adelantado al comienzo, no es más que una estructura de datos, y como su nombre indica, se trata de una cadena de bloques. Esta estructura de datos está formada por bloques, que cada uno contiene transacciones y metainformación. Esta metainformación contiene una estampa de tiempo de creación del bloque, el número de bloque dentro de la red, un *hash* asociado al bloque y el *hash* del bloque anterior, de tal forma que están ordenados, además de otra posible información. Concretamente, el *hash* se refiere a un número igual o menor a 256 bits que permite identificar el bloque de forma única.

Hasta ahora, se ha definido que una red Blockchain está formada por bloques y cada bloque tiene una dirección y la dirección del bloque anterior. Además, cada bloque contiene información, en concreto, transacciones. Las transacciones son esencialmente el envío o transferencia de valor, como puede ser la criptomoneda Bitcoin, entre dos partes. Por lo tanto, hay un destinatario y un mensajero. Para que una persona pueda enviar y recibir tal valor, es

necesario el uso de las *wallet* o monedero, que son las cuentas con las que se maneja la criptomoneda en cuestión por parte de los usuarios. Cada *wallet* tiene un *hash* asociado que la permite Identificarse de forma única y solo es necesario el *hash* de otra cuenta para enviar el dinero. Aquí es donde entra la parte anónima de la simulación de pago en efectivo, solamente la persona que envía el dinero conoce al destinatario, ya que solo se reflejará el *hash* de la transacción.

Otra parte fundamental de la Blockchain es el *ledger* o el libro mayor, el cual recoge todas y cada una de las transacciones realizadas dentro de la red. Puede considerarse como un libro mayor de cuentas de toda la red.

Por último, cabe definir los nodos, una parte muy importante dentro de la red. Anteriormente, se han explicado lo que son los bloques, pero no como se generan. Una red Blockchain comienza con el bloque 0 o génesis y el resto de los bloques se van generando poco a poco. Aquí es donde entran los nodos, los cuales tienen dos tareas principales. La primera es la de generar bloques nuevos a cabio de recibir una cantidad de criptomonedas de esta, de tal forma nuevos bloques se crean y la Blockchain crece. La segunda, es que confirman transacciones, es decir, indican si una transacción la consideran verdadera y la confirman. En caso de que crean que la transacción es mentira y que está intentado estafar dentro de la red, es rechazada por el nodo. Por lo tanto, es necesario tener un incentivo para que se creen nodos, y es aquí donde nacen las criptomonedas. Estas son entregadas a aquellas personas que creen nodos y ayuden al funcionamiento de la red. En consecuencia, se genera una competencia por ver que nodo será el que generé el nuevo bloque de la estructura de datos.

En este punto entra el carácter democrático natural de la Blockchain, y es que los nodos tienen que ponerse de acuerdo para admitir o no nuevas transacciones, y para ello, cada transacción precisa del 51% de los nodos a favor para que la transacción sea verificada y admitida. En caso de que no llegue a tal umbral, será desechada. Otro punto importante, es que todas y cada una de las transacciones son inmutables, por lo que toda la información queda dentro de la red para siempre, siendo imposible eliminarla.

Además, para que el valor de la criptomoneda tenga valor, es necesario invertir, ya sea en trabajo y tiempo o a través de acumular una gran cantidad de criptomonedas, para que los incentivos mantengan el ecosistema y el valor de la criptomoneda se mantenga en el tiempo y no decaiga. También ayuda el hecho de que el suministro está limitado, por lo que, una vez generados todos los bloques de la red, no será posible crear más criptomonedas. Al no "imprimir" nuevas monedas de la nada, se evita la inflación.

Es en esa inversión de los nodos la que permite generar nuevos bloques y confirmar transacciones, por lo que es necesario que la gran cantidad de nodos en todo el mundo que puedan coexistir se pongan de acuerdo y sigan un consenso. Actualmente, hay dos consensos dominantes, *PoS* y *PoW* [23]. *Proof of Stake* (*PoS*) o prueba de autoridad consiste en que los nodos almacenen una gran cantidad de criptomonedas de la red bloqueados para que se le permita. De esta forma, se puede confiar en tal nodo y en caso de que mienta en alguna transacción, se procederá a retirar sus criptomonedas y se eliminará el nodo de la red. *Proof of Work* (*PoW*) o prueba de trabajo consiste en adivinar el *hash* del nuevo bloque a ser generado resolviendo un problema matemático muy complejo el cual requiere de una alta potencia computacional. Por ello, se está invirtiendo una gran cantidad de recursos como hardware y

electricidad para generar un nuevo bloque y conseguir la recompensa. A este proceso se le denomina comúnmente como minería de criptomonedas, emulando así a la extracción de oro de una mina.

Resumiendo, una red Blockchain es una estructura de datos formada por bloques que están conectados entre sí y albergan transacciones inmutables. Los bloques son generados por los nodos y las transacciones verificadas por los mismos, ambos a cambio de una cantidad de la criptomoneda asociada a la red. Existen dos consensos principales para los nodos: *PoW* y *PoS*. Los consensos permiten que todos los ataques realizados a la red deban superar un umbral por encima del 50%, normalmente, para que pueda ser posible comenzar el hackeo, que no completarlo, ya que más adelante seguiría la competencia entre nodos por ver que bloques y transacciones son correctos y verdaderos.

3.1.1 Ethereum

Para el desarrollo del proyecto, la red Blockchain seleccionada ha sido Ethereum, concretamente la red de pruebas Rinkeby, la cual fue construida sobre Ethereum. Esto se debe a que la red principal de Ethereum tiene unas comisiones ciertamente altas a la hora de usarla, por lo que se utilizará una red de pruebas que emulará su comportamiento.

Ethereum fue desarrollada por Vitálik Buterin durante el 2014 y desplegada en el año 2015. Se trata de una red Blockchain *PoW* combinada con *PoS*, ya que durante el 2021 comenzó este cambio de consenso debido al impacto medioambiental que producen las redes *PoW*, las cuales hacen uso de una gran cantidad de electricidad. Este cambio se completará al final de 2022. También, esta red tiene la criptomoneda Ether para apoyar e incentivar la participación en la red [10].

La característica añadida que presenta esta red y por la que ha sido escogida es porque permite la ejecución de código dentro de los bloques de la red, no solamente realizar transacciones. Los programas ejecutados dentro de la red se denominan *SmartContracts* o contratos inteligentes, que permiten que dos o más partes lleguen a un acuerdo que será ejecutado de manera automática sin necesidad de un agente intermedio que regule o supervise el acuerdo. Por lo tanto, las posibilidades son infinitas.

Para escribir los contratos inteligentes, Ethereum presenta un lenguaje de programación propio, llamado Solidity, desarrollado por Gavin Wood con el equipo de Ethereum en el año 2015 [24]. Es un lenguaje fuertemente tipado que se ejecutado en la Ethereum Virtual Machine (EVM) que funciona sobre la red Blockchain de Ethereum.

Para el desarrollo de este proyecto se ha utilizado la red Rinkeby de pruebas y Solidity para el desarrollo del contrato inteligente un rango de versiones soportadas [v0.7.0-v0.9.0].

3.2 Sistema distribuido de almacenamiento

Básicamente, es una red de nodos o computadores conectados entre sí. Los datos almacenados serán guardados en más de un nodo. Por lo tanto, no existe un solo punto donde toda la información será almacenada, si no que se realizará varios puntos distanciados físicamente.

Las principales razones por las que hace uso de este sistema son [25]:

- Gran flexibilidad y escalabilidad: permite una rápida y sencilla manera de incrementar la capacidad de almacenamiento ya que permite el escalar horizontal y verticalmente.
- Mayor rendimiento: cada servidor o nodo dispone de su propio hardware, es decir, propia CPU, espacio de almacenamiento y banda de red, pero todo el sistema se comporta como un solo grupo. Por lo que cuando un nuevo nodo se une, los atributos del sistema como el rendimiento o la velocidad aumentan.
- Mayor resiliencia: los datos almacenados no se guardan en un solo punto si no en varios. Esto ofrece una mayor seguridad al salvaguardar los datos y genera copias de seguridad de los mismo contantemente. En caso de que un nodo de la red deje de funcionar o se pare, el resto lo respaldarán y podrán cubrir su caída.
- Mayor disponibilidad: la existencia de diversos nodos en diferentes espacios geográficos incrementa el nivel de disponibilidad con respecto a modelos en los que solo existe un único punto en donde se alberga toda la información y deba estar funcionando sin parar. De esta manera, si se produce que un nodo está mal funcionando o desactivo, la red podrá seguir activa ofreciendo el mismo servicio.
- Menos vulnerable a ataques informáticos: la existencia de diversos nodos en diferentes espacios geográficos incrementa el nivel de seguridad con respecto a modelos en los que solo existe un único punto en donde se alberga toda la información. De esta manera, los ataques informáticos deberán darse en diversos puntos a la vez para poder vulnerar el sistema.

3.2.1 IPFS

Concretamente, se hará uso del InterPlanetary File System (IPFS) o Sistema de archivos interplanetario diseñado por Juan Benet en el año 2015. Esta plataforma es un sistema distribuido de almacenaje de contenido basado en el método P2P (*Peer-to-Peer*) [3].

Este método P2P consiste en una red de ordenadores o nodos que intercambian la información entre sí sin necesidad de un cliente o servidor fijo, ya que los mismos nodos de la red interactúan como tal. Por lo tanto, los datos fluyen directamente de un ordenador a otro, sin necesidad de un servidor intermediario donde la información está almacenada. Un ejemplo claro de esto lo encontramos en la piratería de películas y series, cuando se usa un Torrent o cualquiera de las páginas de descarga existentes. El mecanismo funciona de la siguiente manera: cuando el usuario quiere descargar la película que quiere ver, el sistema busca un nodo cercano operando en la red el cual tenga disponible el contenido para descargar.

Entonces, este nodo con el contenido envía los datos al ordenador del usuario que quiere descargar la película directamente, sin pasar por un servidor o un intermediario [26].

Una vez explicado en que consiste el método P2P, ya puede entenderse que IPFS busca los datos por contenido, no por localización, y este su gran cambio revolucionario. No pide la información un servidor central o distribuido, si no que busca que nodo más cercano es el que tiene la información solicitada y es el mismo nodo quien la envía.

Una vez entendido el carácter distribuido de IPFS y su funcionamiento como red, cabe adentrase en los atributos de seguridad, privacidad, resiliencia y velocidad de descarga. En cuanto a la seguridad, IPFS divide el documento en porciones, las cuales son encriptadas, replicadas y repartidas a través de toda la red de nodos. A este documento se le asigna una huella para poder ser identificado posteriormente. En cuanto a la privacidad, IPFS no tiene ninguna restricción a la hora de añadir un nuevo contenido a la red, por lo que está protegido contra cualquier tipo de censura. En cuanto a la resiliencia, IPFS puede ofrecer el contenido a pesar de ciertos nodos de la red estén caídos o desactivados. Por último, IPFS ofrece una gran rapidez de descarga para aquel contenido en un nodo cercano al ordenador del usuario. Si se diera el caso en el que está en una localización alejada solo se demoraría en el primer acceso [27].

3.2.1.1 Web3storage

Para facilitar la interacción con IPFS, se hará uso de la API Web3Storage ya que ofrece una conexión rápida y muy sencilla de implementar. Además, está enfocada para ser usada con el lenguaje de programación JavaScript, escogido para este proyecto. Se utilizará la versión v3.5.7 para este proyecto [28].

3.3 Encriptación

La encriptación es una parte fundamental de este proyecto ya que el cifrado de los documentos es algo esencial para garantizar la seguridad de estos.

3.3.1 Hash

El *hash* se refiere a una función o algoritmo matemático que genera una huella unívoca de una entrada de datos, como un documento, archivo, imagen… Esta huella se trata de una serie de caracteres o dígitos de longitud fija. Dada una entrada de datos, este algoritmo siempre devolverá la misma salida, es decir, dado un documento al que se le aplica esta función, siempre generará la misma salida si este no sufre ninguna modificación. En cuanto se produce una mínima modificación, se generará una salida diferente. Esto permite salvaguardar la integridad de los documentos y verificar si han sido alterados [29].

Para la generación de la salida se ha seleccionado el algoritmo SHA256, el cual genera una huella alfanumérica de 256 bits.

3.3.2 Encriptación por clave privada

Este mecanismo de encriptación consiste en encriptar un archivo mediante el uso de una clave privada. Se realiza con el algoritmo AES-256, que encripta los datos a través de una serie de procedimientos matemáticos dependiendo uno de ellos de la clave privada administrada al algoritmo. Esta clave privada será conocida solo por el administrador del sistema. Fue desarrollado por Joan Daemen y Vincent Rijmen y publicado por primera vez en 1998 [30].

3.4 Protocolo HTTP/HTTPS

Hypertext Transfer Protocol (HTTP) o Protocolo de transferencia de hipertexto es un protocolo de comunicaciones el cual permite transferir información a través de documentos, como pueden ser HTML o XML. Este protocolo fue publicado en 1999 y desarrollado por la World Wide Web Consortium y la Internet Engineering Task Force de forma conjunta. Para este proyecto, será usada la versión HTTP v1.1. Por otro lado, Hypertext Transfer Protocol Secure (HTTPS) o protocolo seguro de transferencia de hipertexto se caracteriza por ser la versión segura de HTTP, ya que cifra toda la conexión de un punto a otro [31].

3.5 API

Una *Application Programming Interface* (API) o interfaz de programación de aplicaciones es un conjunto de definiciones, métodos y protocolos para integrar software en aplicaciones y programas. En términos generales, una API permite a un software interactuar con un sistema para obtener datos o ejecutar una función [32]. Como ejemplo de API dentro del proyecto, se encuentra Web3Storage.

3.6 Aplicación web

Una aplicación web no es más que un programa ejecutado en el navegador y los usuarios pueden acceder a él a través de internet mediante un navegador [33]. Normalmente, las aplicaciones web tienen una arquitectura cliente-servidor. Básicamente, el cliente realiza peticiones al servidor, el cual da una respuesta [34].

3.6.1 Lado cliente

El lado cliente o *frontend* consiste en una aplicación informática ejecutada en el navegador la cual consume a través de red de internet un servicio proporcionado por el lado servidor. En esta parte de la arquitectura se incluye lo que ve el usuario como imágenes, vídeos o textos [35].

Entre los lenguajes de marcado interpretados por el navegador, están HTML y CSS. Además, los procesos del lado cliente son escritos en JavaScript principalmente. Estos tres serán los utilizados para el proyecto.

HyperText Markup Language (HTML) o lenguaje de marcado de hipertexto define la estructura del contenido en la web. El término hipertexto hace referencia a los enlaces que conectan páginas web entre sí. Su creación se remonta a 1980 cuando Tim Berners-Lee lo creó [36].

Cascading Style Sheets (CSS) o hojas de estilo en cascada es el lenguaje destinado a describir la presentación de documentos tipo HTML o XML. Desarrollado por la W3C en 1996 [37].

JavaScript es un lenguaje de programación ligero interpretado y ejecutado por la mayoría de los navegadores web. Principalmente, permite ejecutar procesos en el lado cliente dentro del navegador. Este lenguaje de programación fue creado por Brendan Eich en Netscape Communications Corporation, para el navegador web Netscape Navigator.

3.6.2 Lado servidor

En el lado cliente o *backend* consiste en una aplicación que ejecuta programas de forma remota en un servidor. Recibe peticiones del lado cliente y envía de vuelta una respuesta adecuada. Para está comunicación entre lado cliente y lado servidor se hace uso del protocolo HTTP [38].

El lenguaje escogido para programar en el lado servidor ha sido JavaScript. Además, se complementará con los siguientes frameworks y bibliotecas.

3.6.2.1 JavaScript

Se trata de un lenguaje de programación interpretado, imperativo, de alto nivel, orientado a objetos, débilmente tipado y dinámico de propósito general lanzado en 1995 por Netscape Communications [39].

3.6.2.2 Nodejs

Se trata de un entorno de ejecución para JavaScript caracterizado por ser asíncrono, multiplataforma y con una arquitectura orientada a eventos. Principalmente, está enfocado a la ejecución de programas en el lado servidor.

Fue creado por Ryan Lienhart Dahl en el año 2009 [40]. Para este proyecto se hará uso de la versión v14.18.0.

3.6.2.3 Expressjs

Se trata de un marco de trabajo destinado a la creación de aplicaciones web y APIs dentro de Nodejs. Fue creado por TJ Holowaychuk y publicado en 2010 por el mismo [41]. Para este trabajo se hará uso de la versión v4.17.3.

3.6.2.4 Trufflejs

Se trata de un entorno de desarrollo destinado a facilitar el despliegue de contratos inteligentes y la interacción con la red Blockchain de Ethereum. Fue desarrollado y publicado por la compañía ConsenSys Software Inc. en el año 2016 [42].

3.6.2.5 Infura

Se trata de una API con diferentes herramientas la cual facilita la conexión con la red de Ethereum desde cualquier aplicación a través de conexiones HTTP y WebSocket (WSS). Fue desarrollado en 2016 [43].

3.6.2.6 Web3js

Se trata de una colección de librerías que permite la interacción con nodos de la red de Ethereum a través del protocolo HTTP. Se hará uso de la versión v1.7.4.

3.6.2.7 MongoDB

MongoDB es un sistema de base de datos NoSQL, no relacional, orientado a documentos y de código abierto lanzada en 2009 por la empresa MongoDB [44]. Permite una gran escalabilidad y facilidad de manejo de datos. Se hará uso de la versión v5.0.9.

3.7 Aspectos teóricos sobre la integración en la empresa

Este sistema ha sido integrado dentro del gestor documental Neodoc de la empresa Neosystems, el cual está implementado con las siguientes tecnologías de desarrollo.

3.7.1 Ruby

Se trata de un lenguaje de programación interpretado, de propósito general, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro “Matz” Matsumoto. Fue publicado en 1995.Dentro de la empresa, ha sido utilizado para desarrollar toda la lógica del producto en la parte *backend* [45].

3.7.1.1 Ruby on Rails

Se trata del framework más importante dentro del lenguaje de programación Ruby. Está enfocado al desarrollo de aplicaciones web de una manera ágil siguiendo el patrón modelo vista controlador. Fue desarrollado por David Heinemeier Hansson y lanzado en el año 2005 [46].

3.7.2 JavaScript

Ya se ha definido en el apartado 3.6.2.1.

Dentro de la empresa, este lenguaje se usa para el *frontend*. Para la integración en este proyecto, se hará uso también del framework JQuery definido en el apartado 3.8.4.

3.7.3 PostgreSQL

Se trata de un sistema de gestión de bases de datos relacional orientado a objetos lanzado en 1996 por el profesor Michael Stonebraker. Dentro de la empresa, se ha utilizado como sistema para gestionar la base de datos [47].

3.8 Otros aspectos teóricos

A continuación, se engloban otras plataformas, frameworks, librerías y tecnologías utilizadas que fueron de ayuda para el desarrollo del proyecto.

3.8.1 Git

Se trata de un software de control de versiones desarrollado por Linus Torvalds lanzando en 2007 [48], [49].

3.8.2 GitHub

Se trata de una plataforma de almacenamiento de código basada en *git* creada en 2008 por Tom PrestonWerner, Chris Wanstrath y PJ Hyett [50].

3.8.3 Crypto

Se trata de una librería para JavaScript destinada para cifrar contenido a través de diferentes algoritmos. Creada por Irakli Gozalishvili [51].

3.8.4 JQuery

Se trata de una librería de JavaScript destinada a manipular componentes HTML. Lanzada en el año 2006 y creada por John Resig [52].

3.8.5 Dotenv

Se trata de una manera de configurar variables de entorno a través de un fichero. Creado por Scott Motte [53].

3.8.6 PostMan

Se trata de una plataforma para desarrollar, diseñar y sobre todo para probar APIs creada por Abhinav Asthana [54].

3.8.7 Bcrypt

Se trata de una función *hash* para contraseñas diseñado por Niels Provos y David Mazières [55].

3.8.8 Bootstrap

Se trata de un conjunto de herramientas que contiene plantillas de diseño basado en HTML y CSS. Creado por Mark Otto y Jacob Thornton de la compañía Twitter en 2011 [56].

3.8.9 Ejs

Se trata de un lenguaje de plantillas que permite generar un documento HTML y ejecutar JavaScript en texto plano. Creado por Matthew Eernisse. [57]

3.8.10 Jwt

Se trata de un estándar basado en JSON para el manejo de autenticación de usuarios y creación de roles. Fue propuesto por la Internet Engineering Task Force. [58]

3.8.11 Express-session

Se trata de un *middleware* que almacena los datos de sesión del usuario en el servidor [59].

3.8.12 Express-fileUpload

Se trata de un *middleware* que facilita el manejo de archivos dentro de una aplicación con Expressjs [60].

3.8.13 Mocha

Se trata de una framework para realizar pruebas de forma sencilla y rápida [61].

3.8.14 Supertest

Se trata de una librería enfocada a las pruebas de código para servicios HTTP [62].

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

Una vez definidos los aspectos teóricos, los objetivos y el alcance del proyecto, se presentará la planificación y presupuesto inicial para el desarrollo de todo el sistema. La planificación define las tareas a realizar y estima los plazos de desarrollo del proyecto y de entrega del producto, así como los recursos humanos y roles necesarios. Por otro lado, el presupuesto describe el dinero estimado para que el proyecto se lleve a cabo en cuanto a personal y material.

4.1 Planificación Inicial

En esta sección se especifican las tareas en las que está divido el proyecto desde su inicio hasta su entrega final. Con el fin de estimar los plazos, se hará uso de la herramienta Microsoft Project con la finalidad de construir un diagrama de Gantt. Esta planificación inicial también incluye los roles y personal necesarios. Cabe destacar que esta planificación inicial puede verse afectada a medida que avance el proyecto.

4.1.1 Cronograma

El proyecto tiene el 1 de febrero de 2022 como fecha de comienzo y el 11 de julio de 2022 como fecha de finalización, concretamente 5 meses y 10 días. El horario de trabajo será de unas 5 horas al día de lunes a viernes de 08:30 a 13:30, quedando un total de 114 días reales de trabajo. En total, el proyecto suma 945,1 horas de trabajo entre todos los roles dentro del proyecto. Se han adjudicado 6,4 días de trabajo para revisar el proyecto y otro día entero de trabajo que sobraría. Este tiempo sobrante estaría destinado para imprevistos en cuanto a la planificación, como el retraso de una tarea.

4.1.2 Roles

Para el desarrollo de este proyecto serán necesarios los siguientes roles: arquitecto software, desarrollador Full-Stack, diseñador software, consultor de tecnología y jefe de proyecto. Todos estos roles se verán simplificados en una sola persona quien realizará todo el proyecto, que será el autor de este documento.

Horas de trabajo a realizar según el rol:

- Arquitecto software: 232,2 horas.
- Desarrollador Full-Stack: 190,1 horas.
- Diseñador software: 272,3 horas.
- Consultor de tecnología: 33,1 horas.

- Jefe de proyecto: 217,3 horas.

4.1.2.1 OBS

A continuación, se presenta el organigrama de la estructura de roles de personal del proyecto de forma jerárquica:

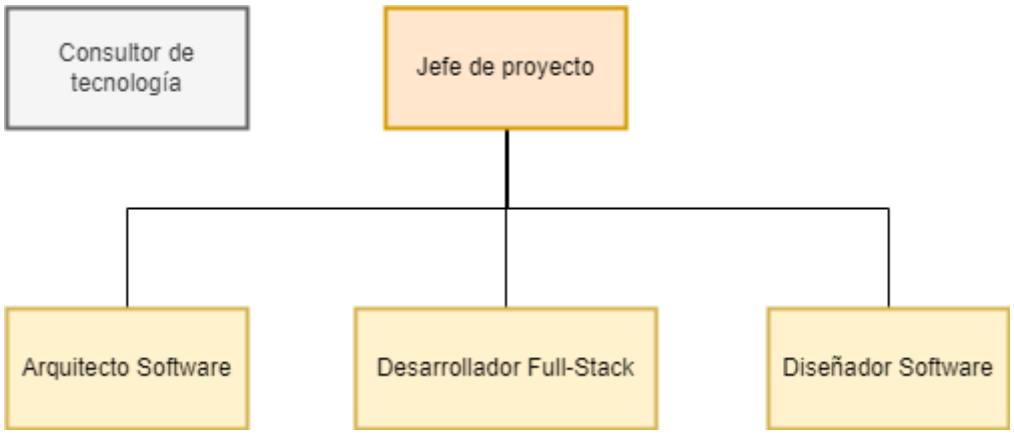

Figura 4.1 Organigrama OBS

4.1.3 Agrupación de tareas

Para planificar el proyecto, se han identificado 5 tareas principales que engloban un conjunto de subtareas propio:

- **Estudio del sistema**: en este apartado se engloban las tareas de comienzo del proyecto destinadas a investigar sobre las tecnologías, profundizar sobre la situación actual del tema escogido a tratar, así como la parte de análisis en el que se agrupa la definición de alcance y objetivos del sistema a parte de la determinación de diagramas y casos de uso.
- **Diseño del sistema**: en este apartado se realizan todos los diagramas necesarios para la elaboración del proyecto y el diseño de todas las partes del sistema.
- **Implementación**: en este apartado se implementa el sistema realmente.
- **Pruebas**: en este apartado se realizan todas las pruebas del proyecto, en concreto las de usabilidad, accesibilidad, integración y unitarias.
- **Manuales del sistema**: en este apartado se elaboran todos los manuales de uso, de instalación, de ejecución, de usuario y del programador.

4.1.4 Diagrama de Gantt

A continuación, se expone el diagrama de Gantt de la planificación del proyecto con las tareas, su duración y los recursos asignados a cada una.

Outline Number	Task Name	Duration	Start	Finish	Pr
1	**Sistema de almacenaje distribuido**	**114 days**	**Tue 01/02/22**	**Mon 11/07/22**	
1.1	Reunión de arranque	0,22 days	Tue 01/02/22	Tue 01/02/22	
1.2	**Estudio del sistema**	**36,84 days**	**Wed 02/02/22**	**Thu 24/03/22**	**2**
1.2.1	Desarrollo de la idea para el proyecto	2 days	Wed 02/02/22	Thu 03/02/22	
1.2.2	**Realización del estudio previo**	**9,4 days**	**Fri 04/02/22**	**Thu 17/02/22**	**4**
1.2.2.1	Estudio de la situación actual	3 days	Fri 04/02/22	Tue 08/02/22	
1.2.2.2	Investigación de tecnologías	6,4 days	Wed 09/02/22	Thu 17/02/22	6
1.2.3	**Analisis del sistema**	**24,44 days**	**Thu 17/02/22**	**Wed 23/03/22**	**5**
1.2.3.1	Definición del alcance	3,2 days	Thu 17/02/22	Tue 22/02/22	
1.2.3.2	Definición de los objetivos del proyecto	3,2 days	Tue 22/02/22	Fri 25/02/22	9
1.2.3.3	Análisis de casos de uso	11,2 days	Fri 25/02/22	Mon 14/03/22	10
1.2.3.4	Análisis y determinación de diagramas	6,84 days	Tue 15/03/22	Wed 23/03/22	11
1.2.4	Documentación del estudio del sistema	1 day	Wed 23/03/22	Thu 24/03/22	8
1.3	**Diseño del sistema**	**30,2 days**	**Thu 24/03/22**	**Fri 06/05/22**	**3**
1.3.1	Realización de diagramas relacionados con el diseño	3 days	Thu 24/03/22	Tue 29/03/22	
1.3.2	**Diseño de Blockchain de Ethereum**	**9,6 days**	**Tue 29/03/22**	**Tue 12/04/22**	**15**
1.3.2.1	Diseño de la interacción con la red Ethereum	4,8 days	Tue 29/03/22	Tue 05/04/22	
1.3.2.2	Diseño del contrato inteligente	4,8 days	Tue 05/04/22	Tue 12/04/22	17
1.3.3	Diseño de la interacción con IPFS	6,4 days	Tue 12/04/22	Wed 20/04/22	16
1.3.4	Diseño de la aplicación web	4,8 days	Wed 20/04/22	Wed 27/04/22	19
1.3.5	Diseño de la interfaz de usuario	5,4 days	Wed 27/04/22	Thu 05/05/22	20
1.3.6	Documentación diseño del sistema	1 day	Thu 05/05/22	Fri 06/05/22	21
1.4	**Implementación**	**21,6 days**	**Fri 06/05/22**	**Mon 06/06/22**	**21**
1.4.1	**Implementación del lado servidor (backend)**	**12,8 days**	**Fri 06/05/22**	**Tue 24/05/22**	**14**
1.4.1.1	Integración de la API de Web3Storage	3,2 days	Fri 06/05/22	Wed 11/05/22	
1.4.1.2	Implementación de la API	3,2 days	Wed 11/05/22	Mon 16/05/22	25
1.4.1.3	Implementación del contrato inteligente	3,2 days	Mon 16/05/22	Thu 19/05/22	26
1.4.1.4	Implementación de la encriptación de los archivos	1,6 days	Thu 19/05/22	Mon 23/05/22	27
1.4.1.5	**Implementación del registro de usuario**	**1,6 days**	**Mon 23/05/22**	**Tue 24/05/22**	**28**
1.4.1.5.1	Creación e integración de la base de datos	1,6 days	Mon 23/05/22	Tue 24/05/22	
1.4.2	**Implementación del lado cliente (frontend)**	**7,8 days**	**Tue 24/05/22**	**Fri 03/06/22**	**24**
1.4.2.1	**Implementación de la interfaz de usuario**	**7,8 days**	**Tue 24/05/22**	**Fri 03/06/22**	
1.4.2.1.1	Interfaz de registro de usuario	4 days	Tue 24/05/22	Mon 30/05/22	
1.4.2.1.2	Interfaz de manejo de archivos	3,8 days	Mon 30/05/22	Fri 03/06/22	33
1.4.3	Documentación de la implementación	1 day	Fri 03/06/22	Mon 06/06/22	31
1.5	**Pruebas**	**11,2 days**	**Mon 06/06/22**	**Tue 21/06/22**	**23**
1.5.1	**Pruebas unitarias**	**4,8 days**	**Mon 06/06/22**	**Mon 13/06/22**	
1.5.1.1	Desarrollo de pruebas para la API del sistema	2,4 days	Mon 06/06/22	Thu 09/06/22	
1.5.1.2	Desarrollo de pruebas para el registro de usuarios	2,4 days	Thu 09/06/22	Mon 13/06/22	38
1.5.2	Pruebas de integración	3,2 days	Mon 13/06/22	Thu 16/06/22	37
1.5.3	Pruebas de usabilidad	1 day	Thu 16/06/22	Fri 17/06/22	40
1.5.4	Pruebas de accesibilidad	1,2 days	Fri 17/06/22	Mon 20/06/22	41
1.5.5	Documentación de la pruebas	1 day	Mon 20/06/22	Tue 21/06/22	42
1.6	**Manuales del sistema**	**6,4 days**	**Tue 21/06/22**	**Thu 30/06/22**	**36**
1.6.1	Manual de intalación	1,6 days	Tue 21/06/22	Thu 23/06/22	
1.6.2	Manual de ejecución	1,6 days	Thu 23/06/22	Mon 27/06/22	45
1.6.3	Manual de usuario	1,6 days	Mon 27/06/22	Tue 28/06/22	46
1.6.4	Manual del programador	1,6 days	Tue 28/06/22	Thu 30/06/22	47
1.7	Revisión final del proyecto	6,4 days	Thu 30/06/22	Fri 08/07/22	44
1.8	Entrega del proyecto	0 days	Mon 11/07/22	Mon 11/07/22	

Figura 4.2 Diagrama de Gantt parte 2

4.2 Presupuesto Inicial

A continuación, se tratarán los costes de realizar el proyecto para la empresa con la finalidad de conseguir un presupuesto inicial al que mostrar al cliente.

4.2.1 Desarrollo de Presupuesto Detallado (Empresa)

Para el desarrollo del proyecto, la empresa deberá hacer frente a una serie de costes necesarios de recursos humanos, de software, de hardware y el coste del desarrollo de la implementación.

La empresa contratará 5 trabajadores para el proyecto. Para ello, asumirá los siguientes costes con el fin de contratarlos:

- Desarrollador Full-Stack: 25€/hora.
- Jefe de proyecto: 45€/hora.
- Consultor de tecnología: 40€/hora.
- Diseñador software: 25€/hora.
- Arquitecto software: 30€/hora.

Con respecto al software utilizado, la mayor parte es software libre excepto las licencias de Microsoft Project, Microsoft Windows 10 y Microsoft Office 365, que presentan un gasto anual sin IVA de 100,80€, 145€ y 61,20€ respectivamente por unidad. Para la licencia de Windows 10 se estima una vida útil de 10 años mientras que para las otras dos una vida útil de 1 año. Durante un año se estiman 1265 de horas de trabajo contando con un horario de 5 horas al día y quitando fines de semana y festivos. El proyecto presenta una duración de 5 meses y 11 días, en concreto 570 horas quitando fines de semana con 5 horas de trabajo por día.

Cálculo de la amortización de Microsoft Windows 10:

Amortización = 570 / 12650 = 0,045 = 4,5%

Cálculo de la amortización de Microsoft Project:

Amortización = 570 / 1265 = 0,45 = 45%

Cálculo de la amortización de Microsoft Office 365:

Amortización = 570 / 1265 = 0,45 = 45%

En el apartado de recursos de hardware, es necesario el cálculo de la amortización de los portátiles, cables y periféricos:

Cálculo de la amortización de un portátil con una vida útil de 5 años:

Amortización = 570 / 6325 = 0,09 = 9%

Cálculo de la amortización de los periféricos con una vida útil de 5 años:

Amortización = 570 / 6325 = 0,09 = 9%

Cálculo de la amortización de los cables con una vida útil de 10 años:

Amortización = 570 / 12650 = 0,045 = 4,5%

Cálculo de la amortización de un router con una vida útil de 7 años:

Amortización = 570 / 8855 = 0,064 = 6,4%

Al presupuesto hay que añadirle un 21% de IVA en el año 2022 y un 25% sobre el coste de beneficio para la empresa por haber realizado el proyecto.

En la siguiente tabla se desglosa el presupuesto con los gastos de la empresa:

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	*Implementación*				
1.1	Estudio del sistema	1	100%	14.165,00 €	14.165,00 €
1.2	Diseño del sistema	1	100%	8.305,00 €	8.305,00 €
1.3	Implementación	1	100%	2.700,00 €	2.700,00 €
1.4	Pruebas	1	100%	1.400,00 €	1.400,00 €
1.5	Manuales del sistema	1	100%	1.440,00 €	1.440,00 €
1.6	Revisión del proyecto	1	100%	1.440,00 €	1.440,00 €
2	*Recursos Software*				
2.1	Microsoft Windows 10	5	4,5%	145,00 €	32,63 €
2.2	Microsoft Project	1	45%	100,80 €	45,36 €
2.3	Microsoft Office 365	5	45%	61,20 €	137,70 €
2.4	Visual Studio Code	5	100%	0,00 €	0,00 €
2.5	Nodejs	1	100%	0,00 €	0,00 €
2.6	Expressjs	1	100%	0,00 €	0,00 €
2.7	Trufflejs	1	100%	0,00 €	0,00 €
2.8	IPFS	1	100%	0,00 €	0,00 €
2.9	MongoDB	1	100%	0,00 €	0,00 €
2.10	Ethereum Rinkeby	1	100%	0,00 €	0,00 €
3	*Recursos Hardware*				
3.1	Cable	30	4,5%	3,50 €	4,73 €
3.2	Periféricos	10	9%	25,00 €	22,50 €

3.3	Router	1	6,4%	40,00 €	2,56 €
3.4	Portátil	5	9%	600,00 €	270,00 €
4	*Gastos indirectos*				
4.1	Papel	50	50%	0,06 €	1,50 €
4.2	Bolígrafo	8	50%	0,60 €	2,40 €
4.3	Lápiz	13	50%	0,40 €	2,60 €
4.4	Electricidad	1	100%	2.004,00 €	2.004,00 €
4.5	Internet por Wifi	1	100%	360,00 €	360,00 €
4.6	Instalaciones	1	100%	2.230,00 €	2.230,00 €
Subtotal					34.565,97 €
Beneficio (25%)					8.641,49 €
IVA (21%)					7.258,85 €
TOTAL					**50.466,32 €**

Tabla 4.1 Presupuesto inicial empresa

4.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

A continuación, se muestra la tabla con el presupuesto para el cliente de una manera simplificada y clara:

Concepto	Cantidad	Precio unitario (€)	Coste total (€)
Estudio del sistema	1	20.281,69 €	20.281,69 €
Diseño del sistema	1	11.684,50 €	11.684,50 €
Implementación	1	3.662,52 €	3.662,52 €
Pruebas	1	1.853,98 €	1.853,98 €
Manuales del sistema	1	2.112,64 €	2.112,64 €
Revisión del proyecto	1	2.112,64 €	2.112,64 €
		Subtotal	41.707,96 €
		IVA (21%)	8.758,67 €
		TOTAL	**50.466,64 €**

Tabla 4.2 Presupuesto inicial cliente

Capítulo 5. Análisis

Este capítulo engloba todas las funcionalidades que el sistema debe tener, concretamente a través de la determinación del alcance, la especificación de requisitos y definición de casos de uso.

5.1 Definición del Sistema

Esta sección comprende el alcance del proyecto, definiendo las limitaciones del sistema y hasta donde se llegará para que se pueda extraer una idea general de todo el proyecto.

5.1.1 Determinación del Alcance del Sistema

Como ya se ha explicado anteriormente en los apartados 1.1 y 2.1, el sistema consiste en una forma distinta de almacenaje en la nube utilizando el concepto de red distribuida, apartado 0, combinada con tecnología Blockchain.

La principal finalidad del proyecto es que los archivos serán almacenados en la red distribuida IPFS, ya explicada en el apartado 3.2.1, de una forma descentralizada repartidos a lo largo de unos nodos. Para asegurar la privacidad y seguridad de los archivos, estos se encriptarán antes del almacenaje con el algoritmo de encriptación AES-256, mecanismo ya definido en el apartado 3.3.2.

Por otro lado, se debe asegurar también la integridad de los archivos, es decir, poder verificar al usuario que el documento que ha subido no ha sido modificado o alterado de ninguna manera por parte del sistema. Para ello, se calculará el *hash* de cada documento, explicado en el apartado 3.3.1, antes de ser almacenado en la red, de tal forma que en el momento de su descarga este deba tener el mismo *hash*. Para almacenar estos *hashes* de una forma inmutable y segura, se hará uso de una red Blockchain, apartado 3.1, concretamente la red Ethereum, apartado 3.1.1, implementando un contrato inteligente que permitirá, a través de transacciones inmutables, almacenar dentro de la red Blockchain los *hashes* de los documentos. A la hora de la descarga de un documento, se comprobará si el archivo es íntegro a través de una consulta al contrato inteligente desplegado en la red Blockchain. En caso afirmativo, el usuario recibirá su archivo, en caso contrario, se le notificará que no es el original y que ha sido alterado de alguna manera.

El último componente del sistema será una aplicación web para que cada usuario pueda tener su cuenta personal con sus documentos propios. De tal manera que, en la base de datos MongoDB, apartado 3.6.2.7, se almacenarán los datos de inicio de sesión junto con el identificador que IPFS provee de un archivo al almacenarlo para poder encontrarlo posteriormente.

Las conexiones que han de realizarse entre los distintos servicios del sistema, aplicación web con IPFS y la red Blockchain, se realizarán a través del protocolo seguro HTPPS, ya definido en el apartado 3.4.

De forma adicional, se ofrecerán las herramientas necesarias a través de una API, apartado 3.5, para que cualquier tercero, como el gestor documental Neodoc en el que se integrará este proyecto, apartado 1.1. La integración de este sistema dentro la plataforma se limita a comprobar solamente la integridad de los archivos, sin la necesidad de guardar los archivos dentro de la plataforma IPFS. De esta forma, la plataforma será la que salvaguarde los documentos y a través de la integración de la API desarrollada en este proyecto, se guarden los *hashes* de documentos específicos de los que se quiera comprobar su integridad dentro de la red Blockchain. En cuanto el usuario quiera comprobar la integridad de su documento, desde Neodoc se enviará el documento a comprobar, se genera su *hash* y se comprueba si existe dentro de la red Blockchain.

5.2 Requisitos del Sistema

En esta sección se recoge de maneral formal la especificación del sistema y las funcionalidades necesarias que debe tener.

5.2.1 Obtención de los Requisitos del Sistema

A continuación, se reúnen los requisitos del sistema, tanto funcionales como no funcionales que el sistema debe cumplir.

5.2.1.1 Requisitos funcionales

Se identifican en este apartado los requisitos funcionales según el tipo de usuario que utilice el sistema.

5.2.1.1.1 Usuario no registrado

RFregistro1. El sistema permitirá a los usuarios registrarse en la aplicación.

RFregistro1.1. Indicando un nombre de usuario.

RFregistro1.1.1. Obligatoriamente.
RFregistro1.1.2. Debe contener caracteres alfabéticos.
RFregistro1.1.3. Debe ser único en la base de datos.

RFregistro1.2. Indicando una contraseña.

RFregistro1.2.1. Obligatoriamente.

RFregistro1.2.2. Debe contener al menos PASS_MIN_CARACTERES caracteres alfanuméricos.

RFregistro1.2.2. Puede contener símbolos.

RFregistro1.3. Indicando una confirmación de la contraseña.

RFregistro1.3.1. Deberá coincidir con RFregistro1.2.

RFregistro1.3.2. En caso de que RFregistro1.2 y RFregistro1.3 no coincidan, el sistema mostrará un mensaje de error.

RFregistro1.4. En caso de que los datos RFregistro1.1, RFregistro1.2 y RFregistro1.3 tengan un formato válido, el sistema guardará ambos datos en la base de datos.

RFregistro1.5. En caso de que los datos RFregistro1.1, RFregistro1.2 o RFregistro1.3 no tengan un formato válido, el sistema mostrará un mensaje de error.

RFregistro1.5.1. El usuario deberá indicar los datos RFregistro1.1, RFregistro1.2 y RFregistro1.3 de nuevo.

5.2.1.1.2 Usuario registrado no autenticado

RFiniSesion1. El sistema permitirá a los usuarios iniciar sesión.

RFiniSesion1.1. Indicando un nombre de usuario.
RFiniSesion1.1.1. Obligatoriamente.
RFiniSesion1.2. Indicando una contraseña.
RFiniSesion1.2.1. Obligatoriamente.

RFiniSesion1.3. El sistema comprobará si los datos RFiniSesion1.1 y RFiniSesion1.2 existen en la base de datos.

RFiniSesion1.3.1. En caso de que coincidan con un usuario registrado, el sistema iniciará sesión del usuario.

RFiniSesion1.3.2. En caso de que no coincidan con un usuario registrado, el sistema mostrará un mensaje de error.

5.2.1.1.3 Usuario registrado y autenticado

RFsesionIniciada1. El sistema permitirá al usuario almacenar archivos.

RFsesionIniciada1.1. Indicando el archivo.

RFsesionIniciada1.1.1. Obligatoriamente.

RFsesionIniciada1.2. El sistema generará un *hash* a partir de RFsesionIniciada1.1.

RFsesionIniciada1.2.1. Se especifica en RNFseguridad2 el mecanismo de encriptación.

RFsesionIniciada1.3. El sistema encriptará el archivo de RFsesionIniciada1.1.

RFsesionIniciada1.3.1. Se especifica en RNFseguridad1 el tipo de encriptación.

RFsesionIniciada1.4. En caso de que no se realicen satisfactoriamente RFsesionIniciada1.3 y RFsesionIniciada1.2, el sistema mostrará un mensaje de error.

RFsesionIniciada1.4.1. El usuario deberá volver a indicar el archivo RFsesionIniciada1.1.

RFsesionIniciada1.5. En caso de que se realicen satisfactoriamente RFsesionIniciada1.3 y RFsesionIniciada1.2, el sistema enviará el archivo encriptado de RFsesionIniciada1.3 a IPFS para que se almacene.

RFsesionIniciada1.5.1. En caso de que IPFS almacene correctamente el archivo encriptado, retornará al sistema un identificador del documento almacenado.

RFsesionIniciada1.5.2. En caso de que IPFS no almacene correctamente el archivo, retornará al sistema un mensaje de error.

RFsesionIniciada1.5.2.1. El sistema mostrará un mensaje de error.

RFsesionIniciada1.5.2.2. El usuario deberá volver a indicar el archivo RFsesionIniciada1.1.

RFsesionInicida1.6. En caso de que cumpla RFsesionInicida1.5.1, el sistema generará una transacción en la red Blockchain.

RFsesionInicida1.6.1. La transacción contendrá información.

RFsesionInicida1.6.1. Una estampa de tiempo del momento en el que se produjo.

RFsesionInicida1.6.2. El *hash* del archivo RFsesionIniciada1.2.

RFsesionInicida1.6.3. La dirección *hash* de la wallet que la produjo.

RFsesionIniciada1.7. En caso de que RFsesionIniciada1.6 no complete la transacción, se mostrará un mensaje de error.

RFsesionIniciada1.7.1. El usuario deberá volver a indicar el archivo RFsesionIniciada1.1.

RFsesionIniciada1.8. En caso de que RFsesionIniciada1.6 complete la transacción, el sistema guardará en la base de datos.

RFsesionIniciada1.8.1. El valor retornado por IPFS en RFsesionIniciada1.5.1.

RFsesionIniciada1.8.2. El nombre del documento de RFsesionIniciada1.1.

RFsesionIniciada1.8.3. Los datos de inicio de sesión RFiniSesion1.1 y RFiniSesion1.2.

RFsesionIniciada1.8.1.1. En caso de error al introducir en la base de datos, se mostrará un mensaje de error.

RFsesionIniciada1.8.1.1.1. El usuario deberá volver a indicar el archivo RFsesionIniciada1.1.
RFsesionIniciada2. El sistema permitirá al usuario visualizar los documentos subidos.

RFsesionIniciada2.1. El sistema mostrará el nombre de los documentos.

RFsesionIniciada3. El sistema permitirá descargar al usuario los documentos asociados a su cuenta.

RFsesionIniciada3.1. Seleccionando un archivo de los mostrados en RFsesionIniciada2.

RFsesionIniciada3.2. El sistema buscará en la base de datos el identificador de IPFS almacenado del archivo RFsesionIniciada3.1.

RFsesionIniciada3.2.1. En caso de que no lo encuentre, mostrará un mensaje de error.

RFsesionIniciada3.2.2. En caso de que lo encuentre, el sistema descargará el archivo utilizando el identificador RFsesionIniciada1.5.1.

RFsesionIniciada3.3. El sistema desencriptará el archivo recibido en RFsesionIniciada3.2.

RFsesionIniciada3.4. El sistema calculará el *hash* del archivo desencriptado en RFsesionIniciada3.3.

RFsesionIniciada3.5. El sistema realizará una consulta a la red Blockchain con el *hash* de RFsesionIniciada3.4.

RFsesionIniciada3.5.1. En caso de que el *hash* calculado no exista dentro de la red Blockchain, se mostrará un mensaje de error.

RFsesionIniciada3.5.1.1. Se indicará que ha ocurrido una alteración en el archivo.

RFsesionIniciada3.5.2. En caso de que exista, el sistema mostrará el archivo descargado.

RFsesionIniciada4. El sistema permitirá al usuario eliminar su cuenta.

5.2.1.2 Requisitos no funcionales

Se identifican en este apartado los requisitos no funcionales relacionados con la seguridad, …

5.2.1.2.1 De seguridad

RNFseguridad1. Antes de enviar un archivo a IPFS, este deberá ser encriptado.

RNFseguridad1.1. Se aplicará en algoritmo de encriptación AES-256.

RNFseguridad2. El *hash* de un archivo será generado con el algoritmo SHA-256.

5.2.1.2.2 Tecnológicos

RNFArquitectura1. El sistema seguirá el uso de la arquitectura de microservicios.

RNFArquitectura1.1. Microservicio de la red Blockchain.

RNFArquitectura1.2. Microservicio del sistema distribuido de almacenaje.

RNFArquitectura1.3. Microservicio aplicación web.

RNFRed1. La red Blockchain especificada en RNFArquitectura1.1 debe de ser pública.

RNFRed1.2. Se implementará el sistema en la red Blockchain Rinkeby, red de pruebas de Ethereum.

RNFComunicacion1. Se hará uso de una API para la comunicación entre los microservicios de RNFArquitectura1.

RNFComunicacion1.2. Se utilizarán protocolos de transmisión de datos para la comunicación.

RNFComunicacion1.2.1. Protocolo HTTP/HTTPS.

RNFNavegador1. La aplicación web será soportada por navegadores web.

RNFNavegador1.1. Google Chrome.
RNFNavegador1.2. Mozilla Firefox.
RNFNavegador1.3. Microsoft Edge.

RNFDocumento1. El sistema soportará distintos tipos de extensiones de documentos.

RNFDocumento1.1. PDF.
RNFDocumento1.3. JPG.
RNFDocumento1.4. PNG.
RNFDocumento1.5. ZIP.
RNFDocumento1.6. RAR.
RNFDocumento1.7. DOC.
RNFDocumento1.8. TXT.

RNFIntegración1. El sistema deberá permitir la integración del sistema dentro de terceros.

RNFIntegración1.1. A través de la API del sistema.
RNFIntegración1.2. Utilizando el protocolo HTPP/HTTPS.

5.2.2 Identificación de Actores del Sistema

El sistema está compuesto principalmente por 5 actores que interactúan con el sistema. Los actores identificados son:

5.2.2.1 Usuario

Se corresponde a cualquier usuario que interacciones de alguna manera con el sistema. Pueden iniciar sesión, registrarse, subir documentos y descargarlos.

5.2.2.1.1 Usuario no autenticado

Se corresponde a cualquier usuario que no ha iniciado sesión en una cuenta, pudiendo solo interactuar con el usuario para iniciar sesión o registrarse.

5.2.2.1.2 Usuario autenticado

Se corresponde a cualquier usuario que ha iniciado sesión en una cuenta. Este actor podrá subir documentos, visualizarlos en la aplicación web y descargarlos.

5.2.2.2 Administrador del sistema

Se trata del administrador de la base de datos, encargado de resolver problemas de los usuarios. También, es el encargado de salvaguardar y modificar la clave privada de encriptación.

5.2.2.3 IPFS

Es el sistema distribuido de almacenaje ya definido en el apartado 3.2.1.

5.2.2.4 Red Blockchain Ethereum Rinkeby

Es la red Blockchain con la que interactuará el sistema, ya definida en el apartado 3.1.1.

5.2.2.5 Base de datos

Es la base de datos donde se almacenarán los datos de la aplicación web.

5.2.2.6 Neodoc

Es la plataforma de la empresa Neosystems, definida en el apartado 1.1, en la que se integrará parte de este sistema.

5.2.3 Especificación de Casos de Uso

Esta sección está destinada a la descripción de los casos de uso para mostrar la interacción entre los actores del sistema.

5.2.3.1 Casos de uso de usuario no autenticado

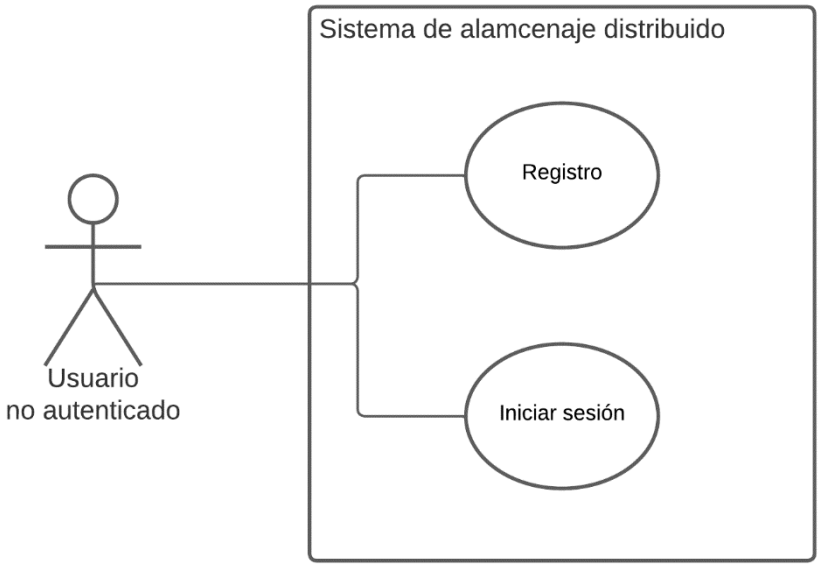

Figura 5.1 Caso de uso de usuario no autenticado

Nombre del Caso de Uso
Registro
Descripción
Un usuario no registrado en la aplicación podrá crearse una cuenta nueva. Para ello, deberá indicar un nombre de usuario y una contraseña asociada.

Tabla 5.1 Caso de uso Registro

Nombre del Caso de Uso
Iniciar sesión
Descripción
Un usuario registrado no autenticado podrá iniciar sesión en una cuenta existente introduciendo un nombre de usuario y su contraseña asociada.

Tabla 5.2 Caso de uso Iniciar Sesión

5.2.3.2 Casos de uso usuario autenticado

A continuación, se define un diagrama de contexto general del sistema con relación al actor usuario autenticado exponiendo todas las funcionales para más adelante explorar cada caso de uso específico.

5.2.3.2.1 Diagrama de contexto

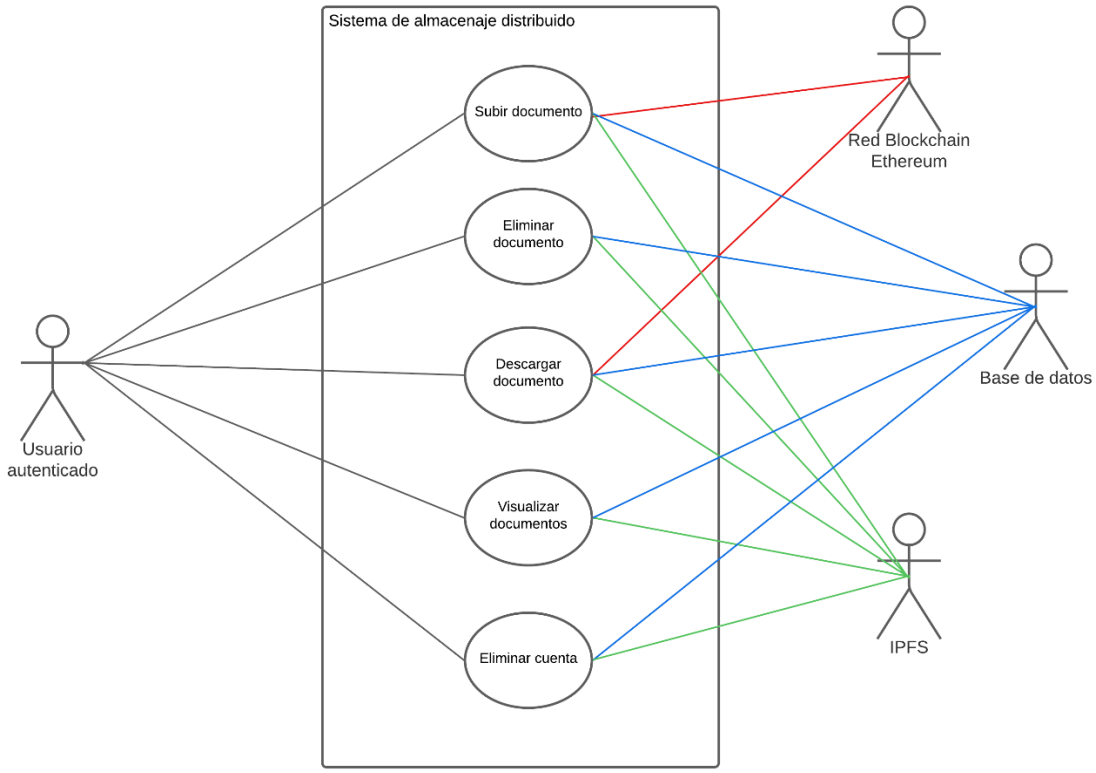

Figura 5.2 Diagrama de contexto de usuario autenticado

5.2.3.2.2 Caso de uso "Subir documento"

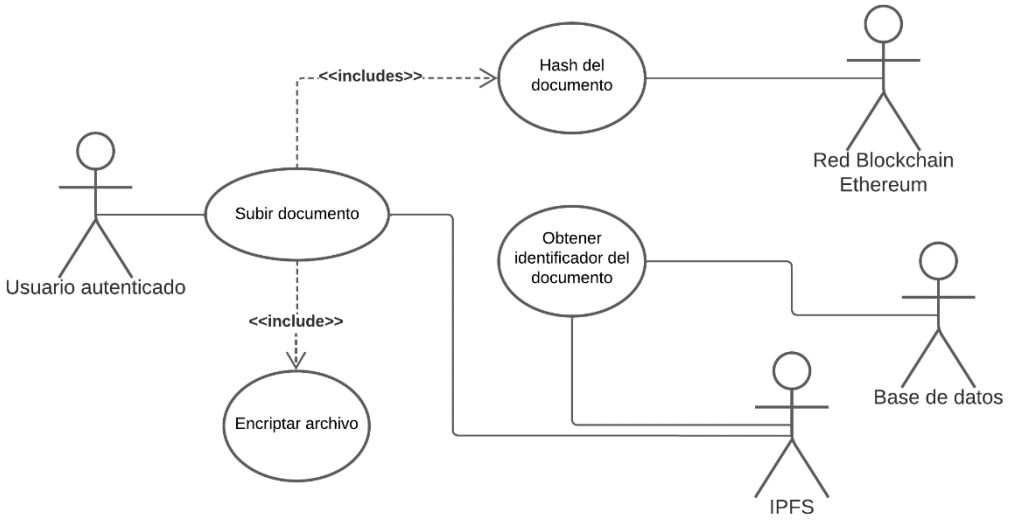

Figura 5.3 Caso de uso de Subir documento

Nombre del Caso de Uso
Subir documento
Descripción
El usuario sube un documento al sistema con la finalidad de que se almacene de forma distribuida y segura. Para ello, el sistema obtiene el *hash* del documento para comprobar su integridad a la hora de la descarga en el caso de uso 0. Por otro lado, el archivo ha de encriptarse previa la subida a IPFS. Por último, se obtiene el identificador del documento en IPFS para guardarlo en la base de datos.

Tabla 5.3 Caso de uso Subir documento

5.2.3.2.3 Caso de uso "Eliminar documento"

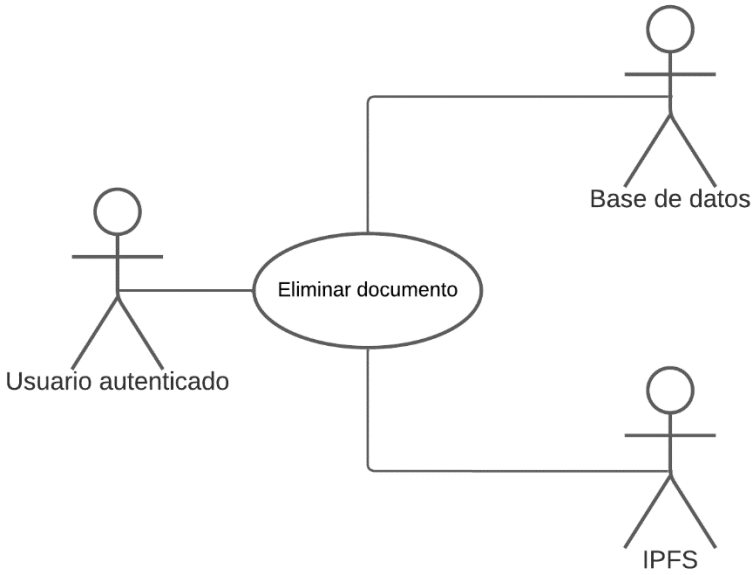

Figura 5.4 Caso de uso Eliminar documento

Nombre del Caso de Uso
Eliminar documento
Descripción
El usuario quiere borrar un documento, por lo que lo elimina de la base de datos y de la plataforma IPFS.

Tabla 5.4 Caso de uso Eliminar documento

5.2.3.2.4 Caso de uso "Descargar documento"

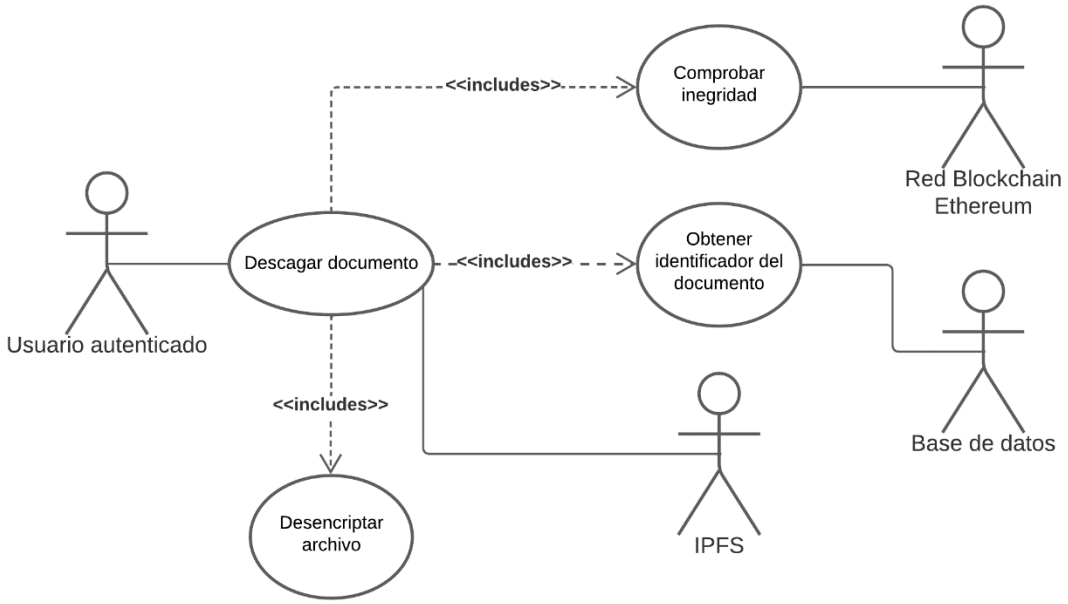

Figura 5.5 Caso de uso de descargar un documento

Nombre del Caso de Uso
Descargar documento
Descripción
El usuario quiere descargar un archivo. Para ello, el sistema deberá obtener de la base de datos el identificador de archivo y buscar con ese ID en IPFS el documento. Una vez obtenido, el sistema desencriptará el archivo y calculará el *hash* de su contenido con la finalidad de comprobar la integridad de este usando la red Blockchain Ethereum. Finalmente, el usuario obtendrá la descarga del documento.

Tabla 5.5 Caso de uso Descargar documento

5.2.3.2.5 Caso de uso "Visualizar documentos"

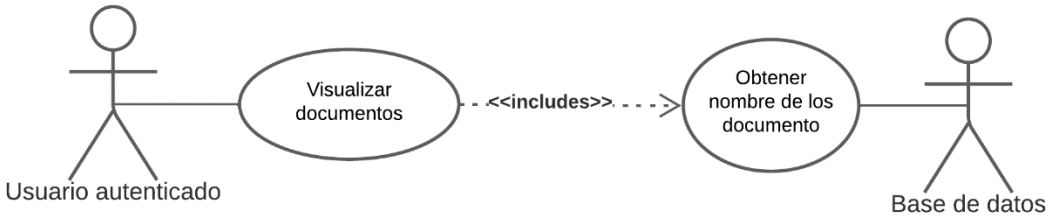

Figura 5.6 Caso de uso de visualizar los documentos del usuario

Nombre del Caso de Uso
Visualizar documentos
Descripción
El usuario quiere consultar los archivos dentro del sistema, por lo que se carga desde la base de datos el nombre de los documentos a la cuenta asociada del usuario para que pueda visualizarlos.

Tabla 5.6 Caso de uso Visualizar documentos

5.2.3.2.6 Caso de uso "Eliminar cuenta"

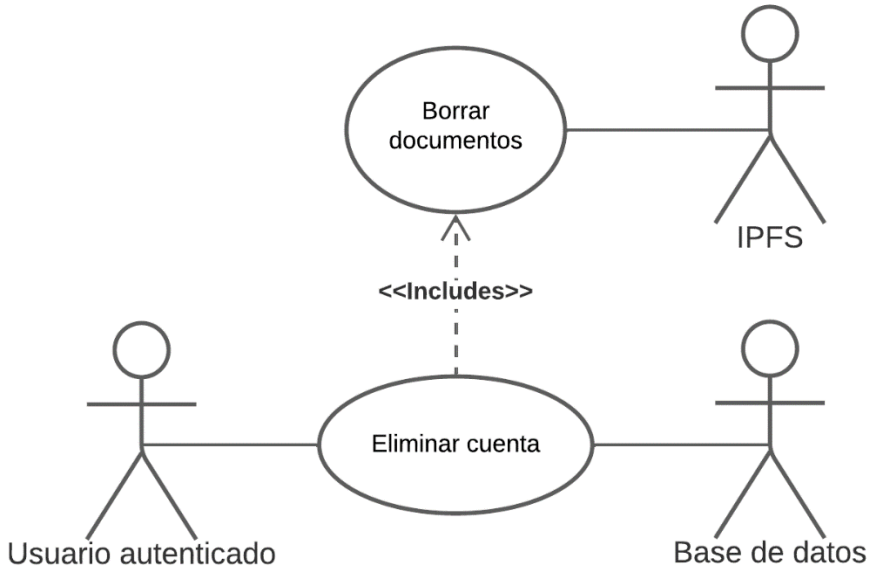

Figura 5.7 Caso de uso de eliminar cuenta

Nombre del Caso de Uso
Eliminar cuenta
Descripción
El usuario quiere borrar su cuenta por lo que se eliminan todos los datos de la base de datos junto con los documentos en IPFS.

Tabla 5.7 Caso de uso Eliminar cuenta

5.2.3.3 Caso de uso de administrador de sistema

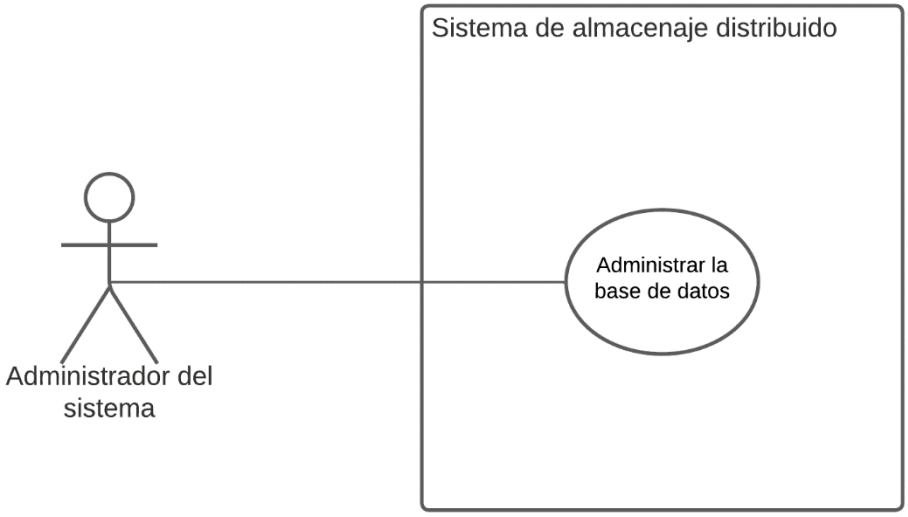

Figura 5.8 Caso de uso administrador del sistema

Nombre del Caso de Uso
Administrar base de datos
Descripción
El administrador obtendrá las credenciales de la base de datos para cualquier conflicto que haya que resolver o modificación a realizar de cualquier tipo.

Tabla 5.8 Caso deuso Administrar base de datos

5.2.3.4 Casos de uso Neodoc

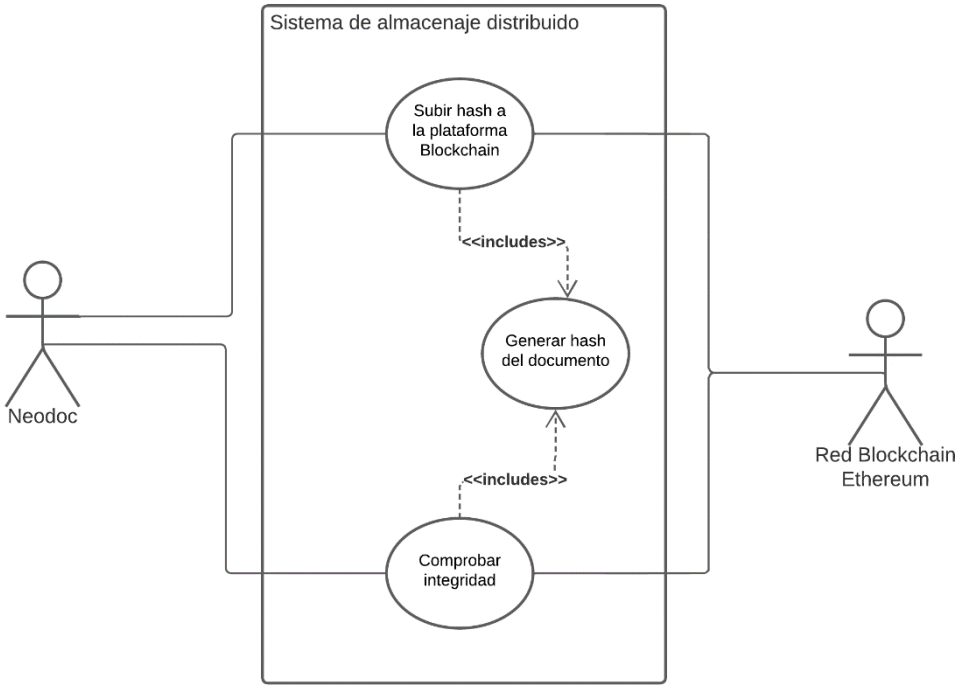

Figura 5.9 Casos de uso Neodoc

Nombre del Caso de Uso
Subir *hash* a la red Blockchain
Descripción
Neodoc se comunicará con el sistema desarrollado para que se almacene en la red Blockchain el *hash* del documento del que se quiere tener un seguimiento del documento y comprobar su integridad cuando se quiera.

Tabla 5.9 Caso de uso Subir hash a la Blockchain

Nombre del Caso de Uso
Comprobar integridad
Descripción
Neodoc se comunicará con el sistema desarrollado enviando un documento del que se quiere comprobar su integridad, por lo que se genera su *hash* y se comprueba si existe dentro de la red Blockchain.

Tabla 5.10 Caso de uso Comprobar integridad

5.3 Identificación de los Subsistemas en la Fase de Análisis

Esta sección trata de analizar el sistema y descomponerlo en subsistemas de tal manera que se facilite su análisis posterior.

5.3.1 Descripción de los Subsistemas

A continuación, se especifican los subsistemas identificados:

5.3.1.1 Cliente

El subsistema de la parte cliente engloba todos los archivos HTML, CSS y JavaScript relativos a la parte *frontend* del sistema que se comunicará con el *backend*, es decir, el 5.3.1.2 Servidor API. Además, este subsistema conforma la interfaz con la que el usuario final interactuará.

5.3.1.2 Servidor

El subsistema servidor está compuesto por una API que conforma una parte fundamental del sistema. Conforma la comunicación entre el cliente y el resto de los subsistemas que forman el conjunto global de la aplicación. Por un lado, realiza toda la conexión con la base de datos para manejar a los usuarios y por otro, se comunica con las interfaces externas de IPFS y de la red Blockchain Ethereum.

5.3.1.3 SmartContract

Este subsistema conforma la implementación, configuración y despliegue del contrato inteligente o *SmartContract* para interaccionar con la red Blockchain. Este contrato inteligente especificará como se interactuará con la red y que datos salvaguardará.

5.3.1.4 Base de datos

Este subsistema conforma el almacenaje de los datos de la aplicación dentro de la base de datos, en concreto las credenciales de usuario y el identificador de los documentos asociados a esa cuenta de usuario. La comunicación con la base de datos y manejo de los datos se realiza desde la parte servidor de la aplicación.

5.3.1.5 Encriptación

Este subsistema engloba los dos algoritmos de cifrado existentes dentro del proyecto. Primero, la encriptación de un documento aplicando el algoritmo de clave privada AES-256. Segundo, la obtención del *hash* de un documento aplicando el algoritmo SHA-256 en el mismo archivo.

5.3.2 Descripción de los Interfaces entre Subsistemas

Una vez identificados los subsistemas, se definen a continuación los medios de comunicación entre los distintos subsistemas.

El cliente se comunicará exclusivamente con el servidor a través de peticiones a una API utilizando el protocolo seguro HTTPS. Además, este mismo servidor establecerá comunicaciones con las interfaces externas de IPFS y de la red Blockchain utilizando el mismo protocolo.

El subsistema *SmartContract* se comunicará con la red Blockchain a través del protocolo HTPP/HTPPS.

La comunicación de la base de datos se realiza a través del protocolo HTTP/HTTPS de manera local, ya que la base de datos es *on-premise*.

Por último, la comunicación entre el subsistema de encriptación y el servidor se realiza de manera local, ya que ambos subsistemas se ejecutan en la misma máquina.

5.4 Diagrama de Clases Preliminar del Análisis

Esta sección concreta tiene como finalidad la creación de un diagrama de clases aproximado del sistema, además de una descripción de todas las clases presentes.

5.4.1 Diagrama de Clases

A continuación, se muestra un diagrama de clases preliminar del análisis, por lo que puede no corresponderse totalmente con el del sistema final.

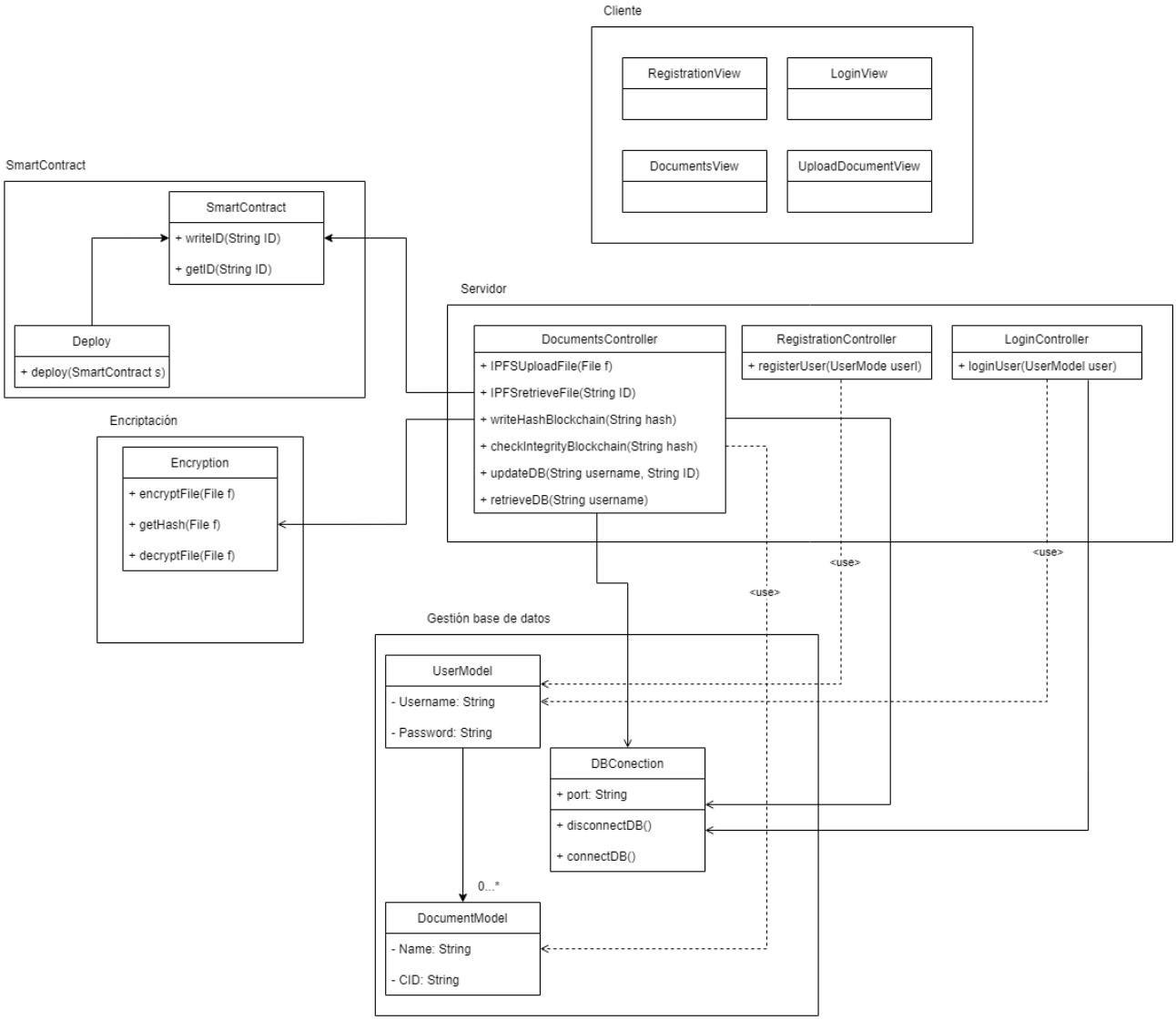

Figura 5.10 Diagrama de clases preliminar del análisis

5.4.2 Descripción de las Clases

Siguiendo la especificación de subsistemas identificados previamente en el apartado 5.3.1, se procederá a describir las clases identificadas en el apartado 5.4.1.

5.4.2.1 Cliente

Nombre de la Clase
RegistrationView
Descripción
Se trata de la vista del registro de usuario.
Responsabilidades
Mostrar al usuario la pantalla de registro para que pueda interaccionar con ella.
Atributos Propuestos
-
Métodos Propuestos
-

Tabla 5.11 Clase RegistrationView

Nombre de la Clase
LoginView
Descripción
Se trata de la vista de inicio de sesión de usuario.
Responsabilidades
Mostrar al usuario la pantalla de inicio de sesión para que pueda interaccionar con ella.
Atributos Propuestos
-
Métodos Propuestos
-

Tabla 5.12 Clase LoginView

Nombre de la Clase
UploadDocumentView
Descripción
Se trata de la vista de subida de un documento para el usuario.
Responsabilidades
Mostrar al usuario la pantalla de subir un documento.
Atributos Propuestos
-
Métodos Propuestos
-

Tabla 5.13 Clase UploadDocumentView

Nombre de la Clase
DocumentsView
Descripción
Se trata de la vista general de todos los documentos del usuario.
Responsabilidades

Mostrar al usuario la pantalla de descarga y visualización de sus documentos.
Atributos Propuestos
-
Métodos Propuestos
-

Tabla 5.14 Clase DocumentsView

5.4.2.2 Servidor

Nombre de la Clase
DocumentsController
Descripción
Esta clase está destinada a albergar todos los métodos relativos a la comunicación con las interfaces externas de IPFS, de la red Blockchain y comunicación con las vistas. Además, también maneja la base de datos.
Responsabilidades
Subir un documento a IPFS. Descargar documentos de IPFS. Añadir *hash* a la red Blockchain. Comprobar la integridad de un *hash* en la red Blockchain. Añadir el ID de un documento a la base de datos. Conseguir los IDs de los documentos de un usuario. Comunicarse con las vistas y mostrar la información de los documentos.
Atributos Propuestos
-
Métodos Propuestos
IPFSUpload: permite subir un archivo a IPFS. **IPFSRetrieve:** permite descargar un archivo de IPFS. **WriteHashBlockchain:** permite escribir el *hash* de un documento dentro de la red Blockchain. **CheckIntegrityBlockchain:** permite comprobar la integridad de un documento mediante la red Blockchain. **UpdateDB:** produce un cambio en la base de datos. **RetrieveDB:** permite coger datos de la base de datos.

Tabla 5.15 Clase DocumentsController

Nombre de la Clase
LoginController
Descripción
Es el controlador de la parte de inicio de sesión para el inicio de sesión de usuarios.
Responsabilidades
Mantener la lógica relativa al inicio de sesión del sistema encapsulada.
Atributos Propuestos
-
Métodos Propuestos
LoginUser: inicio de sesión de un usuario.

Tabla 5.16 Clase LoginController

Nombre de la Clase
RegistrationController
Descripción
Es el controlador de la parte de registro de un usuario.
Responsabilidades
Mantener la lógica relativa al registro del sistema encapsulada.
Atributos Propuestos
-
Métodos Propuestos
RegisterUser: registro de un usuario.

Tabla 5.17 Clase RegistrationController

5.4.2.3 SmartContract

Nombre de la Clase
SmartContract
Descripción
Se trata del código ejecutado dentro de la red Blockchain.
Responsabilidades
Almacenar el *hash* de los documentos almacenados y permitir comprobar si un *hash* existe dentro de la red o no.
Atributos Propuestos
-
Métodos Propuestos
WriteMap: escribe en una estructura de datos *map* el *hash* del documento. **GetCID:** obtiene si el *hash* del documento existe o no dentro de la red.

Tabla 5.18 Clase SmartContract

Nombre de la Clase
Deploy
Descripción
Se trata del despliegue del SmartContract.
Responsabilidades
Desplegar el contrato inteligente en la red Blockchain.
Atributos Propuestos
-
Métodos Propuestos
Deploy: despliega en la red Blockchain el SmartContract.

Tabla 5.19 Clase Deploy

5.4.2.4 Base de datos

Nombre de la Clase
DBConnection
Descripción
Esta clase maneja las conexione con la base de datos.
Responsabilidades
Establecer la conexión y desconexión con la base de datos.

Atributos Propuestos
Port: indica el puerto en el que se realizará conexión con la base de datos.
Métodos Propuestos
ConnectDB: establece la conexión con la base de datos. **DisconnectDB:** para la conexión con la base de datos.

Tabla 5.20 Clase DBConnection

Nombre de la Clase
UserModel
Descripción
Se trata de un modelo de abstracción de un usuario dentro del sistema.
Responsabilidades
Permite tratar con usuarios en otras clases con facilidad.
Atributos Propuestos
Username: el nombre de usuario. **Password:** la contraseña para acceder a la cuenta. **Documents:** lista de documentos.
Métodos Propuestos
-

Tabla 5.21 Clase UserModel

Nombre de la Clase
DocumentModel
Descripción
Se trata de un modelo de abstracción de un documento dentro del sistema.
Responsabilidades
Permite tratar con documentos con facilidad en otras clases.
Atributos Propuestos
Name: el nombre de usuario. **CID:** la contraseña para acceder a la cuenta. **User:** usuario al que pertenece el documento.
Métodos Propuestos
-

Tabla 5.22 Clase DocumentModel

5.4.2.5 Encriptación

Nombre de la Clase
Encryption
Descripción
Se trata de la clase encargada de todas las operaciones relacionado con la encriptación de archivos del proyecto.
Responsabilidades
Encriptar y desencriptar archivos. Generar el *hash* de un documento.
Atributos Propuestos
-
Métodos Propuestos
EncryptFile: encripta un archivo.

DecryptFile: desencripta un archivo. **GetHash**: genera el *hash* de un archivo.

Tabla 5.23 Clase Encryption

5.5 Análisis de Casos de Uso y Escenarios

En esta sección se detallan en escenarios los casos de uso identificados en el apartado 0

5.5.1 Caso de uso registro de usuario

Registro de usuario	
Precondiciones	-
Postcondiciones	Existirá dentro de la base de datos un nuevo usuario único válido con los datos especificados por el usuario no autenticado.
Actores	Iniciado por el usuario no autenticado y finalizado por el usuario autenticado.
Descripción	1. El usuario accede a la pantalla de registro. 2. Indica un nombre de usuario y una contraseña. 3. El sistema procederá a validar que los datos son válidos. 4. El usuario ya autenticado se le redirigirá la pantalla de documentos.

Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el nombre de usuario introducido por el usuario no autenticado ya existe dentro de la base de datos.
	  ○ El usuario deberá volver al paso 2 para indicar de nuevo los datos.
	• **Escenario alternativo 2:** la contraseña introducida por el usuario no tiene el formato correcto.
	  ○ El usuario deberá volver al paso 2 para indicar de nuevo los datos.
	• **Escenario alternativo 3:** el usuario no ha introducido todos los datos obligatorios.
	  ○ El usuario deberá volver al paso 2 para indicar de nuevo los datos.

Excepciones	• **La base de datos no está disponible:** no se permite el registro debido a que la base de datos no está en funcionamiento. ○ Se notifica del error indicando que se intente de nuevo más adelante.
Notas	-

Tabla 5.24 Escenario de caso de uso registro de usuario

5.5.2 Caso de uso inicio de sesión

Iniciar sesión	
Precondiciones	El usuario debe tener ya una cuenta registrada pero no estar autenticado.
Postcondiciones	El usuario pasa a estar autenticado en la aplicación.
Actores	Iniciado por el usuario no autenticado y terminado por el usuario autenticado.
Descripción	1. El usuario accede a la pantalla de inicio de sesión. 2. El usuario indica el nombre de usuario de la cuenta y la contraseña. 3. El sistema procederá a validar que los datos son correctos. 4. El usuario es redirigido a la pantalla de documentos.

Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario no introduce todos los datos obligatorios.
	  ○ El usuario no autenticado deberá volver al paso 2.
	• **Escenario alternativo 2:** los datos introducidos por el usuario no concuerdan con un usuario existente en la base de datos.
	  ○ El usuario no autenticado deberá volver al paso 2.

Excepciones	• **La base de datos no está disponible:** no se permite el registro debido a que la base de datos no está en funcionamiento. ○ Se notifica del error indicando que se intente de nuevo más adelante.
Notas	-

Tabla 5.25 Escenario de caso de uso inicio de sesión

5.5.3 Caso de uso subir documento

Subir un documento	
Precondiciones	El usuario deberá estar autenticado en la aplicación.
Postcondiciones	El documento deberá estar subido en la plataforma IPFS, guardado su ID en la base de datos en la cuenta de usuario asociada y el *hash* del documento dentro de la red Blockchain.
Actores	Usuario autenticado en la aplicación.
Descripción	1. El sistema muestra la pantalla de subir documentos. 2. El usuario selecciona que archivo quiere subir. 3. El usuario pulsa el botón de subir archivo. 4. El sistema comprueba que el documento es de un formato correcto. 5. El sistema genera el *hash* del archivo. 6. El sistema encripta el archivo. 7. El archivo es subido a IPFS. 8. Se realiza una transacción para introducir en la red Blockchain

	Ethereum el *hash* generado por el documento. 9. El ID del documento retornado por IPFS es guardado en la base de datos a la cuenta asociada. 10. Se redirigirá al usuario a la pantalla de documentos.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el documento seleccionado no está soportado por el sistema ya que el sistema solo permite una serie de extensiones. ○ Se notifica el error al usuario y deberá volver al paso 2. • **Escenario alternativo 2:** el usuario se equivoca de documento a subir. El usuario podrá volver al paso 2 para escoger otro documento.

Excepciones	• **La red Ethereum Rinkeby está congestionada:** la red Blockchain podría estar soportando una gran cantidad de transacciones al mismo tiempo lo que podría demorar o incluso rechazar la transacción.
	  ○ Se notifica el error al usuario y deberá volver al paso 2.
	• **La plataforma IPFS no está disponible:** los nodos de la red IPFS no se encuentran disponible por mantenimiento.
	  ○ Se notifica el error y se indica que se intente en otro momento.

Notas	-

Tabla 5.26 Escenario de caso de uso subir un documento

5.5.4 Caso de uso eliminar documento

Eliminar un documento	
Precondiciones	Haber subido un documento previamente y que exista al momento de eliminarlo.
Postcondiciones	El documento es eliminado de la base del sistema completamente, tanto de IPFS como de la base de datos.
Actores	Usuario ya autenticado.
Descripción	1. El sistema muestra la pantalla de los documentos. 2. El usuario selecciona el documento que quiere que sea eliminado. 3. El usuario clica en el botón de “eliminar”. 4. El sistema se comunica con IPFS indicándole que se quiere eliminar el documento con el ID del documento seleccionado por el usuario. 5. El sistema elimina de la base de datos el ID del documento seleccionado.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario se equivoca al seleccionar el archivo a ser eliminado. El usuario cancela la acción y regresa al paso 2.

Excepciones	• **La plataforma IPFS no está disponible:** los nodos de la red IPFS no se encuentran disponible por mantenimiento. ○ Se notifica el error y se indica que se intente en otro momento.
Notas	-

Tabla 5.27 Escenario de caso de uso eliminar un documento

5.5.5 Caso de uso descargar documento

Descargar un documento	
Precondiciones	Debe haberse subido un documento al sistema y existir a la hora de su descarga.
Postcondiciones	El usuario tendrá descargado el documento seleccionado.
Actores	El usuario autenticado.
Descripción	1. El sistema muestra la pantalla de los documentos. 2. El usuario selecciona uno de sus documentos entre los mostrados. 3. El usuario confirma clicando en un botón que quiere descargarlo. 4. El sistema coge el ID del documento de la base de datos. 5. Busca en IPFS con ese ID el documento y lo descarga. 6. El sistema desencripta el archivo. 7. El sistema genera el *hash* del archivo. 8. Se busca dentro de la red Blockchain que ese *hash* existe. 9. Se produce la descarga para el usuario.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario se equivoca al seleccionar el archivo a ser descargado. El usuario cancela la acción y regresa al paso 2. • **Escenario alternativo 2:** el sistema comprueba el *hash* del archivo descargado de IPFS y resulta que no existe dentro de la red Blockchain. ○ Se procede a la descarga del documento para el usuario, pero se le indica que el documento ha sido alterado de alguna manera y que no es íntegro.
Excepciones	• **La plataforma IPFS no está disponible:** los nodos de la red IPFS no se encuentran disponible por mantenimiento. ○ Se notifica el error y se indica que se intente en otro momento.
Notas	-

Tabla 5.28 Escenario de caso de uso descargar un documento

5.5.6 Caso de uso visualizar documentos

Visualizar documentos	
Precondiciones	El usuario ha de estar autenticado.
Postcondiciones	-
Actores	Usuario autenticado.
Descripción	1. El sistema muestra la pantalla de documentos. 2. Se carga de la base de datos el nombre de los documentos. 3. El usuario podrá visualizar el nombre sus documentos.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario no ha subido ningún documento previamente por lo que se muestra una pantalla vacía.
Excepciones	• **La base de datos no está disponible:** no se permite el registro debido a que la base de datos no está en funcionamiento. ○ Se notifica del error indicando que se intente de nuevo más adelante.
Notas	-

Tabla 5.29 Escenario de caso de uso visualizar un documento

5.5.7 Caso de uso eliminar cuenta

Eliminar cuenta	
Precondiciones	El usuario debe estar autenticado en el sistema.
Postcondiciones	Se eliminará la cuenta de usuario con toda su información asociada de la base de datos.
Actores	Usuario autenticado.
Descripción	1. Se muestra al usuario la pantalla de documentos. 2. Selecciona el botón de eliminar cuenta y confirma. 3. El sistema elimina de IPFS todos los documentos asociados a la cuenta. 4. El sistema elimina de la base de datos toda la información asociada a la cuenta.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario se arrepiente de haber borrado la cuenta. Deberá volver a crearse una cuenta en el sistema. • **Escenario alternativo 2:** el usuario selecciona el botón de borrar cuenta, pero luego se arrepiente. Clica en cancelar en vez de confirmar el borrado y vuelve al paso 1.
Excepciones	• **La base de datos no está disponible:** no se permite el registro

	debido a que la base de datos no está en funcionamiento. ○ Se notifica del error indicando que se intente de nuevo más adelante.
Notas	-

Tabla 5.30 Escenario de caso de uso eliminar una cuenta

5.5.8 Caso de uso administrar la base de datos

Administrar la base de datos	
Precondiciones	El administrador debe tener las credenciales para acceder a la base de datos.
Postcondiciones	-
Actores	Administrador del sistema
Descripción	1. El administrador accede con las credenciales a la base de datos. 2. Realiza las operaciones oportunas para resolver conflictos o introducir usuarios nuevos… 3. Cierra la conexión la base de datos.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el administrador no tiene acceso a la base de datos. Deberá pedir acceso al desarrollador o la empresa desarrolladora del sistema.
Excepciones	• **El administrador se equivoca:** realiza una operación que corrompe la base de datos.
Notas	-

Tabla 5.31 Escenario de caso de uso administrar la base de datos

5.5.9 Caso de uso Neodoc subir hash a la Blockchain

Subir *hash* a la red Blockchain	
Precondiciones	El sistema debe estar integrado dentro de la plataforma Neodoc y el usuario debe estar autenticado en esa plataforma.
Postcondiciones	-
Actores	Neodoc y usuario autenticado en Neodoc
Descripción	1. Se muestra la pantalla de la plataforma Neodoc. 2. El usuario autenticado selecciona un archivo del que quiere que se haga un seguimiento. 3. Se envía le archivo al sistema. 4. Se genera le *hash* del archivo. 5. Se genera una transacción en a la red Blockchain y se guarda el *hash* del documento en ella. 6. Se elimina el documento del sistema, ya que solo se necesita

	para generar el *hash*.
Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el usuario se equivoca al seleccionar el archivo a ser subido. El usuario cancela la acción y regresa al paso 2.
Excepciones	• **La red Ethereum Rinkeby está congestionada:** la red Blockchain podría estar soportando una gran cantidad de transacciones al mismo tiempo lo que podría demorar o incluso rechazar la transacción. ○ Se notifica el error al usuario y deberá volver al paso 2.
Notas	-

Tabla 5.32 Caso de uso Neodoc subir hash a la Blockchain

5.5.10 Caso de uso Neodoc comprobar integridad

Comprobar integridad	
Precondiciones	El sistema debe estar integrado dentro de la plataforma Neodoc y el usuario debe estar autenticado en esa plataforma.
Postcondiciones	-
Actores	Neodoc y usuario autenticado en Neodoc.
Descripción	1. Se muestra la pantalla de la plataforma Neodoc. 2. El usuario autenticado selecciona el archivo que quiere comprobar su integridad. 3. El sistema genera el *hash* del documento. 4. El sistema comprueba si el *hash* existe dentro de la red Blockchain.

Variaciones (escenarios secundarios)	• **Escenario alternativo 1:** el sistema comprueba el *hash* del archivo y resulta que no existe dentro de la red Blockchain.
	  ○ Se le indica al usuario que el documento ha sido alterado de alguna manera y que no es íntegro.
	• **Escenario alternativo 2:** el usuario selecciona un archivo que nunca ha sido subido su *hash* a la red.
	  ○ Se le indica que nunca había sido subido a la red o se le impide realizar esta acción directamente.

Excepciones	• **La red Ethereum Rinkeby está congestionada:** la red Blockchain podría estar soportando una gran cantidad de transacciones al mismo tiempo lo que podría demorar o incluso rechazar la transacción. ○ Se notifica el error al usuario y deberá volver al paso 2.
Notas	-

Tabla 5.33 Caso de uso Neodoc comprobar integridad

5.6 Análisis de Interfaces de Usuario

Se ha diseñado una interfaz de usuario sencilla cuya finalidad es la de permitir al usuario interaccionar con el sistema desarrollado, de tal forma que la aplicación pueda usarse con una pequeña cantidad de clics.

Concretamente, existirán 4 pantallas dentro de la aplicación: pantalla de registro, pantalla de inicio de sesión, pantalla de documentos y pantalla de subir un documento. Para cada una de ellas se desarrollará un diseño concreto.

5.6.1 Descripción de la Interfaz

Para la creación del diseño de las interfaces siguientes se ha utilizado la aplicación dedicada a la creación de mockups Balsamiq.

5.6.1.1 Pantalla de registro de usuario

Esta pantalla está destinada para el registro de nuevos usuarios, de tal forma que creen una cuenta nueva en el sistema. En un registro sencillo en el que se piden nombre de usuario y contraseña. Además, permite redirigirse al inicio de sesión.

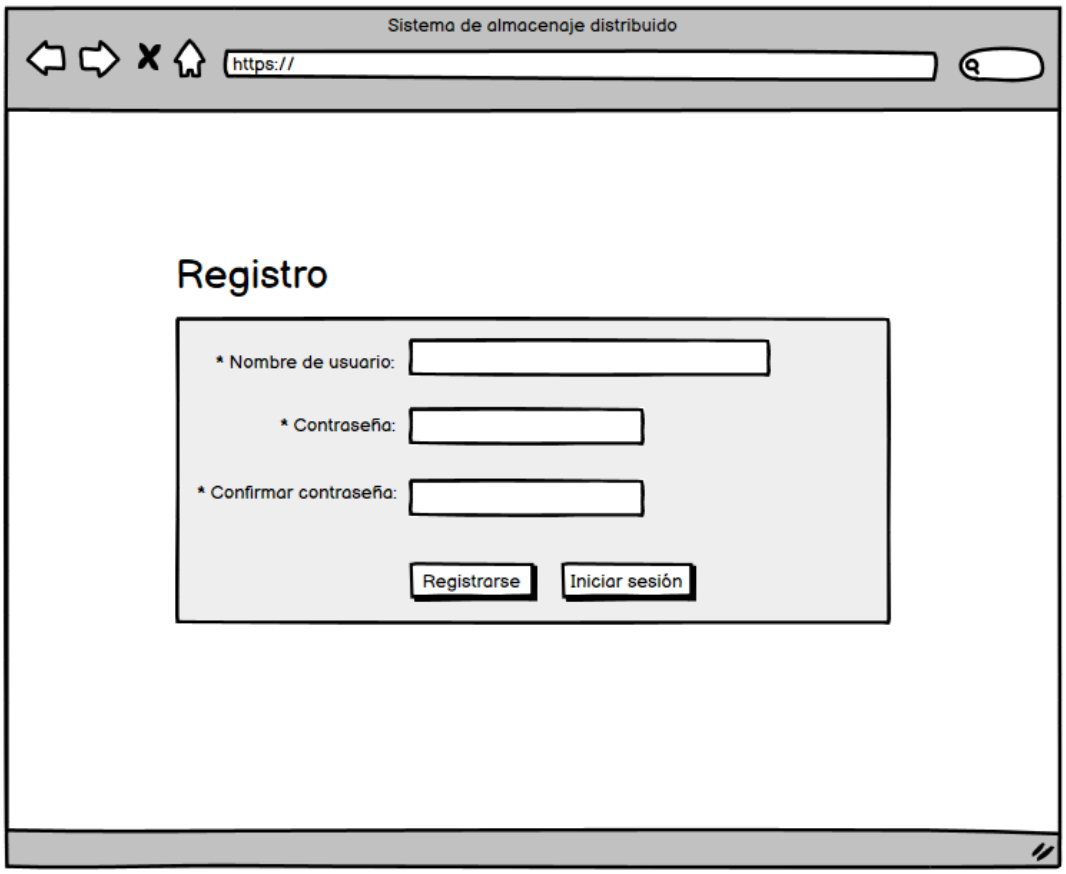

Figura 5.11 Pantalla de registro de usuario

5.6.1.2 Pantalla de inicio de sesión

Esta pantalla está destinada para el inicio de sesión en una cuenta por parte del usuario, de tal manera que se inicie la sesión. Es un inicio de sesión sencillo que pide dos datos, nombre y contraseña y que también permite redirigirse a la pantalla de registro.

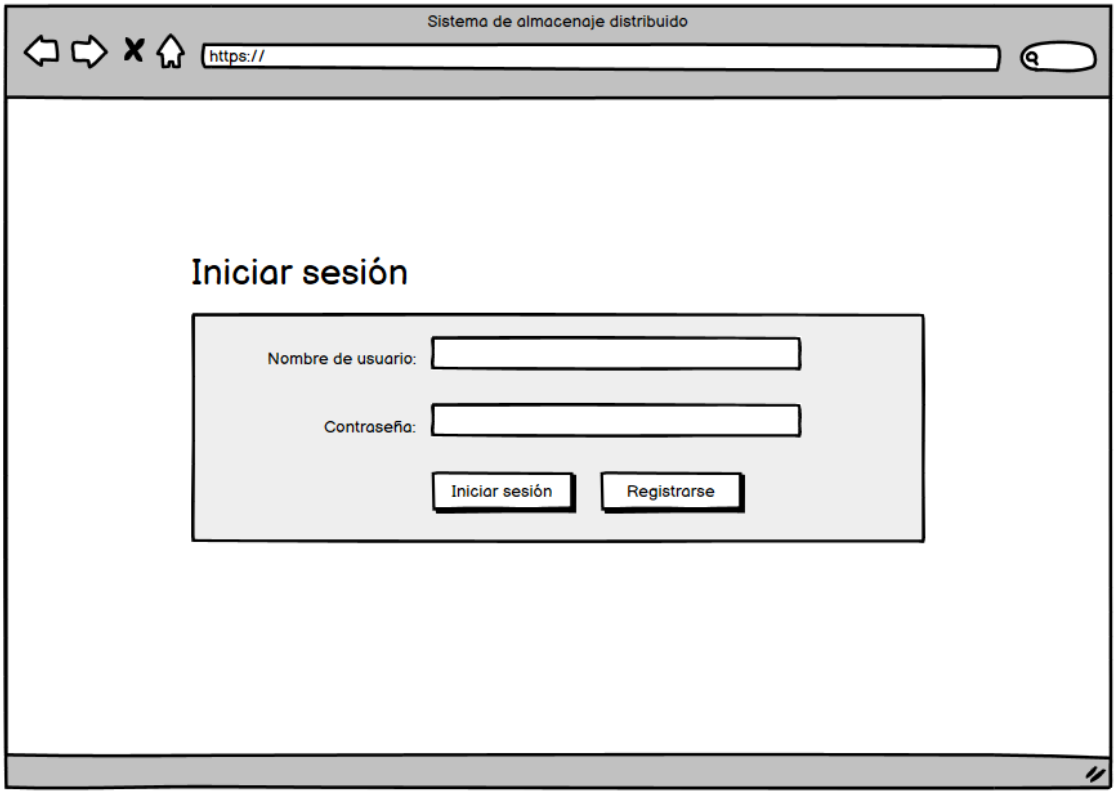

Figura 5.12 Pantalla de inicio de sesión

5.6.1.3 Pantalla de documentos

Esta pantalla está destinada para visualizar los documentos presentes en la cuenta donde se ha iniciado la sesión. Se muestran todos los documentos con sus respectivos nombres. Además, está presente un botón de eliminar cuenta y otro de cerrar sesión arriba a la derecha. Existe un *header* en el que se encuentra a la izquierda el logo, seguido de un botón que redirige a la pantalla subir un documento y un botón para eliminar el documento seleccionado actualmente, mediante un diálogo, y a la derecha un saludo al usuario con un icono de abstracción del usuario.

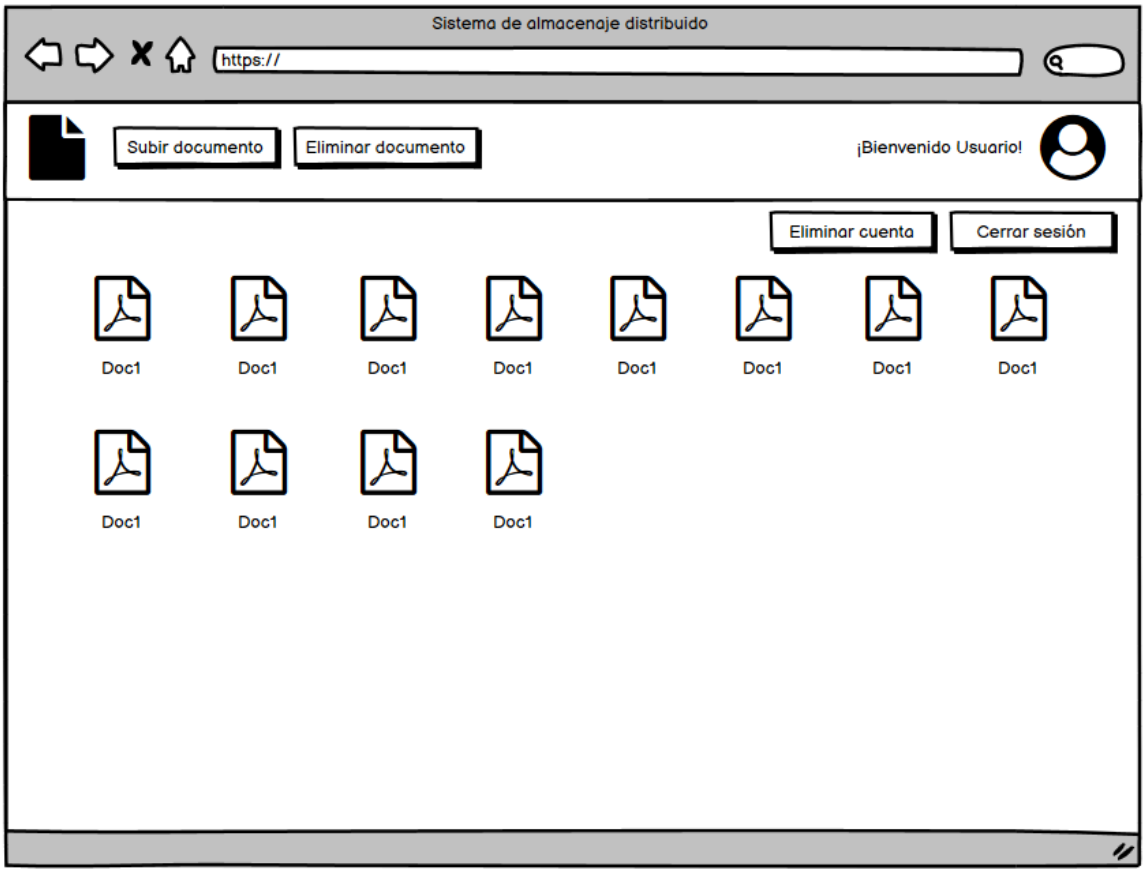

Figura 5.13 Pantalla de documentos

5.6.1.3.1 Botón Eliminar cuenta

Este botón abrirá el siguiente diálogo:

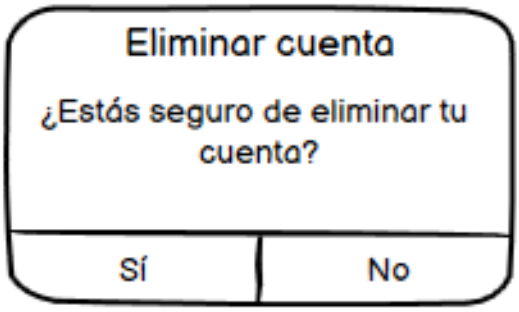

Figura 5.14 Pantalla de documentos

5.6.1.3.2 Botón Eliminar documento

Este botón abrirá el siguiente diálogo:

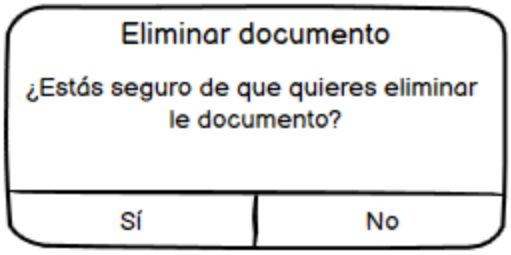

Figura 5.15 Diálogo eliminar documento

5.6.1.4 Pantalla de subir documento

Esta pantalla está destinada a ofrecer la funcionalidad de subir un documento al sistema para un usuario que ya inició sesión. Como se observa a continuación, hay dos maneras de adjuntar un archivo, arrastrándolo o clicando en un botón para abrir el explorar de archivo y seleccionar el archivo. Arriba a la izquierda está presente un botón Ver documentos que redirige a la pantalla de visualizar documentos.

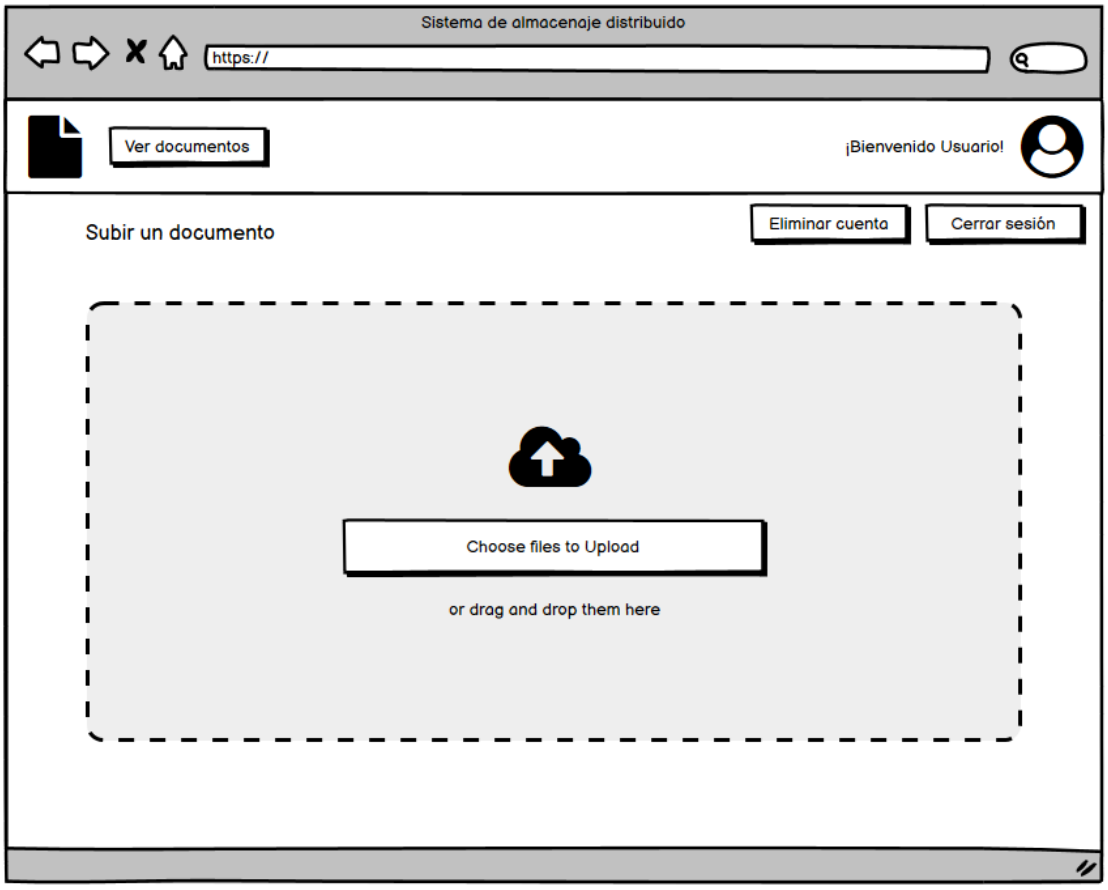

Figura 5.16 Pantalla subir un documento

5.6.2 Descripción de la interfaz Neodoc

A continuación, se muestran unas capturas del navegador donde se introdujo con el *Inspector* los componentes que conformarían la integración de este sistema dentro del producto.

5.6.2.1 Pantalla de visualización documentos

Se muestran los documentos del usuario con una serie de campos. Dentro del campo del nombre se muestra un escudo que representa que esos documentos, a petición del usuario y a través de la integración de este sistema se generó su *hash* y se guardó dentro de la red Blockchain. Cada vez que se carga la pantalla, se comprueba la integridad de los documentos. Se muestra un escudo si es íntegro y una exclamación, como también puede observarse en la siguiente figura, cuando se detecta una alteración el documento.

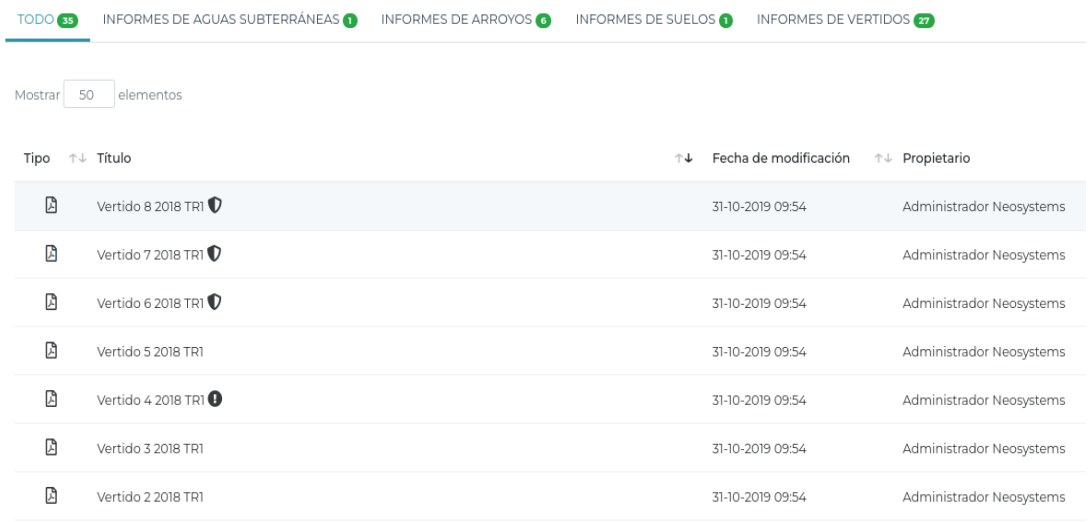

Figura 5.17 Pantalla visualización de documentos

En la parte de la derecha de la pantalla de la figura anterior, se puede abrir un menú de opciones en el que se encuentra la opción Integridad. Esta opción permite la operación de subir el *hash* a la red Blockchain de los archivos seleccionado comunicándose con el sistema desarrollado.

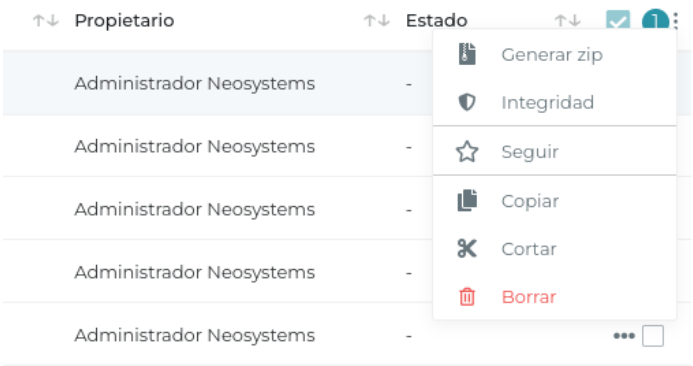

Figura 5.18 Pantalla visualización de documentos con desplegable de opciones

5.6.3 Descripción del Comportamiento de la Interfaz

Los errores de validación como longitud o tipo de dato introducido por el usuario serán identificados dentro del cliente para evitar carga de trabajo para el servidor. De esta manera, se prevén los errores en los datos en las etapas más tempranas. Sin embargo, identificar la unicidad del nombre de usuario o la existencia de una cuenta de usuario precisan de una consulta a la base de datos así que será necesario consultar al servidor y que el cliente reciba una respuesta con el error.

Por otro lado, los errores producidos por la navegación, como intentar acceder a una dirección Url no existente, utilizarán el código de error estandarizado de HTTP 404 *not found*.

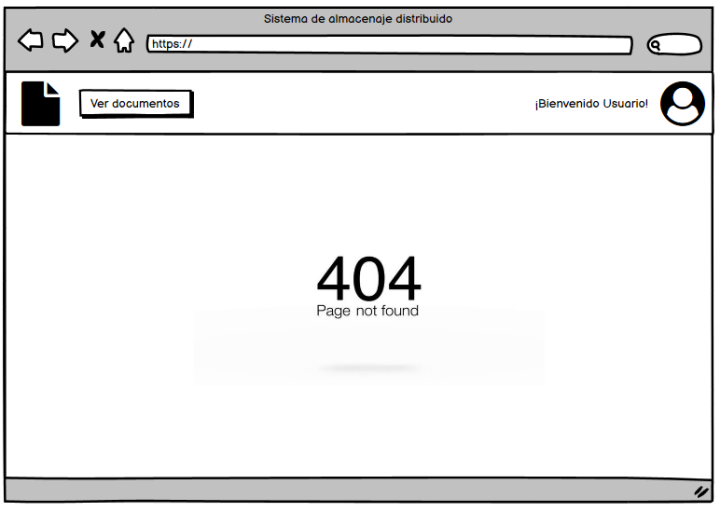

Figura 5.19 Figura de página no existente

Cuando el error sea producido por el servidor, se mostrará el código de error estándar HTTP 500 *internal server error*.

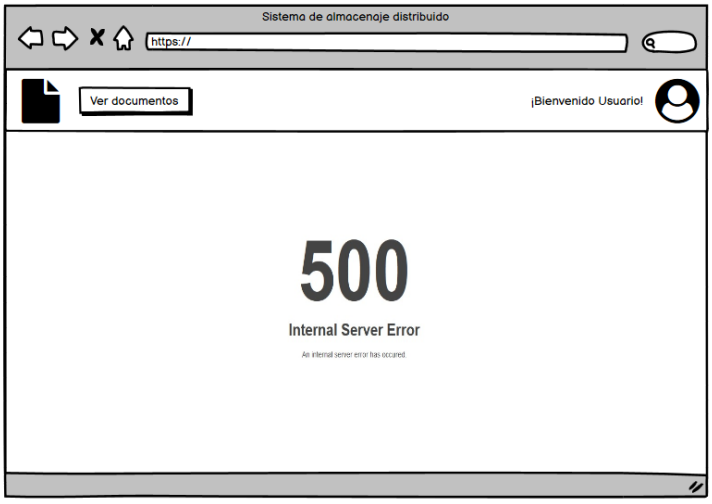

Figura 5.20 Figura de página no existente

5.6.4 Diagrama de Navegabilidad

En este apartado se mostrará un diagrama de navegabilidad de la aplicación. La idea es que la aplicación comience en la pantalla Inicio Sesión y de ahí comience la navegación entre las pantallas.

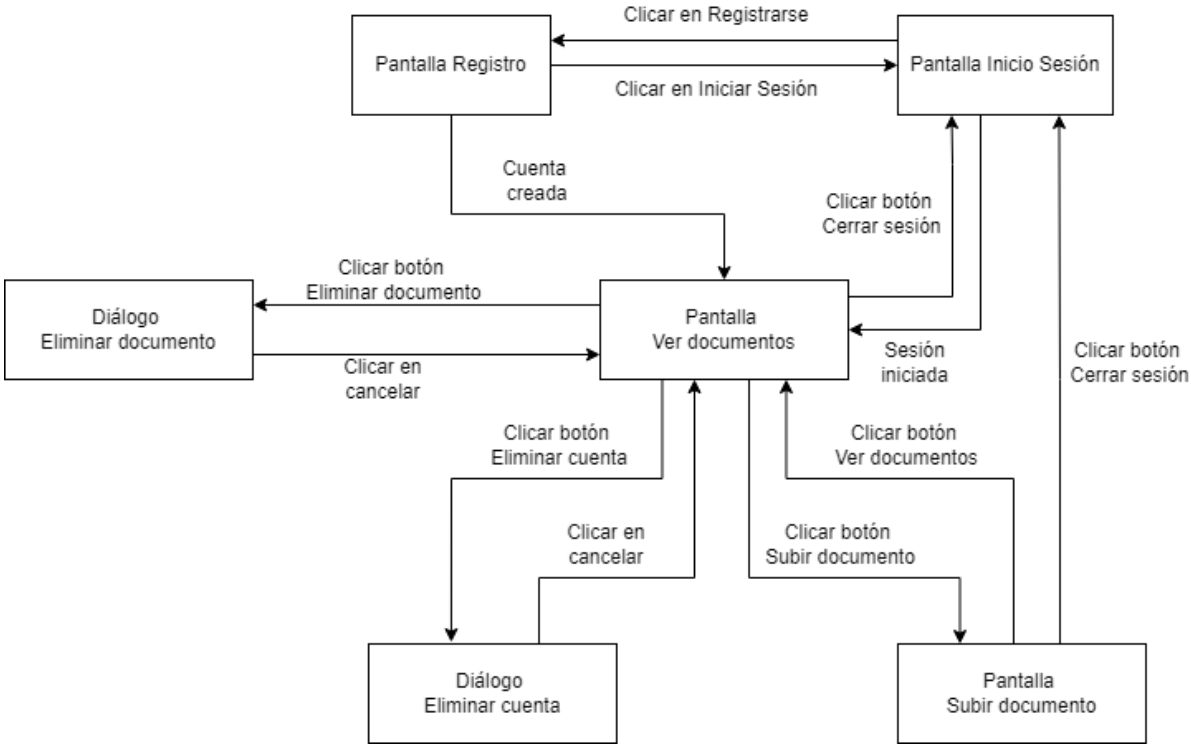

Figura 5.21 Figura diagrama de navegabilidad

5.7 Especificación del Plan de Pruebas

Esta sección consiste en la planificación de todas aquellas pruebas que ayuden a detectar errores, a comprobar el correcto funcionamiento de la aplicación web y de toda la lógica detrás de esta. Para idear las pruebas, es necesario comprender que se espera del sistema y como tiene que comportarse. Por lo tanto, se comprobará que la comunicación entre los sistemas es correcta, que las funciones del *SmartContract* funcionen correctamente, además de comprobar la usabilidad y accesibilidad de las interfaces.

5.7.1 Pruebas unitarias

Las pruebas unitarias consisten en un mecanismo para probar que las clases individuales cumplen con sus funciones correctamente. Estas pruebas irán enfocadas sobre todo a dos partes del proyecto, una para probar el *SmartContract* y la segunda serán realizadas sobre la API. También se realizarán sobre la generación de *hashes*.

Para la realización de estas pruebas se utilizarán las siguientes herramientas:

- Trufflejs para probar el *SmartContract*.
- PostMan para comprobar el correcto funcionamiento de la API.
- El framewok Mocha y Supertest para realizar pruebas unitarias sobre la API.
- Asserts para comprobar la generación de *hashes* y la encriptación.

5.7.1.1 Pruebas individuales según los casos de uso

Caso de Uso 1: Registro de usuario	
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario y contraseña correctos.	El sistema posee un usuario más.
Prueba	**Resultado Esperado**
Un usuario se intenta registrar, pero la contraseña tiene un formato incorrecto.	El sistema no posee un usuario más y se muestra un dialogo notificando el error.
Prueba	**Resultado Esperado**
Un usuario se intenta registrar, pero el usuario introducido ya existe dentro de la base de datos.	El sistema no posee un usuario más y se muestra un dialogo notificando el error.

Tabla 5.34 Caso de uso 1: Registro de usuario

Caso de Uso 2: Inicio de sesión	
Prueba	**Resultado Esperado**
Un usuario inicia sesión en una cuenta existente.	El sistema inicia sesión.
Prueba	**Resultado Esperado**
Un usuario intenta acceder a una cuenta, pero esta no existe.	El sistema no inicia sesión y notifica el problema.
Prueba	**Resultado Esperado**
Un usuario intenta iniciar sesión, pero no rellena un campo obligatorio, la contraseña.	El sistema no inicia sesión y notifica el problema.

Tabla 5.35 Caso de Uso 2: Inicio de sesión

Caso de Uso 3: Subir un documento	
Prueba	**Resultado Esperado**
Se sube un documento al sistema.	El sistema añade a la base de datos el identificador del documento.
Prueba	**Resultado Esperado**
El usuario selecciona un archivo de tipo no soportado por el sistema.	El sistema no sube el documento y notifica del problema.
Prueba	**Resultado Esperado**

El usuario se confunde y selecciona y archivo erróneo. Cancela la acción.	No sucede nada en el sistema.

Tabla 5.36 Caso de Uso 3: Subir un documento

Caso de Uso 4: Eliminar un documento	
Prueba	**Resultado Esperado**
El documento se elimina correctamente.	El sistema elimina de la base de datos los datos sobre ese documento.
Prueba	**Resultado Esperado**
El usuario se equivoca al seleccionar el archivo a eliminar. Cancela la acción.	No sucede nada en el sistema.

Tabla 5.37 Caso de Uso 4: Eliminar un documento

Caso de Uso 5: Descargar un documento	
Prueba	**Resultado Esperado**
El documento se descarga correctamente.	El sistema recibe la petición y descarga el archivo para el usuario.
Prueba	**Resultado Esperado**
El usuario se equivoca y selecciona un archivo erróneo a descargar. Cancela la acción.	No sucede nada en el sistema.
Prueba	**Resultado Esperado**
El documento se descargar de IPFS, pero se detecta que ha sido alterado.	El sistema notifica al usuario de que su documento ha sido alterado y descarga el archivo para el usuario.

Tabla 5.38 Caso de Uso 5: Descargar un documento

Caso de Uso 6: Visualizar documentos	
Prueba	**Resultado Esperado**
Se muestran los documentos correctamente.	El sistema carga el nombre de los documentos asociados al usuario iniciado en la sesión correctamente.
Prueba	**Resultado Esperado**
EL usuario no tiene ningún documento.	No se muestra ningún documento en la pantalla.

Tabla 5.39 Caso de Uso 6: Visualizar documentos

Caso de Uso 7: Eliminar cuenta	
Prueba	**Resultado Esperado**
El sistema elimina la cuenta correctamente.	El sistema elimina un usuario del sistema. El sistema elimina todos los documentos asociados a esa cuenta.
Prueba	**Resultado Esperado**
El usuario selecciona eliminar cuenta, pero luego se arrepiente. Cancela la acción.	El sistema no hace nada.

Tabla 5.40 Caso de Uso 7: Eliminar cuenta

Caso de Uso 8 (Neodoc): Subir hash a la red	
Prueba	**Resultado Esperado**
El *hash* se calcula y se sube correctamente.	El sistema manda la petición a la red Blockchain y se añade el *hash* a la red.
Prueba	**Resultado Esperado**
El usuario se equivoca y selecciona un archivo erróneo a subir. Cancela la acción.	No sucede nada en el sistema.

Tabla 5.41 Caso de Uso 8 (Neodoc): Subir hash a la red

Caso de Uso 8 (Neodoc): Comprobar integridad	
Prueba	**Resultado Esperado**
El documento se comprueba correctamente ya que es el original.	El sistema comprueba en la red que el *hash* existe.
Prueba	**Resultado Esperado**
El usuario quiere comprobar un *hash* de un archivo que ha sido subido a la red.	Se le notifica al usuario el problema y no le deja realizar la acción.
Prueba	**Resultado Esperado**
El documento que va a ser comprobado no coincide con ningún *hash* dentro de la red Blockchain.	Se le notifica al usuario que el documento ha sido el alterado.

Tabla 5.42 Caso de Uso 8 (Neodoc): Comprobar integridad

5.7.2 Pruebas integración

Las pruebas de integración tienen como objetivo comprobar que el sistema funciona correctamente en su conjunto. Estas pruebas se realizarán de manera manual haciendo uso de la aplicación real. Se comprobará que los componentes del sistema no presentan errores de comunicación y realizan las operaciones esperadas.

5.7.3 Pruebas de usabilidad

Las pruebas de usabilidad están destinadas a medir la satisfacción de los usuarios finales con el sistema final desarrollado. Con la finalidad de comprobar esta parte, se les pedirá a usuarios externos al desarrollo de la plataforma rellenar una plantilla para conocer su opinión sobre el sistema.

5.7.4 Pruebas de accesibilidad

En cuanto a las pruebas de accesibilidad, se utilizará la herramienta de *Google Lighthouse*, integrada de manera nativa dentro del navegador Google Chrome, con el fin de analizar la

accesibilidad de la aplicación y el programa Oracle Color para analizar la aplicación con vista de personas daltónicas.

5.7.5 Pruebas de rendimiento

Para las pruebas de rendimiento, se utilizará el analizador de *Google Lighthouse* para analizar la aplicación.

Capítulo 6. Diseño del Sistema

Tras el análisis inicial realizado en el capítulo anterior, se procede ahora a mostrar el diseño del sistema ampliando y adentrándose más a fondo con respecto a lo expresado en el Capítulo 5.

6.1 Arquitectura del Sistema

En esta sección se muestran los diagramas de paquetes, de componentes y de despliegue de la aplicación.

6.1.1 Diagramas de Paquetes

Se muestra a continuación el diseño del diagrama de paquetes de la aplicación.

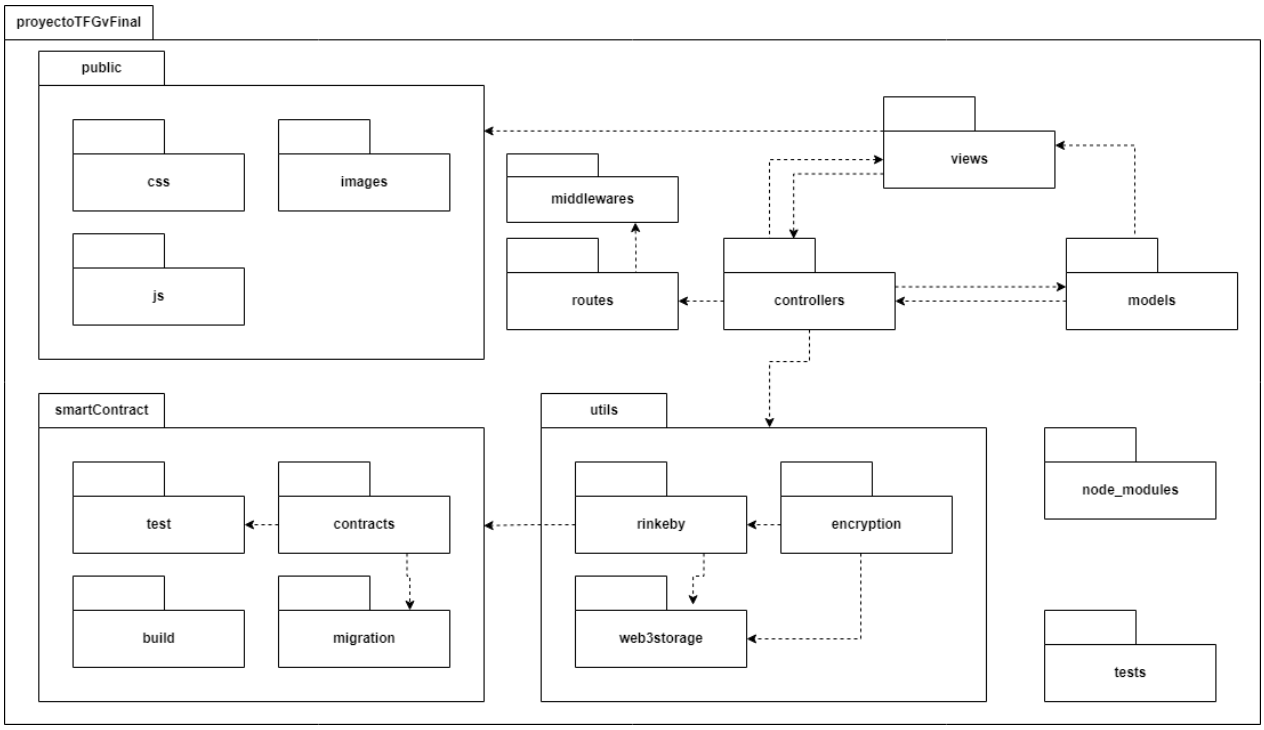

Figura 6.1 Diagrama de paquetes

6.1.1.1 Public

En este paquete se encuentran los paquetes con contenido estático, concretamente los siguientes:

- **Css**: contiene los estilos de los elementos del HTML que se aplicarán a las vistas.
- **Js**: contiene las funciones para interaccionar con los elementos del HTML de manera dinámica desde el cliente.
- ***Images***: contiene las imágenes que son utilizadas en las vistas.

6.1.1.2 Node_modules

Este paquete contiene todos los módulos de terceros usados en este proyecto, como la librería Express-fileUpload o el framework Expressjs.

6.1.1.3 Test

Este paquete contiene las pruebas de la aplicación.

6.1.1.4 Controllers

Este paquete contiene la lógica de la aplicación para servir al cliente con las funcionalidades de la aplicación.

6.1.1.5 Views

Este paquete contiene las vistas para del cliente que el navegador web mostrará al usuario que interacciona con la aplicación web. Está formada así mismo por un paquete denominado *layout* el cual encapsula partes de vistas compartidas entre ellas, evitando así repetir código.

6.1.1.6 Models

Este paquete encapsula las clases que sirven de abstracción para usar los modelos de la base de datos dentro del restos de la aplicación.

6.1.1.7 Routes

Este paquete contiene definidas todas las rutas presentes en la aplicación.

6.1.1.8 Middlewares

Este paquete contiene las acciones a realizar antes de ejecutar una ruta del controlador.

6.1.1.9 Utils

Este paquete contiene toda la lógica relacionada con la encriptación de archivos, generación de *hashes* y comunicación con interfaces externas. Concretamente, incluye:

- **Rinkeby**: contiene la forma de interactuar con la red Blockchain Ethereum Rinkeby.
- ***Encryption***: contiene todas las encriptaciones de archivos y generación de *hashes* de la aplicación.
- **Web3storage**: contiene la lógica de subir y descargar archivos de IPFS.

6.1.1.10 SmartContract

Este paquete alberga toda la implementación relacionada con la implementación y despliegue del contrato inteligente, así como la comunicación con la red pública Blockchain Ethereum Rinkeby. Concretamente incluye:

- ***Build***: contiene las clases con la información de los contratos desplegados.
- ***Test***: contiene la clase con las pruebas del contrato inteligente.
- ***Contracts***: contiene los contratos implementados.
- ***Migration***: este paquete contiene la clase encargada del despliegue del contrato inteligente dentro de la red Blockchain.

6.1.2 Diagramas de Componentes

Se muestra a continuación el diseño del diagrama de componentes de la aplicación.

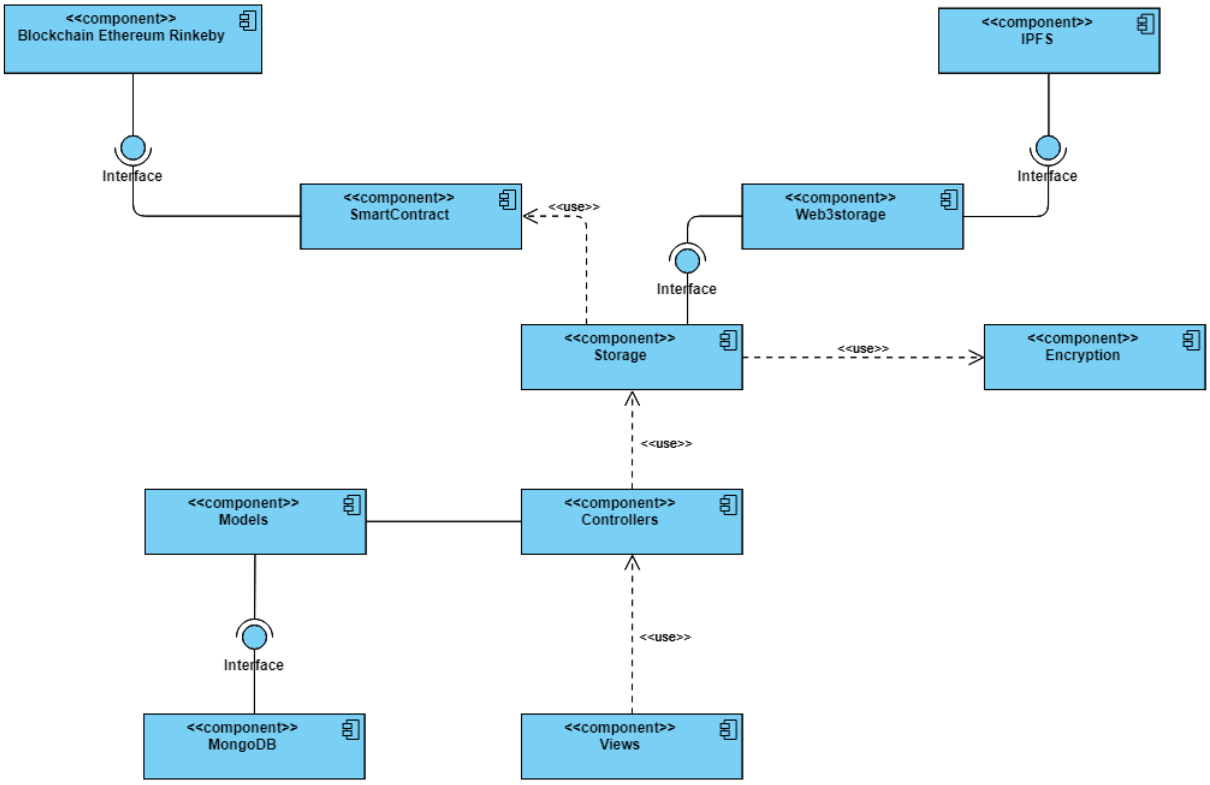

Figura 6.2 Diagrama de componentes

6.1.2.1 MongoDB

El componente MongoDB indica la base de datos en sí que alberga los datos de los usuarios, así como los documentos de cada uno, relacionando ambos modelos entre sí.

6.1.2.2 Storage

Se trata de un servicio principal de la aplicación por el que los controladores obtienen información acerca de las interfaces externas que interactúan con la aplicación para llevar a cabo funcionalidades como subir documentos o descargarlos. También se comunica con el componente *encryption* para manejar la encriptación de archivos y generación de *hashes*.

6.1.2.3 Encryption

Este componente alberga toda la lógica detrás de la encriptación de archivos y generación de *hashes* de documentos.

6.1.2.4 Interfaces externas

A continuación, se describen las interfaces externas que proveen a la aplicación.

6.1.2.4.1 IPFS

Se trata de la red de almacenaje distribuido donde los documentos irán almacenados. Se comunicará con la interfaz web3Storage a través de su interfaz externa.

6.1.2.4.2 Web3Storage

Se trata de la una API que hará de intermediario entre el sistema y la plataforma IPFS. Proveerá de una interfaz externa a la aplicación manejar los documentos.

6.1.2.4.3 Ethereum Blockchain

Se trata de la red Blockchain con la que se interactuará a través del *SmartContract* para almacenar *hashes* de documentos.

6.1.2.5 SmartContract

Engloba todos los aspectos respecto al contrato inteligente, tanto su implementación como despliegue en la red.

6.1.2.6 Patrón Modelo Vista Controlador

Como ya se pudo ir observando, la aplicación web está formada con una arquitectura Modelo Vista Controlador, también denominada como MVC. Esta arquitectura divide la aplicación en diferentes objetos o componentes que interactúan entre sí. De esta forma, las *views* o vistas contienen las pantallas HTML que el usuario final verá e interactuará con ellas.

Por otro lado, los *models* o modelos contienen la abstracción de las tablas de una base de datos, de esta manera se puede interactuar dentro de la aplicación con una modelo de la base de datos de una manera sencilla y eficiente.

Por último, los *controllers* o controladores albergan toda la lógica de la aplicación, envían información a las *views*, se comunican con la base de datos y otras interfaces externas para llevar a cabo las diferentes funcionalidades de la aplicación.

6.1.3 Diagramas de Despliegue

Se muestra a continuación el diseño del diagrama de despliegue de la aplicación.

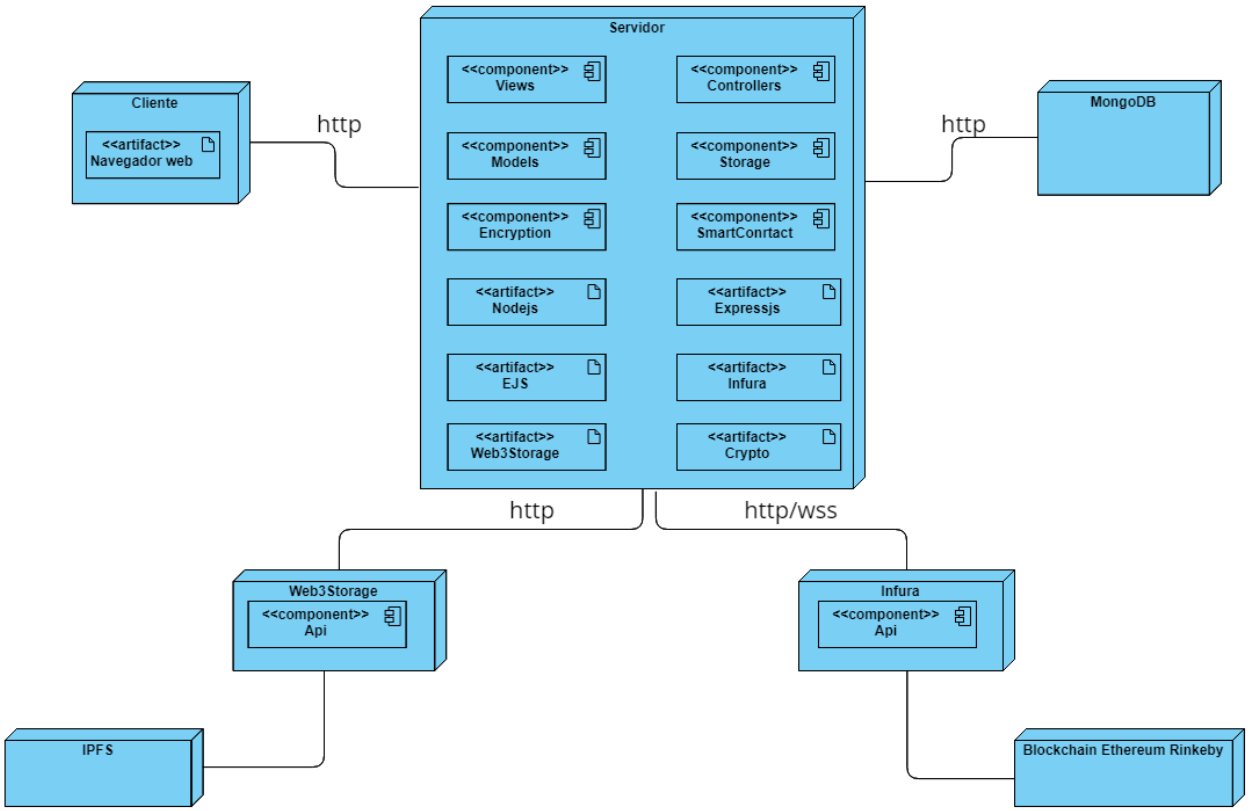

Figura 6.3 Diagrama de despliegue

6.1.3.1 Cliente

Se trata de la máquina del usuario, como podría ser un ordenador en el que a través de un navegador web y el protocolo HTTP podrá acceder al sistema desarrollado.

6.1.3.2 Servidor

Se trata del eje central del sistema. Alberga desde las vistas que se envían al cliente, hasta toda la lógica de la aplicación para llevar a cabo las funcionalidades del sistema. Se representan los componentes y, además, los principales artefactos, como Nodejs y Expressjs.

6.1.3.3 Web3Storage

Se trata de una infraestructura intermedia entre el servidor y la plataforma IPFS. Se comunica con el servidor a través del protocolo HTTP.

6.1.3.4 Infura

Se trata de una infraestructura intermedia entre el servidor y la red Blockchain Ethereum Rinkeby. Se comunica con el servidor a través del protocolo HTTP.

6.1.3.5 MongoDB

Se trata de la base de datos conectada con el servidor para almacenar los datos. La comunicación con este se realizará mediante el protocolo HTTP.

6.1.3.6 IPFS

Se trata de la infraestructura de almacenaje de los archivos.

6.1.3.7 Blockchain Ethereum Rinkeby

Se trata de la red Blockchain.

6.2 Diseño de Clases

En esta sección se presentará un diagrama global de todas las clases del sistema como parte del diseño la aplicación. Este diagrama puede diferir en ciertos puntos con el desarrollado en el análisis, ya que añade variaciones y ampliaciones.

6.2.1 Diagrama de Clases

Se muestra a continuación el diagrama de clases del sistema.

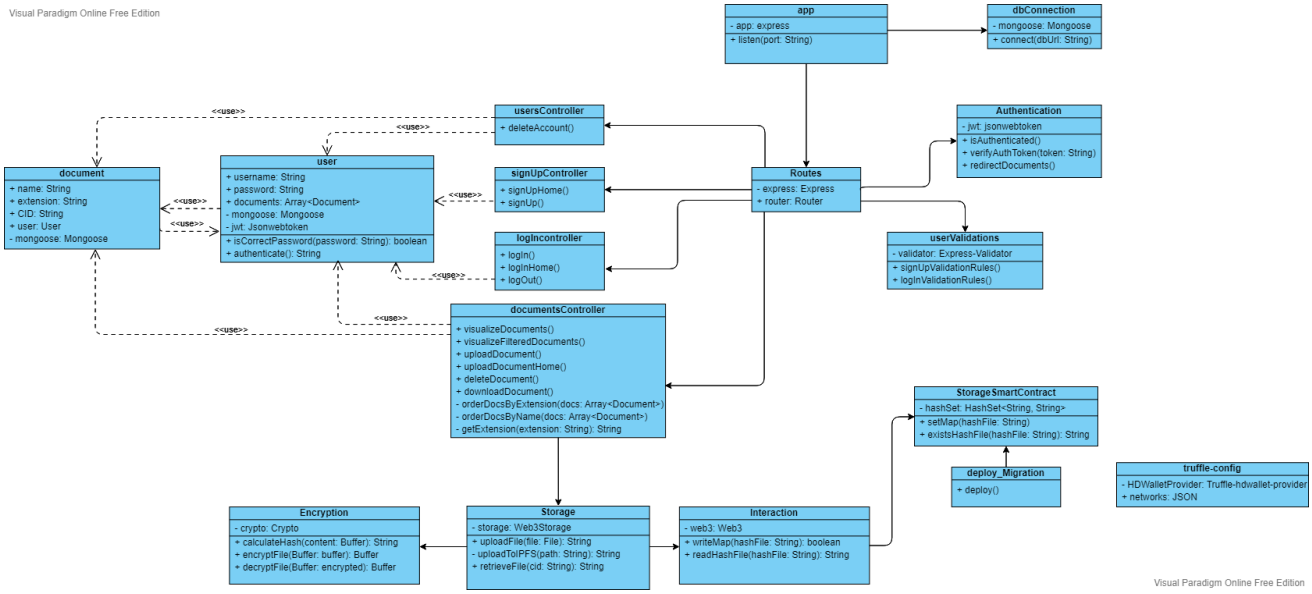

Figura 6.4 Diagrama de clases

Como puede observarse en el diagrama de clases anterior, las clases tienen una finalidad propia y clara de tal forma que cada una encapsula una parte del sistema para llevar a cabo una acción.

En cuanto a la relación de este diagrama de clases con el diagrama de paquetes, mostrado en el apartado 6.1.1, el paquete 6.1.1.4 *Controllers* contiene los controladores *usersController*, *documentsController*, *logInController* y *signUpController*. El paquete 6.1.1.7 *Routes* contiene la clase *routes*, que define todas las rutas de la aplicación, como "/login" o "/documents". El paquete 6.1.1.6 *Models*, contiene los modelos *User* y *Document*. Por otro lado, el paquete 6.1.1.8 *Middlewares* contiene a la clase *middleware*. El paquete 6.1.1.9 *Utils* contiene las clases *Interaction*, *Storage* y *Encryption*. Por último, el paquete 6.1.1.10 *SmartContract* engloba las clases *truffle-config*, *deploy_migration* y *StorageSmartConrtact*. El resto de las clases como *app* y *dbConnection* se encontrarían en el paquete raíz ProyectoTFGvFinal.

6.2.1.1 App

Se encarga de inicializar la aplicación en un puerto concreto, configurar la aplicación, inicializar los *middlewares* e iniciar la conexión a la base de datos.

6.2.1.2 DbConnection

Esta clase se encarga de conectar la app a la base de datos a través del método "connect(dbUrl: String)".

6.2.1.3 Routes

Esta clase se encarga de albergar y definir todas las rutas existentes de la aplicación y definir los métodos HTTP que usarán las mismas.

6.2.1.4 Authentication

Se trata de una clase que actúa como un *middleware*, es decir, realizará una acción antes de que una ruta se ejecute. Concretamente, verifica que el usuario tiene la autorización para acceder a una ruta específica, es decir, debe estar autenticado en la aplicación.

6.2.1.5 UserValidations

Esta clase se encarga de definir las validaciones de los campos que el usuario debe introducir para iniciar sesión y registrar una nueva cuenta.

6.2.1.6 StorageSmartContract

Esta clase se trata de un contrato inteligente escrito con el lenguaje Solidity el cual actuará como una base de datos de *hashes*.

6.2.1.7 Deploy_migration

Esta clase se encara de desplegar el contrato inteligente en la red Blockchain.

6.2.1.8 Truffle-config

Esta clase alberga las configuraciones de la red o redes donde se quiere desplegar el contrato inteligente.

6.2.1.9 Storage

Esta clase es el eje central de comunicación entre las clases *Encryption* e *Interaction* con el controlador *documentsController*. Alberga todas las funcionalidades de ambas clases además de la subido del archivo a IPFS, de tal forma que el controlador solo necesita de esta clase y no tener que usar las tres a la vez.

6.2.1.10 Encryption

Esta clase alberga todos los métodos de encriptación. Desde el cálculo del *hash* de un archivo hasta la encriptación de un archivo y su desencriptación.

6.2.1.11 Interaction

Esta clase se comunica con el contrato inteligente desplegado en la red Blockchain, tanto para leer un *hash* como subir uno.

6.2.1.12 User

Se trata de un modelo de abstracción de un usuario de la base de datos. Además, alberga métodos de autenticación de usuario y de comprobación de contraseñas.

6.2.1.13 Document

Se trata de un modelo de abstracción de un documento de la base datos.

6.2.1.14 DocumentsController

Este controlador alberga todos los métodos relacionados con el manejo de documentos, desde el visualizado de todos los documentos, la subida de un documento, la descarga de este hasta su borrado en la base de datos. También, encapsula métodos privados para ordenador los documentos y que estos sean mostrados al usuario de cierta manera, por orden alfabético o por tipo de archivo.

6.2.1.15 UsersController

Este controlador simplemente encapsula el borrado de un usuario de la base de datos, y, por ende, el borrado de todos sus documentos asociados.

6.2.1.16 SignUpController

Este controlador tiene como objetivo tratar el registro de un nuevo usuario.

6.2.1.17 LogInController

Este controlador tiene como objetivo tratar el inicio de sesión de un usuario.

6.3 Diagramas de secuencia

En esta sección se mostrarán los diagramas de las principales funcionalidades del sistema con el fin de ayudar en la implementación del sistema.

6.3.1 Subir un archivo

Se reflejará el comportamiento del sistema a la hora de subir un archivo al sistema.

6.3.1.1 Diagramas de Secuencia

Se muestra a continuación el diagrama de secuencia de subir un archivo al sistema.

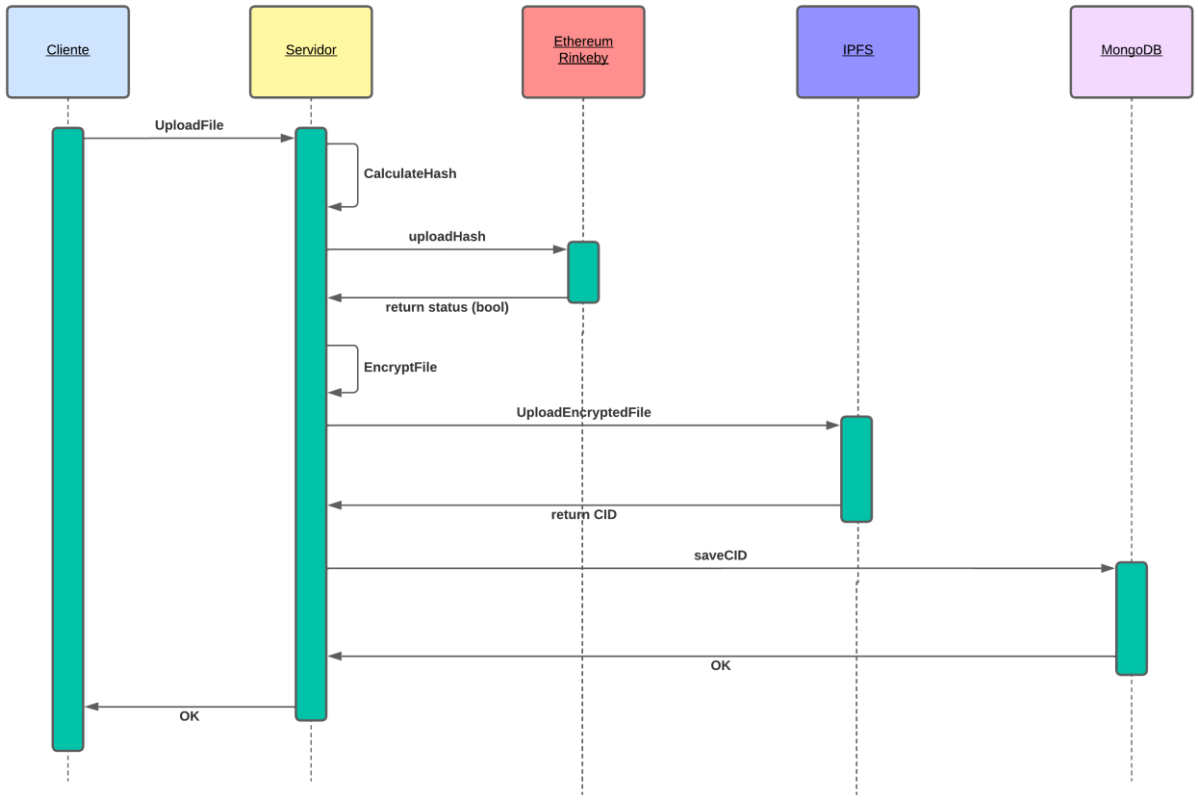

Figura 6.5 Diagrama de secuencia Subir un archivo

Primero de todo, se entiende que el usuario ya está autenticado y decide subir un archivo. Por ello, desde el cliente se envía le archivo al servidor, donde se calculará el *hash* del archivo y se almacenará dentro de la red Blockchain. A continuación, se encriptará el archivo y será subid dentro de IPFS. IPFS retornará un identificador que será almacenado dentro de la base de datos para que en un futuro se pueda descargar el archivo. Una vez guardado en la base de datos, se responderá al cliente con un *OK* de que todo ha funcionado correctamente.

6.3.2 Descargar un archivo

Se reflejará el comportamiento del sistema a la hora de descargar un archivo al sistema.

6.3.2.1 Diagramas de Secuencia

Se muestra a continuación el diagrama de secuencia de descargar un archivo al sistema.

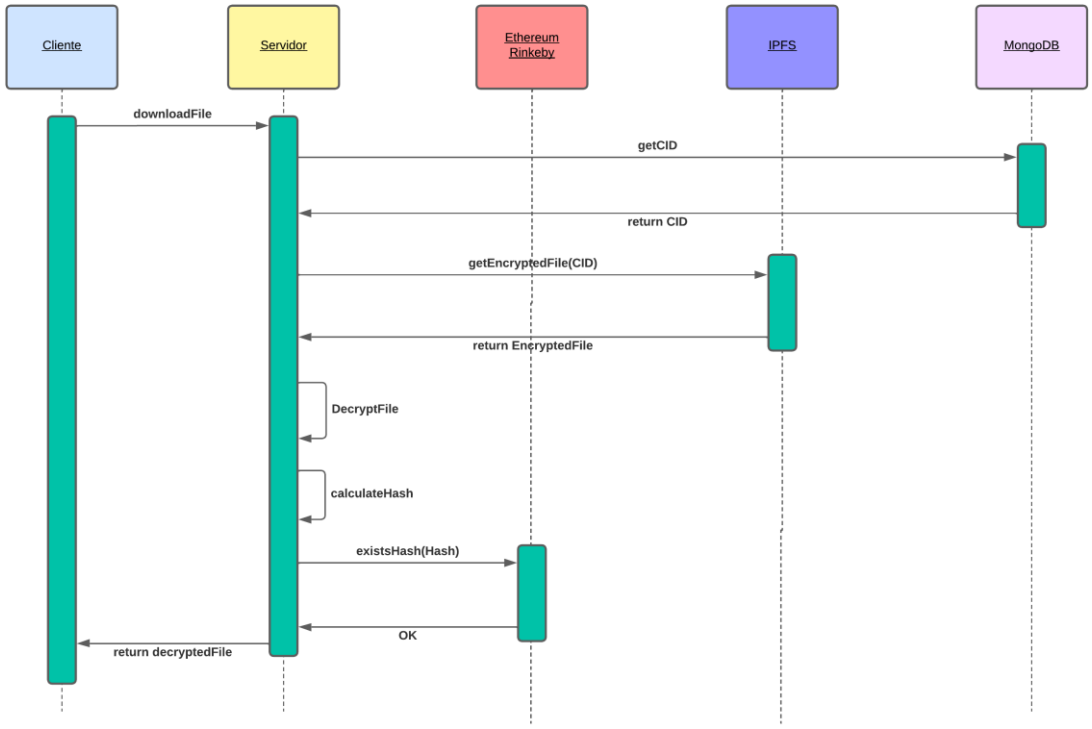

Figura 6.6 Diagrama de secuencia Descargar un archivo

Primero de todo, se entiende que le usuario ya está autenticado y decide descargar uno de sus archivos. Una vez seleccionado, se coge de la base de datos el CID de ese documento para más adelante descargar el archivo de IPFS con ese mismo CID. Una vez descargado, se debe desencriptar y calcular su hash. Con ese hash, se comprueba dentro de la red Blockchain que existe y si se recibe el *OK* se envía al cliente el archivo desencriptado.

6.3.3 Eliminar un archivo

Se muestra a continuación el diagrama de secuencia de eliminar un archivo del sistema.

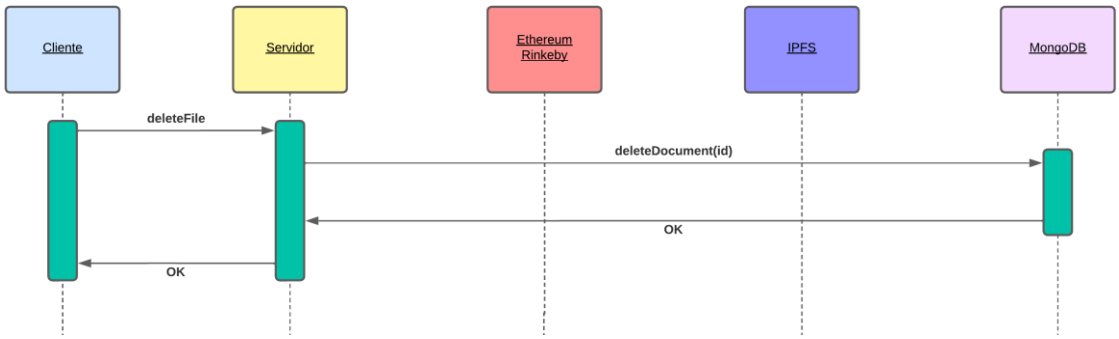

Figura 6.7 Diagrama de secuencia Eliminar un archivo

Primero de todo, se entiende que le usuario ya está autenticado y decide eliminar uno de sus archivos. Por lo tanto, se coge el id del documento a eliminar y se comunica con la base de datos el servidor para eliminar el documento. Se devuelve un *OK* que se propaga hasta el cliente. El *hash* en la Blockchain no es necesario eliminarlo y el documento de IPFS tampoco, ya que la plataforma lo dejará de indexar entre sus nodos por inactividad y sin el CID no se puede recuperar tal archivo.

6.3.4 Eliminar un usuario

Se muestra a continuación el diagrama de secuencia de eliminar un usuario del sistema.

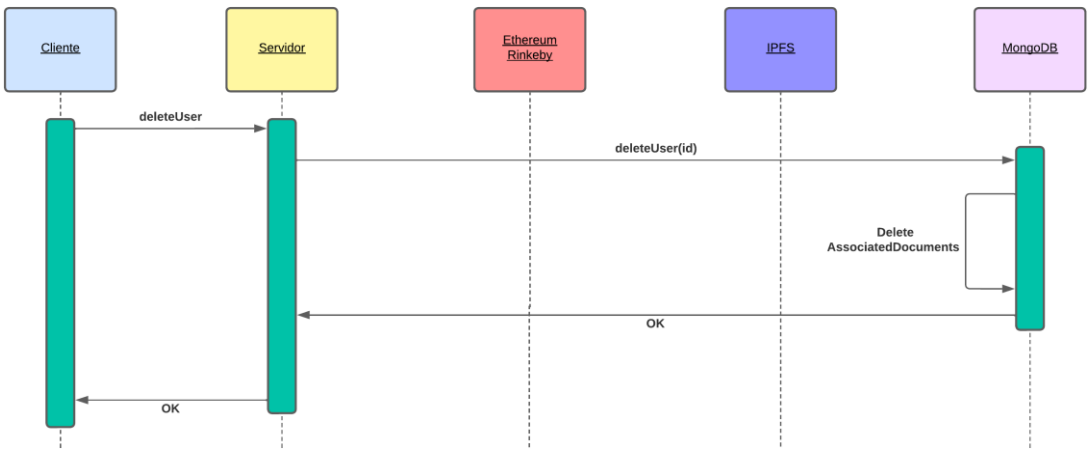

Figura 6.8 Diagrama de secuencia Eliminar un usuario

Primero de todo, se entiende que le usuario ya está autenticado y decide eliminar su cuenta. Por lo tanto, se comunica el servidor con la base de datos y le envía el id del usuario a eliminar. Se eliminan todos los documentos asociados al usuario, así como la cuenta en sí y se devuelve el *OK* que es propagado hasta el cliente.

6.3.5 Filtrar archivos

Se muestra a continuación el diagrama de secuencia de filtrar archivos del sistema.

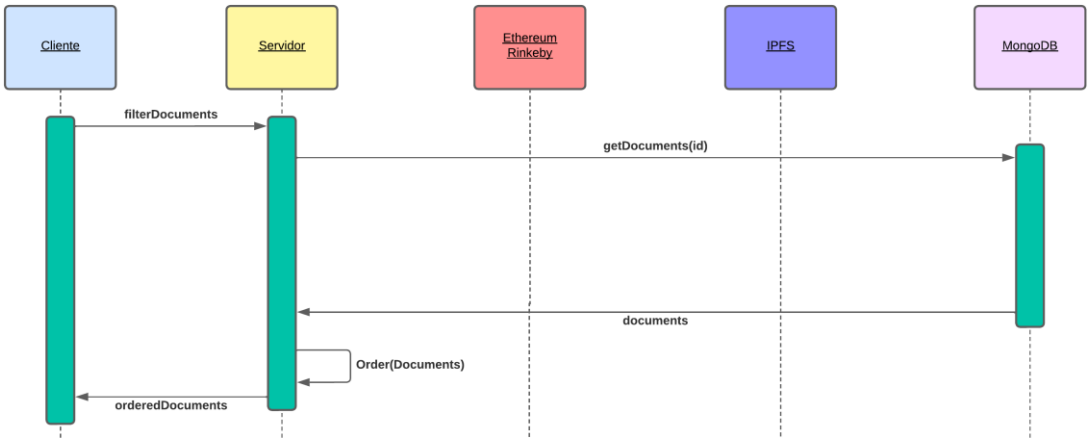

Figura 6.9 Diagrama de secuencia Filtrar archivos

Primero de todo, se entiende que le usuario ya está autenticado y decide visualizar sus documentos de una manera ordenada. Por lo tanto, se cogen de la base de datos los documentos asociados a la cuenta del usuario y se ordenan en el servidor para más adelantes enviaros al cliente.

6.3.6 Registrar nuevo usuario

Se muestra a continuación el diagrama de secuencia registrar un nuevo usuario en el sistema.

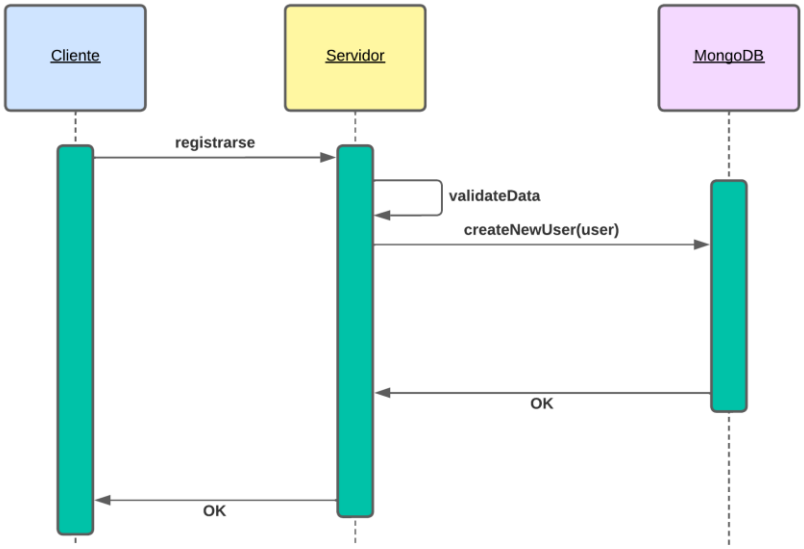

Figura 6.10 Diagrama de secuencia Registrarse

6.3.7 Iniciar sesión

Se muestra a continuación el diagrama de secuencia iniciar sesión en el sistema.

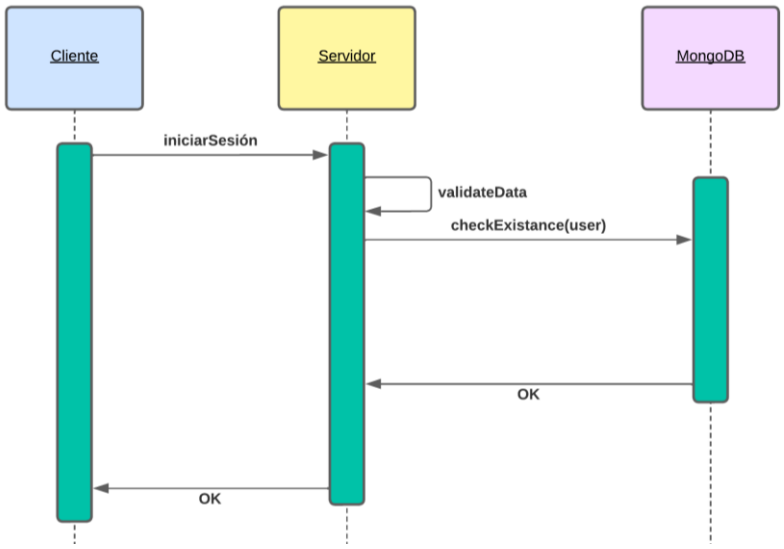

Figura 6.11 Diagrama de secuencia Iniciar sesión

6.4 Diagramas de Actividades

En esta sección se muestra el comportamiento del sistema a través de diagramas de actividades.

6.4.1 Subir documento

Se muestra a continuación el diagrama de actividades subir documento al sistema.

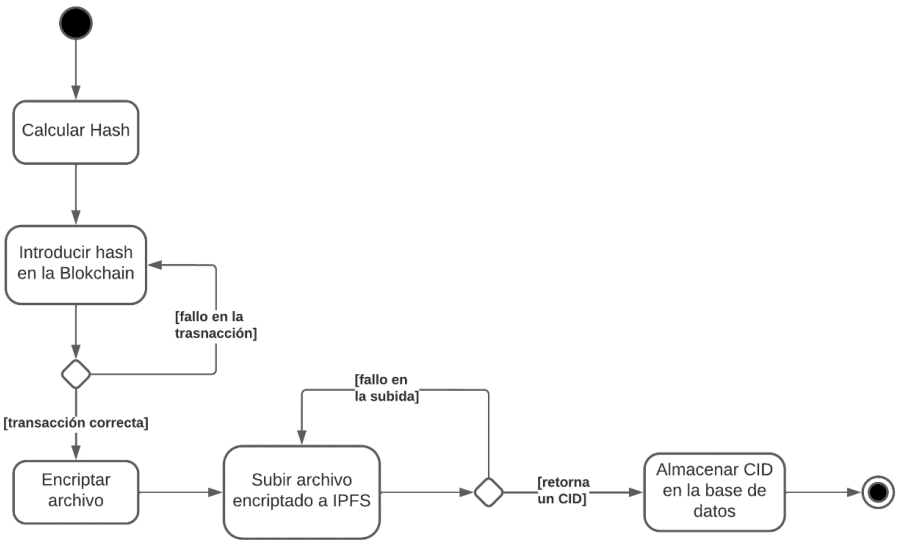

Figura 6.12 Diagrama de actividades Subir un archivo

Primero de todo, el usuario que realice el inicio de esta sección al adjuntar un documento debe estar autenticado en el sistema. Luego, se calculará el *hash* del archivo, se introducirá este dentro de la red Blockchain. En caso negativo, se volverá a intentar este paso y en caso favorable se continuará y se encriptará el archivo. Se subirá el archivo encriptado a IPFS, en caso de que no se pueda a subir se volverá al inicio de esta actividad y si es correcto, devolverá un identificador CID y este se almacenará en la base de datos para terminar con la ejecución.

6.5 Diseño de la Base de Datos

La base de datos es una parte fundamental de este sistema. Conforme tanto el manejo y creación de usuario como el almacenaje de los documentos con su identificador CID para descargarlos de la plataforma IPFS. Aun así, el diseño de la base de datos es bastante simple y pequeño.

6.5.1 Descripción del SGBD Usado

Para el desarrollo de esta aplicación se ha utilizado una base de datos NoSQL, no relacional, concretamente MongoDB. Esta base de datos guarda la información en documentos con un formato propio, muy similar al JSON, denominado BSON. Además, la base de datos se divide en colecciones, una por cada entidad que se quiera definir.

6.5.2 Integración del SGBD en Nuestro Sistema

La integración de la base de datos se realiza mediante una biblioteca de JavaScript denominada como Mongoose, la cual permite conectar, a través del protocolo HTTP, con la base de datos. En este caso, la base de datos es *on-premise* y se conectaría mediante la Url "mongodb://localhost/proyectoTFG". De todas formas, su despliegue en la nube utilizando MongoDB Atlas es muy sencillo de realizar, cambiando así a una infraestructura *cloud*.

6.5.3 Diagrama de la base de datos

Se muestra a continuación el diagrama representativo de la base de datos.

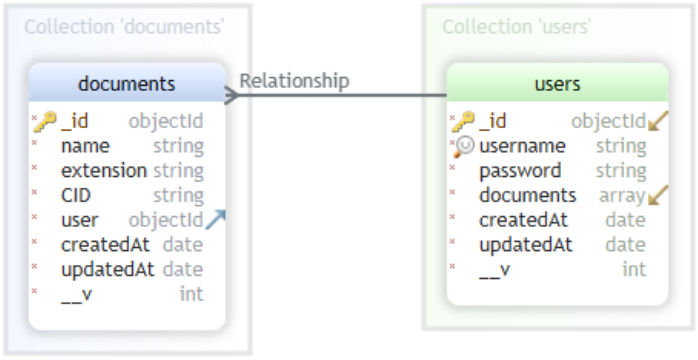

Figura 6.13 Diagrama de la base de datos

Como puede observarse, la base de datos solo presenta dos entidades. Por un lado, los usuarios, que tienen un id único, un *username*, una *password*, siendo estos necesarios para la creación de un nuevo usuario, y una lista de documentos no obligatorios. Por otro lado, los documentos que tienen un id, un *name*, una *extension* que representa el tipo de documento y un *user*, que sería el usuario al que pertenece el documento. Todos estos campos son obligatorios para la creación de un nuevo documento. Además, ambas entidades tienen dos campos similares, *createdAt* y *updatedAt*, que representan la fecha de creación y la fecha en la que se produjo alguna modificación en algún campo de la entidad.

6.6 Diseño de la Interfaz

En esta sección se mostrará la interfaz definitiva que la aplicación tendrá. Se observará una gran evolución respecto a la ya diseñada en el análisis, la cual sirvió para sentar las bases del diseño y se ha mejorado. Cabe destacar que toda la interfaz será *responsive*, y se adaptará a la pantalla en la que se esté utilizando la aplicación.

6.6.1 Inicio de sesión

El inicio de sesión es bastante simple. Se muestran el campo de nombre de usuario y contraseña además de dos botones, a la izquierda para aceptar el inicio y el de la derecha redirige al registro.

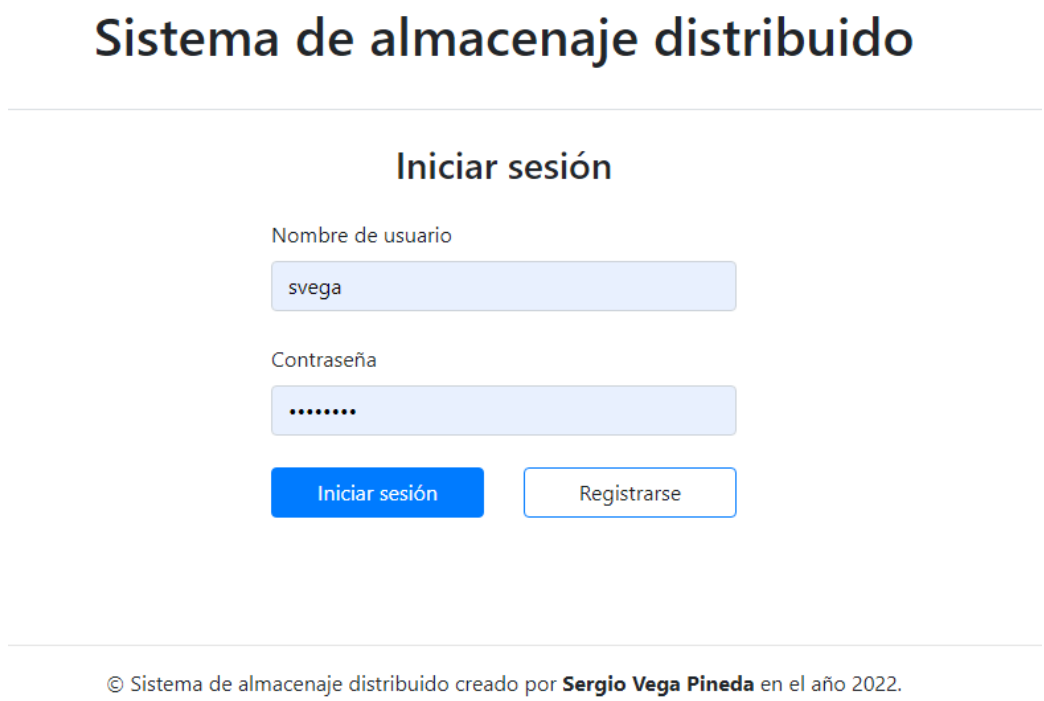

Figura 6.14 Pantalla de inicio de sesión

Si el usuario introduce mal los datos y estos no pasan las validaciones, se mostrarán alertas de error.

Figura 6.15 Pantalla Inicio de sesión con alerta

6.6.2 Registro

El registro también es bastante simple. Se muestran el campo de nombre de usuario, contraseña y confirmación de contraseña además de dos botones, a la izquierda para aceptar el registro y el de la derecha redirige al inicio de sesión.

Figura 6.16 Pantalla registro de usuario

Si el usuario introduce mal los datos y estos no pasan las validaciones, se mostrarán alertas de error.

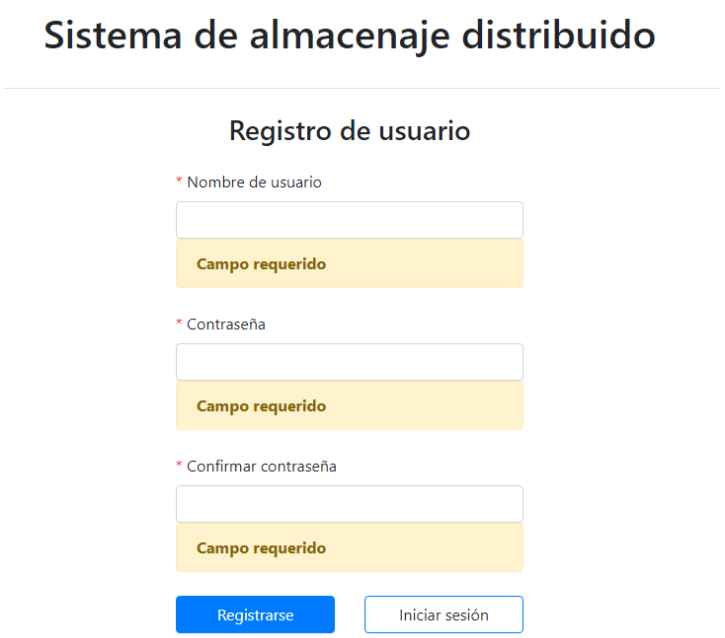

Figura 6.17 Pantalla Registro con alertas de campos vacíos

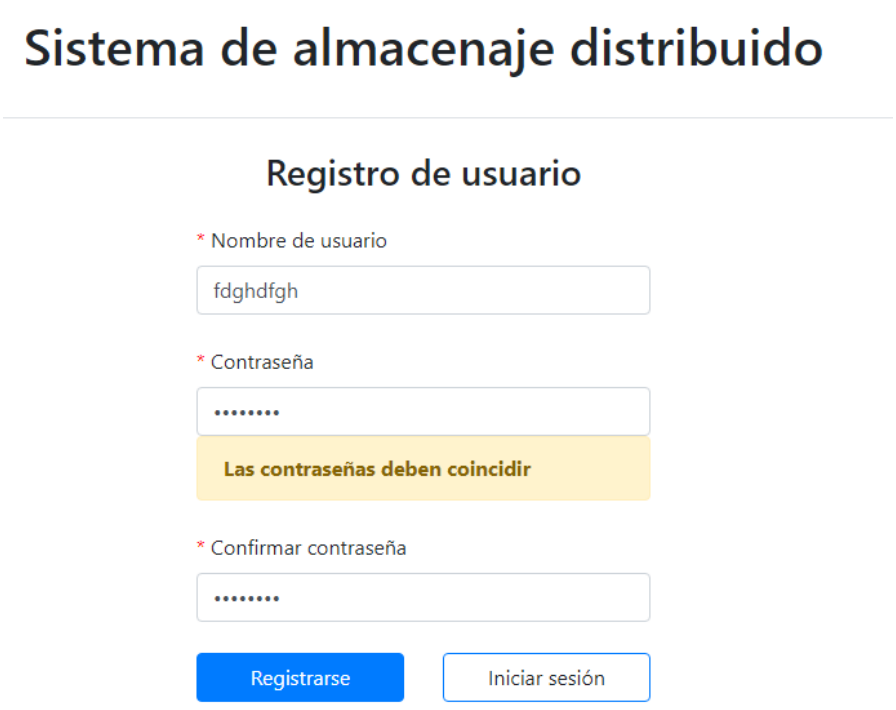

Figura 6.18 Pantalla Registro con alerta de contraseñas diferentes

6.6.3 Barra de navegación superior

Esta barra está compuesta por un logo a la izquierda, seguida por las páginas de la aplicación web y en la parte derecha un icono representando el perfil del usuario, el cual despliega una lista con dos opciones, la de cerrar sesión y la eliminar una cuenta.

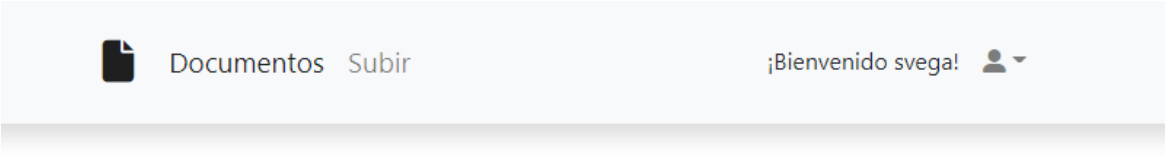

Figura 6.19 Barra de navegación superior

Dentro del icono de perfil, se despliegan dos opciones.

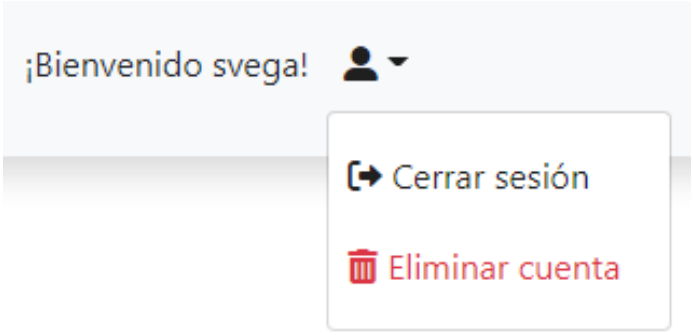

Figura 6.20 Desplegable del icono perfil

En caso de presionar eliminar cuenta, se abrirá un diálogo modal que permite confirmar o cancelar la acción.

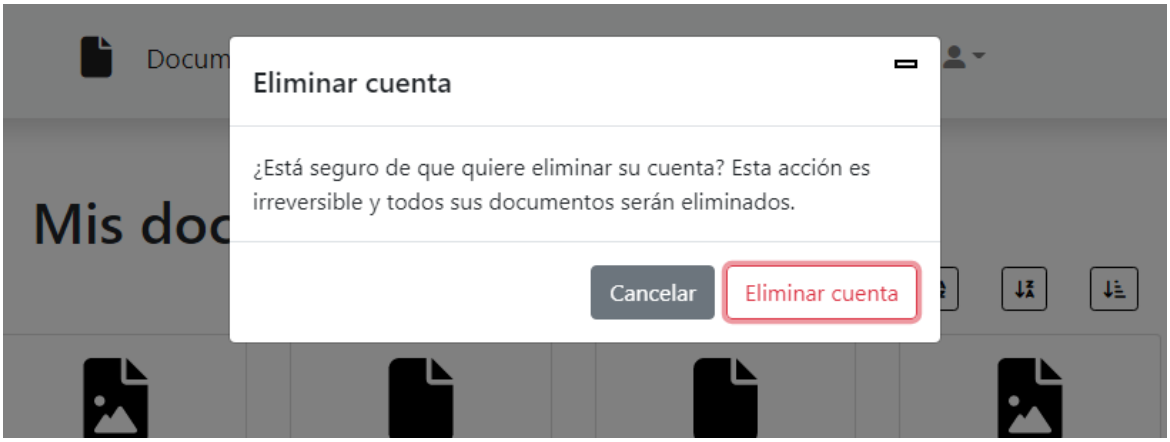

Figura 6.21 Confirmación de borrado de cuenta

6.6.4 Pie de página

El pie de página se trata de un texto sencillo explicando que es una aplicación de almacenaje creada por Sergio Vega Pineda en el año 2022.

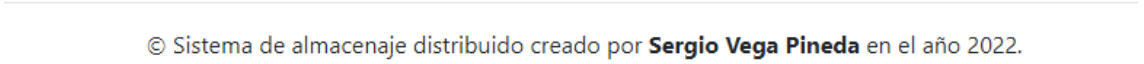

Figura 6.22 Pie de página

6.6.5 Pantalla de Documentos

Esta página está destinada a mostrar los documentos y las acciones relativas a estos. Concretamente, cada tarjeta contiene un documento con su nombre y dos acciones, descargar el documento y eliminarlo. Por otro lado, en la parte derecha superior, se encuentran 3 filtros que pueden aplicarse a los documentos de tal forma que puedan ordenarse por orden alfabético de la A-Z, de la Z-A y por tipo de documento.

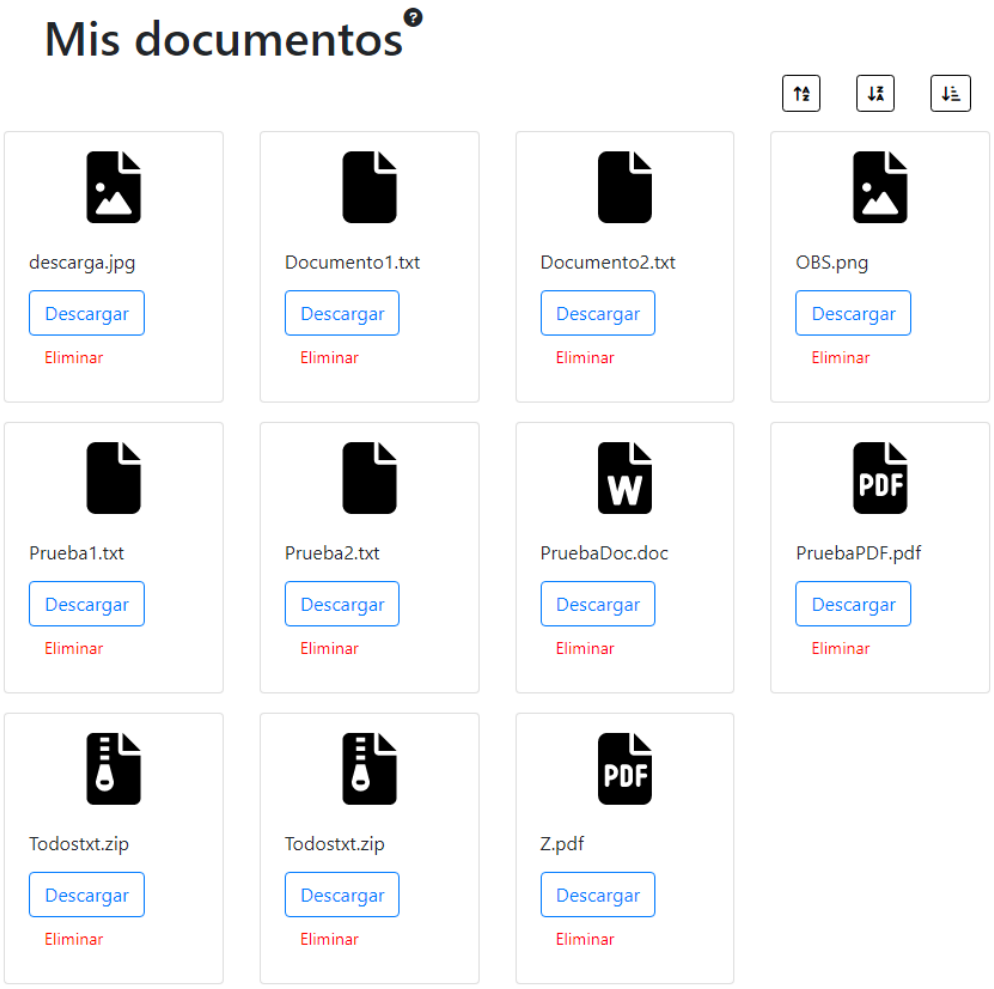

Figura 6.23 Pantalla Visualizar documentos

Si se decide eliminar un documento, aparecerá un diálogo modal preguntando si está seguro de realizar esta acción.

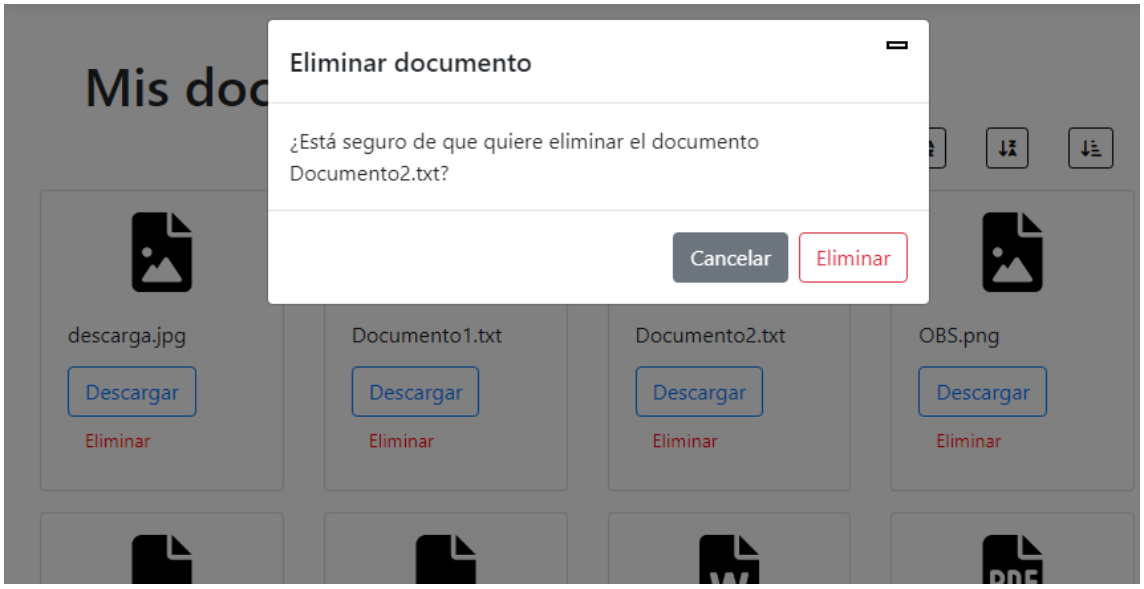

Figura 6.24 Modal para confirmar la eliminación de un documento

Si se presiona descargar, se descargará el documento dentro de la máquina del usuario.

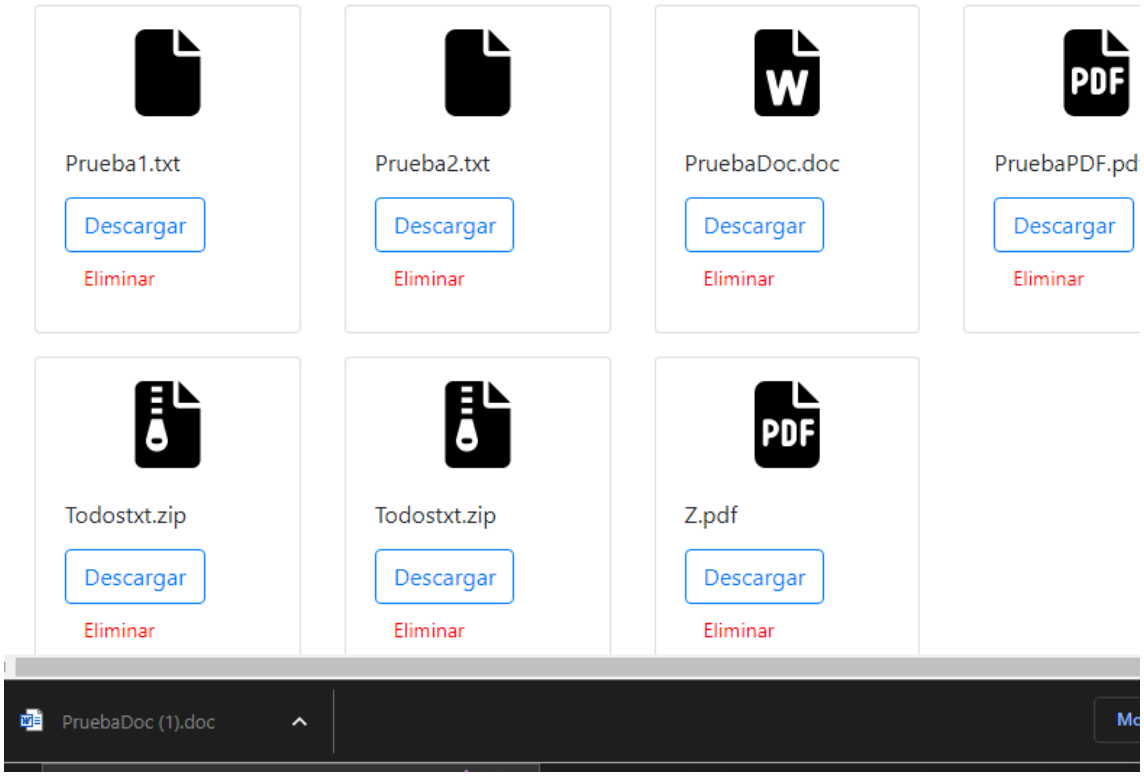

Figura 6.25 Pantalla Documentos con una descarga

6.6.6 Página Subir un documento

La pantalla de subir un documento es una pantalla simple que permite adjuntar un archivo en la aplicación y que se suba dentro del sistema.

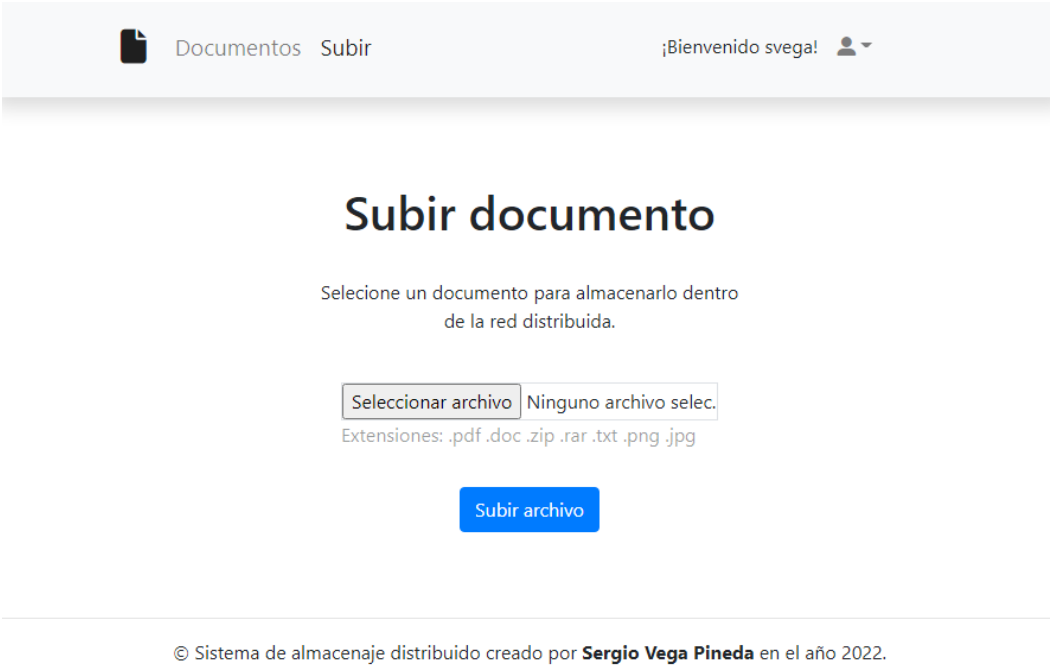

Figura 6.26 Pantalla Subir un documento

Una vez presionado el botón subir archivo con un documento adjunto, aparecerá un icono para simular que se está cargando.

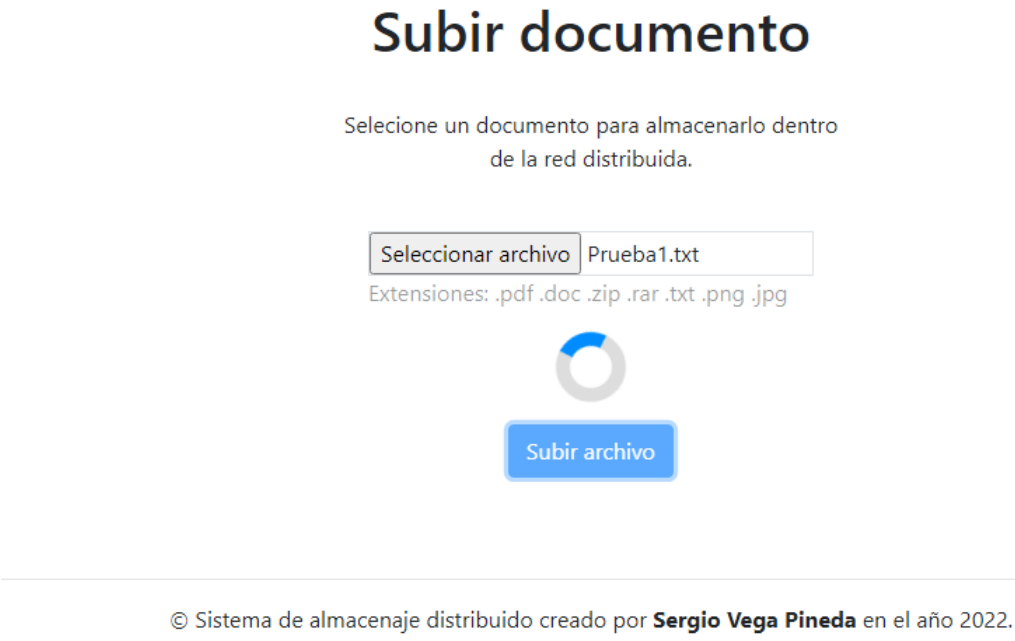

Figura 6.27 Pantalla Subir documento cargando

6.7 Especificación Técnica del Plan de Pruebas

Esta sección consiste en especificar de una forma detallada las pruebas que se van a realizar sobre el sistema a lo largo del proyecto. Todas las pruebas se realizarán sobre la siguiente máquina:

- CPU: Intel Core i7-8700.
- Ram: 16 GB.
- Sistema operativo: Windows 10 pro de 64 bits.
- Navegador web: Google Chrome.

6.7.1 Pruebas Unitarias

Las pruebas unitarias realizadas en este sistema se han centrado, sobre todo, en la parte del desarrollo del contrato inteligente y del correcto comportamiento de las rutas de la aplicación.

Para realizar las pruebas del *SmartContract* se utilizará el framework Mocha. Por otro lado, para las pruebas de las rutas, se utilizará PostMan y el framework Mocha y la librería Supertest.

Registro de usuario	
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario y contraseña correctos.	El sistema posee un usuario más. El servidor responde al cliente con un código 200. Se genera un token guardado en la *cookie* del navegador para que se sepa que el usuario está autenticado.
Prueba	**Resultado Esperado**
Un usuario se intenta registrar, pero la contraseña tiene un formato incorrecto.	El sistema no posee un usuario más y se muestra un dialogo notificando el error. El servidor responde con un código 400.
Prueba	**Resultado Esperado**
Un usuario se intenta registrar, pero el usuario introducido ya existe dentro de la base de datos.	El sistema no posee un usuario más y se muestra un dialogo notificando el error. El servidor responde con un código 400.

Tabla 6.1 Pruebas unitarias Registro de usuario

Inicio de sesión	
Prueba	**Resultado Esperado**
Un usuario inicia sesión en una cuenta existente.	El sistema inicia sesión. El servidor responde con un código 200. Se genera un token guardado en la *cookie* del navegador para que se sepa que el usuario está autenticado.
Prueba	**Resultado Esperado**
Un usuario intenta acceder a una cuenta, pero esta no existe.	El sistema no inicia sesión y notifica el problema al cliente. El servidor responde con un código 400.
Prueba	**Resultado Esperado**
Un usuario intenta iniciar	El sistema no inicia sesión y notifica el problema. El servidor

sesión, pero no rellena un campo obligatorio, la contraseña.	responde con un código 400.

Tabla 6.2 Pruebas unitarias Inicio de sesión

Subir un documento	
Prueba	**Resultado Esperado**
Se sube un documento al sistema.	El sistema añade a la base de datos el identificador del documento. Responde al cliente con un código 302 para redirigir al usuario a la pantalla de documentos.
Prueba	**Resultado Esperado**
El usuario selecciona un archivo de tipo no soportado por el sistema.	El sistema no sube el documento y notifica del problema. El servidor responde con un código 409.
Prueba	**Resultado Esperado**
El usuario se confunde y selecciona y archivo erróneo. Cancela la acción.	No sucede nada en el sistema. El servidor no genera respuesta alguna.
Prueba	**Resultado Esperado**
El usuario intenta acceder a esta ruta, pero no está autenticado.	Se redirige el usuario a la pantalla de inicio de sesión. El servidor genera un código de respuesta 302.

Tabla 6.3 Pruebas unitarias Subir un documento

Eliminar un documento	
Prueba	**Resultado Esperado**
El documento se elimina correctamente.	El sistema elimina de la base de datos los datos sobre ese documento. El servidor responde con un código 200.
Prueba	**Resultado Esperado**
El usuario se equivoca al seleccionar el archivo a eliminar. Cancela la acción.	No sucede nada en el sistema. El servidor no responde de ningún modo.

Tabla 6.4 Pruebas unitarias Eliminar un documento

Descargar un documento	
Prueba	**Resultado Esperado**
El documento se descarga correctamente.	El sistema recibe la petición y descarga el archivo para el usuario.
Prueba	**Resultado Esperado**
El documento se descargar de IPFS, pero se detecta que ha sido alterado.	El sistema notifica al usuario de que su documento ha sido alterado y descarga el archivo para el usuario. El servidor responde con un código 200.

Tabla 6.5 Pruebas unitarias Descargar un documento

Visualizar documentos	
Prueba	**Resultado Esperado**
Se muestran los documentos correctamente.	El sistema carga el nombre de los documentos asociados al usuario iniciado en la sesión correctamente. El servidor responde con un código 200.
Prueba	**Resultado Esperado**
EL usuario no tiene ningún documento.	No se muestra ningún documento en la pantalla. El servidor responde con un código 200.

Tabla 6.6 Pruebas unitarias Visualizar documentos

Eliminar cuenta	
Prueba	**Resultado Esperado**
El sistema elimina la cuenta correctamente.	El sistema elimina un usuario del sistema. El sistema elimina todos los documentos asociados a esa cuenta. El servidor responde con un código 200.
Prueba	**Resultado Esperado**
El usuario selecciona eliminar cuenta, pero luego se arrepiente. Cancela la acción.	El sistema no hace nada. El servidor no responde de ningún amanera.

Tabla 6.7 Pruebas unitarias Eliminar cuenta

SmartContract	
Prueba	**Resultado Esperado**
Se sube un *hash* a la red.	Se comprueba que el *HashSet* del contrato inteligente ha aumentado en 1.
Prueba	**Resultado Esperado**
Se comprueba que un *hash* existe dentro del contrato inteligente y es correcto.	El *SmartContract* devuelve el hash de vuelta, comprobando así que existe.
Prueba	**Resultado Esperado**
Se comprueba que un *hash* existe dentro del contrato inteligente, pero no existe.	El *SmartContract* devuelve una cadena vacía.

Tabla 6.8 Pruebas unitarias SmartContract

Generación de hash	
Prueba	**Resultado Esperado**
Se genera dos *hashes* correctamente del mismo texto.	Ambos *hashes* coinciden.
Prueba	**Resultado Esperado**
Se genera el *hash* correctamente de dos textos distintos.	No coinciden los *hashes*.

Tabla 6.9 Resultado pruebas unitarias generar Hash

Encriptación	
Prueba	**Resultado Esperado**
Se encriptan dos textos distintos.	El resultado de las encriptaciones debe ser distinto.
Prueba	**Resultado Esperado**
Se encripta y desencripta un texto y se compara la desencriptación el contenido original sin haber sido encriptado.	Ambos contenidos deben coincidir.
Prueba	**Resultado Esperado**
Se encripta y desencripta el mismo contenido dos veces. Ambas desencriptaciones se comparan.	Ambos contenidos desencriptados deben coincidir.

Tabla 6.10 Resultado pruebas unitarias de encriptación

6.7.2 Pruebas de Integración y del Sistema

Una vez desarrollado el sistema, se probará manualmente de forma extensiva con varios usuarios para comprobar el correcto funcionamiento por parte de personas ajenas al desarrollador del sistema.

Registro de usuario	
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio” y contraseñas “pass1234”.	El sistema redirige al usuario a la página de documentos, que estará vacía ya que es un nuevo usuario y no tiene ningún documento en su cuenta.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio”, contraseña “pass1234” y confirmación de contraseña “DFG”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta indicando que las contraseñas no coinciden.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “”, contraseña “” y confirmación de contraseña “”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta que los campos nombre usuario, contraseña y confirmación de contraseña están vacío y son requeridos.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio”, contraseña “pass1234” y confirmación de contraseña “pass1234”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta que ya existe una cuenta con ese usuario.

Tabla 6.11 Pruebas integración Registro de usuario

Inicio de sesión	
Prueba	**Resultado Esperado**
Un usuario inicia sesión en una cuenta existente con el nombre de usuario “Sergio” y contraseña “pass1234”.	El sistema inicia sesión y redirige al usuario a la pantalla Documentos para mostrar los documentos que tena en esa cuenta asociados. .
Prueba	**Resultado Esperado**
Un usuario intenta acceder a una cuenta, con los datos “Lausip” y contraseña “fdgsdfg23”, pero esta no existe.	El sistema no inicia sesión y se mantiene en la pantalla de inicio de sesión. Además, muestra un error indicando que el usuario o contraseña son incorrectos.
Prueba	**Resultado Esperado**
Un usuario intenta iniciar sesión, pero no rellena el campo obligatorio contraseña, pero si usuario con “Sergio”.	El sistema no inicia sesión y se mantiene en la página de inicio de sesión. Muestra un error indicando que la contraseña o el usuario son incorrectos.

Tabla 6.12 Pruebas integración Inicio de sesión

Subir un documento	
Prueba	**Resultado Esperado**
Se sube el documento “Documento1.txt” al sistema.	El sistema añade el documento, mientras esto se realiza se muestra un icono de cargando y cuando termina se redirige al usuario a la pantalla documentos donde observará el nuevo documento junto al resto.
Prueba	**Resultado Esperado**
El usuario quiere subir un archivo .jpeg, pero este no está soportado por el sistema.	No se muestra el archivo desde el explorador de archivos abierto desde la aplicación, ya que esa extensión no está soportada.
Prueba	**Resultado Esperado**
El usuario se confunde y selecciona un archivo erróneo “Documento1.txt”. Cancela la acción.	No sucede nada en el sistema. Se mantiene en la pantalla subir un documento.
Prueba	**Resultado Esperado**
El usuario intenta acceder a esta ruta, pero no está autenticado.	Se redirige el usuario a la pantalla de inicio de sesión.
Prueba	**Resultado Esperado**
El usuario no selecciona ningún documento, pero presiona el botón de subir un documento.	Se mantiene en la página de subir un documento y se muestra una alerta de que no se ha selecciona ningún archivo.

Tabla 6.13 Pruebas integración Subir un documento

Eliminar un documento	
Prueba	**Resultado Esperado**
El documento "Documento1.txt" se elimina correctamente.	El sistema elimina el documento y carga la página documentos de nuevo sin el documento "Documento1.txt".
Prueba	**Resultado Esperado**
El usuario se equivoca al seleccionar el archivo "Documento3.txt" a eliminar. Cancela la acción.	Se abre el modal y el usuario presiona cancelar. Se mantiene en la página documentos.

Tabla 6.14 Pruebas integración Eliminar un documento

Descargar un documento	
Prueba	**Resultado Esperado**
El documento "Documento1.txt" se descarga correctamente.	Se muestra el archivo descargado en la pantalla inferior del navegador y también el archivo dentro de la carpeta descargas del equipo. Se mantiene en la página de documentos.
Prueba	**Resultado Esperado**
El documento "Documento2.txt" se descargar de IPFS, pero se detecta que ha sido alterado.	El sistema notifica al usuario de que su documento ha sido alterado con una alerta y descarga el archivo para el usuario. Se mantiene en la página de documentos.

Tabla 6.15 Pruebas integración Descargar un documento

Visualizar documentos	
Prueba	**Resultado Esperado**
Se muestran los documentos correctamente del usuario "Sergio" que inició sesión.	El sistema carga el nombre de los documentos asociados al usuario junto con un icono indicando el tipo de archivo y un botón de descargar y otro de eliminar documento.
Prueba	**Resultado Esperado**
El usuario "Rubén" no tiene ningún documento.	No se muestra ningún documento en la pantalla documentos.
Prueba	**Resultado Esperado**
Se pulsa en el icono de filtrar por extensión el usuario "Sergio".	Los documentos del usuario se juntan por tipo de documentos y son mostrados en grupos según su tipo. Se mantiene en la página de documentos.

Tabla 6.16 Pruebas integración Visualizar documentos

Eliminar cuenta	
Prueba	**Resultado Esperado**
El sistema elimina la cuenta correctamente de nombre de usuario "Sergio".	El sistema elimina la cuenta con nombre de usuario "Sergio" junto a todos sus documentos asociados. Se redirige al usuario a la pantalla de inicio de sesión.
Prueba	**Resultado Esperado**
El usuario selecciona eliminar cuenta de "Lausip", pero luego se arrepiente. Cancela la acción.	Se abre un diálogo de confirmación que el usuario presiona en cancelar. Se mantiene en la pantalla actual.

Tabla 6.17 Pruebas integración Eliminar cuenta

6.7.3 Pruebas de Usabilidad y Accesibilidad

En esta sección se definirán las pruebas de usabilidad y accesibilidad que se llevarán a cabo para evaluar la aplicación.

6.7.3.1 Pruebas de usabilidad

Para la realización de pruebas de usabilidad se hará uso de una serie de cuestionarios con el fin de conocer la opinión de los usuarios acerca del sistema. Este cuestionario será rellenado con respecto al sistema descrito anteriormente con todas y cada una de las pantallas.

Las indicaciones previas al uso de la aplicación han sido una pequeña introducción sobre el sistema, es decir, comentar simplemente que es un sistema de almacenaje de archivos y sirve para almacenar archivos.

Para el desarrollo de la interacción del usuario con la aplicación, se utilizará la propia máquina con la que se desarrolló el sistema y se iniciará de manera local la aplicación. En cuanto el grupo de personas a probar la aplicación, se encuentran dos mujeres, una joven de unos 22 años, con conocimientos avanzados de interfaces y buen manejo de tecnologías, y otra mayor cerca de los 60, con una buena relación con la tecnología. Por otro lado, lo probará un hombre de 22 años. De esta manera, tenemos una gran variedad de usuarios y puntos de vista.

6.7.3.2 Diseño de Cuestionarios

A continuación, se detalla el contenido del cuestionario. Cabe destacar que su realización no durará más de 15 minutos por parte del usuario.

6.7.3.2.1 Cuestionario de Evaluación

El cuestionario estará formado por los siguientes puntos:

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
- **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
- **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

6.7.3.2.2 Cuestionario para el Responsable de las Pruebas

Por otro lado, mientras los usuarios interactúan con la aplicación, el responsable de pruebas realizará un estudio y rellenará un cuestionario propio para evaluar las acciones de los usuarios. Este apartado tiene como finalidad estudiar a los usuarios y su comunicación con el usuario con el objetivo de comprender como interactúan, los puntos flacos de la aplicación, aspectos a mejorar… De esta manera se obtiene información adicional sobre los usuarios.

6.7.3.3 Actividades de las Pruebas de Usabilidad

A continuación, se muestran las actividades relacionadas con las pruebas de usabilidad.

6.7.3.3.1 Preguntas de carácter general

Las siguientes preguntas forman parte del inicio del cuestionario y están destinadas a aportar información sobre el perfil del usuario que probará la aplicación.

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Utiliza gestores documentales frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Le preocupa la privacidad y seguridad de sus documentos almacenados en la nube?
1. Me es indiferente 2. Me preocupa un poco 3. Me preocupa 4. Me preocupa mucho

Tabla 6.18 Definición cuestionario preguntas de carácter general

6.7.3.3.2 Actividades guiadas

En este apartado de las pruebas, se incluye una lista de actividades que el usuario debe realizar para probar correctamente la aplicación en su totalidad.

- Registrarse en la aplicación.
- Fallar en el registro:
  - Falta de campos requeridos.
  - Las contraseñas no coinciden.
  - La longitud de la contraseña no es la adecuada.
- Iniciar sesión
- Fallar en el inicio de sesión:
  - Falta de campos requeridos.
  - Intentar entrar con un usuario y contraseña inventados.
- Visualizar la pantalla de documentos sin haber subido ningún documento.
- Subir un documento.
- Intentar subir un documento sin adjuntar uno.
- Subir varios documentos y visualizarlos en la pantalla de documentos.
- Ordenar los documentos por orden alfabético de la A-Z.
- Ordenar los documentos por orden alfabético de la Z-A.
- Ordenar los documentos por tipo de archivo.
- Cerrar sesión y volver a iniciar sesión.
- Borrar un documento.
- Borrar un documento, pero cancelar la acción.
- Borrar la cuenta.
- Borrar la cuenta, pero cancelar la acción.

6.7.3.3.3 Preguntas Cortas sobre la Aplicación y Observaciones

En esta sección se muestra el cuestionario de preguntas cortas dirigidas al usuario. Se trata de un compendio de preguntas relacionado con la interacción del usuario con la aplicación, su opinión de la interfaz y pantallas y otras preguntas.

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
¿Se ha perdido en algún momento dentro de la aplicación?				
¿La barra de navegación le parece fácil de usar?				
¿Le parece que los elementos de la aplicación son claros?				

¿Ha identificado la finalidad de cada botón desde un inicio?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Identifica las funcionalidades de la aplicación fácilmente?				
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
¿La información sobre los documentos le ha parecido suficiente?				
¿Le ha parecido útil la disposición de los documentos en tarjetas con botones individuales?				
¿Los filtros para ordenar documentos le parecen suficientes?				
¿Siente que la búsqueda de un documento es rápida una vez se tenga una gran cantidad de estos?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
La posición de los botones es				
Como se visualizan los documentos es				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
¿Cree que debería tener más explicaciones sobre las funcionalidades la aplicación?				
Observaciones				
Cualquier comentario que el usuario quisiera compartir				

Tabla 6.19 Definición cuestionario preguntas cortas sobre la aplicación y observaciones

6.7.3.3.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	

Errores graves cometidos	
Dudas que el usuario presenta	
El usuario tiene siempre un objetivo bien definido usando la aplicación	
El usuario no parece perdido utilizando la aplicación	

Tabla 6.20 Definición cuestionario para el responsable

6.7.4 Pruebas de Accesibilidad

En cuanto a las pruebas de accesibilidad, se utilizará la herramienta de *Google Lighthouse* [63], integrada de manera nativa dentro del navegador Google Chrome, con el fin de analizar la accesibilidad de la aplicación y el programa Oracle Color [64] para analizar la aplicación con vista de personas daltónicas.

6.7.5 Pruebas de Rendimiento

Para las pruebas de rendimiento se utilizará el analizador de *Google Lighthouse* para analizar la aplicación.

Capítulo 7. Implementación del Sistema

Este capítulo trata sobre las tecnologías, estándares y normas utilizados para el desarrollo de la aplicación. Además, se comentarán los posibles problemas encontrados durante el desarrollo.

7.1 Estándares y Normas Seguidos

Este apartado tiene como objetivo definir los estándares y normas seguidos en el desarrollo del sistema. Lo primero a mencionar es que la tecnología Blockchain no presenta un estándar como tal ya que es una tecnología muy joven y pleno desarrollo. Aun así, para otros aspectos de la aplicación como el desarrollo web y la encriptación de archivos y generación de *hashes* sí que se han utilizado estándares.

Para el desarrollo de la aplicación web, se ha utilizado el patrón, bien conocido, Modelo-Vista-Controlador. Este patrón encapsula en diferentes objetos partes del desarrollo de la aplicación. Siguiendo con este apartado, la transmisión de datos se realiza mediante el protocolo HTTP, un estándar técnico que define como un cliente se comunica con un servidor. Para el envío de datos entre cliente y servidor se ha utilizado el formato JSON (*JavaScript Object Notation*), un formato ligero de intercambio de datos nativo de JavaScript. Para la parte del desarrollo de la interfaz, se ha utilizado Bootstrap, de tal manera que el resultado de esta es *responsive*, es decir, se adapta según el dispositivo donde se utiliza.

Por otro lado, MongoDB utiliza una norma propia de intercambio de datos para el almacenamiento de datos y transferencia de estos a la base de datos. Se trata del formato BSON (*Binary JavaScript Object Notation*), el cual está basado en el formato JSON.

Por último, para realizar la encriptación se los archivos se ha utilizado el estándar AES-256. Se trata de uno de los algoritmos de cifrado más utilizados y seguros que actualmente existen. En cuanto a la generación de los *hashes*, se ha utilizado el algoritmo estandarizado SHA-256.

7.2 Lenguajes

A continuación, se explican los lenguajes de programación utilizados para el desarrollo del sistema, así como su versión.

7.2.1 Lenguajes de Programación

Este apartado engloba los lenguajes de programación utilizados para el desarrollo de la aplicación.

7.2.1.1 JavaScript

JavaScript es el lenguaje principal del desarrollo de la aplicación, sobre todo en la parte del *backend*, englobando tanto la parte de encriptación como la de creación del servidor y manejo de rutas. Ya se ha descrito este lenguaje en el apartado 3.6.2.1.

7.2.1.2 Solidity

Solidity se trata de un lenguaje enfocado únicamente a la creación de contratos inteligentes para la red Blockchain Ethereum. Ya se ha descrito en el apartado 3.1.1. El sistema utilizará la versión v0.8.15.

7.2.2 Lenguajes de estilos

Este apartado engloba los lenguajes de programación utilizados para el diseño gráfico de la aplicación.

7.2.2.1 CSS

Se trata de un lenguaje de diseño gráfico muy utilizado con el objetivo de definir los estilos de un documento escrito en un lenguaje de etiquetas, como el HTML. Ya ha sido definido en el apartado 3.6.1. El sistema utilizará la versión 3.

7.2.3 Lenguajes de marcado

Este apartado describe los lenguajes de etiquetas para elaborar la estructura de un documento.

7.2.3.1 HTML

HTML será el lenguaje de marcado utilizado para diseñar y estructurar todas las pantallas de la aplicación. Ya ha sido definido en el apartado 3.6.1. Se hará uso de la versión 5.

7.2.3.2 EJS

Se trata un lenguaje de marcado que combina HTML junto con JavaScript, de tal forma que se puede generar un documento con un lenguaje de marcado en el que se puede ejecutar JavaScript plano y ofrecer una generación más dinámica de los documentos. Ya ha sido definido en el apartado 3.8.9. Se hará uso de la versión v3.1.8.

7.3 Herramientas y Programas Usados para el Desarrollo

A continuación, se definirán todas aquellas herramientas y programas utilizados para el desarrollo del sistema, así como las versiones utilizadas.

7.3.1 MongoDB

Se trata de la base de datos utilizada dentro de este proyecto. Ya ha sido definida en el apartado 3.6.2.7. Se ha escogido la versión v5.0.9.

7.3.2 MongoDB Compass

Se trata de una herramienta sencilla para visualizar la base de datos mediante una interfaz. Permite también interactuar con ella, escribir datos, modificarlos y eliminarlos, entre otras funcionalidades. Se ha escogido la versión v1.32.2.

7.3.3 Visual Studio Code

Se trata del editor de código fuente, desarrollado por Windows, escogido para la elaboración de este proyecto en su versión v1.69.0. Visual Studio Code ofrece una serie de *plugins* para instalar en el editor de tal manera que es el propio programador el que adecua el entorno de desarrollo a sus gustos.

7.3.4 PostMan

Postman es un programa que permite realizar peticiones HTTP sobre una API. Se utiliza para probar la APIs de una manera sencilla y rápida. Fue ya definido en el apartado 3.8.6. Dentro de este proyecto se ha utilizado la versión v5.5.5 del programa.

7.3.5 Nodejs

Se trata de un entorno de desarrollo de JavaScript para desarrollar aplicación de servidor principalmente. Ya fue definido en el apartado 3.6.2.2. Se ha utilizado en su versión v14.18.0.

7.3.6 Expressjs

Se trata de un framework para manejar las rutas de una aplicación, así como la creación e inicialización de un servidor de forma ágil en Nodejs. Ya ha sido definido en el apartado 3.6.2.3. Se ha usado la versión v4.18.1.

7.3.7 Trufflejs

Se trata de un framework que facilita y agiliza el desarrollo y despliegue de contratos inteligente dentro una red Blockchain. Ya ha sido definido en el apartado 3.6.2.4. Se hará uso de la versión v 5.5.21.

7.3.8 Bootstrap

Se trata de una biblioteca para el desarrollo del diseño de las interfaces de aplicaciones. Ya fue definido en el apartado 3.8.8. Se hará uso de la versión v4.3.1.

7.3.9 NPM

Se trata de un gestor de paquetes asociado a Nodejs con la finalidad de instalar y compartir paquetes de manera sencilla y ágil. Se hará uso de la versión v6.14.15.

7.4 Creación del Sistema

A pesar de haber definido un análisis y diseño de la aplicación, el surgimiento de imprevistos, problemas y momentos de atascamiento son algo natural durante el desarrollo de un proyecto. Por lo tanto, esta sección está destinada a identificar estos problemas que han sido encontrados y por último definir la descripción de las clases.

7.4.1 Problemas Encontrados

A continuación, se enumeran una serie de problemas que se han encontrados durante el desarrollo del proyecto.

7.4.1.1 Nuevos lenguajes y frameworks

Para la realización de este proyecto se han utilizado una serie de tecnologías con las que el autor no estaba muy familiarizado con ellas desde un principio.

Debido a que el autor nunca había desarrollado un proyecto utilizando el entorno de Nodejs junto Expressjs, uno de los primeros inconvenientes fue el estructurado del proyecto en carpetas, encapsulando así las diferentes partes del proyecto. Esto se debió, más que nada, por desconocimiento del entorno, pero fue solucionado con cierta agilidad.

Por otro lado, el uso de tecnologías novedosas supone un aprendizaje sobre la marcha extra. Plataformas como IPFS y la red Blockchain, requieren de una gran capacidad de abstracción sumado a horas de entendimiento y estudio sobre ellas para conseguir comprenderlas perfectamente y como estas pueden encajar dentro de un proyecto como este.

En general, la mayor parte del proyecto ha supuesto algo novedoso para el autor ya que desconocía la mayoría de las herramientas utilizadas.

7.4.1.2 Aplicación de buenas prácticas de programación

Una de las mayores preocupaciones por parte del autor es el uso de buenas prácticas en el desarrollo del proyecto. Es por ello, por lo que se ha llevado a cabo un estudio previo sobre la estructura de las clases, sobre el patrón Modelo Vista Controlador dentro de Nodejs, encapsulamiento de rutas y funcionalidades como la encriptación en clases individuales…

7.4.1.3 Comportamiento asíncrono de JavaScript

Al ser el lenguaje JavaScript un lenguaje con el que el autor no estaba familiarizado desde un principio, una serie de problemas con respecto a su naturaleza asíncrona ha surgido a lo largo del proyecto. Esta serie de problemas van desde el uso y entendimiento de los denominados *callbacks* y de cómo se debe realizar el código para que una línea no se ejecute hasta que la anterior termine su ejecución.

Uno de los mayores problemas ha sido la escritura en archivos en el lado servidor, ya que el autor se encontraba con que el cliente recibía sus archivos vacíos, pero en ciertos casos la escritura sí que realizaba. Esto se debía a que la escritura se realizaba de forma asíncrona y el cliente descargaba el archivo antes de que este fuera escrito completamente.

7.4.1.4 Diseño de la interfaz

La parte *Frontend* ha sido un gran reto ya que el autor de este documento no tiene como fuerte este aspecto del mundo de la programación y le gusta más enfocarse a la parte de *backend*. Aun así, la biblioteca de Bootstrap, junto con ejemplos, ha sido de gran ayuda para el total desarrollo del proyecto de manera satisfactoria.

7.4.2 Descripción Detallada de las Clases

Las clases creadas para la realización del proyecto pueden encontrarse dentro del código adjunto al documento. Todas las clases que conforman la lógica del sistema están documentadas con el autor definido en la parte superior del archivo. Además, todos los métodos públicos están debidamente comentados para que sea posible su entendimiento.

Capítulo 8. Desarrollo de las Pruebas

En este capítulo se muestran los resultados de las pruebas definidas previamente en el diseño.

8.1 Pruebas Unitarias

En esta sección se muestran los resultados de las pruebas unitarias realizadas con PostMan y con el framework Mocha y la librearía Supertest.

Registro de usuario		
Prueba	**Resultado Esperado**	**Superada**
Un usuario se registra con nombre de usuario y contraseña correctos.	El sistema posee un usuario más. El servidor responde al cliente con un código 200. Se genera un token guardado en la *cookie* del navegador para que se sepa que el usuario está autenticado.	Sí.
Prueba	**Resultado Esperado**	
Un usuario se intenta registrar, pero la contraseña tiene un formato incorrecto.	El sistema no posee un usuario más y se muestra un dialogo notificando el error. El servidor responde con un código 400.	Sí.
Prueba	**Resultado Esperado**	
Un usuario se intenta registrar, pero el usuario introducido ya existe dentro de la base de datos.	El sistema no posee un usuario más y se muestra un dialogo notificando el error. El servidor responde con un código 400.	Sí.

Tabla 8.1 Resultado pruebas unitarias Registro de usuario

Inicio de sesión		
Prueba	**Resultado Esperado**	**Superada**
Un usuario inicia sesión en una cuenta existente.	El sistema inicia sesión. El servidor responde con un código 200. Se genera un token guardado en la *cookie* del navegador para que se sepa que el usuario está autenticado.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Un usuario intenta acceder a una cuenta, pero esta no existe.	El sistema no inicia sesión y notifica el problema al cliente. El servidor responde con un código 400.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Un usuario intenta iniciar sesión, pero no rellena un campo obligatorio, la contraseña.	El sistema no inicia sesión y notifica el problema. El servidor responde con un código 400.	Sí.

Tabla 8.2 Resultado pruebas unitarias Inicio de sesión

Subir un documento		
Prueba	**Resultado Esperado**	**Superada**
Se sube un documento al sistema.	El sistema añade a la base de datos el identificador del documento. Responde al cliente con un código 302 para redirigir al usuario a la pantalla de documentos.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El usuario selecciona un archivo de tipo no soportado por el sistema.	El sistema no sube el documento y notifica del problema. El servidor responde con un código 409.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El usuario se confunde y selecciona y archivo erróneo. Cancela la acción.	No sucede nada en el sistema. El servidor no genera respuesta alguna.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El usuario intenta acceder a esta ruta, pero no está autenticado.	Se redirige el usuario a la pantalla de inicio de sesión. El servidor genera un código de respuesta 302.	Sí.

Tabla 8.3 Resultado pruebas unitarias Subir un documento

Eliminar un documento		
Prueba	**Resultado Esperado**	**Superada**
El documento se elimina correctamente.	El sistema elimina de la base de datos los datos sobre ese documento. El servidor responde con un código 200.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El usuario se equivoca al seleccionar el archivo a eliminar. Cancela la acción.	No sucede nada en el sistema. El servidor no responde de ningún modo.	Sí.

Tabla 8.4 Resultado pruebas unitarias Eliminar un documento

Descargar un documento		
Prueba	**Resultado Esperado**	**Superada**
El documento se descarga correctamente.	El sistema recibe la petición y descarga el archivo para el usuario.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El documento se descargar de IPFS, pero se detecta que ha sido alterado.	El sistema notifica al usuario de que su documento ha sido alterado y descarga el archivo para el usuario. El servidor responde con un código 200.	No. No se ha encontrado forma de modificar el archivo en IPFS, ya que está encriptado. Y si se vuelve a subir encriptado, su identificador sería diferente de la versión alterada.

Tabla 8.5 Resultado pruebas unitarias Descargar un documento

Visualizar documentos		
Prueba	**Resultado Esperado**	**Superada**
Se muestran los documentos correctamente.	El sistema carga el nombre de los documentos asociados al usuario iniciado en la sesión correctamente. El servidor responde con un código 200.	Sí.
Prueba	**Resultado Esperado**	**Superada**
EL usuario no tiene ningún documento.	No se muestra ningún documento en la pantalla. El servidor responde con un código 200.	Sí.

Tabla 8.6 Resultado pruebas unitarias Visualizar documentos

Eliminar cuenta		
Prueba	**Resultado Esperado**	**Superada**
El sistema elimina la cuenta correctamente.	El sistema elimina un usuario del sistema. El sistema elimina todos los documentos asociados a esa cuenta. El servidor responde con un código 200.	Sí.
Prueba	**Resultado Esperado**	**Superada**
El usuario selecciona eliminar cuenta, pero luego se arrepiente. Cancela la acción.	El sistema no hace nada. El servidor no responde de ningún amanera.	Sí.

Tabla 8.7 Resultado pruebas unitarias Eliminar cuenta

SmartContract		
Prueba	**Resultado Esperado**	**Superada**
Se sube un *hash* a la red.	Se comprueba que el *HashSet* del contrato inteligente ha aumentado en 1.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Se comprueba que un *hash* existe	El *SmartContract* devuelve el *hash* de vuelta, comprobando así que existe.	Sí.

dentro del contrato inteligente y es correcto.		
Prueba	**Resultado Esperado**	**Superada**
Se comprueba que un *hash* existe dentro del contrato inteligente, pero en realidad no existe.	El *SmartContract* devuelve una cadena vacía.	Sí.

Tabla 8.8 Resultado pruebas unitarias SmartContract

Generación de hash		
Prueba	**Resultado Esperado**	**Superada**
Se genera dos *hashes* correctamente del mismo texto.	Ambos *hashes* coinciden.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Se genera el *hash* correctamente de dos textos distintos.	No coinciden los *hashes*.	Sí.

Tabla 8.9 Resultado pruebas unitarias generar hashes

Encriptación		
Prueba	**Resultado Esperado**	**Superada**
Se encriptan dos textos distintos.	El resultado de las encriptaciones debe ser distinto.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Se encripta y desencripta un texto y se compara la desencriptación el contenido original sin haber sido encriptado.	Ambos contenidos deben coincidir.	Sí.
Prueba	**Resultado Esperado**	**Superada**
Se encripta y desencripta el mismo contenido dos veces. Ambas desencriptaciones se comparan.	Ambos contenidos desencriptados deben coincidir.	Sí.

Tabla 8.10 Resultado pruebas unitarias de encriptación

Uno de los principales problemas encontrados fue con la biblioteca Supertest y Mocha, debido a que las rutas debían incorporar un token de sesión para probar ciertas acciones en las que se requería de autenticación. Como estos casos no pudieron ser cubiertos mediante estas pruebas, se realizaron mediante PostMan a medida que se desarrollaba el proyecto y se

guardaron en una carpeta dentro de la herramienta para futuras pruebas. Esta herramienta ofrece una manera ágil y fácil de comprobar el correcto funcionamiento de la aplicación.

Se muestran a continuación las pruebas unitarias realizadas:

8.1.1.1 Resultado con PostMan

A continuación, se muestra una pantalla con las rutas utilizadas para probar toda la aplicación a medida que se iba desarrollando el sistema. Por lo tanto, no se avanzaba a una nueva funcionalidad sin comprobar el correcto funcionamiento de este.

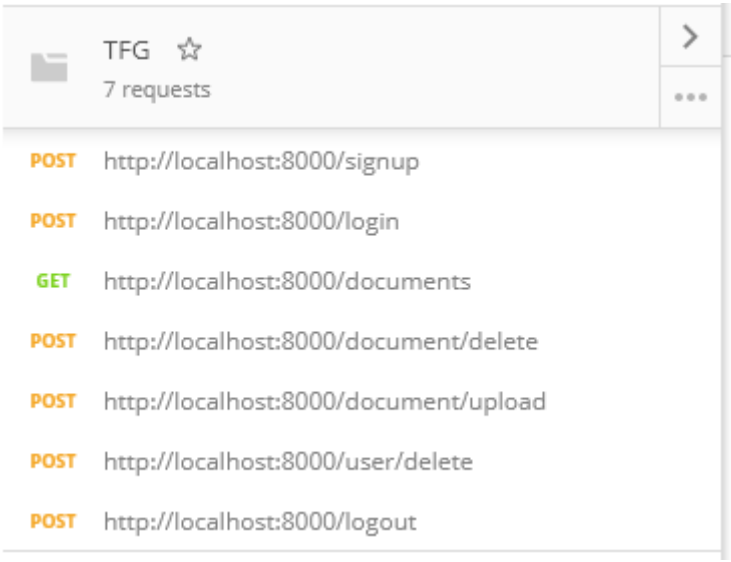

Figura 8.1 Resultado pruebas unitarias rutas PostMan

8.1.1.2 Resultado pruebas unitarias SmartContract

A continuación, se muestra un pantallazo sobre las pruebas sobre el *SmartContract*:

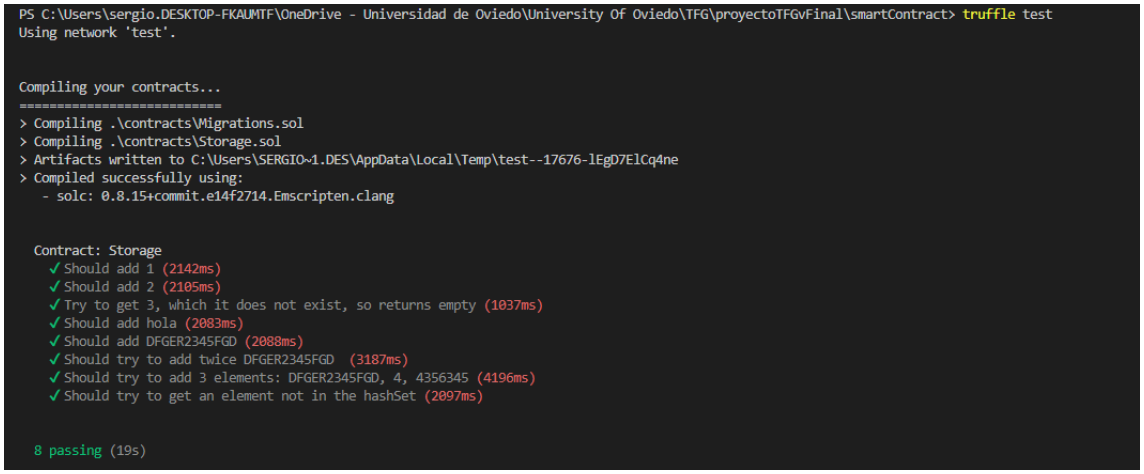

Figura 8.2 Resultado pruebas unitarias SmartContract

8.1.1.3 Resultado pruebas unitarias rutas y hashes

A continuación, se muestra un pantallazo sobre las pruebas sobre las rutas y generación de *hashes*:

```
  GET /
GET / 200 12.159 ms - 3450
    ✓ should respond a 200 status code (38ms)

  GET /login
GET /login 200 3.057 ms - 3450
    ✓ should respond a 200 status code

  GET /signup
GET /signup 200 2.425 ms - 4086
    ✓ should respond a 200 status code

  GET /sdfgsdfg
GET /sdfgsdfg 404 1.913 ms - 1051
    ✓ should respond a 404 status code

  GET /documents
token:undefined
GET /documents 302 1.850 ms - 28
    ✓ should respond a 302 status code, redirecction to login, because there is no user authenticated

  GET /documents/order/:filter
token:undefined
GET /documents/order/alphabeticDesc? 302 0.955 ms - 28
    ✓ should respond a 302 status code, redirecction to login, because there is no user authenticated

  GET /document/upload
token:undefined
GET /document/upload 302 0.821 ms - 28
    ✓ should respond a 302 status code, redirecction to login, because there is no user authenticated

  POST /document/download
token:undefined
POST /document/download 302 1.232 ms - 28
    ✓ should respond a 302 status code, redirecction to login, because there is no user authenticated

  Testing hash generation
Content: This is the content to get a hash
File hash: 609cd2ec14554632e897fe0fc3f3641ccfd298889d5b696b539a5e6d172a1d50
Content: This is the content to get a hash
File hash: 609cd2ec14554632e897fe0fc3f3641ccfd298889d5b696b539a5e6d172a1d50
    ✓ should generate the hash from the content twice, and both should match
Content: This is the content to get a hash
File hash: 609cd2ec14554632e897fe0fc3f3641ccfd298889d5b696b539a5e6d172a1d50
Content: This is another content to get a hash
File hash: 19d4027f8870d2338008e732671942bf652500fb34f8e4e92259b57b923be3be
    ✓ should generate the hash from the different contents, and should not match
    ✓ should generate the encryption from different contents, and should not match
    ✓ should encrypt and decrypt content twice, and both decrypted should match
    ✓ should encrypt content, and compare with the original so they should match


  13 passing (121ms)
```

Figura 8.3 Resultado pruebas unitarias rutas y hashes

8.2 Pruebas de Integración y del Sistema

En esta sección se llevarán a cabo las pruebas de integración y se comprobará su resultado con el esperado.

Registro de usuario	
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio” y contraseñas “pass1234”.	El sistema redirige al usuario a la página de documentos, que estará vacía ya que es un nuevo usuario y no tiene ningún documento en su cuenta.
	Resultado obtenido
	Se comprueba que verdemente no existe ningún usuario con ese nombre de usuario en la base de datos y lo crea. Redirige al usuario a la pantalla de documentos y no se muestra ningún documento ya que no tiene ninguno asociado.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio”, contraseña “pass1234” y confirmación de contraseña “DFG”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta indicando que las contraseñas no coinciden.
	Resultado obtenido
	El sistema comprueba que las contraseñas no coinciden, por lo que mantiene al usuario en la pantalla de registro y muestra una alerta indicando que las contraseñas no coinciden.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “”, contraseña “” y confirmación de contraseña “”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta que los campos nombre usuario, contraseña y confirmación de contraseña están vacío y son requeridos.
	Resultado obtenido
	El sistema mantiene al usuario en la pantalla de registro y muestra un mensaje de alerta indicando que los campos nombre de usuario, contraseña y confirmación de contraseña son requeridos.
Prueba	**Resultado Esperado**
Un usuario se registra con nombre de usuario “Sergio”, contraseña “pass1234” y confirmación de contraseña “pass1234”.	El sistema se mantiene en la pantalla de registro y muestra un mensaje de alerta que ya existe una cuenta con ese usuario.
	Resultado obtenido
	El sistema comprueba si el nombre de usuario ya existe en la base de datos. Este ya existe por lo que no puede crearse un nuevo usuario con ese nombre. El sistema mantiene al usuario en la pantalla de registro y le indica que el nombre de usuario ya existe.

Tabla 8.11 Resultado pruebas integración Registro de usuario

Inicio de sesión	
Prueba	**Resultado Esperado**
Un usuario inicia sesión en una cuenta existente con el nombre de usuario “Sergio” y contraseña “pass1234”.	El sistema inicia sesión y redirige al usuario a la pantalla Documentos para mostrar los documentos que tena en esa cuenta asociados.
	Resultado obtenido
	El sistema inicia sesión en la cuenta del usuario ya que comprueba que los datos existen en la base de datos y le redirige a la pantalla de documentos donde podrá visualizar sus documentos.
Prueba	**Resultado Esperado**
Un usuario intenta acceder a una cuenta, con los datos “Lausip” y contraseña “fdgsdfg23”, pero esta no existe.	El sistema no inicia sesión y se mantiene en la pantalla de inicio de sesión. Además, muestra un error indicando que el usuario o contraseña son incorrectos.
	Resultado obtenido
	El sistema mantiene al usuario en la pantalla de inicio de sesión debido a que comprueba que no existe ningún usuario con esas credenciales. Muestra una alerta indicando que el usuario o contraseña son incorrectos.
Prueba	**Resultado Esperado**
Un usuario intenta iniciar sesión, pero no rellena el campo obligatorio contraseña, pero si usuario con “Sergio”.	El sistema no inicia sesión y se mantiene en la página de inicio de sesión. Muestra un error indicando que la contraseña o el usuario son incorrectos.
	Resultado obtenido
	El sistema mantiene al usuario en la pantalla de inicio de sesión ya que no pueden quedar campos vacíos. El sistema muestra una alerta al usuario indicando que la contraseña es un campo requerido.

Tabla 8.12 Resultado pruebas integración Inicio de sesión

Subir un documento	
Prueba	**Resultado Esperado**
Se sube el documento “Documento1.txt” al sistema.	El sistema añade el documento, mientras esto se realiza se muestra un icono de cargando y cuando termina se redirige al usuario a la pantalla documentos donde observará el nuevo documento junto al resto.
	Resultado obtenido
	El sistema se comunica con la base de datos para que esta añada el documento adjuntado por el usuario. El sistema redirige al usuario a la pantalla documentos donde verá su nuevo documento subido.
Prueba	**Resultado Esperado**
El usuario quiere subir un archivo .jpeg, pero este no está soportado por el	No se muestra el archivo desde el explorador de archivos abierto desde la aplicación, ya que esa extensión no está soportada.

sistema.	
	Resultado obtenido
	El sistema permanece en la pantalla de subir un documento.
Prueba	**Resultado Esperado**
El usuario se confunde y selecciona un archivo erróneo "Documento1.txt". Cancela la acción.	No sucede nada en el sistema. Se mantiene en la pantalla subir un documento.
	Resultado obtenido
	El sistema permanece en la pantalla de subir un documento.
Prueba	**Resultado Esperado**
El usuario intenta acceder a esta ruta, pero no está autenticado.	Se redirige el usuario a la pantalla de inicio de sesión.
	Resultado obtenido
	El sistema redirige al usuario al inicio de sesión porque intento acceder a esta ruta sin estar autenticado.
Prueba	**Resultado Esperado**
El usuario no selecciona ningún documento, pero presiona el botón de subir un documento.	Se mantiene en la página de subir un documento y se muestra una alerta de que no se ha selecciona ningún archivo.
	Resultado obtenido
	El sistema permanece en la pantalla de subir un documento y muestra una alerta al usuario de que no se ha seleccionado ningún documento.

Tabla 8.13 Resultado pruebas integración Subir un documento

Eliminar un documento	
Prueba	**Resultado Esperado**
El documento "Documento1.txt" se elimina correctamente.	El sistema elimina el documento y carga la página documentos de nuevo sin el documento "Documento1.txt".
	Resultado obtenido
	El sistema se comunica con la base de datos, elimina el documento seleccionado y ese documento asociado al modelo de usuario. Se vuelve a cargar la pantalla de documentos y el documento borrado ya no aparece.
Prueba	**Resultado Esperado**
El usuario se equivoca al seleccionar el archivo "Documento3.txt" a eliminar. Cancela la acción.	Se abre el modal y el usuario presiona cancelar. Se mantiene en la página documentos.
	Resultado obtenido
	El sistema mantiene al usuario en la pantalla documentos.

Tabla 8.14 Resultado pruebas integración Eliminar un documento

Descargar un documento	
Prueba	**Resultado Esperado**
El documento "Documento1.txt" se descarga correctamente.	Se muestra el archivo descargado en la pantalla inferior del navegador y también el archivo dentro de la carpeta descargas del equipo. Se mantiene en la página de documentos.
	Resultado obtenido
	El sistema envía el archivo al cliente y se descarga en la máquina del usuario. El usuario puede ver su contenido correctamente. El sistema se mantiene en la pantalla de documentos.
Prueba	**Resultado Esperado**
El documento "Documento2.txt" se descargar de IPFS, pero se detecta que ha sido alterado.	El sistema notifica al usuario de que su documento ha sido alterado con una alerta y descarga el archivo para el usuario. Se mantiene en la página de documentos.
	Resultado obtenido
	No se ha podido realizar este caso ya que no se ha encontrado manera de modificar el archivo encriptado dentro de IPFS. En caso de que sucedería, sucedería lo que se espera.

Tabla 8.15 Resultado pruebas integración Descargar un documento

Visualizar documentos	
Prueba	**Resultado Esperado**
Se muestran los documentos correctamente del usuario "Sergio" que inició sesión.	El sistema carga el nombre de los documentos asociados al usuario junto con un icono indicando el tipo de archivo y un botón de descargar y otro de eliminar documento.
	Resultado obtenido
	El sistema muestra el nombre junto con un icono representando la extensión del archivo y dos botones de acción de los documentos asociados a la cuenta encontrados en la base de datos.
Prueba	**Resultado Esperado**
El usuario "Rubén" no tiene ningún documento.	No se muestra ningún documento en la pantalla documentos.
	Resultado obtenido
	No se muestra ningún documento porque en la base de datos no tiene ningún documento asociado.
Prueba	**Resultado Esperado**
Se pulsa en el icono de filtrar por extensión el usuario "Sergio".	Los documentos del usuario se juntan por tipo de documentos y son mostrados en grupos según su tipo. Se mantiene en la página de documentos.
	Resultado obtenido
	Se recarga la pantalla de documentos y se muestran estos agrupados según su tipo.

Tabla 8.16 Resultado pruebas integración Visualizar documentos

Eliminar cuenta	
Prueba	**Resultado Esperado**
El sistema elimina la cuenta correctamente de nombre de usuario "Sergio".	El sistema elimina la cuenta con nombre de usuario "Sergio" junto a todos sus documentos asociados. Se redirige al usuario a la pantalla de inicio de sesión.
	Resultado obtenido
	El sistema elimina la cuenta de usuario de la base de datos junto con todos los documentos que esa cuenta tenía asociados. Además, se redirige al usuario a la pantalla de inicio de sesión.
Prueba	**Resultado Esperado**
El usuario selecciona eliminar cuenta de "Lausip", pero luego se arrepiente. Cancela la acción.	Se abre un diálogo de confirmación que el usuario presiona en cancelar. Se mantiene en la pantalla actual.
	Resultado obtenido
	El sistema mantiene al usuario en la pantalla actual en la que estaba. No sucede nada.

Tabla 8.17 Resultado pruebas integración Eliminar cuenta

8.3 Pruebas de Usabilidad y Accesibilidad

Esta sección está destinada para mostrar los resultados de las pruebas de accesibilidad y usabilidad.

8.3.1 Pruebas de Usabilidad

A partir de los cuestionarios que se diseñaron anteriormente, en el apartado 6.7.3.2, y de los procedimientos explicados, se muestran, a continuación, los cuestionarios rellenados por los usuarios.

8.3.1.1 Persona 1

A continuación, se le pidió a un usuario con conocimientos avanzados en informática, que se encuentra familiarizado con el uso de un ordenador a diario.

8.3.1.1.1 Datos personales

- Ocupación: estudiante.
- Edad: 22.
- Sexo: mujer.

8.3.1.1.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días **X** 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión **X** 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar **X** 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo **X** 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Utiliza gestores documentales frecuentemente?
1. Todos los días 2. Varias veces a la semana **X** 3. Ocasionalmente 4. Nunca o casi nunca
¿Le preocupa la privacidad y seguridad de sus documentos almacenados en la nube?
1. Me es indiferente 2. Me preocupa un poco 3. Me preocupa **X** 4. Me preocupa mucho

Tabla 8.18 Cuestionario rellenado 1 de preguntas de carácter general

8.3.1.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	X			
¿Existe ayuda para las funciones en caso de que tenga dudas?	X			
¿Le resulta sencillo el uso de la aplicación?	X			
¿Se ha perdido en algún momento dentro de la aplicación?				X
¿La barra de navegación le parece fácil de usar?	X			
¿Le parece que los elementos de la aplicación son claros?	X			
¿Ha identificado la finalidad de cada botón desde un inicio?		X		
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Identifica las funcionalidades de la aplicación fácilmente?	X			
¿Funciona cada tarea como Vd. Espera?	X			
¿El tiempo de respuesta de la aplicación es muy grande?		X		
¿La información sobre los documentos le ha parecido suficiente?			X	
¿Le ha parecido útil la disposición de los documentos en tarjetas con botones individuales?	X			
¿Los filtros para ordenar documentos le parecen suficientes?	X			
¿Siente que la búsqueda de un documento es rápida una vez se tenga una gran cantidad de estos?				X

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es		X		
Los iconos e imágenes usados son	X			
Los colores empleados son		X		
La posición de los botones es		X		
Como se visualizan los documentos es		X		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?				X

¿Cree que el programa está bien estructurado?	X		
¿Cree que debería tener más explicaciones sobre las funcionalidades la aplicación?	X		
Observaciones			
A la hora de descargar un documento, se debería mostrar un *feedback* al usuario sobre si el documento está siendo descargado. Me gustaría que hubiera un buscador de documentos por nombre. Me gustaría que se mostrase la fecha de subida del documento.			

Tabla 8.19 Cuestionario rellenado 1 preguntas cortas sobre la aplicación y observaciones

8.3.1.1.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Comienza de una forma ágil y con un objetivo claro.
Tiempo en realizar cada tarea	Manejo rápido de los componentes de la aplicación.
Errores leves cometidos	Presionar el botón subir el archivo sin haber seleccionado un archivo adjunto previamente.
Errores graves cometidos	Ninguno.
Dudas que el usuario presenta	Ninguna.
El usuario tiene siempre un objetivo bien definido usando la aplicación	Sí.
El usuario no parece perdido utilizando la aplicación	El usuario se maneja en todo momento sabiendo donde está situado.

Tabla 8.20 Cuestionario rellenado 1 responsable de las pruebas

8.3.1.2 Persona 2

A continuación, se le pidió a un usuario con conocimientos medio en tecnología y ofimática, que se encuentra familiarizado con el uso de un ordenador a diario.

8.3.1.2.1 Datos personales

- Ocupación: activa.
- Edad: 50.
- Sexo: mujer.

8.3.1.2.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días X 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?

1. Es parte de mi trabajo o profesión **X** 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar **X** 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias **X**
¿Utiliza gestores documentales frecuentemente?
1. Todos los días 2. Varias veces a la semana **X** 3. Ocasionalmente 4. Nunca o casi nunca
¿Le preocupa la privacidad y seguridad de sus documentos almacenados en la nube?
1. Me es indiferente 2. Me preocupa un poco 3. Me preocupa **X** 4. Me preocupa mucho

Tabla 8.21 Cuestionario rellenado 2 de preguntas de carácter general

8.3.1.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	**X**			
¿Existe ayuda para las funciones en caso de que tenga dudas?			**X**	
¿Le resulta sencillo el uso de la aplicación?		**X**		
¿Se ha perdido en algún momento dentro de la aplicación?				**X**
¿La barra de navegación le parece fácil de usar?		**X**		
¿Le parece que los elementos de la aplicación son claros?		**X**		
¿Ha identificado la finalidad de			**X**	

cada botón desde un inicio?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Identifica las funcionalidades de la aplicación fácilmente?		X		
¿Funciona cada tarea como Vd. Espera?		X		
¿El tiempo de respuesta de la aplicación es muy grande?	X			
¿La información sobre los documentos le ha parecido suficiente?				X
¿Le ha parecido útil la disposición de los documentos en tarjetas con botones individuales?	X			
¿Los filtros para ordenar documentos le parecen suficientes?			X	
¿Siente que la búsqueda de un documento es rápida una vez se tenga una gran cantidad de estos?				X

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es			X	
Los iconos e imágenes usados son		X		
Los colores empleados son		X		
La posición de los botones es		X		
Como se visualizan los documentos es		X		

Diseño de la Interfaz	**Si**	**No**	**A veces**
¿Le resulta fácil de usar?	X		
¿El diseño de las pantallas es claro y atractivo?		X	
¿Cree que el programa está bien estructurado?	X		
¿Cree que debería tener más explicaciones sobre las funcionalidades la aplicación?			X

Observaciones
El principal defecto que encuentro, a nivel personal, es la estética de la página, ya que la encuentro un tanto fea. Las funcionalidades presentes me parecen correctas pero la búsqueda de un documento y los filtros ofrecidos me parecen insuficientes. También me gustaría que los tiempos de subida de un documento fueran más cortos.

Tabla 8.22 Cuestionario rellenado 2 preguntas cortas sobre la aplicación y observaciones

8.3.1.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Sí, se nota que está adecuado a este tipo de aplicaciones.
Tiempo en realizar cada tarea	Manejo rápido de los componentes de la aplicación.
Errores leves cometidos	Ninguno.
Errores graves cometidos	Ninguno.
Dudas que el usuario presenta	Ninguna.
El usuario tiene siempre un objetivo bien definido usando la aplicación	Sí, siempre sabe que quiere hacer.
El usuario no parece perdido utilizando la aplicación	El usuario se maneja en todo momento sabiendo donde está situado.

Tabla 8.23 Cuestionario rellenado 2 responsable de las pruebas

8.3.1.3 Persona 3

A continuación, se le pidió a un usuario con conocimientos medios en informática, que se encuentra familiarizado con el uso de un ordenador de manera ocasional.

8.3.1.3.1 Datos personales

- Ocupación: estudiante.
- Edad: 22.
- Sexo: hombre.

8.3.1.3.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente **X** 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio **X** 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca **X**
¿Qué busca Vd. Principalmente en un programa?

1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido **X** 4. Que tenga todas las funciones necesarias
¿Utiliza gestores documentales frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca **X**
¿Le preocupa la privacidad y seguridad de sus documentos almacenados en la nube?
1. Me es indiferente 2. Me preocupa un poco **X** 3. Me preocupa 4. Me preocupa mucho

Tabla 8.24 Cuestionario rellenado 3 de preguntas de carácter general

8.3.1.3.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?	**X**			
¿Existe ayuda para las funciones en caso de que tenga dudas?			**X**	
¿Le resulta sencillo el uso de la aplicación?		**X**		
¿Se ha perdido en algún momento dentro de la aplicación?				**X**
¿La barra de navegación le parece fácil de usar?	**X**			
¿Le parece que los elementos de la aplicación son claros?		**X**		
¿Ha identificado la finalidad de cada botón desde un inicio?			**X**	
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Identifica las funcionalidades de la aplicación fácilmente?		**X**		
¿Funciona cada tarea como Vd. Espera?	**X**			
¿El tiempo de respuesta de la aplicación es muy grande?		**X**		
¿La información sobre los documentos le ha parecido suficiente?		**X**		
¿Le ha parecido útil la disposición	**X**			

de los documentos en tarjetas con botones individuales?				
¿Los filtros para ordenar documentos le parecen suficientes?		X		
¿Siente que la búsqueda de un documento es rápida una vez se tenga una gran cantidad de estos?			X	
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son		X		
La posición de los botones es	X			
Como se visualizan los documentos es		X		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?				X
¿Cree que el programa está bien estructurado?		X		
¿Cree que debería tener más explicaciones sobre las funcionalidades la aplicación?		X		
Observaciones				
Es fácil navegar por la aplicación, pero las funcionalidades, si no conoces de antemano de que trata la aplicación puedes estar un tanto despistado, ya que no expresa explícitamente que hace la aplicación. Me hubiera gustado haber visto alguna introducción sobre la aplicación en algún momento.				

Tabla 8.25 Cuestionario rellenado 3 preguntas cortas sobre la aplicación y observaciones

8.3.1.3.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	Presenta dificultades al principio, pero al ser una aplicación sencilla se maneja mejor con el paso del tiempo.
Tiempo en realizar cada tarea	El esperado para este tipo de perfil poco tecnológico.
Errores leves cometidos	No ser paciente a la hora de la descarga de un archivo.
Errores graves cometidos	Ninguno.
Dudas que el usuario presenta	Ninguna.
El usuario tiene siempre un objetivo bien definido usando la aplicación	Sí, pero en un inicio se presenta algo perdido sobre de que trata la aplicación.
El usuario no parece perdido utilizando la aplicación	Siempre sabe dónde está en cada momento, pero en el momento inicial puede no haber identificado el objetivo de toda la aplicación, en la parte del registro.

Tabla 8.26 Cuestionario rellenado 3 responsable de las pruebas

8.3.1.4 Resultados de los cuestionarios

En líneas generales, los usuarios se han manejado de una manera correcta utilizando la aplicación, alguno de una manera más veloz que otro. No han encontrado problema alguno a la hora de subir y visualizar los documentos. El icono del perfil colocado en la parte superior a la derecha es muy intuitivo por ser un factor común con la mayoría de las páginas web, lo cual ayuda a la adaptabilidad del usuario a una nueva página. Aun así, botones de ordenar documentos pueden confundir, aunque el *tooltip* que presentan ayudan generosamente a entender su propósito.

En cuanto a los cambios realizados en la aplicación, está la introducción de un párrafo de introducción sobre la aplicación dentro de la pantalla de registro, para explicar la finalidad de la aplicación y sus objetivos generales. También, sirve como eslogan y gancho para nuevos usuarios. Se trata de la oración “Tu gestor de archivos seguro, privado y descentralizado”.

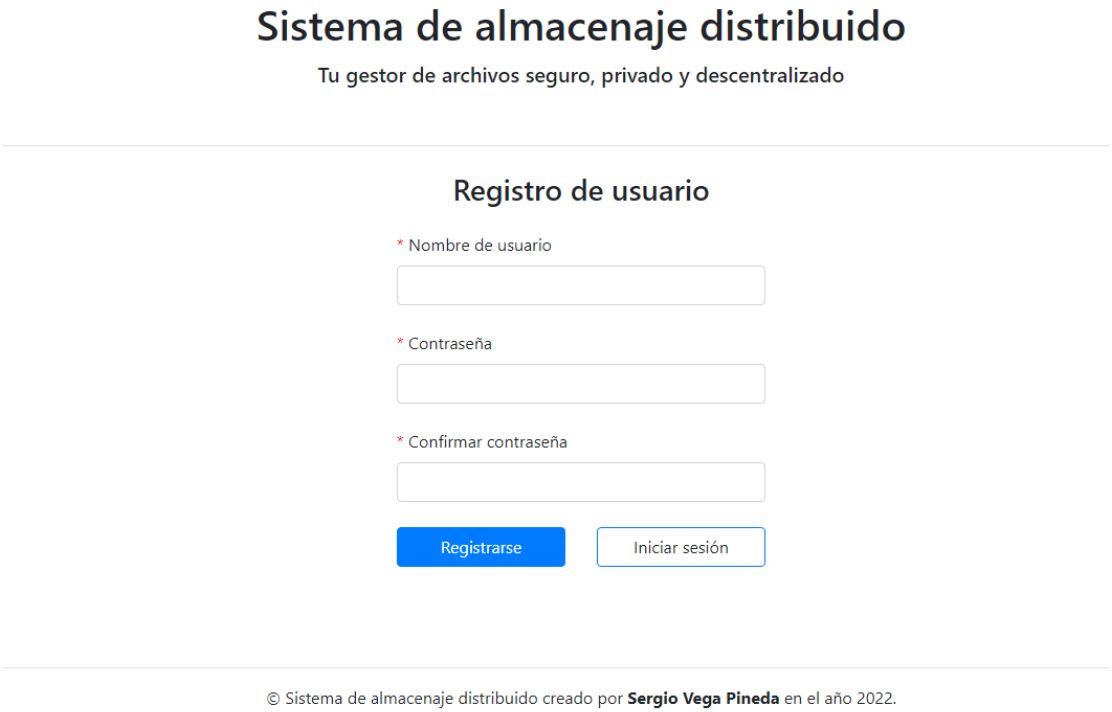

Figura 8.4 Nueva pantalla de registro

Por otro lado, se ha indicado un texto pequeño indicando que no hay ningún documento subido por parte del usuario en la plataforma. Se muestra la oración “No tiene ningún documento subido actualmente”.

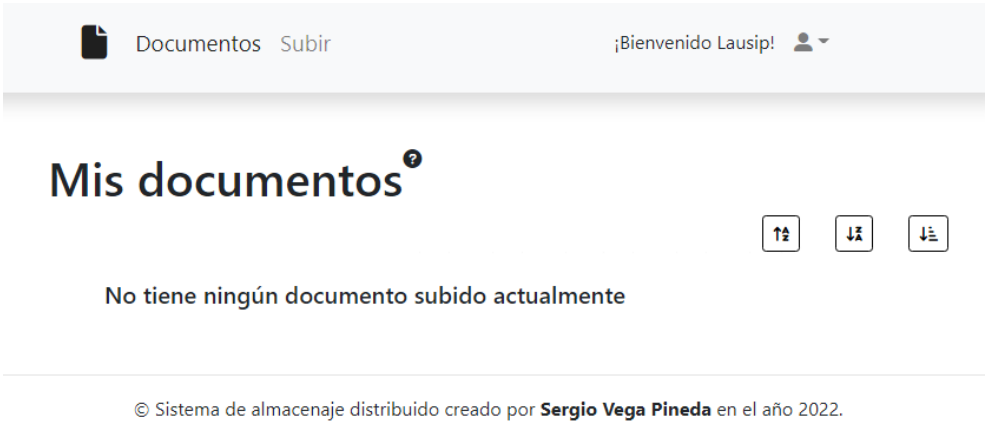

Figura 8.5 Pantalla documentos sin documentos subidos

La realización de un buscador es una muy buena recomendación que se tomará en cuenta para en un futuro ser implementada. Lo mismo para la introducción de más información sobre los documentos, como la fecha de subida o su tamaño.

También, se ha notado que la subida de documentos es un poco pesada, ya que hay que realizarlo de uno en uno. Por lo tanto, subir varios documentos de manera simultánea en un futuro es una funcionalidad que se podría implementar.

8.3.2 Pruebas de Accesibilidad

Las pruebas de accesibilidad se dividirán en dos partes. Por un lado, las pruebas realizadas con la herramienta *Google Lighthouse,* y por otro, pruebas de daltonismo mediante Oracle Color.

8.3.2.1 Análisis con Google Lighthouse

Debido a que el número de pantallas es bastante bajo, se puede pasar el analizador *Google Lighthouse* en cada una de ellas para obtener una puntuación de accesibilidad, entre otras.

8.3.2.1.1 Pantalla registro

La puntuación de accesibilidad ofrecida por la herramienta es de 93 sobre 100. Además, se han reportado dos inconvenientes:

- Los colores de fondo y de primer plano no tienen una relación de contraste adecuada.
- Los elementos de título no aparecen en orden secuencial descendente.

Figura 8.6 Puntuación de accesibilidad pantalla registro

8.3.2.1.2 Pantalla inicio de sesión

La puntuación de accesibilidad ofrecida por la herramienta es de 93 sobre 100. Además, se han reportado dos inconvenientes:

- Los colores de fondo y de primer plano no tienen una relación de contraste adecuada.
- Los elementos de título no aparecen en orden secuencial descendente.

Figura 8.7 Puntuación de accesibilidad pantalla iniciar sesión

8.3.2.1.3 Pantalla de visualización de documentos

La puntuación de accesibilidad ofrecida por la herramienta es de 86 sobre 100. Además, se han reportado cuatro inconvenientes:

- Los atributos [aria-*] no se corresponden con sus funciones.
- Los colores de fondo y de primer plano no tienen una relación de contraste adecuada.
- Los enlaces no tienen nombres reconocibles.
- Las listas no contienen únicamente elementos <li> y elementos que admiten secuencias de comandos (<script> y <template>).

Figura 8.8 Puntuación de accesibilidad pantalla documentos

8.3.2.1.4 Pantalla de subir un documento

La puntuación de accesibilidad ofrecida por la herramienta es de 86 sobre 100. Además, se han reportado cuatro inconvenientes:

- Los elementos de formulario no tienen ninguna etiqueta asociada.
- Los colores de fondo y de primer plano no tienen una relación de contraste adecuada.
- Los enlaces no tienen nombres reconocibles.
- Las listas no contienen únicamente elementos <li> y elementos que admiten secuencias de comandos (<script> y <template>).

Figura 8.9 Puntuación de accesibilidad pantalla subir documento

8.3.2.2 Mejoras aplicadas a Google Lighthouse

A continuación, se procede a mostrar las mejoras aplicadas a los resultados previos del *Google Lighthouse*.

8.3.2.2.1 Mejora de la pantalla de inicio de sesión y registro

Tras aplicar las siguientes mejoras, se ha obtenido una puntuación de 100 sobre 100 de accesibilidad con la herramienta *Google Lighthouse* en las pantallas de inicio de sesión y registro. Para resolver los problemas, se han realizado los siguientes pasos:

- Se han ajustado las etiquetas <h>, de tal manera que los títulos se indican de manera descendente empezando por el 6. Previamente se hacía un salto de más de 1 valor desde el título principal <h1>, hasta el siguiente que existiera, siendo este <h5>.
    - Se ha cambiado la combinación de colores del botón principal izquierdo, Iniciar sesión y Registrarse. De esta manera, se obtiene un nivel de accesibilidad AAA, establecido por la norma WCAG 2.1.

Figura 8.10 Mejora de accesibilidad pantallas Inicio sesión y registro

A continuación, se muestra la nueva apariencia en cuanto al estilo de la página se refiere.

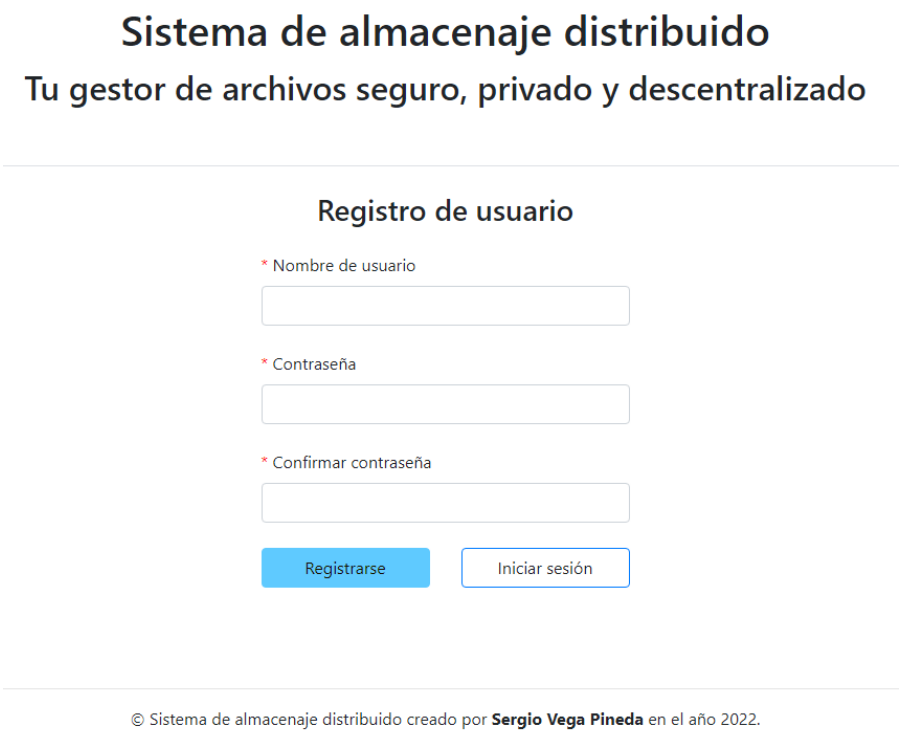

Figura 8.11 Pantalla de registro con mejoras de accesibilidad

8.3.2.2.2 Mejora de la pantalla de documentos

Tras aplicar las siguientes mejoras, se ha obtenido una puntuación de 90 sobre 100 de accesibilidad con la herramienta *Google Lighthouse* en la pantalla de documentos. Para resolver los problemas, se han realizado los siguientes pasos:

- Se ha cambiado el color de la letra del botón de descargar de todos los documentos para hacerlo negro para obtener el nivel de accesibilidad AAA.

- Se ha cambiado el rojo del botón eliminar documentos y eliminar cuenta de la barra de navegación para obtener el nivel de accesibilidad AAA.
- Se ha disminuido la opacidad de la letra de los elementos de la barra de navegación y aumentado su grado de oscuridad para llegar a obtener el nivel de accesibilidad AAA, establecido por la norma WCAG 2.1.

Figura 8.12 Mejora de accesibilidad pantalla documentos

A continuación, se muestra la nueva interfaz de la pantalla documentos:

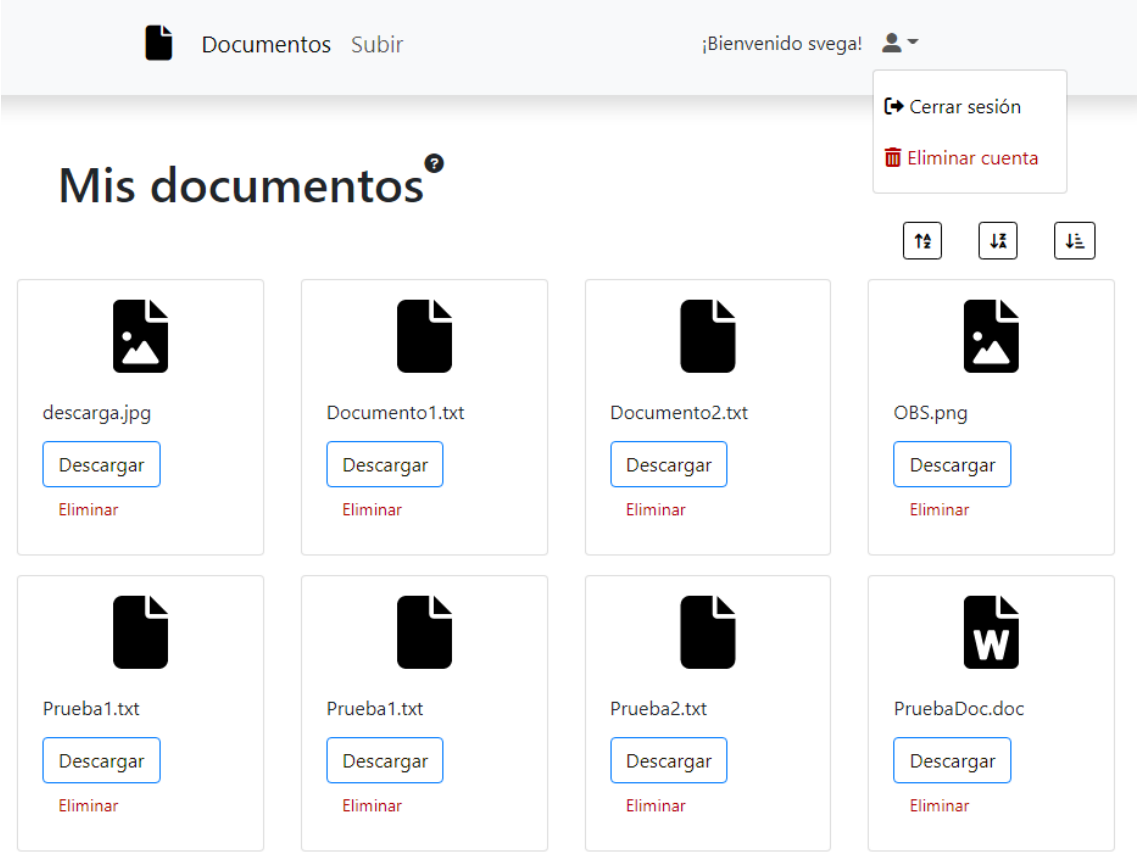

Figura 8.13 Mejora pantalla documentos

8.3.2.2.3 Mejora de la pantalla de subir documento

Tras aplicar las siguientes mejoras, se ha obtenido una puntuación de 90 sobre 100 de accesibilidad con la herramienta *Google Lighthouse* en la pantalla de documentos. Para resolver los problemas, se han realizado los siguientes pasos:

- Se ha cambiado el color de la letra del botón subir archivo para obtener el nivel de accesibilidad AAA.
- Se han realizado los mismos cambios obre la barra de navegación que en el apartado anterior 8.3.2.2.2.
- Se ha quitado la opacidad del elemento “Extensiones: .pdf .doc .zip .rar .txt .png .jpg” y se ha configurado el color a más oscuro de tal manera que se obtenga el nivel de accesibilidad AAA.

A continuación, se muestra la nueva apariencia en cuanto al estilo de la página se refiere.

Figura 8.14 Mejora subir documento

8.3.2.3 Análisis con Oracle Color

Por otro lado, también se han realizado pruebas con el programa Oracle Color, el cual permite aplicar filtros de colores simulando diferentes versiones de daltonismo: deuteranopia (dificultad para distinguir el color verde), protanopia (dificultad para distinguir el color rojo), y tritanopia (dificultad para distinguir el azul).

8.3.2.3.1 Deuteranopia

Se ha aplicado el filtro de deuteranopia al inicio de sesión y a la pantalla de documentos.

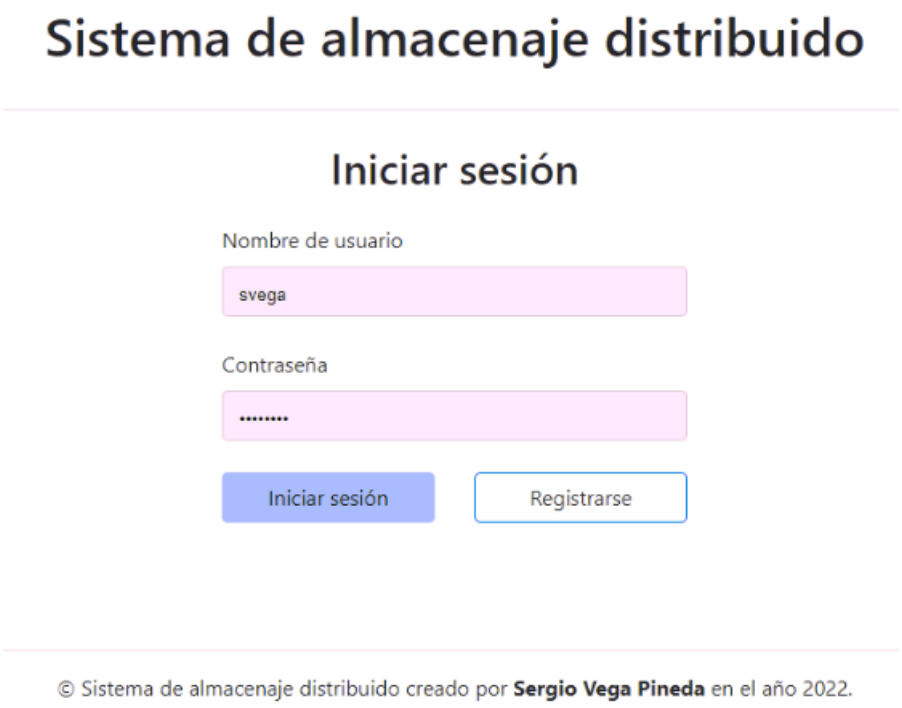

Figura 8.15 Filtro deuteranopia aplicado en la pantalla inicio de sesión

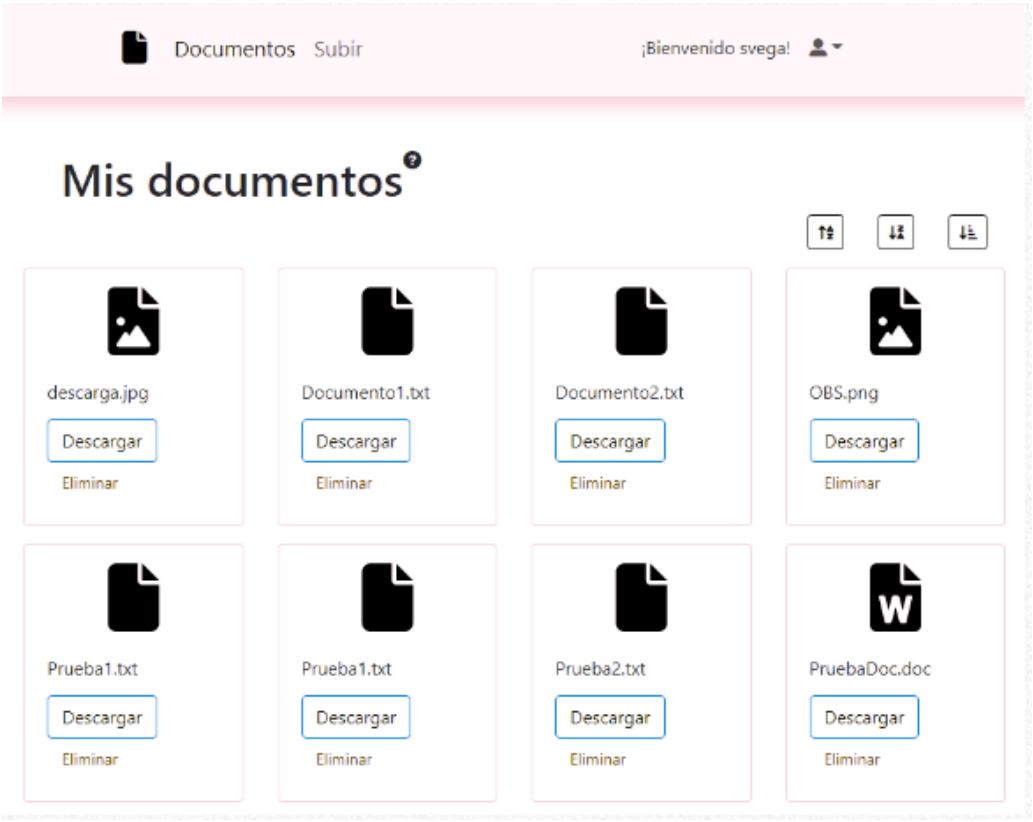

Figura 8.16 Filtro deuteranopia aplicado en la pantalla documentos

8.3.2.3.2 Protanopia

Se ha aplicado el filtro de protanopia al registro y a la pantalla de documentos.

Figura 8.17 Filtro protanopia aplicado a la pantalla registro

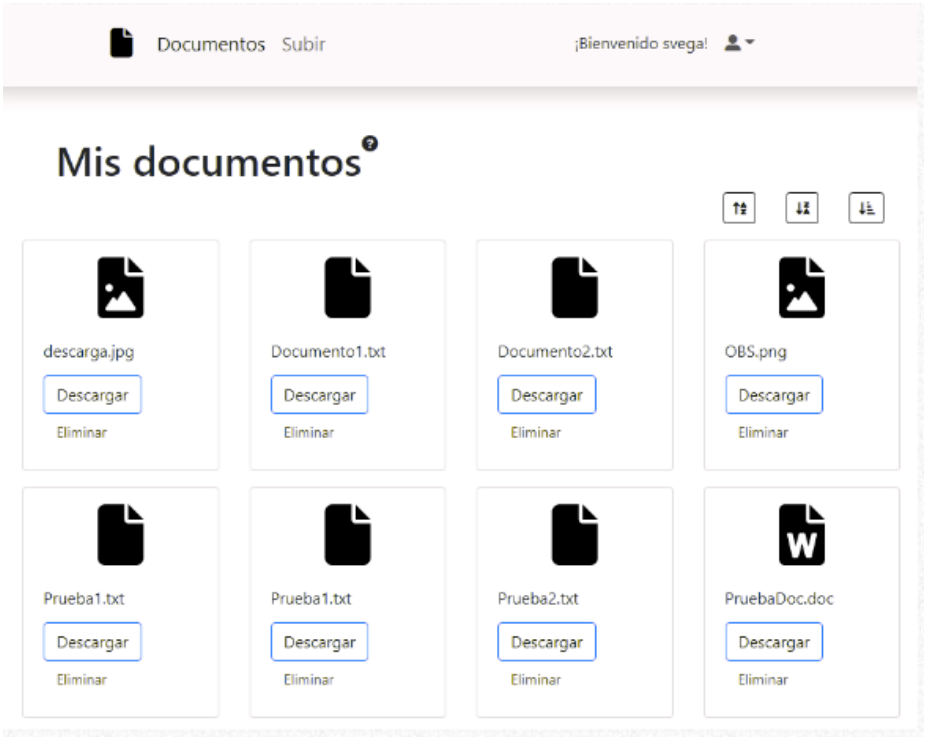

Figura 8.18 Filtro protanopia aplicado a la pantalla documentos

8.3.2.3.3 Tritanopia

Se ha aplicado el filtro de tritanopia a la pantalla de documentos y a la pantalla subir un documento.

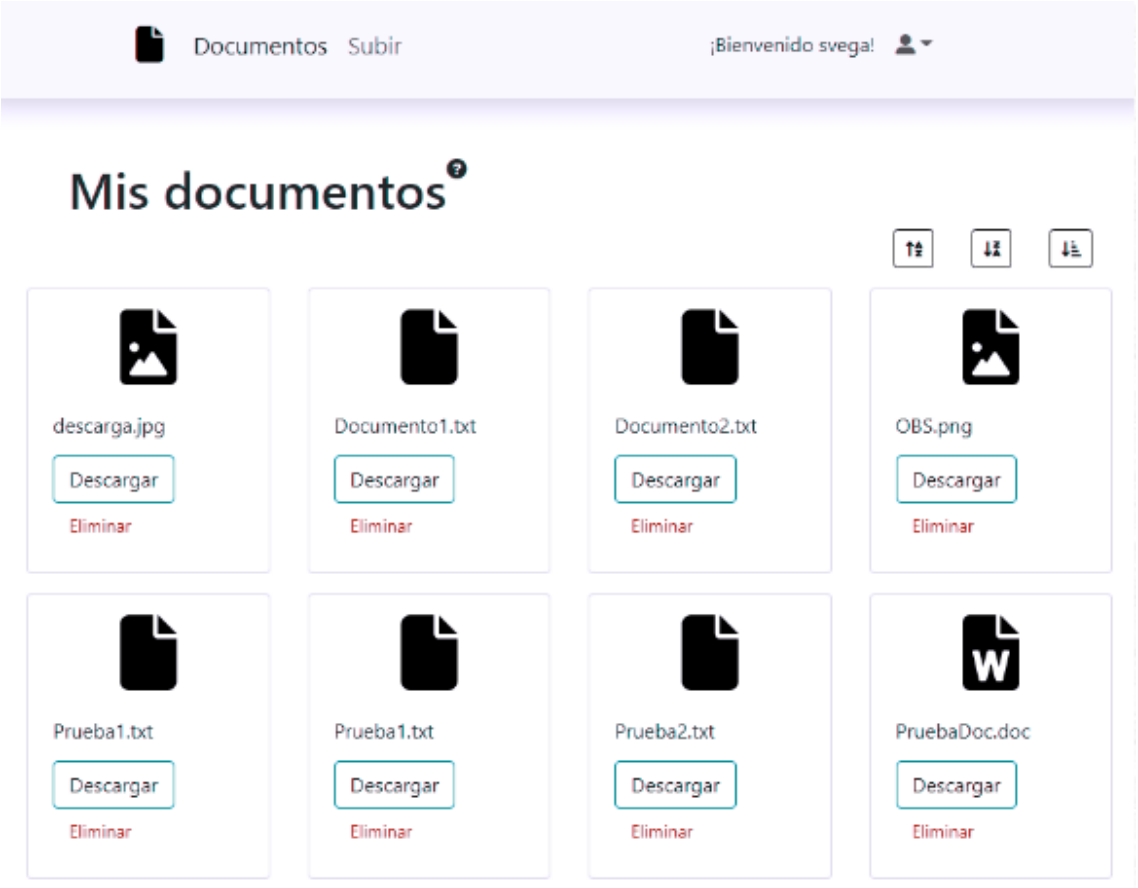

Figura 8.19 Filtro tritanopia aplicado a la pantalla documentos

Figura 8.20 Filtro tritanopia aplicado a la pantalla subir documento

8.3.2.4 Accesibilidad con Dispositivos Móviles

Cabe destacar que la aplicación web es *responsive*, por lo que se adapta al ancho del dispositivo en el que se está mostrando. Esto se debe, principalmente, gracias al uso de la biblioteca Boostrap y a ciertos cambios puntuales en el HTML y CSS por parte del autor.

8.4 Pruebas de Rendimiento

Para la elaboración de las pruebas de rendimiento se volverá a hacer uso de la herramienta *Google lighthouse*. Esta herramienta indica una serie de métricas con respecto al rendimiento que ayuda a entender este atributo dentro de la aplicación desarrollada.

8.4.1 Pruebas de Rendimiento

A continuación, se muestran las métricas de rendimiento explicadas en *Google Lighthouse*.

Figura 8.21 Métricas de rendimiento Google Lighthouse

8.4.2 Pantalla Iniciar sesión

En esta pantalla se ha obtenido una puntuación 100 de 100.

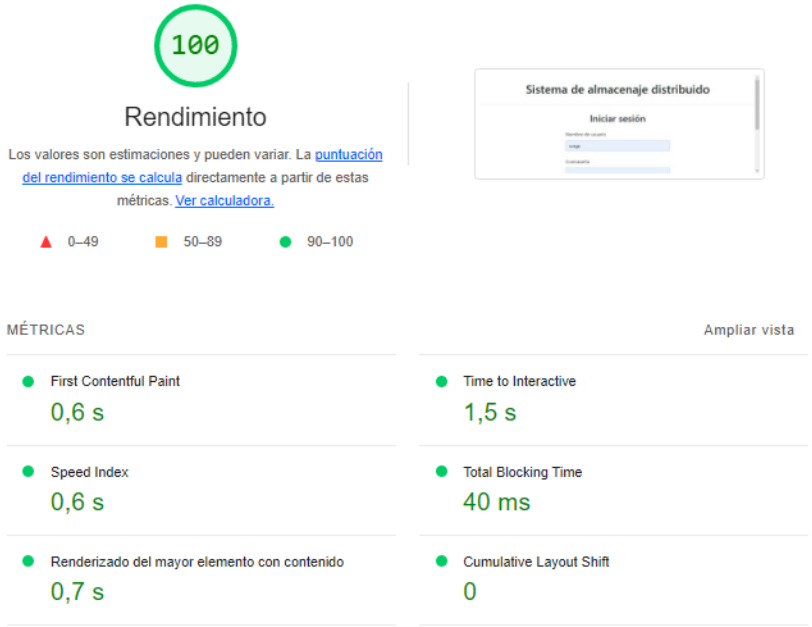

Figura 8.22 Rendimiento pantalla iniciar sesión

8.4.3 Pantalla Registro

En esta pantalla se ha obtenido una puntuación 100 de 100.

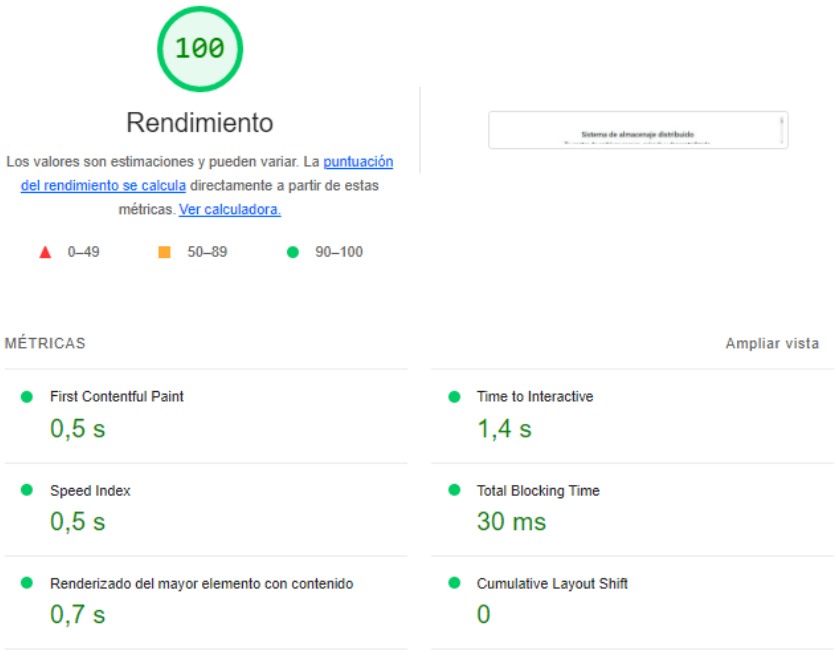

Figura 8.23 Rendimiento pantalla registro

8.4.4 Pantalla Documentos

En esta pantalla se ha obtenido una puntuación 97 de 100.

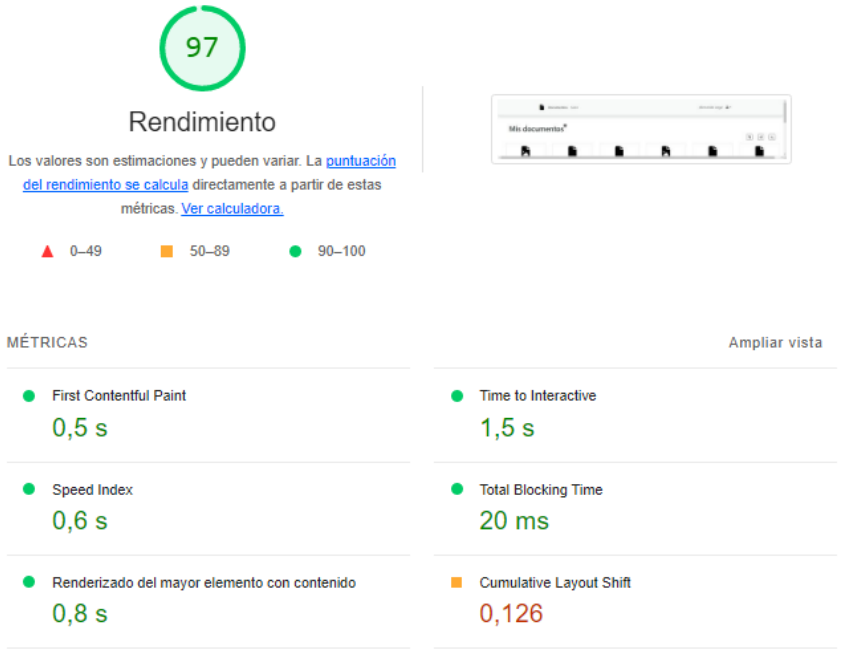

Figura 8.24 Rendimiento pantalla documentos

8.4.5 Pantalla Subir documento

En esta pantalla se ha obtenido una puntuación 99 de 100.

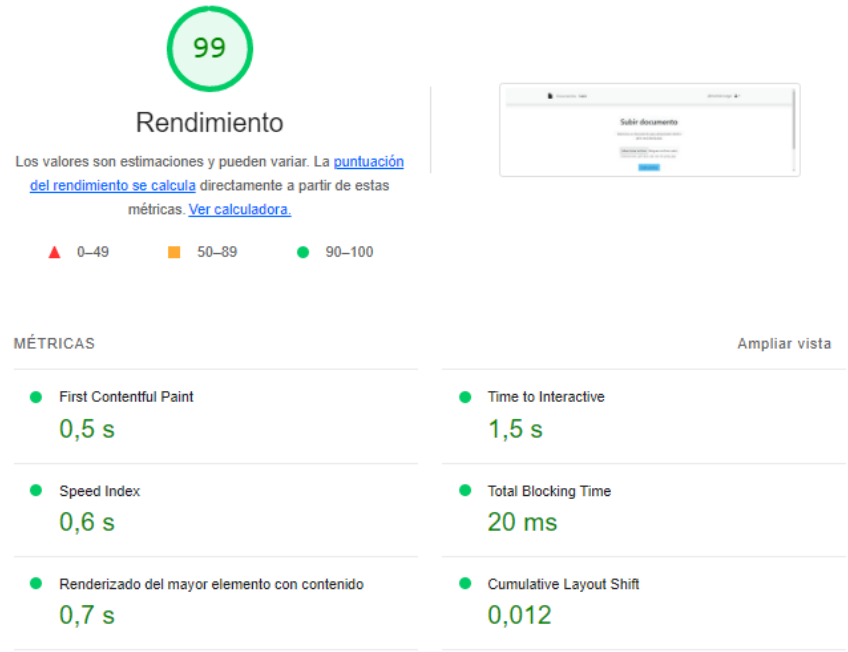

Figura 8.25 Rendimiento pantalla subir un documento

8.4.6 Resultados

Como ha podido observarse en los resultados anteriores, el rendimiento general de todas las pantallas de la aplicación es prácticamente perfecto, con una puntuación mínima de 97 en una de las pantallas.

Capítulo 9. Manuales del Sistema

En este capítulo se indicarán las herramientas y programas necesarios para instalar el sistema, un manual de ejecución de este y un manual para los usuarios que interactúen con la aplicación.

9.1 Manual de Instalación

En esta sección se expondrán las herramientas y programas que deben estar instalados, así como las configuraciones necesarias para poder, más adelante, ejecutar el proyecto.

Primero de todo, las siguientes instrucciones están elaboradas sobre un sistema operativo Windows 10 de 64 bits, pero este proyecto debería poder instalarse y ejecutarse en otros tipos de sistemas operativos como Ubuntu. Una vez dicho esto, se necesitará instalar en la máquina los siguientes programas y herramientas:

- Npm v6.14.15: https://www.npmjs.com/
- MongoDB v5.0.9: https://www.mongodb.com/try/download/enterprise
- Visual Studio Code (cualquier versión, también se puede usar cualquier otro editor de texto que se prefiera): https://code.visualstudio.com/
- Nodejs v14.18.0 (como mínimo): https://nodejs.org/es/download/
- Código fuente adjunto a la documentación descargado y descomprimido.

Una vez ya instalado todo, se debe abrir, dentro del editor de texto, en este caso Visual Studio Code, la carpeta que contiene el proyecto. Dentro de Visual Studio Code, se debe abrir una nueva terminal y escribir en la línea de comando el siguiente comando “npm i”. Este comando instalará dentro del proyecto todas las librerías y frameworks necesarias para la ejecución del proyecto, las cuales están definidas dentro del archivo “package.json”.

Por otro lado, si se quiere obtener una vista gráfica de la base de datos, se puede instalar la herramienta MongoDB Compass, https://www.mongodb.com/try/download/compass, en su versión v1.32.2 (*stable*). En caso de que una vez instalada la base de datos MongoDB y el proyecto no sea capaz de conectarse y MongoDB Compass tampoco, será necesario iniciarla. Para ello, se escribirá en el buscador de Windows “servicios”. Se abrirá entonces una nueva pestaña, en la cual debemos encontrar la aplicación de MongoDB. Una vez encontrada, será necesario clicar botón derecho sobre le nombre e iniciar. De esta manera se iniciará y ejecutará la base de datos de forma local dentro de la máquina en la que se instaló.

Una vez realizada toda la instalación, solo quedarían las configuraciones. Para ello, será necesario abrir el archivo “. env” del proyecto. Este archivo contiene todas las variables globales necesarias para el correcto funcionamiento de la aplicación. Aunque no sería necesario cambiar nada, se explicarán las diferentes variables y de donde se consiguen.

Dentro de este archivo, se podrán configurar todas las claves secretas que se usan para encriptar los archivos y generar *hashes*, aunque debe ser cuidadoso sobre cómo y cuándo se

realiza esta acción. Por otro lado, se ha de configurar el puerto en el que se quiera lanzar la aplicación y la Url de la base de datos, que en principio será una Url *localhost*, ya que presenta una arquitectura *on-premise*.

Por otro lado, está la variable "RINKEBY_NETWORK", la cual contiene la API que ofrece Infura. Para ello, será necesario registrarse en esta plataforma, crear un proyecto Ethereum y obtener el Url de la red Rinkeby de tal proyecto. De esta manera se puede comunicar el proyecto con la red Blockchain de Ethereum Rinkeby. Lo mismo con la variable "WEB3_STORAGE", la cual se trata de una Url que conecta el proyecto con la plataforma Web3Storage. Para obtenerla, será necesaria una cuenta en esta plataforma y obtener la Url desde la página API tokens de Web3Storage.

Por último, es necesaria una cuenta o *wallet*, en la red de Ethereum. Para ello, será necesario instalar dentro del navegador de Google Chrome la extensión Metamask. Una vez instalada, se deberá iniciar y seguir los pasos para crear una *wallet*, y dentro de la variable "MNEMONIC" se introducirán las palabras que Metamask indican que sirven para recuperar la *wallet*. Además, una vez ya en posesión de la *wallet*, se deberá copiar la dirección de la cuenta dentro de la variable "ETH_ACCOUNT" y la clave privada de la cuenta, que se consigue dentro de detalles de la cuenta, dentro de la variable "ETH_PK". De esta manera, ya solo quedaría lena de fondos la *wallet* para poder hacer uso de la aplicación, ya que cada transacción dentro de la red cuesta dinero. Por lo tanto, será necesario copiar la dirección de la wallet y seguir los pasos de la siguiente página https://rinkebyfaucet.com/.

Como algo adicional, si se quiere comenzar la aplicación con la base de datos llena, se deberá consultar el archivo "HowToImport.txt" dentro del Directorio "MongoDB" del archivo adjunto. Ahí se explica como importar los datos y las credenciales de inicio de sesión para visualizar los datos.

9.2 Manual de Ejecución

En esta sección se expondrán las pautas para ejecutar la aplicación. Para poder arrancar el servidor, se deberá abrir la carpeta contenedora del proyecto dentro de Visual Studio Code. Una vez dentro, será necesario abrir una nueva terminal y en ella ejecutar el comando "node app.js", desde el directorio raíz del proyecto. Esto hará que se ejecuta y arranque el servidor.

Para el lado cliente, será necesario un navegador web, como Google Chrome. En el buscador se pegará la ruta donde el servidor está ejecutándose, en este caso, "http://localhost:8000/". De esta manera, el servidor estará ejecutándose al mismo que el cliente mostrará el contenido del sistema para que el usuario interactúe con ella.

9.3 Manual de Usuario

En esta sección se expondrá el manual de uso de la aplicación para usuarios.

9.3.1 Iniciar sesión y registro

Para comenzar con la aplicación, la ruta más sencilla existente es "/", la cual abre la pantalla de inicio de sesión.

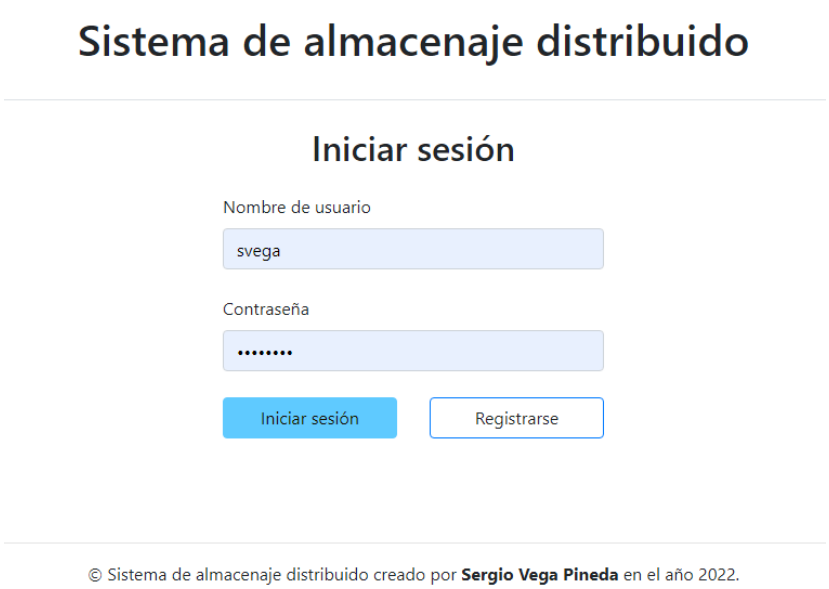

Figura 9.1 Pantalla Inicio sesión

Como puede observarse, existen dos botones y dos campos rellenar obligatorios. En caso de que el usuario tenga una cuenta ya registrada, deberá introducir sus credenciales y clicar en el botón iniciar sesión. En caso de que no tenga una cuenta ya registrada, deberá pulsar el botón registrarse, el cual le llevará a la pantalla de registro. En caso de que el usuario presione el botón iniciar sesión y los datos introducidos sean erróneos, se le notificará del error a través de alertas.

Figura 9.2 Pantalla alertas inicio de sesión

Dentro de la pantalla de registro, se muestran 3 campos obligatorios a rellenar por parte del usuario. Una vez rellenados, podrá pulsar el botón registrarse para crear la nueva cuenta o el botón iniciar sesión para volver a la pantalla de iniciar sesión, explicada en la figura anterior. Si algún dato es erróneo, se le notificará de este a través de una alerta.

Figura 9.3 Pantalla registro

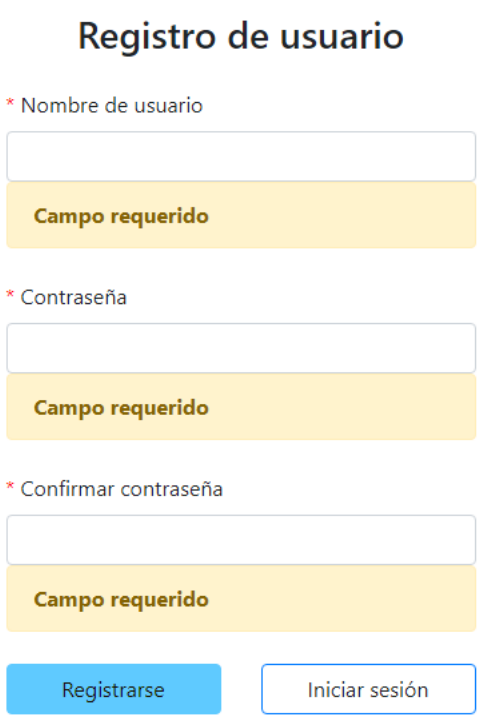

Figura 9.4 Pantalla alertas de registro

Ya sea desde el registro o inicio de sesión, si el usuario introduce datos correctos se le enviará a la pantalla de visualizar documentos.

9.3.2 Barra de navegación

La barra de navegación está presente en todas las pantallas en las que el usuario está autenticado. Muestra en la parte izquierda las pantallas que pueden accederse a través del clic en su nombre, más el icono del archivo, que redirige a la pantalla de documentos también. En la parte derecha se encuentra el perfil de usuario, como un icono de usuario, y su derecha, un texto de bienvenida personalizado para cada usuario.

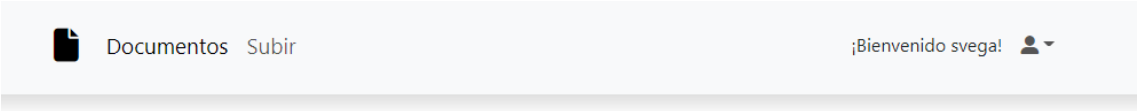

Figura 9.5 Barra de navegación

Dentro del icono del perfil de usuario, se pueden realizar dos acciones, una cerrar sesión, al cual redirigirá al usuario a la pantalla de inicio de sesión, o la acción de eliminar la cuenta, que abrirá un modal para que el usuario confirme la acción o la cancele.

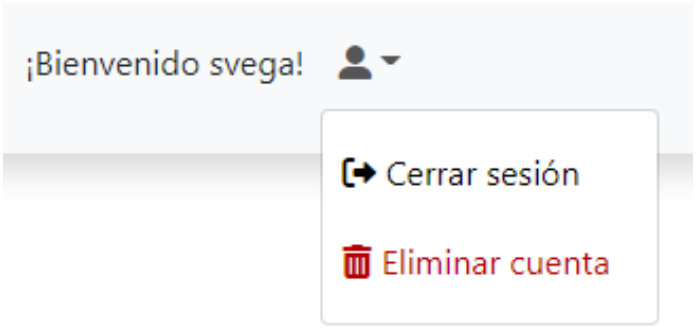

Figura 9.6 Desplegable con acciones del perfil

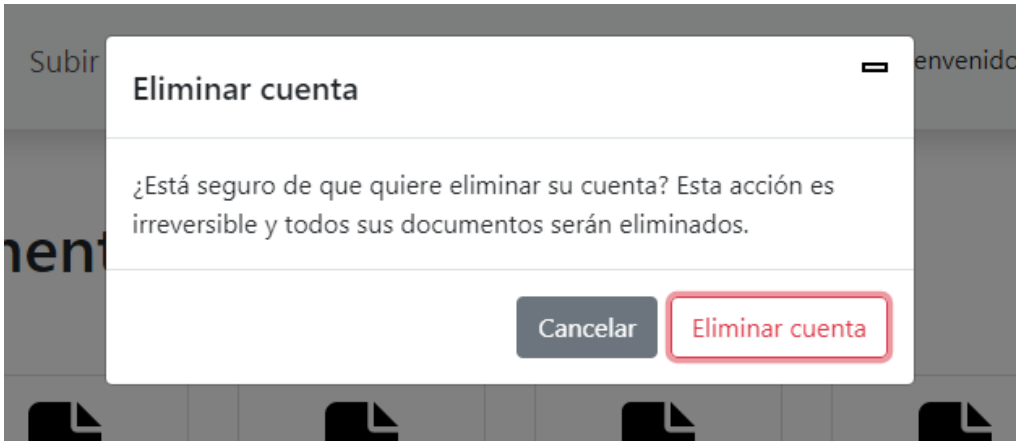

Figura 9.7 diálogo para confirmar la eliminación de la cuenta

9.3.3 Visualizar documentos

Dentro de la pantalla de documentos, se podrán visualizar los documentos, descararlos, eliminarlos y ordenarlos según tipo, nombre de manera alfabética de la A-Z y de la Z-A. Cada documento está formado por una tarjeta con dos botones, descargar y eliminar. El de descargar el documento directamente en la máquina y el de eliminar abre un diálogo modal de confirmación. Por otro lado, los botones de la derecha ordenan los documentos. El más a la derecha ordena los documentos según el tipo del archivo, el del centro ordena según el nombre del documento de la Z-A y el más a la izquierda ordena de la A-Z.

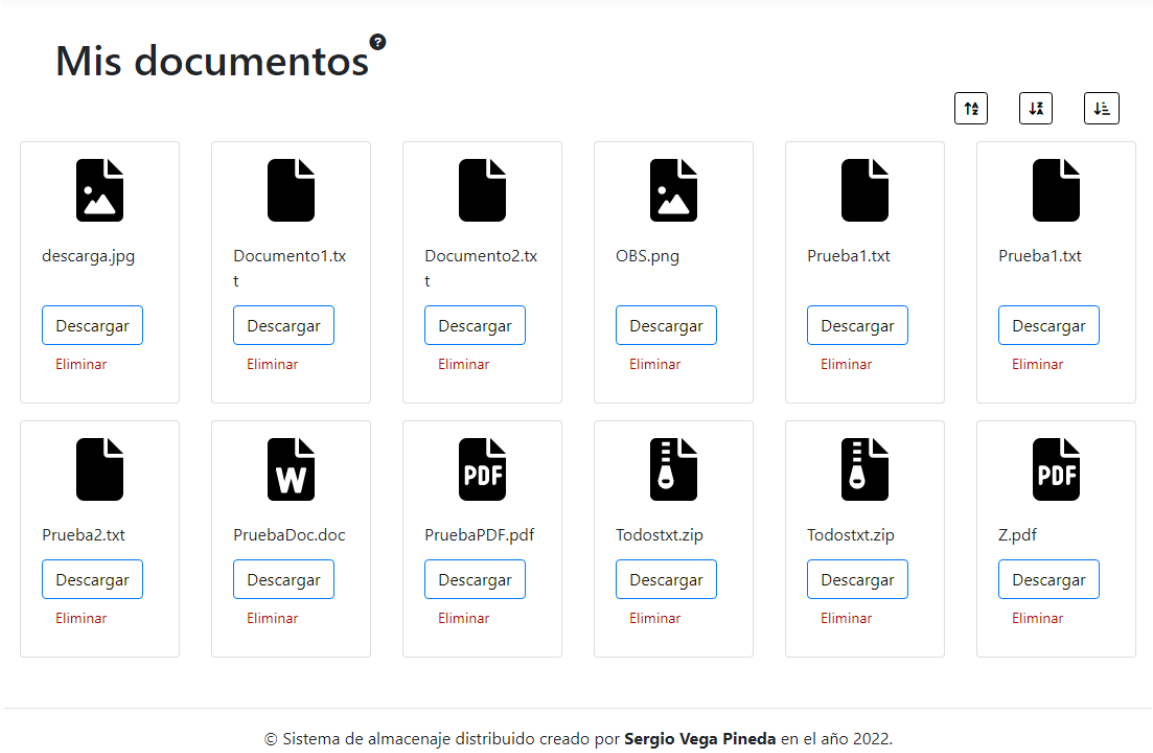

Figura 9.8 Pantalla documentos ordenada por nombre A-Z

Figura 9.9 Pantalla documentos ordenada por tipo de archivo

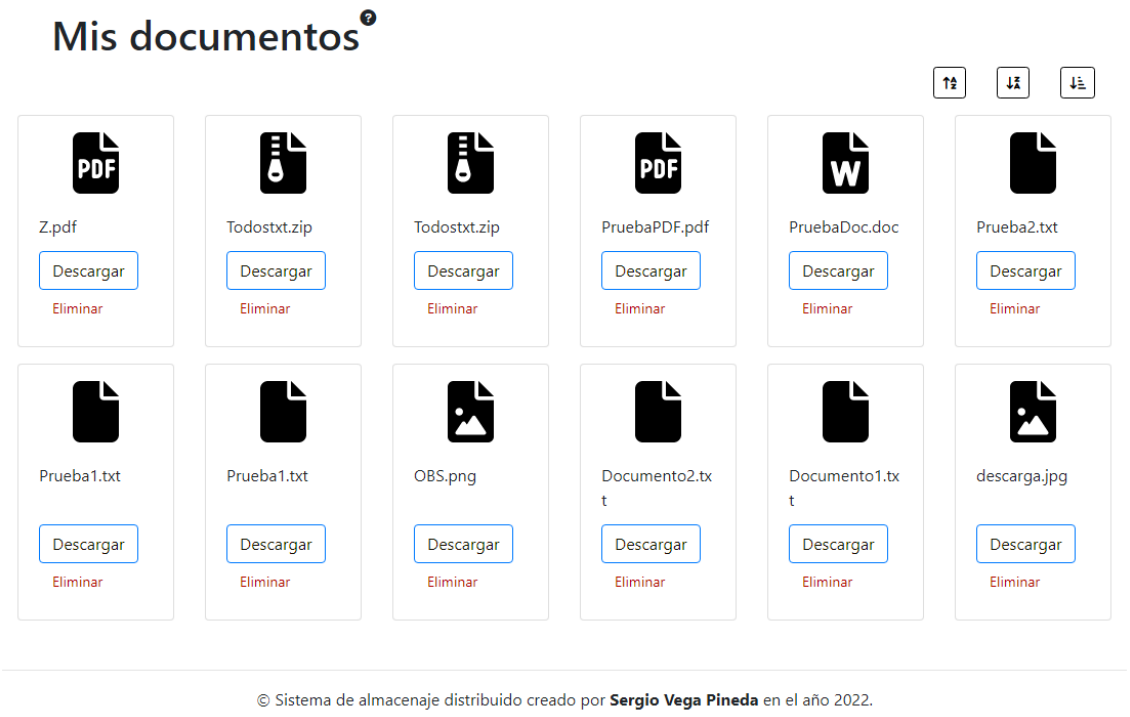

Figura 9.10 Pantalla documentos ordenada por nombre Z-A

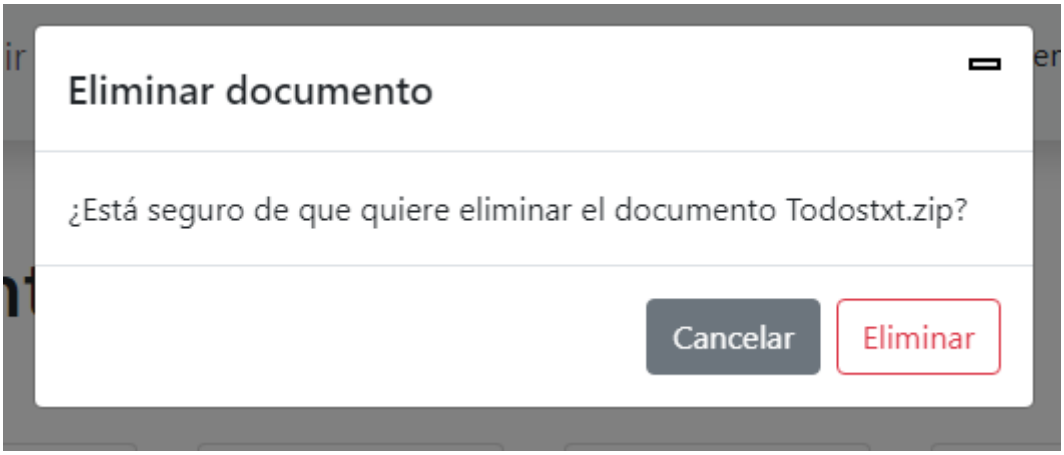

Figura 9.11 Modal de eliminación de un archivo

9.3.4 Subir documento

Esta pantalla está formada simplemente por un botón seleccionar archivo, el cual abrirá el explorador de archivos del sistema operativo para que el usuario pueda escoger un documento que subir al sistema. Por otro lado, en caso de que exista cualquier inconveniente, el usuario recibirá una alerta indicando que ha habido un error. Las extensiones de archivos soportadas por el sistema y que el usuario puede subir son la siguientes: .pdf, .doc, .zip, .rar, .txt, .png y .jpg.

Figura 9.12 Pantalla subir documento

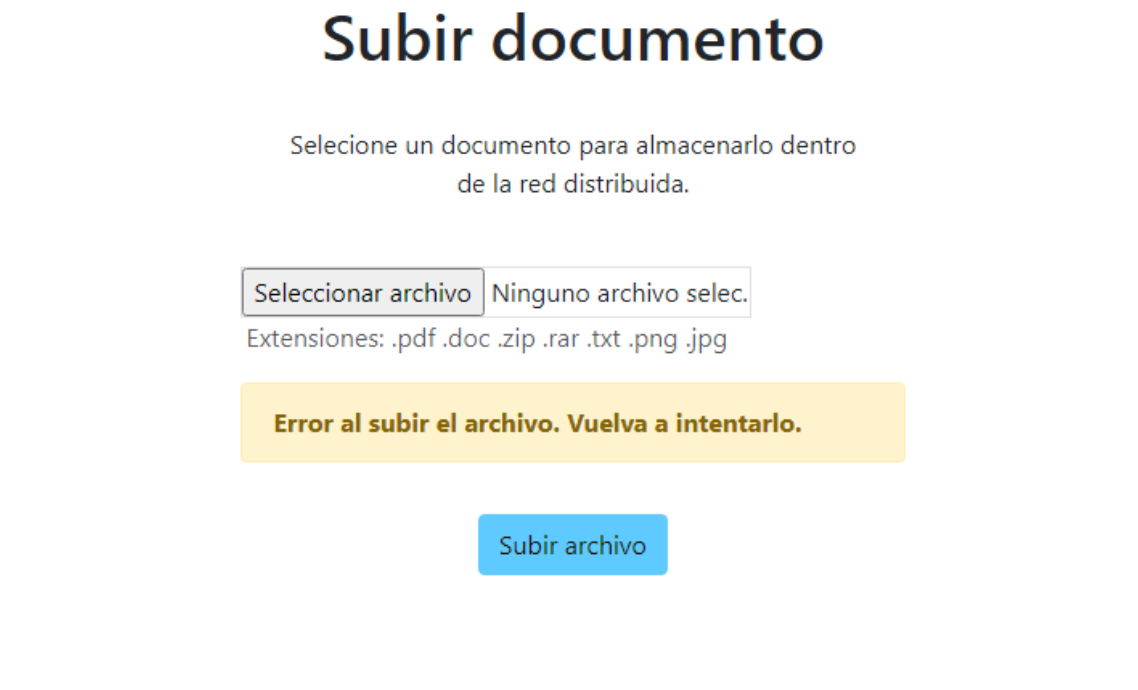

Figura 9.13 Pantalla alerta subir documento

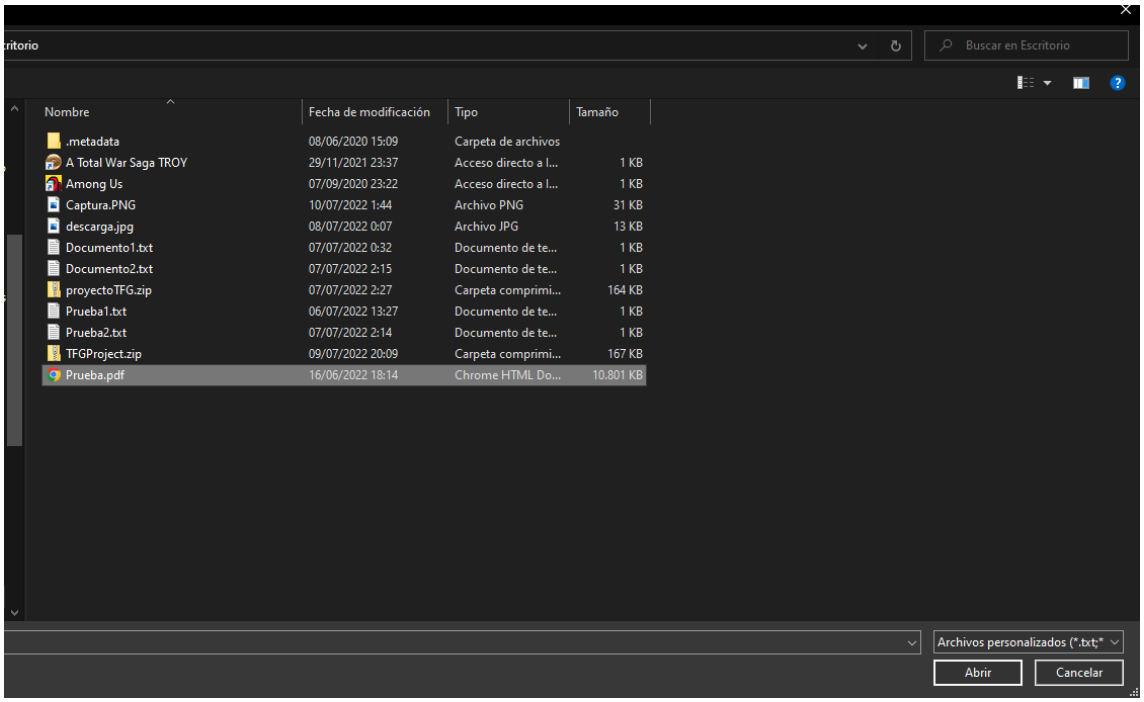

Figura 9.14 Explorador de archivos de Windows 10

9.4 Manual del Programador

En esta sección se expondrá el manual de uso de la aplicación para programadores.

9.4.1 Ampliar el esquema de la base de datos

Para introducir un nuevo modelo en la base de datos, se deberá crear en la carpeta *models* una nueva clase y crear un nuevo esquema de manera similar a los otros dos ya existentes, que sirven como ejemplo y guía para la creación de un nuevo modelo.

9.4.2 Ampliar encriptaciones

Existe una carpeta *encryption* la cual engloba cualquier sistema o método de encriptación o relacionado con ella que el sistema pueda tener.

9.4.3 Variables globales

Todas las variables globales que deban ser secretas deberán ir dentro del archivo .env.

Capítulo 10. Conclusiones y Ampliaciones

Este capítulo engloba las conclusiones sobre el presente proyecto, así como las posibles ampliaciones futuras que pueda haber.

10.1 Conclusiones

En líneas generales, se ha desarrollado una aplicación web con un propósito claro, descentralizar el almacenaje de archivos. Y tal objetivo se ha alcanzado, consiguiendo lograr un sistema de almacenamiento de archivos en la nube combinando e integrando en un mismo sistema varias tecnologías novedosas, IPFS y red Blockchain, y otras no tan novedosas como técnicas de encriptación y desarrollo de aplicaciones web. El resultado final, un gestor documental de naturaleza distribuida, segura y privada.

Aun así, el camino para llegar a una versión final usable no ha sido fácil, ya que la creación de un gestor documental desde 0, más la integración de diferentes tecnologías de manera que se sincronicen y trabajen juntas no es una tarea sencilla. De todas formas, la creación de un gestor documental no era realmente el objetivo del proyecto, sino más bien una manera visual para mostrar el sistema logrado. Por lo tanto, la simpleza presente en la interfaz y funcionalidades dentro del mismo es justificable.

Pero la fase más complicada y a la que se le ha dedicado más esfuerzo fue la del estudio de tecnologías y herramientas junto con la fase de diseño. Esto se debe principalmente a dos factores. El primero, el desconocimiento sobre estas tecnologías y escasez de documentación relacionada en comparación con otras herramientas. El segundo, la alta complejidad de idear y diseñar como diferentes sistemas, que no tienen mucho en común, interactúan para conseguir un producto final.

Por otro lado, la integración del proyecto dentro del producto Neodoc ha sido un éxito, aunque no sea con 100% de las funcionalidades desarrolladas en este proyecto. Esto se debe a que se ha simplificado el caso de uso para su integración, ya que el propio gestor documental ya presentaba una manera eficiente de almacenar los archivos y no cabía la posibilidad de cambiar el método.

Por lo tanto, la integración consistió en utilizar la tecnología Blockchain para comprobar la integridad de los archivos almacenados dentro del propio gestor documental, sin necesidad de integrar la plataforma IPFS. Aun así, se implementó de tal forma que IPFS funcionase como una especie de copia de seguridad para los archivos del propio gestor documental, en caso de una emergencia ante desastres. De esta manera, se descubrió un nuevo objetivo durante el desarrollo e implementación del proyecto que no se había identificado en las fases de estudio y análisis.

Para concluir, uno de los objetivos no escritos del autor es que los lectores de este documento sean conscientes de dónde y cómo sus archivos son almacenados en la nube y lo que ello implica.

10.2 Ampliaciones

En este apartado se mostrarán posibles ampliaciones futuras que el sistema pueda incorporar para mejorar en aspectos como diseño, usabilidad y rendimiento.

10.2.1 Buscador de documentos

Con la finalidad de mejorar la usabilidad de la visualización de los documentos, se incorporará un buscador de documentos en la que, a través de la escritura del nombre de un documento, se podrá buscar y filtrar documentos por nombre. De esta manera, se consigue agilizar la búsqueda de un documento por parte del usuario.

10.2.2 Organización de documentos por carpetas

Aunque los filtros para ordenar los documentos son bastante útiles para manejarlos, cabe destacar que no son suficientes, sobre todo en el caso en el que se concentre una gran cantidad de documentos en la cuenta de usuario. Por lo tanto, una buena ampliación para el trabajo sería la posibilidad de crear carpetas contenedoras de archivos, de tal manera que los usuarios puedan organizar los documentos como deseen.

10.2.3 Subir varios documentos simultáneamente

Actualmente, la subida de documentos puede hacerse algo tediosa debido a que solo puede realizarse de uno en uno, por lo tanto, se puede implementar una mejora de tal manera que se permita seleccionar varios archivos para subir cada uno de ellos a la plataforma.

10.2.4 Introducción de más información sobre documentos

Tras obtener el *feedback* de los usuarios que han utilizado el sistema, se concluyó que estaría bien ampliar el sistema y mostrar más información acerca de los documentos, como puede ser la fecha de subida de este o el tamaño de este. Esta mejora aportaría más información sobre los documentos del usuario.

10.2.5 Mejora de la interfaz de usuario

Por otro lado, la interfaz de usuario podría presentar mejoras en cuanto al estilo se refiera. Se podría incluir un logo, añadir *branding* y estilos corporativos para que sea una aplicación destacable e identificable de forma única al usuario.

10.2.6 Mejora de rendimiento

Actualmente, la subida de un documento, además de poder hacerse solamente de uno en uno, es bastante lenta. Esto se debe principalmente al uso de la tecnología Blockchain y a la naturaleza distribuida del sistema de almacenaje distribuido utilizado, que lo hace ser un poco más lento. Aunque no se pudiera encontrar una solución a este apartado sobre el rendimiento, sí que puede mejorarse la interfaz para ofrecer un mayor *feedback* al usuario sobre las acciones solicitadas.

Capítulo 11. Planificación del Proyecto y Presupuesto finales

En este capítulo se presentará la realidad sobre la planificación del proyecto y el presupuesto final del proyecto.

11.1 Planificación Final

En esta sección se mostrará la planificación final del proyecto. Para ello, se han realizado modificaciones sobre el diagrama de Gantt presentado en el apartado 4.1.

11.1.1 Cronograma

Finalmente, si se han cumplido con los plazos de inicio y terminación del proyecto inicialmente establecidos, pero dentro del desglose de las diferentes actividades y tareas han sufrido una serie de modificaciones, ya que su planificación no ha sido acertada.

El proyecto tiene el 1 de febrero de 2022 como fecha de comienzo y el 11 de julio de 2022 como fecha de finalización, concretamente 5 meses y 10 días. Se ha mantenido también el horario de trabajo 5 horas al día de lunes a viernes de 08:30 a 13:30, quedando un total de 115 días reales de trabajo. El día extra previamente establecido se ha utilizado en una de las tareas planificadas para cubrir algún retraso y el tiempo de revisión ha caído de 6,4 días a 6 días y será realizado por el jefe de proyecto junto al arquitecto software. En total, el proyecto suma 973,5 horas de trabajo entre todos los roles dentro del proyecto, incrementando así en 28,4 horas las horas totales respecto a la planificación inicial.

11.1.2 Roles

Respecto a los roles, se han mantenido los ya definidos en la planificación inicial: arquitecto software, desarrollador Full-Stack, diseñador software, consultor de tecnología y jefe de proyecto.

Se presentan las horas planificadas inicialmente junto con las reales según el rol:

- Arquitecto software: 232,2 horas (inicial) – 284,1 horas (real).
- Desarrollador Full-Stack: 190,1 horas (inicial) – 191,1 horas (real).
- Diseñador software: 272,3 horas (inicial) – 258,1 horas (real).
- Consultor de tecnología: 33,1 horas (inicial) – 46,1 horas (real).
- Jefe de proyecto: 217,3 horas (inicial) – 194,1 horas (real).

11.1.3 Agrupación de tareas

Para planificar el proyecto, se ha seguido la misma agrupación de tareas definida anteriormente:

- **Estudio del sistema**: en este apartado se engloban las tareas de comienzo del proyecto destinadas a investigar sobre las tecnologías, profundizar sobre la situación actual del tema escogido a tratar, así como la parte de análisis en el que se agrupa la definición de alcance y objetivos del sistema a parte de la determinación de diagramas y casos de uso.
- **Diseño del sistema**: en este apartado se realizan todos los diagramas necesarios para la elaboración del proyecto y el diseño de todas las partes del sistema.
- **Implementación**: en este apartado se implementa el sistema realmente.
- **Pruebas**: en este apartado se realizan todas las pruebas del proyecto, en concreto las de usabilidad, accesibilidad, integración y unitarias.
- **Manuales del sistema**: en este apartado se elaboran todos los manuales de uso, de instalación, de ejecución, de usuario y del programador.

11.1.4 Diagrama de Gantt

A continuación, se expone el diagrama de Gantt de la planificación final del proyecto con las tareas, su duración y los recursos asignados a cada una.

Outline Number	Task Name	Duration	Start	Finish	Pr
1	**Sistema de almacenaje distribuido**	**115 days**	**Tue 01/02/22**	**Mon 11/07/22**	
1.1	Reunión de arranque	0,22 days	Tue 01/02/22	Tue 01/02/22	
1.2	**Estudio del sistema**	**35,2 days**	**Wed 02/02/22**	**Wed 23/03/22**	**2**
1.2.1	Desarrollo de la idea para el proyecto	2 days	Wed 02/02/22	Thu 03/02/22	
1.2.2	**Realización del estudio previo**	**13 days**	**Fri 04/02/22**	**Tue 22/02/22**	**4**
1.2.2.1	Estudio de la situación actual	4 days	Fri 04/02/22	Wed 09/02/22	
1.2.2.2	Investigación de tecnologías	9 days	Thu 10/02/22	Tue 22/02/22	6
1.2.3	**Analisis del sistema**	**19,2 days**	**Wed 23/02/22**	**Tue 22/03/22**	**5**
1.2.3.1	Definición del alcance	3,2 days	Wed 23/02/22	Mon 28/02/22	
1.2.3.2	Definición de los objetivos del proyecto	5 days	Mon 28/02/22	Mon 07/03/22	9
1.2.3.3	Análisis de casos de uso	4 days	Mon 07/03/22	Fri 11/03/22	10
1.2.3.4	Análisis y determinación de diagramas	7 days	Fri 11/03/22	Tue 22/03/22	11
1.2.4	Documentación del estudio del sistema	1 day	Tue 22/03/22	Wed 23/03/22	8
1.3	**Diseño del sistema**	**33,4 days**	**Wed 23/03/22**	**Mon 09/05/22**	**3**
1.3.1	Realización de diagramas relacionados con el diseño	4 days	Wed 23/03/22	Tue 29/03/22	
1.3.2	**Diseño de Blockchain de Ethereum**	**9,6 days**	**Tue 29/03/22**	**Mon 11/04/22**	**15**
1.3.2.1	Diseño de la interacción con la red Ethereum	4,8 days	Tue 29/03/22	Mon 04/04/22	
1.3.2.2	Diseño del contrato inteligente	4,8 days	Tue 05/04/22	Mon 11/04/22	17
1.3.3	Diseño de la interacción con IPFS	6,4 days	Mon 11/04/22	Wed 20/04/22	16
1.3.4	Diseño de la aplicación web	7 days	Wed 20/04/22	Fri 29/04/22	19
1.3.5	Diseño de la interfaz de usuario	5,4 days	Fri 29/04/22	Fri 06/05/22	20
1.3.6	Documentación diseño del sistema	1 day	Fri 06/05/22	Mon 09/05/22	21
1.4	**Implementación**	**23 days**	**Mon 09/05/22**	**Thu 09/06/22**	**21**
1.4.1	**Implementación del lado servidor (backend)**	**14,2 days**	**Mon 09/05/22**	**Fri 27/05/22**	**14**
1.4.1.1	Integración de la API de Web3Storage	3,2 days	Mon 09/05/22	Thu 12/05/22	
1.4.1.2	Implementación de la API	3,2 days	Thu 12/05/22	Tue 17/05/22	25
1.4.1.3	Implementación del contrato inteligente	3,2 days	Wed 18/05/22	Mon 23/05/22	26
1.4.1.4	Implementación de la encriptación de los archivos	3 days	Mon 23/05/22	Thu 26/05/22	27
1.4.1.5	**Implementación del registro de usuario**	**1,6 days**	**Thu 26/05/22**	**Fri 27/05/22**	**28**
1.4.1.5.1	Creación e integración de la base de datos	1,6 days	Thu 26/05/22	Fri 27/05/22	
1.4.2	**Implementación del lado cliente (frontend)**	**7,8 days**	**Fri 27/05/22**	**Wed 08/06/22**	**24**
1.4.2.1	**Implementación de la interfaz de usuario**	**7,8 days**	**Fri 27/05/22**	**Wed 08/06/22**	
1.4.2.1.1	Interfaz de registro de usuario	4 days	Fri 27/05/22	Thu 02/06/22	
1.4.2.1.2	Interfaz de manejo de archivos	3,8 days	Thu 02/06/22	Wed 08/06/22	33
1.4.3	Documentación de la implementación	1 day	Wed 08/06/22	Thu 09/06/22	31
1.5	**Pruebas**	**10 days**	**Thu 09/06/22**	**Thu 23/06/22**	**23**
1.5.1	**Pruebas unitarias**	**4,8 days**	**Thu 09/06/22**	**Thu 16/06/22**	
1.5.1.1	Desarrollo de pruebas para la API del sistema	2,4 days	Thu 09/06/22	Mon 13/06/22	
1.5.1.2	Desarrollo de pruebas para el registro de usuarios	2,4 days	Tue 14/06/22	Thu 16/06/22	38
1.5.2	Pruebas de integración	2 days	Thu 16/06/22	Mon 20/06/22	37
1.5.3	Pruebas de usabilidad	1 day	Mon 20/06/22	Tue 21/06/22	40
1.5.4	Pruebas de accesibilidad	1,2 days	Tue 21/06/22	Wed 22/06/22	41
1.5.5	Documentación de la pruebas	1 day	Wed 22/06/22	Thu 23/06/22	42
1.6	**Manuales del sistema**	**6,4 days**	**Thu 23/06/22**	**Fri 01/07/22**	**36**
1.6.1	Manual de intalación	1,6 days	Thu 23/06/22	Mon 27/06/22	
1.6.2	Manual de ejecución	1,6 days	Mon 27/06/22	Tue 28/06/22	45
1.6.3	Manual de usuario	1,6 days	Tue 28/06/22	Thu 30/06/22	46
1.6.4	Manual del programador	1,6 days	Thu 30/06/22	Fri 01/07/22	47
1.7	Revisión final del proyecto	6 days	Mon 04/07/22	Mon 11/07/22	44
1.8	Entrega del proyecto	0 days	Mon 11/07/22	Mon 11/07/22	

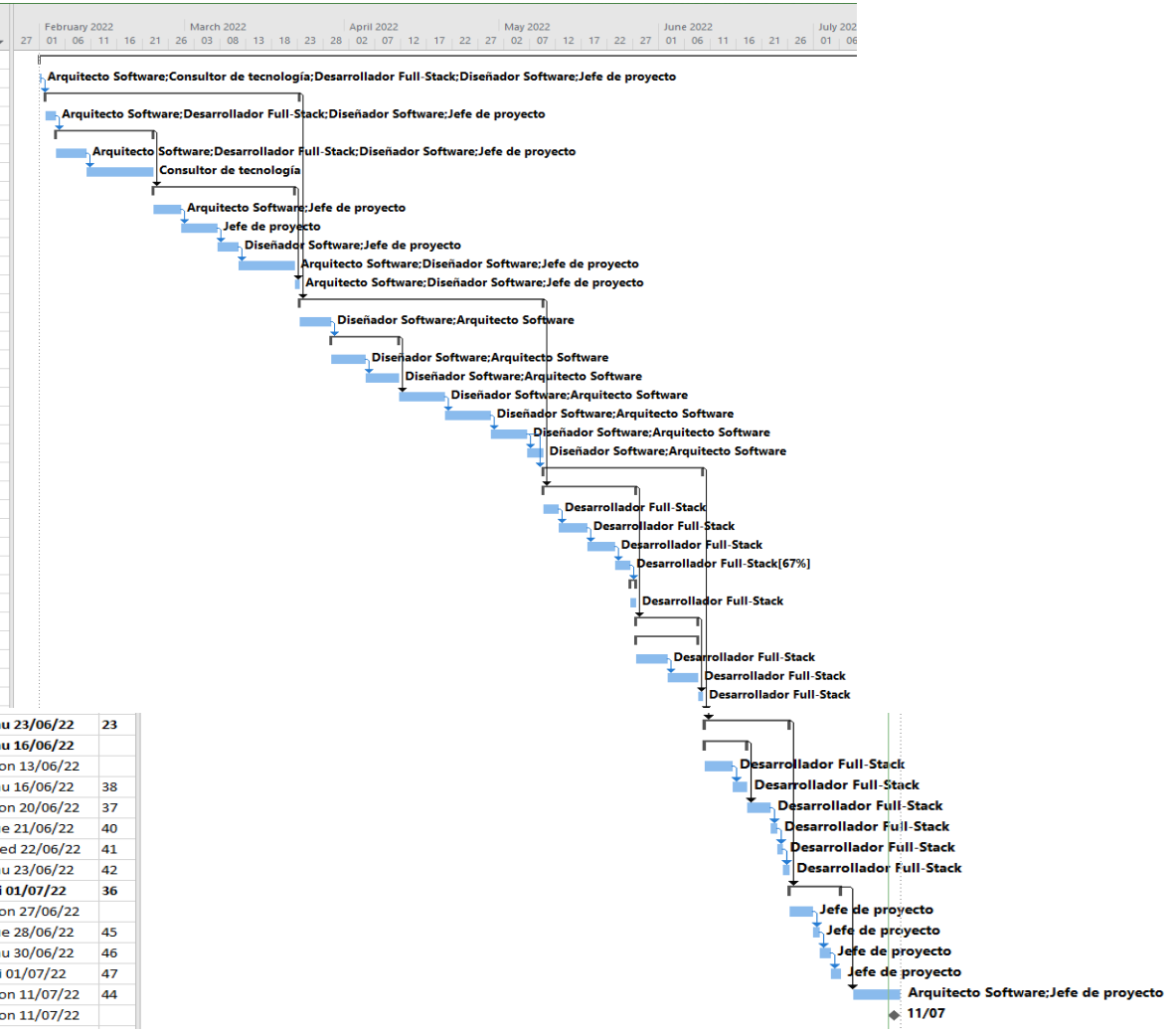

Figura 11.1 Diagrama de Gantt Planificación final

11.1.5 Resultados

Con respecto a la planificación a la planificación inicial, se han obtenido los siguientes resultados en días de trabajo:

Agrupación de tareas	Días iniciales	Días finales
Estudio del sistema	36,84 días	35,2 días
- Estudio previo	9,4 días	13 días
- Análisis	24,44 días	19,2 días
Diseño del sistema	30,2 días	33,4 días
Implementación	21,6 días	23 días
- Lado cliente	7,8 días	7,8 días
- Lado servidor	12,8 días	14,2 días
Pruebas	11,2 días	10 días
Manuales	6,4 días	6,4 días
Revisión del proyecto	6,4 días	6 días

Tabla 11.1 Tabla comparación planificaciones

Como puede observarse en la tabla anterior, se producen una serie de cambios ligeros en la planificación, sobre todo en el reparto de días del estudio del sistema. Aun así, se puede concretar que se ha realizado una buena planificación ya que las fechas estimadas en un primer momento se cumplen y la desviación no es elevada.

11.2 Presupuesto Final

En esta sección se muestran los presupuestos finales aplicables a la planificación final realizada previamente. Cabe destacar que los sueldos serán los mismos a los definidos en el apartado 4.2.

11.2.1 Desarrollo de Presupuesto Detallado (Empresa)

A continuación, se muestra la tabla con el presupuesto de la empresa:

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	*Implementación*				
1.1	Estudio del sistema	1	100%	13.275,00 €	13.275,00 €
1.2	Diseño del sistema	1	100%	9.185,00 €	9.185,00 €
1.3	Implementación	1	100%	2.750,00 €	2.750,00 €
1.4	Pruebas	1	100%	1.250,00 €	1.250,00 €

1.5	Manuales del sistema	1	100%	1.440,00 €	1.440,00 €
1.6	Revisión del proyecto	1	100%	2.250,00 €	2.250,00 €
2	*Recursos Software*				
2.1	Microsoft Windows 10	5	4,5%	145,00 €	32,63 €
2.2	Microsoft Project	1	45%	100,80 €	45,36 €
2.3	Microsoft Office 365	5	45%	61,20 €	137,70 €
2.4	Visual Studio Code	5	100%	0,00 €	0,00 €
2.5	Nodejs	1	100%	0,00 €	0,00 €
2.6	Expressjs	1	100%	0,00 €	0,00 €
2.7	Trufflejs	1	100%	0,00 €	0,00 €
2.8	IPFS	1	100%	0,00 €	0,00 €
2.9	MongoDB	1	100%	0,00 €	0,00 €
2.10	Ethereum Rinkeby	1	100%	0,00 €	0,00 €
3	*Recursos Hardware*				
3.1	Cable	30	4,5%	3,50 €	4,73 €
3.2	Periféricos	10	9%	25,00 €	22,50 €
3.3	Router	1	6,4%	40,00 €	2,56 €
3.4	Portátil	5	9%	600,00 €	270,00 €
4	*Gastos indirectos*				
4.1	Papel	50	35%	0,06 €	1,05 €
4.2	Bolígrafo	8	20%	0,60 €	0,96 €
4.3	Lápiz	13	40%	0,40 €	2,08 €
4.4	Electricidad	1	100%	2.504,00 €	2.504,00 €
4.5	Internet por Wifi	1	100%	360,00 €	360,00 €
4.6	Instalaciones	1	100%	2.230,00 €	2.230,00 €
Subtotal					35.763,56 €
Beneficio (25%)					8.940,89 €
IVA (21%)					7.510,35 €
TOTAL					**52.214,80 €**

Tabla 11.2 Presupuesto empresa final

11.2.2 Desarrollo de Presupuesto Simplificado (Cliente)

A continuación, se muestra la tabla con el presupuesto para el cliente de una manera simplificada y clara:

Concepto	Cantidad	Precio unitario (€)	Coste total sin IVA (€)
Estudio del sistema	1	20.985,69 €	20.985,69 €
Diseño del sistema	1	12.035,50 €	12.035,50 €
Implementación	1	3.762,81 €	3.762,81 €
Pruebas	1	1.652,95 €	1.652,95 €
Manuales del sistema	1	2.112,84 €	2.112,84 €
Revisión del proyecto	1	2.602,94 €	2.602,94 €
		Subtotal	43.152,72 €
		IVA (21%)	9.062,07 €
		TOTAL	**52.214,80 €**

Tabla 11.3 Presupuesto cliente final

11.2.3 Resultados

En cuanto al presupuesto final e inicial, existe una variación de 1.748,48€. Esto se debe al uso del día extra que se había reservado para imprevistos además del aumento en días de tareas donde se encontraban más roles. Además, en la revisión del proyecto se introdujo otro rol, el arquitecto software, lo cual produjo un aumento en sus horas y, por ende, un aumento en el presupuesto final. También, hubo algún cambio pequeño con respecto al material usado, como bolígrafos, lápices… Aun así, la variación es del 3,36%, lo que significa que se ha hecho una buena planificación respecto al presupuesto.

Capítulo 12. Referencias Bibliográficas

A continuación, se enumeran las referencias consultadas para realizar la documentación de este trabajo.

[1] "Neosystems Gijón." https://neosystems.es/ (accessed Feb. 08, 2022).

[2] "Neodoc - Gestor de contenido empresarial." https://neosystems.es/neodoc/ (accessed Feb. 08, 2022).

[3] "IPFS." https://ipfs.io/#how (accessed Feb. 05, 2022).

[4] "Bitcoin - Wikipedia", Accessed: Feb. 06, 2022. [Online]. Available: https://es.wikipedia.org/wiki/Bitcoin

[5] "Historia del precio del Bitcoin," *CoinMarketCap*. https://coinmarketcap.com/es/currencies/bitcoin/ (accessed Jun. 26, 2022).

[6] "El Salvador hace historia al convertirse en el primer país en adoptar el bitcoin como moneda legal," *elEconomista*, Sep. 2021, Accessed: Feb. 06, 2022. [Online]. Available: https://www.eleconomista.es/divisas/noticias/11382210/09/21/El-Salvador-hace-historia-al-convertirse-en-el-primer-pais-en-adoptar-el-bitcoin-como-moneda-legal.html

[7] Enrique Pérez, "No lo llames Facebook, llámalo Meta: la empresa cambia de nombre y lo apuesta todo al metaverso," *Xataka*, Oct. 2021, Accessed: Feb. 06, 2022. [Online]. Available: https://www.xataka.com/empresas-y-economia/no-llames-facebook-llamalo-meta-empresa-cambia-nombre-apuesta-todo-al-metaverso

[8] "Otro país adoptó al Bitcoin como moneda de curso legal," *InfoTechnology*, Apr. 2022, Accessed: Jun. 22, 2022. [Online]. Available: https://www.cronista.com/infotechnology/criptomonedas/otro-pais-adopto-al-bitcoin-como-moneda-de-curso-legal/

[9] "'No valen nada': las tres razones por las que Warren Buffett nunca invirtió en criptomonedas," *InfoTechnology*, Jun. 2022, Accessed: Jun. 21, 2022. [Online]. Available: https://www.cronista.com/infotechnology/criptomonedas/no-valen-nada-las-tres-razones-por-las-que-warren-buffett-nunca-invirtio-en-criptomonedas/

[10] "Ethereum - Wikipedia." https://es.wikipedia.org/wiki/Ethereum (accessed Feb. 05, 2022).

[11] "Historia del precio del Ethereum," *CoinMarketCap*, Accessed: Jun. 24, 2022. [Online]. Available: https://coinmarketcap.com/es/currencies/ethereum/

[12] "Los famosos están adoptando los NFT a lo grande," *Cointelegraph*, Oct. 2021, Accessed: Jun. 21, 2022. [Online]. Available: https://es.cointelegraph.com/news/celebrities-are-embracing-nfts-in-a-big-way

[13] Albert Sanchís, "El 80% de los NFT son un fraude. No lo decimos nosotros, sino la principal plataforma de comercio de NFTs," Feb. 02, 2022. Accessed: Feb. 06, 2022. [Online]. Available: https://magnet.xataka.com/en-diez-minutos/80-nft-fraude-no-decimos-nosotros-sino-principal-plataforma-comercio-nfts

[14] "Dropbox - Wikipedia." https://es.wikipedia.org/wiki/Dropbox (accessed Feb. 04, 2022).

[15] "Dropbox." https://www.dropbox.com/ (accessed Feb. 04, 2022).

[16] Miguel de Icaza, "Dropbox Lack of Security," *Miguel de Icaza`s Blog*, Accessed: Feb. 05, 2022. [Online]. Available: https://tirania.org/blog/archive/2011/Apr-19.html

[17] Samuel Julia Cristobal, "Adiós a tu privacidad: Dropbox comparte tus datos con Google y Amazon," *Blog de Dataprius*, Sep. 2019, Accessed: Feb. 05, 2022. [Online]. Available: https://blog.dataprius.com/index.php/2019/09/13/adios-a-tu-privacidad-dropbox-comparte-tus-datos-con-google-y-amazon/#:~:text=Cuando%20tu%20empresa%20sube%20sus,para%20%E2%80%9Cmejorar%20el%20servicio%E2%80%9D

[18] Marcos Sierra, "Dropbox comparte datos de sus usuarios con Amazon y Google," *Voz Populi*, Aug. 2019, Accessed: Feb. 05, 2022. [Online]. Available: https://www.vozpopuli.com/economia_y_finanzas/dropbox-comparte-usuarios-amazon-google_0_1272773238.html

[19] "Google Drive-Wikipedia." https://es.wikipedia.org/wiki/Google_Drive (accessed Feb. 04, 2022).

[20] "Google Drive." https://www.google.com/intl/es_es/drive/ (accessed Feb. 04, 2022).

[21] Marcos Merino, "Google va a borrar tus archivos de Drive (o incluso banearte) si escanea su contenido y lo encuentra 'inapropiado' (que no ilegal)," *GENBETA*, Jan. 2022, Accessed: Feb. 06, 2022. [Online]. Available: https://www.genbeta.com/actualidad/google-va-a-borrar-tus-archivos-drive-incluso-banearte-escanea-su-contenido-encuentra-inapropiado-que-no-ilegal

[22] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Oct. 31, 2008. Accessed: Feb. 05, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[23] Daniel Jimenez, "¿Cuántos algoritmos de consenso existen para las Blockchain?," *Cointelegraph*, Sep. 2019, Accessed: Feb. 06, 2022. [Online]. Available: https://es.cointelegraph.com/news/cuantos-algoritmos-de-consenso-existen-para-las-blockchain

[24] "Solidity - Wikipedia." https://es.wikipedia.org/wiki/Solidity (accessed Feb. 05, 2022).

[25] ILIMIT, "Ventajas del almacenamiento distribuido," Nov. 24, 2020. Accessed: Feb. 05, 2022. [Online]. Available: https://www.ilimit.com/blog/ventajas-del-almacenamiento-distribuido/

[26] "Peer-to-Peer (P2P) - Wikipedia", Accessed: Feb. 05, 2022. [Online]. Available: https://es.wikipedia.org/wiki/Peer-to-peer

[27] "IPFS - Docs." https://docs.ipfs.io/concepts/what-is-ipfs/#decentralization (accessed Feb. 05, 2022).

[28] "Web3Stroage." https://web3.storage/ (accessed Feb. 05, 2022).

[29] "¿Qué es el *hash* de un archivo y para que nos vale?" https://peritosinformaticos.es/que-es-el-hash-de-un-archivo-y-para-que-nos-vale/ (accessed Feb. 05, 2022).

[30] Javier López, "Así funciona el sistema de cifrado AES-256 bits, ¿es realmente seguro?," *HardZone*, May 2022, Accessed: Feb. 05, 2022. [Online]. Available: https://hardzone.es/tutoriales/rendimiento/cifrado-aes-256-bits-como-funciona/

[31] "Generalidades del protocolo HTTP," *Developer Mozilla*, Accessed: Feb. 06, 2022. [Online]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[32] "Interfaz de programación de aplicaciones (API) - Wikipedia." https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones (accessed Feb. 05, 2022).

[33] "Aplicación Web - Wikipedia", Accessed: Feb. 05, 2022. [Online]. Available: https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

[34] "Modelo Cliente-Servidor - Wikipedia." https://es.wikipedia.org/wiki/Cliente-servidor (accessed Feb. 05, 2022).

[35] "¿Qué significa lado del cliente y lado del servidor? | Lado del cliente vs. Lado del servidor," *Cloudflare*. https://www.cloudflare.com/es-es/learning/serverless/glossary/client-side-vs-server-side/ (accessed Feb. 05, 2022).

[36] "HTML: Lenguaje de etiquetas de hipertexto," *Developer Mozilla*. https://developer.mozilla.org/es/docs/Web/HTML (accessed Feb. 06, 2022).

[37] "CSS," *Developer Mozilla*. https://developer.mozilla.org/es/docs/Web/CSS (accessed Feb. 05, 2022).

[38] "Introducción al lado servidor," *Developer Mozilla*. https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction (accessed Feb. 05, 2022).

[39] "JavaScript - Wikipedia." https://es.wikipedia.org/wiki/JavaScript (accessed Feb. 04, 2022).

[40] "Nodejs - Wikipedia." https://es.wikipedia.org/wiki/Node.js (accessed Feb. 05, 2022).

[41] "Expressjs - Wikipedia", Accessed: Feb. 05, 2022. [Online]. Available: https://expressjs.com/es/

[42] "Truffle." https://trufflesuite.com/ (accessed Feb. 05, 2022).

[43] "Infura." https://infura.io/ (accessed Feb. 04, 2022).

[44] "MongoDB." https://www.mongodb.com/ (accessed Feb. 05, 2022).

[45] "Ruby - Wikipedia." https://es.wikipedia.org/wiki/Ruby (accessed Feb. 04, 2022).

[46] "Ruby on Rails." https://es.wikipedia.org/wiki/Ruby_on_Rails (accessed Feb. 05, 2022).

[47] "PostgreSQL - Wikipedia." https://es.wikipedia.org/wiki/PostgreSQL#:~:text=de%20datos%20Ingres.-,Historia,de%20base%20de%20datos%20relacional. (accessed Feb. 06, 2022).

[48] "Git." https://git-scm.com/ (accessed Feb. 05, 2022).

[49] "Git - Wikipedia", Accessed: Feb. 05, 2022. [Online]. Available: https://es.wikipedia.org/wiki/Git

[50] "GitHub." https://es.wikipedia.org/wiki/GitHub (accessed Feb. 05, 2022).

[51] "cryptojs," *npm*. https://www.npmjs.com/package/crypto-js (Access Ed Feb. 05, 2022).

[52] "JQuery." https://jquery.com/ (accessed Feb. 05, 2022).

[53] "dotenv," *npm*, Accessed: Feb. 06, 2022. [Online]. Available: https://www.npmjs.com/package/dotenv

[54] "PostMan." https://www.postman.com/ (accessed Feb. 05, 2022).

[55] "Bcrypt - Wikipedia", Accessed: Feb. 06, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Bcrypt

[56] "Bootstrap - Wikipedia." https://es.wikipedia.org/wiki/Bootstrap_(framework) (accessed Feb. 08, 2022).

[57] "EJS." EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. (accessed Feb. 06, 2022).

[58] "JWT." https://jwt.io/ (accessed Feb. 05, 2022).

[59] "Express-session." https://www.npmjs.com/package/express-session (accessed Feb. 06, 2022).

[60] "Express-fileUpload." https://www.npmjs.com/package/express-fileupload (accessed Feb. 05, 2022).

[61] “Mocha.” https://mochajs.org/ (accessed Feb. 04, 2022).

[62] “Supertest.” https://www.npmjs.com/package/supertest (accessed Feb. 05, 2022).

[63] “Google Lighthouse”, Accessed: Mar. 09, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Google_Lighthouse

[64] “Oracle Color”, Accessed: Mar. 10, 2022. [Online]. Available: https://colororacle.org/

Capítulo 13. Apéndices

En este capítulo se comentará el glosaría y diccionario de datos, se definirá el contenido entregado en el archivo adjunto y el índice alfabético.

13.1 Glosario y Diccionario de Datos

Descripción breve de algunos de los términos utilizados en el proyecto.

- **API**: se trata de una interfaz de programación de aplicaciones y su objetivo es el intercambio de datos entre programas software.
- ***Backend***: término en inglés que refiere a toda la lógica de una aplicación web. Concretamente, se refiere a toda la arquitectura interna para que la aplicación lleve a cabo todas las funcionalidades necesarias.
- **Blockchain**: término en inglés para referirse a una estructura de datos en la que la información es almacenada dentro de bloques contiguos con metainformación en cada uno de ellos.
- **Framework**: término en inglés que define un esquema o marco de trabajo que engloba una estructura base para elaborar un proyecto con objetivos específicos.
- ***Frontend***: término en inglés que refiere a la parte que el usuario visualiza de la aplicación y con la que interactúa.
- ***Hash***: término en inglés que se trata del resultado de una función que recibe una entrada y aplica una serie de algoritmos que la transforman en una salida distinta.
- **HTTP**: se trata de un protocolo de transferencia de hipertexto estandarizado cuyo objetivo es el de trasmitir datos.

13.2 Contenido Entregado en el Archivo adjunto

A continuación, se resumirán los directorios que se estarán dentro del fichero adjunto con el proyecto elaborado. De esta manera se facilitará la comprensión y la búsqueda de información dentro de este.

13.2.1 Contenidos

El archivo adjunto tiene como nombre "UO271280.zip". El contenido del proyecto estará dentro del directorio raíz "proyectoTFGvFinal". Dentro de este, se encontrarán todos los directorios y clases que conforman el proyecto y permiten su ejecución.

13.2.1.1 Estructura general directorios del Archivo adjunto

Se muestra a continuación una tabla con la organización de la carpeta del archivo adjunto.

Directorio	Contenido
./ Directorio raíz del Archivo adjunto	Contiene un fichero leeme.txt explicando toda esta estructura.
./proyectoTFGvFinal	Contiene toda la estructura de directorios del proyecto para desarrollo.
./MongoDB	Contiene dos ficheros en formato JSON exportados de la base de datos más un fichero .txt explicando como importar los datos en una nueva base de datos.

Tabla 13.1 Estructura general del archivo adjunto

13.2.1.2 Estructura de Directorios de "proyectoTFGvFinal"

Se muestra a continuación una tabla con la organización de carpetas dentro del proyecto desarrollado.

Directorio	Contenido
./ Directorio raíz de "proyectoTFGvFinal"	Contiene los directorios del proyecto y los ficheros .env, *app.js*, *dbConnection.js*, *package.json* y *package-lock.json*.
./controllers	Este directorio engloba todos los controladores de la aplicación y solo ese tipo de clases. Se trata de archivos JavaScript.
./models	Este directorio engloba todos los modelos de la aplicación y solo ese tipo de clases. Se trata de archivos JavaScript.
./views	Este directorio engloba todas las vistas de la aplicación y solo ese tipo de clases. Se trata de archivos EJS.
./tests	Este directorio presenta el fichero con las

	pruebas de las rutas de la aplicación.
./smartConrtact	Este directorio engloba la configuración de Truffle. Además, presente el código fuente relacionado con el contrato inteligente.
./utils	Este directorio presenta los ficheros de código fuente con la lógica de encriptación y guardado en IPFS. Se trata de archivos JavaScript.
./validations	Este directorio engloba todas las validaciones de formularios que existan dentro de la aplicación. Se trata de archivos JavaScript.
./middlewares	Este directorio engloba todos los ficheros de *middlewares* de la aplicación.
./routes	Este directorio incluye un fichero con todas las rutas de la aplicación.
./node_modules	Esta carpeta se genera después de realizar el comando "npm i" en la terminal. Instala todos los módulos necesarios para ejecutar la aplicación y que esta funcione correctamente.
./public	Este directorio contiene el código fuente de la parte del cliente. Contine la hoja de estilos, almacena las fotografías y funciones JavaScript.

Tabla 13.2 Estructura directorios de "*proyectoTFGvFinal*"

13.2.2 Código Ejecutable e Instalación

Previamente, es necesario instalar la base de datos MongoDB, versión v5.0.9 (*stable*), el manejador de paquetes Npm, versión v6.14.15 y Nodejs, versión v14.18.0.

Para poner en marchar la aplicación, basta con abrir el proyecto con un editor de texto, como puede ser Visual Studio Code. Dentro de este, abrir una nueva terminal y ejecutar el comando "npm i", el cual instalará todas las herramientas necesarias. Por otro lado, para realizar la ejecución del proyecto, se utilizará el comando "node app.js" y desde un navegador como Google Chrome, se podrá visitar la aplicación a través del Url "http://localhost:8000/". En caso de que se quiera ejecutar las pruebas, bastaría con "npm test". Todo lo anterior desde el directorio raíz. En caso de que se quiera ejecutar las pruebas del *SmartContract*, será necesario moverse hasta el directorio "smartContract", y desde él, ejecutar el comando desde la terminal "truffle test".

13.2.3 Ficheros de Configuración

El fichero de configuración de la base de datos está dentro del directorio MongoDB en el que se explica la manera de importar contenido a la base de datos. Luego, existe un fichero .env con variables globales con valores por defecto que deberían servir para la ejecución del proyecto.

13.3 Índice Alfabético

13.4 Código Fuente

El código fuente se puede encontrar en el directorio “UO271280/proyectoTFGvFinal”. En el apartado 13.2.1.2, se explica el contenido de la carpeta.