Aplicación descentralizada para recompensar la revisión por pares

Decentralized application to reward peer review

Trabajo de Fin de Grado

Curso 2020-2021

Autores

Alejandro Alarcón Ayllo

Carlos Rodríguez Hernández

Fernando Ruiz García

Directores

Samer Hassan Collado

Ámbar Tenorio-Fornés

Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el software desarrollado.

Alejandro Alarcón Ayllo
Carlos Rodríguez Hernández
Fernando Ruiz García

Agradecimientos

Queremos agradecer, en primer lugar, a nuestros tutores, Samer Hassan Collado y Ámbar Tenorio-Fornés el habernos dado esta oportunidad y haber confiado en nosotros.

Asimismo, nos gustaría agradecer a todos los revisores que nos han dedicado tiempo y esfuerzo para ayudarnos con el desarrollo del proyecto, participando en la fase de experimentación.

También, agradecer a nuestras familias, que nos han apoyado en todo momento y nos han animado hasta al final.

Índice General

Índice de Figuras

Resumen

En la actualidad, la revisión por pares se basa en un sistema altruista y desinteresado en el que el esfuerzo y tiempo dedicado por los revisores no recibe el suficiente reconocimiento.

RevLounge es una plataforma que proporciona un sistema de recompensas a las revisiones de artículos académicos por pares, permitiendo destacar a aquellos revisores que han recibido un mayor número de recompensas y posibilitando la importación de sus revisiones ya realizadas a la misma.

Este proyecto ofrece la posibilidad de corregir la actual falta de incentivos mediante la adición de recompensas del ecosistema Ethereum que incluyen tokens no fungibles, tokens de reputación y donaciones. El objetivo final del proyecto, por tanto, es el de mejorar la calidad de las revisiones y valorar de forma tangible el reconocimiento de estas.

Con este propósito, se ha creado una plataforma en ReactJS, tecnología puntera en el desarrollo de interfaces de usuario, y se ha conectado con un sistema descentralizado en blockchain para la gestión de recompensas. Para ello se ha hecho uso de tecnologías como Web3.js o Semantic UI, haciendo de RevLounge una aplicación descentralizada o DApp.

Se incluye en el proyecto una recopilación de experiencias reales de usuarios que han probado la aplicación final, en la que se valora positivamente el resultado.

Palabras Clave

Blockchain, DApp, Ethereum, NFT, ReactJS, Recompensa, Revisión por Pares, Token, Token No Fungible

Abstract

Currently, peer review is based on an altruistic and disinterested system in which the effort and time spent by reviewers is not sufficiently recognised.

RevLounge is a platform that provides a system of rewards for peer reviews of academic articles, highlighting those reviewers who have received the highest number of rewards and making it possible to import their reviews into the platform.

This project offers the possibility of correcting the current lack of incentives by adding Ethereum ecosystem rewards including non-fungible tokens, reputation tokens and donations. The ultimate goal of the project, therefore, is to improve the quality of reviews and to tangibly value the recognition of reviews.

To this end, a platform has been created in ReactJS, a cutting-edge technology in the development of user interfaces, and connected to a decentralized blockchain system for the management of rewards. To do this, technologies such as Web3.js or Semantic UI have been used, making RevLounge a decentralized application or DApp.

The project includes a compilation of real experiences of users who have tested the final application, in which the result is positively evaluated.

Keywords

Blockchain, DApp, Ethereum, NFT, Non Fungible Token, Peer Review, ReactJS, Reward, Token

Capítulo 1. Introducción

1.1 Antecedentes

Las investigaciones científicas han sentado las bases para que podamos avanzar como sociedad, desde el ámbito tecnológico hasta el ámbito sanitario, pasando por todas las ramas que conforman la ciencia. Para que esto sea posible, todas las investigaciones se rigen por el método científico[1]. Éste se basa en la premisa de la observación de un fenómeno, para poder proponer una hipótesis y comprobarla a través de la experimentación.

Cuando un investigador realiza una aportación en forma de artículo a la comunidad científica, esta debe ser revisada para comprobar su veracidad y calidad. Un estudio debe ser replicable por cualquier sujeto en cualquier lugar, de lo contrario no podríamos tener la certeza de la veracidad del experimento.

El método científico comienza a ser usado por los trabajos del brillante investigador Galileo Galilei provenientes del siglo XVI[2] y, en la actualidad, sigue siendo el método utilizado por todos los investigadores alrededor del mundo. Pero no basta con investigar y redactar un artículo para su posterior publicación en una revista, este debe pasar por un proceso llamado “Revisión por pares” que verifique su veracidad y replicabilidad.

Este proceso resulta esencial para avanzar en los distintos campos científicos.

Cuando un artículo es publicado en una revista, esta es la encargada de encontrar revisores voluntarios que se encarguen de revisar el estudio. De esta manera se da por válida una investigación.[2]

Aunque el ámbito de las revisiones científicas no ha sufrido grandes cambios desde que se empezaran a realizar siguiendo el método por pares, con el gran avance tecnológico que se ha producido en los últimos años, han empezado a surgir diferentes plataformas digitales[3][4] en las que los revisores pueden reunir todas sus revisiones de una manera sencilla además de poder realizar las revisiones a los artículos presentes en ellas. En la actualidad estos revisores hacen su trabajo de una forma completamente altruista y voluntaria, no reciben ninguna recompensa más allá de la realización de contribuir a la comunidad científica y de obtener el prestigio de revisar para ciertas revistas de importancia.

Por otro lado, en la actualidad surgen innovaciones a un ritmo frenético y cada día salen nuevas aplicaciones de la tecnología. Una de las aportaciones más importantes de los últimos años ha sido la tecnología blockchain.

Esta tecnología consiste en una base de datos pública, descentralizada y perpetua que permite almacenar una gran cantidad de datos sin depender de un único servidor. Pueden ser desde datos de transferencias de divisas, como el Bitcoin, hasta cualquier otro tipo de información.

Además de Bitcoin, durante los últimos años, una gran cantidad de proyectos han surgido utilizando este ecosistema para almacenar transferencias de divisas, existen una infinidad de iniciativas, como Ethereum, Stellar o Cardano, de las que hablaremos en el punto "2.7 Altcoins". Este último se caracteriza por implementar un sistema de ejecución de código dentro de la cadena de bloques, permitiendo la implementación de aplicaciones descentralizadas utilizando el lenguaje Solidity.

Ethereum no solo ha abierto un nuevo paradigma dentro de la tecnología blockchain implementando los contratos inteligentes, explicados en el punto "2.6 Ethereum", sino que también implementa diversas funcionalidades como la generación de tokens. Podemos definir un token como un objeto parecido a una moneda, pero sin valor de curso legal, aplicando la tecnología blockchain se permite crearlos, así como almacenar información en ellos o transferirlos entre usuarios[5]. Gracias a esta implementación, surgen los Tokens No Fungibles o NFTs, que permiten generar una entidad digital única tal y como se comenta en el punto '2.8 Contratos ERC y Tokens No Fungibles (NFT)'.

Las aplicaciones de la cadena de bloques son muy diversas, por ejemplo, se utiliza como sistema de almacenamiento en la nube, como sistema de seguridad automatizado o como ecosistema de desarrollo gracias a los contratos inteligentes o Smart Contracts.

Cada vez son más las empresas que deciden adoptar esta tecnología, puesto que ofrece un alto nivel de fiabilidad y seguridad. Por ejemplo, Walmart[6] utiliza el sistema blockchain para realizar el seguimiento de sus productos, aprovecha esta tecnología para mejorar la trazabilidad de estos de una forma descentralizada. También encontramos como la compañía de envíos navieros Maersk[7] implementa el servicio de la cadena de bloques para realizar el seguimiento por todo el mundo de sus contenedores, para ello utilizan la plataforma Tradelens[8]. Otro ejemplo es la aerolínea British Airways, que utiliza la tecnología blockchain para tener un control de los datos de vuelos y viajeros.[9]

Esta tecnología abarca muchos campos y se puede implementar con fines muy variados. Empresas relacionadas con el mundo científico también han empezado a utilizarla para diferentes tareas, es el caso de UnitedHealthCare[10], implementado un sistema para mejorar los directorios médicos.

Surgen también iniciativas para implementar el sistema blockchain en las investigaciones científicas, utilizando la cadena de bloques para crear sistemas de almacenamiento y gestión de artículos y revisiones. Se puede dar así la certeza de que son únicos y no han sido alterados, solventando algunos de los problemas actuales como lo son el plagio y el fraude en su elaboración. Asimismo, actualmente resulta complejo establecer sistemas de control en los datos de investigación, es por esto por lo que la blockchain podría ser la solución al problema, proporcionando la originalidad e inmutabilidad de los datos introducidos.

Por último, encontramos la posibilidad de implementar alicientes en las revisiones por pares, consiguiendo una mejoría en las condiciones de los revisores, así como en la calidad de estas, materia en la que se centra el desarrollo de este proyecto.[11]

1.2 Motivaciones

La principal motivación de este proyecto es tratar de contribuir a la mejora en el reconocimiento de la labor de los revisores de artículos científicos, los cuales apenas reciben algún tipo de recompensa por la revisión de estas publicaciones.

Desde hace un tiempo, la idea de implantar un sistema para poder recompensar la labor de los revisores ha cogido fuerza ya que, en muchas ocasiones, los mejor cualificados rechazan realizarlas debido al arduo trabajo que ello supone, el tiempo que conlleva y el poco incentivo que existe. Esto deriva en que las revisiones en muchas ocasiones acaben siendo realizadas por revisores inadecuadamente cualificados.

En los últimos años han surgido diferentes plataformas que permiten subir los artículos científicos haciéndolos mucho más accesibles, sin embargo, no han supuesto un estímulo suficiente para los revisores.

Por todo ello, aprovechando el auge de las tecnologías descentralizadas se busca poder ayudar a mejorar la situación por medio de la implementación de una plataforma, basada en la tecnología blockchain, que permita dar varios tipos de recompensas a los revisores y que pueda fomentar la realización de revisiones por pares, así como aumentar su calidad.

1.3 Objetivos

Los objetivos principales que persigue este proyecto son:

- Promover las revisiones de artículos científicos por medio de incentivar a los revisores
- Desarrollo de una aplicación web (Front-End) que haga uso del ecosistema Ethereum de Recompensas
- Propuesta de arquitectura que integre recompensas en blockchain con las herramientas de revisión por pares

Chapter 1. Introduction

1.1 Backgrounds

Scientific research has laid the foundations for us to move forward as a society, from technology to health care, and across all branches of science. For this to be possible, all research is governed by the scientific method[1]. This is based on the premise of observing a phenomenon in order to propose a hypothesis and test it through experimentation.

When a researcher makes a contribution in the form of an article to the scientific community, it must be reviewed for veracity and quality. A study must be replicable by any subject anywhere, otherwise we could not be certain of the veracity of the experiment.

The scientific method began to be used in the work of the brilliant researcher Galileo Galilei in the 16th century[2] and, today, it is still the method used by all researchers around the world. But it is not enough to research and write an article for subsequent publication in a journal, it must go through a process called "peer review" to verify its veracity and replicability.

This process is essential for progress in different scientific fields.

When an article is published in a journal, the journal is responsible for finding volunteer reviewers to review the study. In this way, the research is considered valid.[2] Although the field of scientific review has not been a very successful one.

Although the field of scientific reviews has not undergone major changes since they began to be carried out following the peer review method, with the great technological advances that have occurred in recent years, different digital platforms have begun to emerge[3][4] where reviewers can gather all their reviews in a simple way as well as being able to review the articles present in them. Currently, these reviewers do their work in a completely altruistic and voluntary manner, receiving no reward beyond the realisation of contributing to the scientific community and obtaining the prestige of reviewing for certain important journals.

On the other hand, innovations are emerging at a frenetic pace and new applications of technology are emerging every day. One of the most important contributions in recent years has been blockchain technology.

This technology consists of a public, decentralized and perpetual database that allows a large amount of data to be stored without relying on a single server. This can range from currency transfer data, such as Bitcoin, to any other type of information.

In addition to Bitcoin, in recent years, a large number of projects have emerged using this ecosystem to store currency transfers, there are countless initiatives, such as Ethereum, Stellar or Cardano, which we will talk about in point "2.7 Altcoins". The latter

is characterised by implementing a code execution system within the blockchain, allowing the implementation of decentralized applications using the Solidity language.

Ethereum has not only opened a new paradigm within the blockchain technology by implementing smart contracts, explained in point "2.6 Ethereum", but it also implements various functionalities such as the generation of tokens. We can define a token as an object similar to a coin, but without legal tender value, applying blockchain technology allows them to be created, as well as storing information on them or transferring them between users[5]. Thanks to this implementation, Non Fungible Tokens or NFTs arise, which allow the generation of a single digital entity as discussed in point '2.8 ERC Contracts and Non Fungible Tokens (NFTs)'.

The applications of the blockchain are very diverse, for example, it is used as a cloud storage system, as an automated security system or as a development ecosystem thanks to Smart Contracts.

More and more companies are choosing to adopt this technology, as it offers a high level of reliability and security. For example, Walmart[6] uses the blockchain system to track its products, taking advantage of this technology to improve their traceability in a decentralized way. We also find how the shipping company Maersk[7] implements the blockchain service to track its containers around the world, using the Tradelens[8] platform. Another example is British Airways, which uses blockchain technology to keep track of flight and passenger data[9].

This technology covers many fields and can be implemented for a wide variety of purposes. Companies related to the scientific world have also started to use it for different tasks, as in the case of UnitedHealthCare[10], which has implemented a system to improve medical directories.

Initiatives are also emerging to implement the blockchain system in scientific research, using the blockchain to create storage and management systems for articles and reviews. This can provide certainty that they are unique and have not been altered, solving some of the current problems such as plagiarism and fraud in their production. Likewise, it is currently difficult to establish control systems for research data, which is why the blockchain could be the solution to the problem, providing the originality and immutability of the data entered.

Finally, we find the possibility of implementing incentives in peer reviews, achieving an improvement in the conditions of the reviewers, as well as in the quality of these, a matter on which the development of this project is centralized.[11] The blockchain could be the solution to the problem, providing the originality and immutability of the data entered.

1.2 Motivations

The main motivation of this project is to try to contribute to the improvement of the recognition of the work of the reviewers of scientific articles, who hardly receive any kind of reward for the review of these publications.

For some time now, the idea of implementing a system to reward the work of reviewers has been gaining momentum as, on many occasions, the best qualified reviewers refuse to carry out reviews due to the arduous work involved, the time involved and the little incentive that exists. As a result, reviews often end up being carried out by inadequately qualified reviewers.

In recent years, different platforms have emerged that allow scientific articles to be uploaded, making them much more accessible, but they have not provided sufficient stimulus for reviewers.

Therefore, taking advantage of the rise of decentralized technologies, the aim is to help improve the situation by implementing a platform, based on blockchain technology, which allows various types of rewards to be given to reviewers and which can encourage the performance of peer reviews, as well as increase their quality.

1.3 Goals

The main goals of this project are as follows:

- Promote scientific article reviews by providing incentives to reviewers.
- Development of a web application (Front-End) that makes use of the Ethereum Rewards ecosystem.
- Propose an architecture that integrates blockchain rewards with peer review tools.

Capítulo 2. Estado del Arte

2.1 Revisiones por Pares

Para conocer este modelo, es conveniente explicar cómo llegamos a la situación actual en relación con los artículos científicos. La primera revisión previa a una publicación es de 1665, sin embargo, no sería hasta el siglo siguiente que aparecería la primera revisión por pares. Al principio eran escasas, su número no aumentó de manera notable hasta el siglo XX. A partir de entonces, este sistema de revisiones evolucionaría hasta el que conocemos hoy en día. Con esta metodología, varias personas, llamadas revisores, evalúan una investigación de uno o varios autores con el objetivo de constatar la calidad y veracidad del trabajo [12].

Existen tres tipos de revisión por pares:

- Simple-ciega: esta modalidad tiene la característica de que el revisor conoce la identidad del autor, pero el autor no conoce quien es la persona que está revisando su artículo. Actualmente es la modalidad más extendida y usada.
- Abierta: en esta modalidad, el autor conoce la identidad del revisor al igual que el revisor conoce la del autor.
- Doble-ciega: en este caso, ni el autor conoce la identidad del revisor ni el revisor conocer la identidad del autor.

Una revisión por pares tiene como principales objetivos medir la veracidad y la calidad de un artículo científico, pero este proceso debe ser igualmente claro y profesional. Mientras la modalidad doble-ciega guarda el anonimato de los actores involucrados en la revisión, en las otras dos modalidades al menos una de las identidades es conocida por la otra persona. Es por lo que se podría dar el caso de que cuestiones interpersonales tuvieran algún efecto durante el proceso de revisión.

Cuando se realiza la revisión, con el propósito de mejorar la calidad del estudio, el revisor puede señalar los errores que se hayan cometido en la edición del artículo, así como modificaciones pertinentes a realizar.[12]

Finalmente, es importante conocer las ventajas y las desventajas que se pueden encontrar en el modelo de revisión por pares. Por un lado, las principales ventajas son:[12]

- Se consiguen opiniones de calidad y contrastadas acerca de la información incluida en un artículo científico.
- Se otorga confianza al proceso de publicación de los artículos.
- Cuando se trata de una investigación, ayuda a darle veracidad.

Por otro lado, también encontramos diversas desventajas, las más importantes son:[13]

- Puede pasar mucho tiempo desde que el artículo se escribe hasta que se somete al proceso de revisión.
- Número insuficiente de revisores que ralentiza el proceso.
- Los requisitos y cualidades para poder ser revisor no están formalmente definidos por lo que muchas veces los revisores no están suficientemente preparados para la revisión que debe realizar.

2.2 Plataformas de Revisiones por Pares

En la actualidad, existen una gran variedad de revistas científicas en línea en las que se publican artículos para posteriormente ser revisados por pares. Un investigador, tras haber realizado un trabajo de investigación, y haber redactado el correspondiente artículo, se pone en contacto con una de estas revistas para poder publicar su trabajo y, no solo lograr que sea accesible al público general para contribuir con la comunidad científica, sino también esperando que el journal contacte con revisores que se encarguen de revisar y validar la información que ha sido subida a la revista. A través de este procedimiento lo que se busca es darle veracidad y reconocimiento al artículo.

La revisión por pares juega un papel fundamental a la hora de aportar credibilidad y confirmación a una investigación de cualquier ámbito.

Como hemos comentado, existen un número muy grande de journals en los que hacer aportaciones, con la idea de poder localizar y ofrecer identificadores a autores y revisores, nace ORCID[3].

Dentro de estas plataformas, podemos destacar algunas como Publons[4], que da la posibilidad de operar directamente con ORCID. Este portal fue desarrollado en Nueva Zelanda durante el año 2012, con el objetivo de reconocer, registrar y clasificar las revisiones por pares. Permite crearse a los revisores un perfil de usuario en el que incluir su ámbito de investigación, sus revisiones y las revistas para las que ha revisado.

Los perfiles de los revisores son públicos, lo que permite a los mismos acreditar de una forma sencilla las contribuciones que han realizado y, por otro lado, facilita a las revistas encontrar revisores que se ajusten a sus parámetros. Aunque, actualmente, es mayoritario el número de revistas que no ofrecen los datos de los autores de las revisiones por pares, conservando su información personal en el anonimato.

En contraposición a esta situación, han nacido plataformas como F1000[14] o PeerJ[15], que destacan por ofrecer revisiones por pares abiertas, es decir, los datos del investigador, así como los de los revisores son accesibles y públicos.

Parece ser que este es el camino que cada vez más plataformas quieren seguir y que poco a poco se abrirá paso dando lugar a las revisiones por pares Open Access.

Uno de los problemas que tienen ambos tipos de plataformas, es la falta de recompensas a los revisores, es decir, los investigadores que contribuyen revisando artículos, en su conjunto prácticamente total, lo hacen de una manera altruista, sin recibir nada más allá del prestigio por contribuir en una revista con renombre.

Muchas revistas han buscado e investigado acerca de cómo crear un sistema de recompensas o reconocimiento a revisores, debido a esta idea, han nacido plataformas como Peereviewers[16] o Rubriq (cerrada en 2017). Ofreciendo un sistema de recomendación para revisores a los autores de las investigaciones, de esta manera, aquellos que lograran hacer mejor su labor podían destacar entre los demás, consiguiendo así mayor reputación. Algunos grupos editoriales grandes, generalmente los más importantes, llegan a dar algún tipo de incentivo a los revisores por su trabajo como descuentos o libros, no obstante, muchas veces esta actividad se ve de una forma negativa ya que hay investigadores que creen que puede afectar a la calidad y honestidad de las revisiones.

Sin embargo, hasta ahora solo hemos hablado de sistemas de recompensas que hacen uso de sistemas centralizados. Es aquí donde entra el planteamiento de crear un sistema descentralizado de recompensas que puedan implementar las plataformas de revisiones por pares. De esta manera se consigue un sistema honesto e inmutable, que resulta más creíble y transparente al no depender de una plataforma centralizada.

2.3 Bitcoin y sus Orígenes

En el año 2009, surge la primera criptomoneda bajo el nombre Bitcoin[17]. Fue creado bajo el seudónimo de Shatoshi Nakamoto, aunque la identidad real de la persona, o colectivo, es desconocida. Esta criptomoneda se introdujo con la intención de crear una manera de pago totalmente digital, anónima mediante una red descentralizada y que pudiera prescindir de un tercero de confianza en el escenario de una transacción entre dos partes.[18]

La principal característica en su momento de lanzamiento fue la de estar basada en una red descentralizada o blockchain. Esto conllevaba una gran innovación y permitía a Bitcoin incorporar todas las virtudes en materia de seguridad y anonimato mencionadas anteriormente.

Bitcoin es una plataforma estrictamente financiera, es decir dedicado exclusivamente a transacciones de criptodivisas entre cuentas. Como se ha mencionado, uno de los objetivos que tiene esta criptomoneda es el anonimato y la seguridad de las transacciones, por ello introduce su propio sistema para lograrlo. El Bitcoin se define como *una cadena de firmas digitales*[17]. Esto ayuda a entender cómo funciona el sistema Bitcoin, con la cadena de bloques que se ha explicado en el apartado de Blockchain.

Esta criptomoneda ha sufrido un crecimiento vertiginoso desde su lanzamiento y ha dado lugar a que, por un lado, la gente de a pie conozca el ecosistema de las criptomonedas y por otro lado ha ayudado a extender la tecnología blockchain ya que, dado su éxito, muchos profesionales empezaron a interesarse por ella.

Su inicio fue muy discreto, la primera transacción se realizó entre su creador Shatoshi Nakamoto y un desarrollador llamado Hal Finnery quien recibió los primeros 50 bitcoins de la historia.[19]

Desde que se lanzara en 2009 hasta que se fue adoptando, pasaron algunos años, cuando diversas organizaciones reconocidas a nivel mundial empezaron a aceptarlas como forma de pago, aunque la confianza de la sociedad por este tipo de monedas costaría aún más tiempo en consolidarse. Aunque debido a que Bitcoin era un proyecto de código abierto, no tardaron mucho en surgir criptomonedas similares.[19]

Desde el año 2012, su crecimiento fue imparable y cada vez más organizaciones aceptaban pagos en Bitcoins hasta llegar a unos valores de valorización que nadie habría logrado sospechar.

Si bien es cierto que el éxito del Bitcoin es irrefutable, también lo es que tiene unas limitaciones. Su principal limitación es que fue diseñada únicamente como una moneda, es decir, no se puede extender su uso más allá de las transacciones económicas, algo que como veremos, dio lugar a proyectos más ambiciosos, por ejemplo, Ethereum.

2.4 Tipos de Arquitecturas

Existen diferentes tipos de arquitecturas, aunque nos vamos a centrar en las dos que hemos utilizado en este proyecto, la arquitectura centralizada y la arquitectura distribuida. Sus características son totalmente diferentes y, por tanto, están orientadas para fines distintos.

La Arquitectura centralizada es aquella en la que en la red existe un nodo central del que dependen el resto de los nodos. Esto tiene como principal desventaja que, si el nodo central sufre algún tipo de problema, esto se verá reflejado en el resto de nodos de la red.

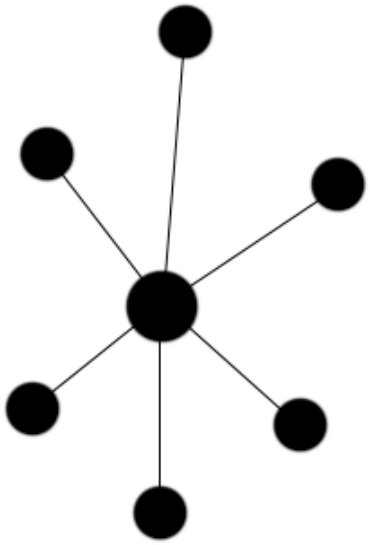


Figura 2.1: Arquitectura Centralizada

Por otro lado, existe la arquitectura distribuida que es aquella en la que no existe un nodo central y en la que todos los nodos se conectan entre sí. En este tipo de arquitectura, todos los nodos son independientes, aunque entre ellos existen unas condiciones equivalentes.

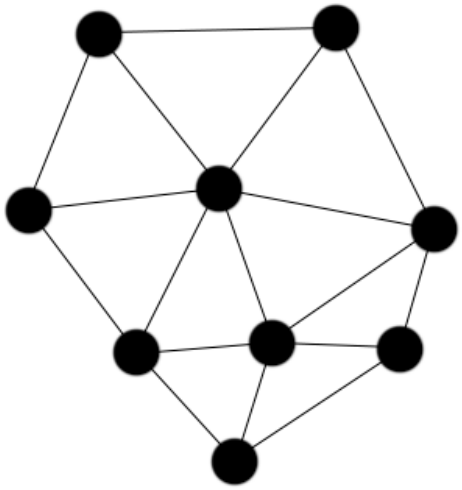


Figura 2.2: Arquitectura Distribuida

Hasta ahora, el tipo de arquitectura centralizada ha sido la más utilizada pero cada vez están surgiendo nuevas tecnologías basándose en una arquitectura distribuida.

2.5 Tecnología Blockchain

La tecnología blockchain, o cadena de bloques, originalmente fue creada para almacenar las transacciones de Bitcoin (Nakamoto, 2008), podemos definirla como una base de datos distribuida e inmutable, siendo la primera divisa digital descentralizada.

Como se ve con más detalle en la siguiente sección, Bitcoin almacena todas sus transacciones en el libro mayor o "ledger". Para poder recoger toda esta información, se creó el sistema de la blockchain que ofrece un sistema seguro y descentralizado.

Comenzaremos hablando por el funcionamiento de la cadena de bloques. Como ya hemos comentado, es una base de datos compartida por un gran número de usuarios de manera peer-to-peer. La información que contiene es pública, compartida por todos los usuarios. Para añadir información nueva a la blockchain, tiene que haber un consenso por parte de la mayoría de las partes que conforman la red. Una vez que la información se escribe en un bloque, es inmutable, esto es, no se puede eliminar ni modificar y quedará escrita para siempre.

El proceso de adición de bloques a la blockchain recibe el nombre de minado. Este proceso es llevado a cabo por los mineros que se encargan de almacenar las últimas transacciones incluyéndolas en el bloque que más tarde se minará y con ello se confirmarán las transacciones. Una vez que el bloque tiene almacenadas todas las transacciones, el minero debe encontrar un hash único para el bloque. Cuando este hash ha sido encontrado, el minero recibe una compensación económica por el minado y añade el bloque a la red blockchain para que el resto de los nodos validen la información.[20]

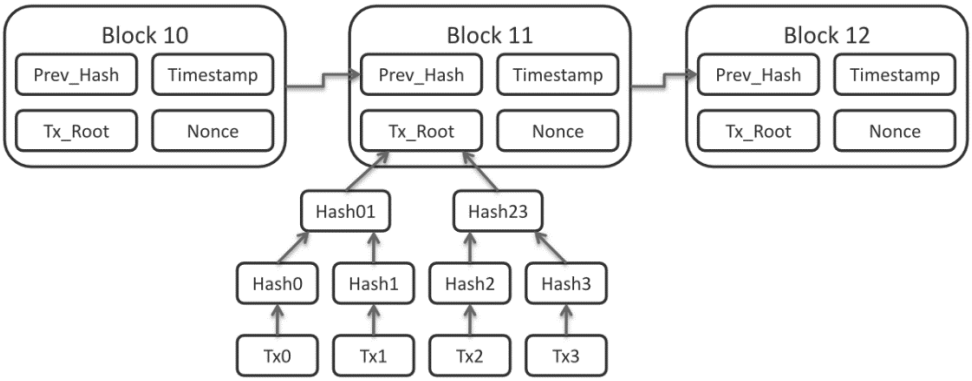


Figura 2.3: Esquema de la red blockchain de Bitcoin

Para poder dar un bloque por minado, existe un procedimiento obligatorio con fines de verificación llamado proof-of-work. Siendo este sistema el que hace prácticamente imposible el borrado de información en bloques altos "ya escritos", y el que aporta seguridad a la blockchain.

La proof-of-work consiste en escanear en busca de un valor que cuando fue hasheado, al igual que con SHA-256, el hash comience con un número de cero bits. El trabajo medio que hace falta es exponencial en el número de cero bits requeridos y puede verificarse ejecutando un único hash.

Los pasos para ejecutar la red son los siguientes:

1) Las transacciones nuevas se transmiten a todos los nodos.

2) Cada nodo recoge todas las transacciones en un bloque.

3) Cada nodo trabaja en resolver una proof-of-work compleja para su bloque.

4) Cuando un nodo resuelve una proof-of-work, transmite el bloque a todos los nodos.

5) Los nodos aceptan el bloque si todas las transacciones en él son válidas y no se han gastado con anterioridad.

6) Los nodos expresan su aceptación del bloque al trabajar en crear el siguiente bloque en la cadena, usando el hash del bloque aceptado como hash previo.[21]

La red blockchain se encuentra distribuida en los nodos, cada uno de ellos tiene una copia exacta de la cadena de bloques. De esta forma se aseguran su inmutabilidad y acceso continuo.

Pese a que sus orígenes son debidos al Bitcoin, esta tecnología se aplica a una gran cantidad de proyectos, muchos de ellos basados en criptodivisas y otros implementando funcionalidades nuevas, como Ethereum o Cardano, que trataremos más adelante, pero también son muchas las empresas que están implementando este sistema, tal y como describimos en la Introducción.

Por lo tanto, podemos de hablar de varias ventajas de la blockchain, entre ellas destacamos: la información nunca se ve alterada ni se pierde, los datos no se pueden falsear y nos proporciona transacciones seguras y muy confiables.

Aunque, en contraposición, podemos nombrar algunas de las desventajas que se encuentran actualmente en debate: la contaminación que puede producir el minado de bloques, el coste de implementación, la madurez de la tecnología, la inmutabilidad de la blockchain, provocando que no pueda haber cambios en caso de error o la variabilidad que tiene la velocidad del procesado de la información, dependiendo, entre otros, de las comisiones pagadas a los mineros[22].

Recientemente han salido diversos estudios apuntando el gran consumo energético que provoca el minado de bloques en la blockchain de Bitcoin. Según el Índice de Consumo Energético del Bitcoin[23] realizado por Cambridge, solo el Bitcoin consume 123.77 TWh (teravatios/hora) al año, más que algunos países. Otras blockchains más modernas, como la que implementa Ethereum, llevan años reduciendo gradualmente su consumo llegando a uno mínimo, gracias a mejoras como el proof-of-stake.

2.6 Ethereum

Ethereum es una plataforma digital que, basándose en la tecnología blockchain y adoptando la idea principal de Bitcoin, implementa su propia criptomoneda, el Ether y grandes mejoras como los Smarts Contracts, marcando un antes y un después en el uso de las blockchains.

Fue creada originalmente por Vitalik Buterin en el año 2015, teniendo en mente desde un principio habilitar a los desarrolladores la posibilidad de crear aplicaciones descentralizadas. Ethereum nace de la ambición de Buterin por llevar el Bitcoin un paso más allá que ser solamente un ecosistema financiero de transacciones.

En un inicio, el lenguaje de programación utilizado para el desarrollo de contratos inteligentes era Serpent[24], consiste en un lenguaje ensamblador que compila código Ethereum Virtual Machine (EVM) con extensiones para de alto nivel. El propio repositorio del proyecto no recomienda su uso para construir aplicaciones debido a que se trata de un lenguaje de bajo nivel.

Seguidamente, en el año 2015, salió Solidity[25] un lenguaje de programación también orientado a compilar código EVM, pero en esta ocasión de alto nivel. Tiene una sintaxis similar a la de JavaScript, lo que facilita su entendimiento y aprendizaje.

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) {
        storedData = x;
    }

    function get() constant returns (uint) {
        return storedData;
    }
}
```

Figura 2.4: Ejemplo sencillo de código en Solidity

Con Ethereum se abre un nuevo paradigma que no existía con el Bitcoin, la creación de aplicaciones descentralizadas. Éstas pueden competir con las aplicaciones centralizadas que han existido hasta ahora para reemplazarlas por un sistema descentralizado, ya que ofrece puede ofrecer diversas ventajas entre las que podemos encontrar: un nivel alto de seguridad debido a que todos los nodos de la red contienen la una copia de la información y esto unido a que para añadir información a la red blockchain es necesario conseguir el consenso de la mayoría de la red, dificulta el éxito de posibles ataques en la red. Otra ventaja es que debido a que no existe un nodo central del que dependen el resto de nodos de la red, tienen lugar menos interrupciones del servicio ya que no hay un nodo que tenga más autoridad dentro de la red. Por otro lado, la transparencia es otra ventaja de las aplicaciones descentralizadas y es que la información es accesible a todos los usuarios lo que complica la censura de información.

Aunque las aplicaciones descentralizadas tienen diversas ventajas frente a las aplicaciones centralizadas, también tienen algunas desventajas, la primera es que son en general más complicadas de desarrollar y de mantener ya que al trabajar con una red distribuida, al realizar una actualización, esta tiene que realizarse en todos los nodos de la red. Otra desventaja es que la seguridad de este tipo de aplicaciones conlleva un enfoque distinto al que puede tener en las aplicaciones centralizadas ya que para que un ataque tenga éxito, debe afectar a la mayoría de los nodos de la red lo que dependiendo del tamaño de esta es muy complicado, por tanto, hay que utilizar mecanismos de seguridad propios para este tipo de tecnología. Finalmente, las aplicaciones descentralizadas no suelen tener una la mejor experiencia de usuario ya que trabaja con campos diferentes a las aplicaciones centralizadas, por ejemplo, en vez de utilizar datos personales del usuario, suele utilizar las claves públicas y privadas de los clientes lo que no las hace tan intuitivas[26].

La moneda que mueve Ethereum se llama Ether y, al igual que como ocurre en el caso del Bitcoin, se paga una cantidad en forma de recompensa a los mineros por cumplir su trabajo. La comisión que hay que pagar para que nuestro código sea ejecutado se llama 'gas'. A mayor complicidad del código, mayor será el gas a pagar por su ejecución. Es por esto que en Solidity es muy importante tratar de tener un código eficiente, que no realice operaciones innecesariamente complejas.

El fin último de Ethereum es conseguir que se puedan desarrollar programas que sean autogobernables, es decir, que realicen transacciones de manera automática en la blockchain.

Todas estas razones lo convierten en una tecnología idónea para utilizar en el proyecto.

2.7 Altcoins

Además del Bitcoin y Ethereum, existen una infinidad de criptodivisas con proyectos muy diversos. Cada día salen a disponibilidad de los usuarios más criptodivisas, son las llamadas Altcoins.

Podemos destacar algunas de ellas que cuentan con proyectos interesantes, para ejemplificar las distintas opciones y los distintos objetivos que pueden llegar a tener.

Ripple[27] (XRP) se trata de una criptodivisa orientada a poder realizar transacciones económicas, pero de una forma rápida y con unas comisiones menores que las que posee el Bitcoin. Este proyecto no se basa en la tecnología blockchain, si no que utiliza su propia plataforma de datos, llamada Algoritmo de Consenso de Protocolo Ripple (RPCA)[28]. Pero no solo se queda ahí, está enfocada a realizar transacciones internacionales rápidas, creando un propio ecosistema de pago en general. Uno de los puntos que más destaca es que está diseñado como un método de pago diario y permite ser cambiado a cualquier moneda o valor con una comisión mínima [29].

Stellar Lumens[30] (XLM) es un protocolo descentralizado de código abierto que sirve principalmente para adoptar el papel de intermediario en la conversión de divisas, esto quiere decir que resulta útil cuando se quiere realizar una conversión de monedas fiat (divisas normales) a criptomonedas y viceversa. Tiene la particularidad de que no es minable, en su inicio se emitieron 100.000 millones de tokens. Las bases del proyecto, protocolo, clientes, etc. forman parte de un total de más de 90 proyectos. Todos ellos bajo licencia Apache 2.0, lo que permite que el código sea utilizado tanto en iniciativas públicas como en privadas[31].

Cardano[32] (ADA) se trata de la primera blockchain desarrollada científicamente, esto indica que, desde sus inicios, toda su etapa de desarrollo se ha basado en el método científico, habiendo sido su "white paper" revisado por pares. Se basa en una serie de

tecnologías que hacen de Cardano una blockchain de alta velocidad. El objetivo que tiene el proyecto es el de convertirse en un token de alto uso, permitiendo transacciones rápidas y, en la próxima actualización "Goguen", integrará la posibilidad de albergar Smart Contracts[33].

2.8 Contratos ERC y Tokens No Fungibles (NFT)

ERC son las siglas en inglés de Ethereum Request for Comments, son documentos que usan los contratos inteligentes de Ethereum para describir una funcionalidad específica y unas reglas concretas acerca del comportamiento en la implementación de tokens dentro del sistema de Ethereum. Para que un ERC se pueda convertir en estándar, debe pasar un proceso de aprobación que se llama EIP, de sus siglas en inglés Ethereum Improvement Proposal.[34]

Dentro del listado de ERCs[35] podemos encontrar diversos estándares que albergan una gran variedad de funcionalidades, vamos a destacar el ERC-20 y el ERC-721, pues son los dos que se han utilizado en este proyecto.

Por un lado, el ERC-20[36] se trata de un protocolo que permite la implementación de una API estándar para la implementación de tokens en el ecosistema Ethereum. Habilita la posibilidad de transferir tokens, así como de aprobarlos para ser gastados por terceras personas no propietarias del token en cuestión. Se creó por la necesidad de tener un estándar que pudiera ser implementado en cualquier proyecto para hacer una gestión de tokens.

Hay distintas implementaciones de este estándar disponibles en la red, cada una con sus ventajas y desventajas, desde priorizar el consumo de gas hasta mejorar la seguridad de este.

Por otro lado, el ERC-721[37] se define como un estándar para Tokens No Fungibles (NFT), esto quiere decir que permite crear tokens únicos e inalterables dentro de los Smart Contracts. El estándar provee de funcionalidades para realizar un seguimiento y transferir NFTs. Éstos pueden representar activos tangibles de la vida real a través de un medio digital, como por ejemplo casas, arte, artículos de colección o incluso deudas.

Nace para suplir las necesidades que no cubre el estándar ERC20, éste es insuficiente cuando hay que realizar un seguimiento de los NFT, ya que cada activo es distinto (no fungible) mientras que cada uno de los tokens es idéntico (fungible).

La popularidad actual de los NFT es debida, en gran parte, a las subastas de arte digital que se han desarrollado durante este año, llegando a alcanzar el primer token no fungible

del artista ‘Beeple’ los 69.3 millones de dólares[38]. Aunque el estándar ERC-721 que utilizó CryptoKitties[39] fue el primero en ser usado en la categoría NFT[40].

Esto ha habilitado que se hayan creado multitud de páginas web que se dedican a vender arte digital en forma de NFT, encontrándonos con proyectos pequeños como CryptoSushis[41] y llegando a mercados enteros de tokens como OpenSea[42] o AtomicHub[43].

Existen muchos otros estándares, como el ERC-1155[44], una variante que ofrece el potencial del ERC-721 pero orientado a tokens semi-fungibles, es decir, permite que un contrato trabaje con múltiples tipos de tokens a la vez, y habilita la posibilidad de que un contrato inteligente gobierne un número ilimitado de tokens, eliminando la necesidad de aprobar contratos de tokens individuales por separado[45].

Capítulo 3. Metodología y Tecnología

3.1 Introducción a las Metodologías Ágiles

El desarrollo de este proyecto conlleva muchas decisiones y modificaciones que hay que realizar de manera dinámica, por ello se requería un modo de trabajo que nos permitiera ajustarnos a estas variaciones de una manera eficaz. Con este fin, el equipo tenía claro que lo mejor sería adoptar una metodología ágil para llevar a cabo el desarrollo.

Las ventajas de las metodologías ágiles para este tipo de proyectos son numerosas, ya que ofrecen unas características que ayudan a agilizar el proceso de desarrollo, por ejemplo, son muy flexibles y permiten adaptarse mejor a los cambios que puedan surgir de manera repentina. Además, la alta colaboración y comunicación entre los integrantes del equipo es otra característica básica de este tipo de metodologías. Pero la principal ventaja que el equipo vio en ellas es que se pueden adaptar al equipo y no al revés, lo que las convierte en una herramienta muy efectiva para el progreso fructífero del proyecto [46].

Dentro de las metodologías ágiles, el equipo optó por utilizar la metodología SCRUM, ya que como se explicará a continuación, debido a sus características y a la experiencia previa que el equipo tenía en su uso, se decidió que era la mejor opción para el proyecto.

3.1.1 SCRUM

Idealmente, esta metodología está orientada a entornos de trabajo que cumplan cuatro características. La primera es la incertidumbre, es decir, el equipo conoce el objetivo al que se quiere llegar, pero no existe un plan detallado de cómo se conseguirá alcanzar. La segunda, es la autoorganización, es decir, no hay roles predefinidos dentro del equipo. La tercera es que el conocimiento adquirido debe ser compartido entre todos los integrantes del equipo y para conseguirlo, todos los miembros deben trabajar en diferentes áreas del proyecto para luego poder compartir sus nuevos aprendizajes. Y finalmente, el control del proyecto es llevado a cabo por todos los integrantes a la vez.[47]

Estas cuatro características son las que desde el principio han estado presentes en nuestro equipo. Desde el principio conocíamos cual era la aplicación que queríamos desarrollar, pero en ningún momento hemos tenido un camino claro que seguir para poder conseguirlo. Además, todos hemos participado en diferentes partes del proyecto, tanto en el proceso de investigación como en el desarrollo y la implementación.

También ha sido muy importante el control que se ha impuesto dentro del propio grupo para que el desarrollo fuese equitativo en las diferentes partes y no centrarnos solo en una en concreto.

Otra característica distintiva de esta metodología es que el proceso de trabajo se basa en periodos de trabajo rápidos llamados “sprints”. Los diferentes sprints llevados a cabo por el equipo venían marcados por las reuniones con los tutores, en las que se revisaban los avances que se habían logrado y se establecían los objetivos a cumplir para el siguiente sprint.

Esta metodología nos ha permitido llevar un ritmo de desarrollo muy fluido y nos ha ayudado a poder solventar los problemas que iban surgiendo gracias a su flexibilidad.

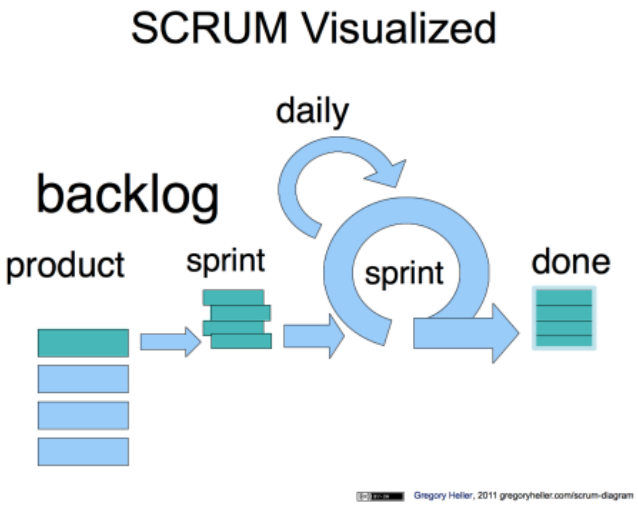


Figura 3.1: Metodología SCRUM representada gráficamente

3.1.2 Trello

Se trata de una herramienta para organizar tareas, al ser online es versátil para realizar una organización en equipo, pues no es necesario descargar ninguna aplicación y permite crear tableros para trabajar en grupo de una forma muy sencilla.

Esta aplicación se basa en el método Kanban de gestión de proyectos, de esta manera todas las actividades de un proyecto se presentan en forma de tarjetas sobre un tablero que es accesible para todo el equipo, lo que facilita la creación y edición de tareas.[48]

Pese a que se base en un método distinto al utilizado, las diferencias entre ambos no suponen un impedimento para utilizar esta herramienta, el Kanban se basa en una cadencia de flujo continuo, con cambios que pueden suceder en cualquier momento y con una entrega continua, en contraposición del método SCRUM explicado en el apartado anterior.[49]

Trello cuenta con distintas suscripciones de pago que implementan funcionalidades exclusivas, no obstante, en el caso de este proyecto se ha utilizado la versión gratuita ya que es suficiente para una correcta organización.

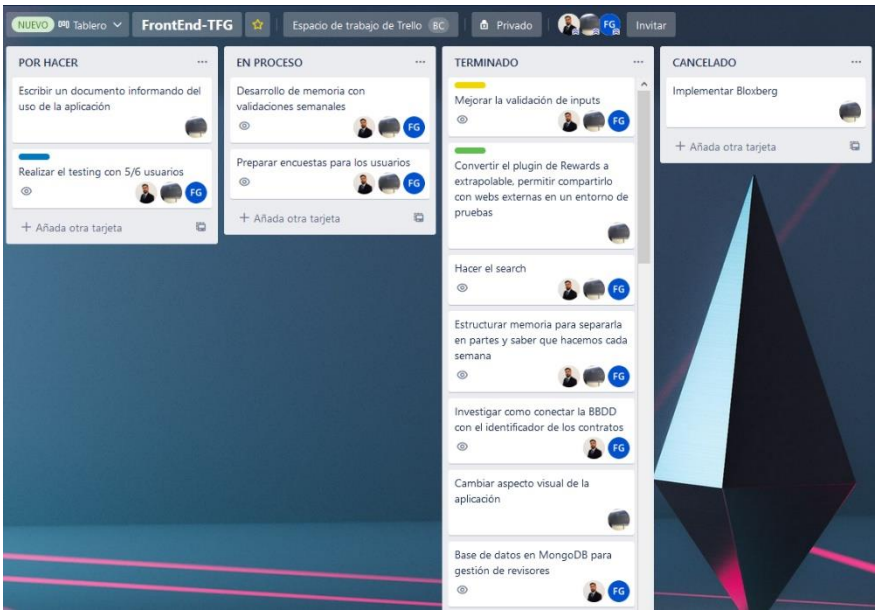


Figura 3.2: Tablero de Trello con método SCRUM

3.2 Plan de Trabajo

Desde el principio, el modo de trabajo del equipo ha pasado por diferentes fases que han sido necesarias para desarrollar la aplicación.

Durante los primeros meses, el objetivo se centró en la investigación y aprendizaje de las herramientas que necesitábamos usar. En esta fase, el equipo se centró en realizar pruebas con estas herramientas, comprobar cómo se conectaban entre ellas, y buscar ejemplos de implementaciones que las utilizaban. Fue la fase más lenta ya que todas las tecnologías eran nuevas para todos los integrantes del grupo.

Tras este periodo de investigación, se avanzó a una segunda fase en la que el equipo empezó a desarrollar los prototipos con los conocimientos adquiridos durante los meses previos. En esta fase, el principal objetivo fue implementar un modelo completamente funcional. Gracias al trabajo en grupo, se pudieron desarrollar los distintos prototipos en los que los esfuerzos se enfocaron en la funcionalidad y no en elementos como la apariencia.

Por otro lado, el equipo continuó investigando nuevas tecnologías que podrían ser de utilidad de cara al objetivo final. Se estuvo trabajando con muchas tecnologías que finalmente no fueron implementadas. Los prototipos sirvieron para comprobar si las investigaciones estaban bien dirigidas o, por el contrario, no tendrían una utilidad real en el proyecto.

La última fase, fue la implementación del proyecto final, desde la especificación de su diseño hasta la conclusión del desarrollo. Aunque esta fase no fue tan lenta como la anterior, gracias a las herramientas y metodologías mencionadas anteriormente, el proceso fue mucho más rápido que las fases anteriores.

Esta fase, a diferencia de las otras, tenía como objetivo, además de ser plenamente funcional, la experiencia del usuario final, por lo que una parte muy importante fue la forma de hacerla más intuitiva y cercana para los usuarios. Esto se hizo gracias a las puestas de ideas en común por parte de los integrantes del equipo.

Finalmente, se realizó una fase de evaluación. En ella entrevistamos y dimos a probar nuestra aplicación a un grupo de revisores. Posteriormente, se analizaron los resultados y conclusiones obtenidas.

La mayoría de las fases se han hecho en común, aunque en el desarrollo de la aplicación final, al haber más funciones y partes del trabajo, se han repartido entre los integrantes del grupo para poder llevar a cabo el desarrollo de una manera más eficiente y fluida. En el capítulo 6, se encuentra la aportación de cada uno de los integrantes en las distintas fases del proyecto.

3.3 Enfoque de Software Libre

Cuando hablamos de software libre, nos estamos refiriendo a cuatro principios básicos o, en este caso, cuatro libertados principales:

- Poder ejecutar el programa indistintamente del propósito.
- Tener libertad para estudiar el código y adaptarlo a las necesidades de cada individuo.
- La libertad para redistribuir copias y ayudar así a la gente.
- Poder modificar el programa para publicarlo y así contribuir a la comunidad.

Cualquier software que incluya estos cuatro principios, se puede denominar Software Libre.

Poder disfrutar de todas estas libertades, también implica no tener que pedir permiso o pagar por ellas. Si se hicieran cambios, no debería ser necesario notificárselo a nadie ni notificarlo de alguna manera en concreto.[50]

No obstante, el software libre también tiene alguna desventaja. La diversidad de distribuciones, licencias de uso o incluso de herramientas que tienen el mismo fin, puede suponer alguna complicación en usuarios menos experimentados[51].

En contraposición al software libre nos encontramos con el software propietario. Podemos nombrar, de una forma general, una serie de desventajas con las que cuenta este tipo de software.

- El código fuente es secreto, por lo que no se puede ver que ocurre detrás de la pantalla.
- No suele estar permitido hacer adaptaciones del software para necesidades particulares[52]

Teniendo todos estos conceptos en mente, se decidió que lo mejor para el proyecto es que fuera parte del software libre, así cada interesado puede experimentar con el mismo

e involucrase en el proceso de investigación, aportando novedades a la comunidad. Por esta misma razón se ha implementado en el proyecto la licencia GNU General Public License (GPL), que permite realizar todas las acciones descritas anteriormente. Se puede encontrar el código de este proyecto en el enlace de Github:

https://github.com/RevLounge/Dapp

3.4 Tecnologías

Nuestro proyecto final se divide en un frontend desarrollado en React conectado a través de la librería de JavaScript web3, a los contratos con los que partimos de un inicio, aunque se vieron ligeramente alterados como veremos más adelante.

A este ecosistema se le añadió un servidor de Mongo conectado a una API para poder hacer consultas sin depender exclusivamente de los contratos de Solidity.

Estas tecnologías, trabajando en conjunto, permiten el correcto funcionamiento de esta aplicación.

- Github

Servicio en la nube que sirve para crear repositorios de proyectos y llevar un control de versiones. Permite trabajar en equipo facilitando el uso compartido de archivos. En este caso particular se trabajó con Github Desktop.[53]

El código de este proyecto se puede consultar en GitHub:

https://github.com/RevLounge/Dapp

- Ethereum

Como ya se ha explicado anteriormente en el capítulo 2, Ethereum es una plataforma creada en 2014 que permite el desarrollo de aplicaciones descentralizadas (dapps) mediante la tecnología blockchain. Nace como una plataforma para solventar las limitaciones de Bitcoin permitiendo a todos los usuarios poder programar su propio código, para ello, se utiliza la denominada Ethereum Virtual Machine (EVM). Esta máquina virtual es la que se encarga de ejecutar todo el código y gracias a eso, se puede utilizar el protocolo de consenso en la blockchain de Ethereum.[54]

Otra parte esencial de esta plataforma son las cuentas Ethereum. Tenemos que diferenciar entre dos tipos, las cuentas de los usuarios y las cuentas de los contratos. Estas cuentas son direcciones de 20 bytes entre las que se realizarán todos los cambios de

estado, es decir, entre las que se va a realizar el intercambio de información, y que pertenecen a un usuario concreto.

Finalmente, se debe tener en cuenta que Ethereum usa su propia criptomoneda, el Ether (ETH), usada en las transacciones entre cuentas.

- Solidity

Es el lenguaje de programación que nos permite crear contratos inteligentes para la red Ethereum. Es un lenguaje de alto nivel cuyo principal objetivo es poder implementar los contratos de una manera sencilla, por ello, su sintaxis es muy similar a la de otros lenguajes de alto nivel como JavaScript.

A modo de introducción, se pueden entender los contratos inteligentes como un conjunto de funciones e instrucciones que, tras ser compiladas, se almacenan en la red blockchain para poder interactuar con ellas. Hay que tener en cuenta que se interactúa con los contratos inteligentes por medio de transacciones que son mensajes que pueden incluir datos o ether.

Para crear los contratos de una manera segura y eficiente Solidity incluye el concepto de gas, que se entiende como la energía necesaria para poder llevar a cabo una transacción. Cada una, tiene dos indicadores de gas. Por un lado, tiene un coste de gas que es la cantidad de energía que conlleva su ejecución y, por otro lado, tiene un límite de gas, cuyo valor es establecido por el creador de la transacción y sirve para limitar la cantidad de trabajo necesario para llevar a cabo la transacción. En caso de que una transacción conlleve más gas que su límite establecido, la operación se revierte y no se realiza.

- Ganache

Ganache[55] es un programa perteneciente a Truffle Suite, es una cadena de bloques personal que facilita el desarrollo de tecnologías blockchain en local, sin necesidad de tener que estar conectado a una red de pruebas en línea.

Despliega por defecto en el ordenador del usuario, y en un solo click, una cadena de bloques con 10 cuentas de Ethereum con 100 ETH en cada cuenta.

Proporciona una clave pública y una clave privada para dirección de cuenta, haciendo accesible el desarrollo en tecnologías blockchain a cualquier desarrollador.

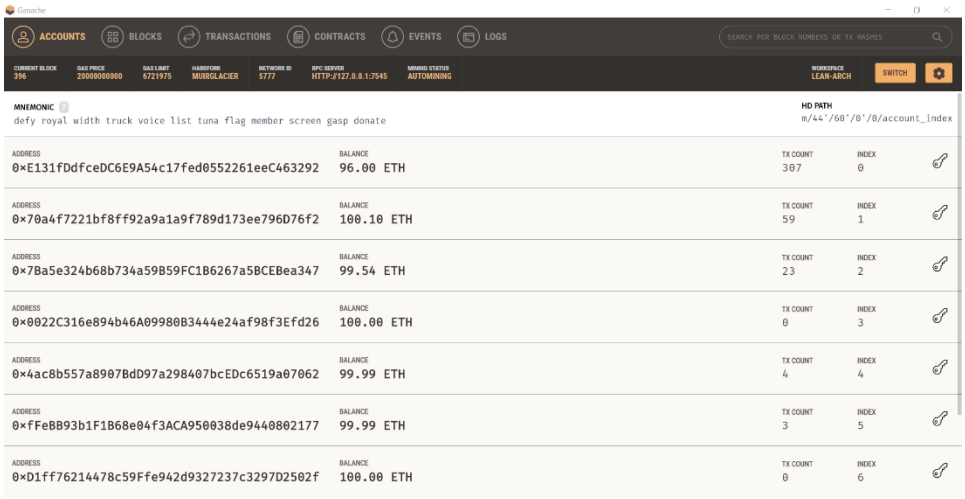


Figura 3.3: Pantalla principal de Ganache

- Metamask

Los navegadores actuales no integran la tecnología blockchain, es por esto que es necesario una extensión que lo convierta en un navegador compatible.

Aunque Metamask permite conectar cuentas personales al navegador para hacer transacciones y operaciones, en este caso nos resulta interesante la posibilidad de importar las cuentas de Ganache y conectarlo a la blockchain que crea en la red local.

Para importar una cuenta, simplemente necesitamos copiar su clave privada en la extensión y elegir a que red queremos conectarla.

De esta manera podemos probar las dApps utilizando las cuentas de prueba, verificando así el correcto funcionamiento de la aplicación.[56]

- ReactJS

ReactJS[57] es una biblioteca de JavaScript que permite crear interfaces de usuario. Permite diseñar vistas para el proyecto y, de forma automática, actualizar y renderizar de nuevo cuando cambie el estado de los datos.

Tiene un sistema basado en componentes encapsulados que manejan de forma individual sus propios estados, de esta forma se pueden convertir en interfaces complejas trabajando en conjunto.

En definitiva, es una tecnología puntera que, una vez aprendida, permite realizar interfaces de usuario completas de una manera sencilla.

En este caso, junto con ReactJS se ha trabajado con Semantic UI[58], un framework de diseño que implementa un conjunto de componentes para ReactJS, permitiendo el ágil desarrollo de aplicaciones web con un formato visual e intuitivo.

```
class HelloMessage extends React.Component {
    render() {
      return (
        <div>
          Hola {this.props.name}
        </div>
      );
    }
  }

  ReactDOM.render(
    <HelloMessage name="Taylor" />,
    document.getElementById('hello-example')
  );
```

Figura 3.4: Ejemplo de código en ReactJS

- Web3.js

Se trata de una API en JavaScript que permite simplificar el proceso de conexión con los contratos de Ethereum en la blockchain para interactuar con ellos.

Como ya se ha comentado, [59]la red de Ethereum se distribuye en nodos, y estos a su vez tienen una copia de la cadena de bloques. Para poder llamar a una función de un Smart Contract, debemos consultar a uno de esos nodos, proporcionándole:

- La dirección del Smart Contract
- La función a la que queremos hacer una llamada
- Las variables que vamos a pasar a dicha función

Los nodos de Ethereum reciben la información en el lenguaje JSON-RPC[60] que resulta especialmente complicado de tratar de forma manual. Es aquí donde web3.js supone una ayuda, ya que permite hacer consultas a los contratos con una interfaz en JavaScript.

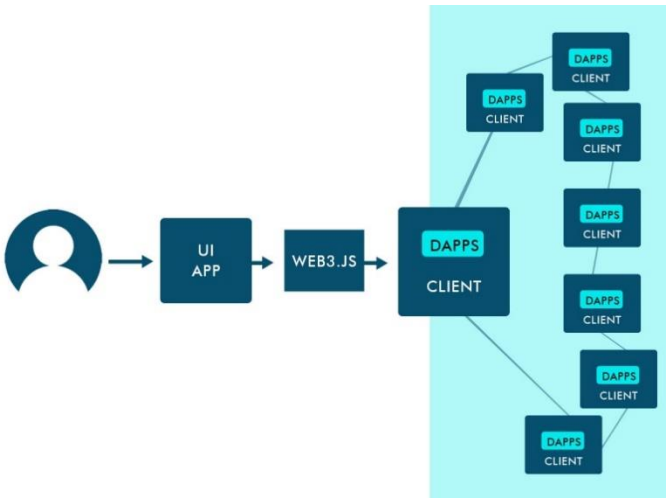


Figura 3.5: Interacción de Web3.js con contratos inteligentes

Fuente: blockchainexpert.uk

- MERN

Son las siglas de MongoDB, Express JS, React JS y Node.JS.[61]

Por un lado, MongoDB es una de las bases de datos NoSQL más utilizadas y extendidas.

Una base de datos de MongoDB[62] se puede utilizar para almacenar los distintos que contiene una aplicación, cada registro es un documento que almacena un par de datos clave-valor, muy parecidos a los objetos con los que trabaja JavaScript.

Por otro lado, Express JS[63] es una librería que facilita los pasos de creación de una API en JavaScript. Es una estructura de aplicaciones Node.js flexible que permite su integración en aplicaciones web y móviles.

Las otras dos tecnologías restantes han sido tratadas en los puntos inmediatamente anteriores.

Esta tecnología se utiliza para poder crear y ejecutar el servidor de la base de datos y su respectiva API.

- IPFS

IPFS[64] es un sistema de ficheros distribuido que funciona similar a una blockchain en el sentido de que muchos ordenadores guardan copias de los ficheros. IPFS utiliza nodos como una blockchain, con esto podemos decidir si crear nuestro propio nodo o conectarnos gratuitamente a uno de ellos.

En nuestro caso, utilizamos Infura[65], una Gateway pública y gratuita sin necesidad de registro para poder subir la representación gráfica de los NFTs a una red distribuida y almacenar su referencia de IPFS en el propio token. Normalmente, debido al coste de guardar la información en la blockchain, solamente se almacena lo más relevante como el autor o el dueño, y se adjunta un link con el contenido del NFT. El problema es que estos enlaces son frágiles[66] y pueden cambiar o no estar disponibles en algún momento. IPFS ayuda a arreglar estos problemas de direcciones puesto que cada pieza de contenido almacenada se le asigna un CID, un identificador único el cual sirve también para acceder a él[67]. De esta manera y gracias al modo en el que se almacena la información, se consigue la permanencia e inmutabilidad de los NFTs.

Se ha utilizado esta tecnología debido a que es una de las herramientas más extendidas para almacenar metadatos en los marketplaces de NFTs[68], mercados donde se realiza la compra y venta de los tokens.

Capítulo 4. Experimentación con las tecnologías

4.1 Primeros pasos

Antes de comenzar con el desarrollo de la plataforma de recompensas para revisores realizamos distintas implementaciones, a modo esquemático y con fines de aprendizaje, para adquirir conocimientos de las tecnologías que más tarde han sido utilizadas en el proyecto final.

Por un lado, empezamos experimentando con una red social implementada en blockchain, que permite crear posts que se registran en el contrato, así como enviar una propina en ether a los distintos posts que han subido los usuarios.

Por otro lado, hicimos una primera aproximación de lo que ahora es la aplicación final, que consiste en una DApp básica pero funcional que permite interactuar al completo con los contratos originales que más tarde usamos en el proyecto final.

4.2 Primera implementación: Social Network

La red social[69] consiste en un portal que permite a sus usuarios subir un post y que éste se quede almacenado en la blockchain. También ofrece la posibilidad de poder enviar una propina a través del contrato al creador de cualquier post que ya se encuentre registrado.

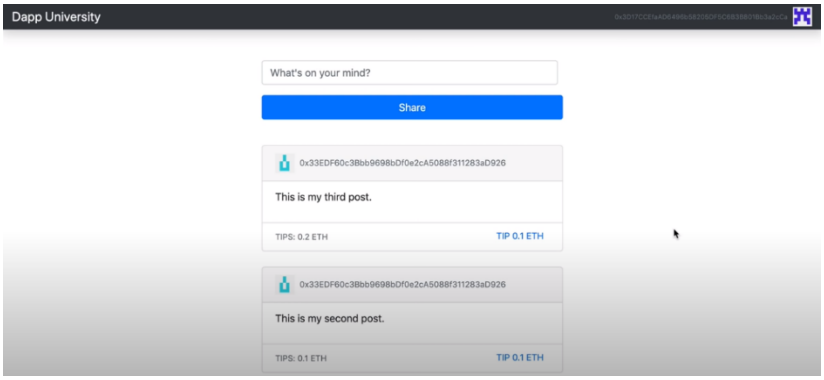


Figura 4.1: La Red Social en Blockchain

La implementación de esta DApp nos sirvió para aprender a programar contratos en Solidity, que nos sería de ayuda a la hora de modificar los contratos que hemos utilizado como base, tal como se ve en el punto “5.8 Modificaciones del Contrato Inteligente”, y como representar la información que se almacena en la blockchain utilizando React. Adquirimos los conocimientos necesarios para construir una aplicación con

características similares a las que deseábamos en nuestro proyecto final. Aprendimos a desplegar nuestra propia red de Ethereum en local utilizando la herramienta Ganache, perteneciente a la suite de Truffle, y a interactuar con las plataformas que implementan este tipo de redes mediante la extensión de navegador MetaMask. Otro punto importante fue la introducción a Web3.js, herramienta que utilizamos para comunicarnos con la red de Ethereum.

La pantalla principal de la aplicación cuenta con una barra superior, donde se nos informa de la cuenta de MetaMask con la que hemos accedido, desde el punto de vista de la aplicación, el usuario que está usándola en ese momento. A raíz de la dirección de la cuenta, se genera una imagen o identicon[1], utilizando la librería identicon.js[70] que permite identificar al usuario de una forma más visual.

Figura 4.2 Identicons

En el centro de la página nos encontramos con un apartado donde podemos escribir el texto que deseamos publicar a modo de post. Cuando el usuario interactúa con el botón "Share", Web3.js llama a la función del contrato de Solidity createPost, que permite grabar en la blockchain la información del post, así como la dirección del emisor.

La aplicación también cuenta con una lista donde se muestran todos los posts que están almacenados en la cadena de bloques junto con un botón de "Tip 0,1 ETH" que se encarga de mandar una propina al creador del post llamando a la función del contrato tipPost. Además, junto a cada post encontramos un contador que nos muestra la cantidad total de propinas que este ha recibido.

La realización de este primer prototipo nos proporcionó una mayor soltura a la hora de entender el funcionamiento y posibilidades de las tecnologías anteriormente mencionadas.

4.3 Segunda implementación: Rewards MVP

Rewards MVP (Minimum Valuable Product) es el proyecto que nos acercó a lo que finalmente se convirtió nuestra aplicación. Como punto de partida tomamos la estructura de la aplicación que habíamos realizado en la primera implementación, así como todas las herramientas que habíamos aprendido a utilizar. En esta implementación, al igual que en el proyecto final, utilizamos contratos de Solidity que estaban siendo desarrollados por

[1] Identicon: se llama así a la representación visual de una cadena de símbolos. Normalmente se utiliza para representar a usuarios dentro de sistemas informáticos.

otro grupo en paralelo[71]. Esto añadió la dificultad de adaptarse a los constantes cambios que los mismos recibían a lo largo del desarrollo de este ejercicio. Para conseguir un diseño consistente se decidió importar Semantic UI para utilizar con React.

La aplicación permitía a los autores crear y registrar sus papers en la blockchain para así poder recibir propinas de cualquier otro usuario. Para ello creamos una página muy simple en la que se listaban los títulos de los artículos, acompañados de datos generados aleatoriamente, que habían sido registrados en el sistema a los que agregamos un botón para realizar una donación de "0.1 ETH". Dicha donación se transfería íntegramente a la cuenta del creador del paper. Tanto las llamadas a las funciones de los contratos, como cargar la información de los bloques se ha realizado a través de Web3.js.

Al lanzar cualquier transacción del contrato se tienen que escribir los datos en el mismo, como ya hemos comentado anteriormente, se ha de pagar la comisión de los mineros, es decir, el coste de lanzar el contrato a la blockchain (gas). Es por esto por lo que cada interacción de escritura en la blockchain es necesario aceptarla en MetaMask y pagar su gas correspondiente.

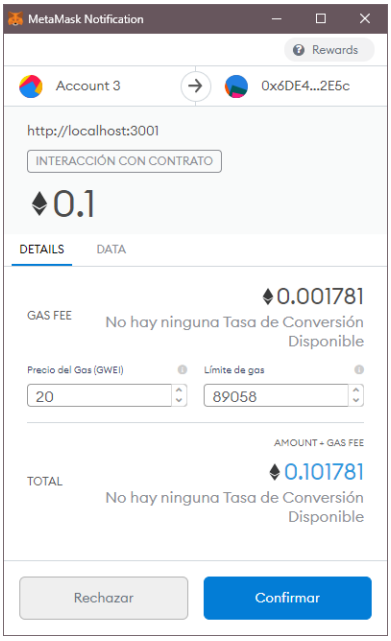


Figura 4.3: Transacción de "0.1" ether en Metamask

En ese momento, realizamos reuniones con los creadores de los contratos que estábamos integrando para compartir opiniones acerca de la estructura de estos. Los nuevos cambios que ellos implementaron incluyeron las revisiones de los artículos y que las recompensas se otorgaban directamente a los revisores del paper. También, nos explicaron los tipos de recompensas que más tarde implementaron: los tokens de reputación, basándose en el ERC-20 y los Awards, empleando el ERC-721. Las recompensas se encuentran detalladas en los puntos 5.5, 5.6 y 5.7.

Los cambios mencionados nos plantearon un nuevo problema con en el que no nos habíamos enfrentado hasta el momento, la creación de más de una página para poder almacenar las revisiones de cada paper. No llevó mucho tiempo encontrar una solución,

React Router. Después de realizar un curso y leer su documentación lo aplicamos en nuestro proyecto.

Ahora desde cada paper se puede acceder a la página de este y ver las revisiones que han sido registradas. Para añadir las revisiones reutilizamos el código desarrollado para la creación de papers, haciendo las modificaciones necesarias. También trasladamos el botón de dar propinas al listado de revisiones y añadimos los dos nuevos botones que más tarde conectamos con los tokens de reputación y los Awards.

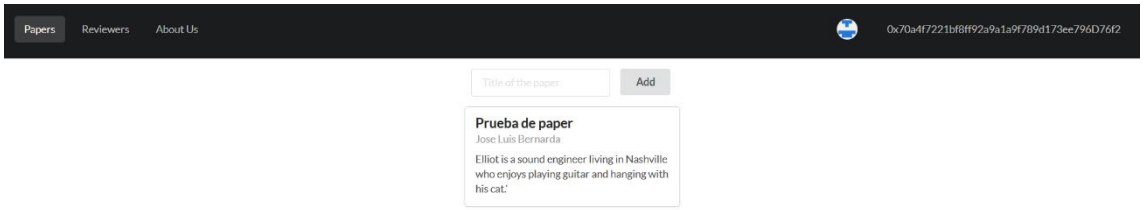


Figura 4.4: Vista de papers

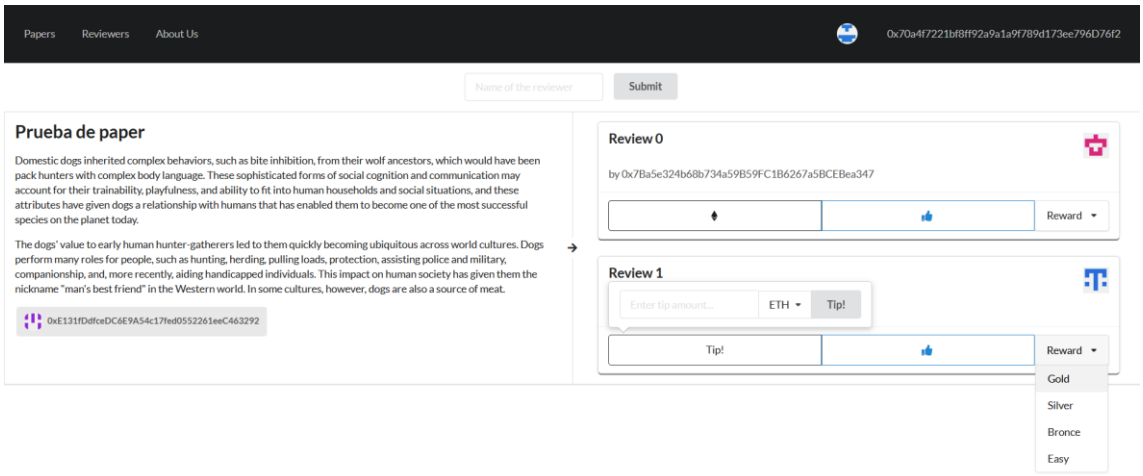


Figura 4.5: Vista de un paper y sus revisiones

Cada revisión cuenta con un componente que consta de tres botones y permiten dar hasta tres tipos de recompensas. Cada uno de ellos implementa una función distinta de los Smart Contracts. En orden de izquierda a derecha, el primero permite enviar una propina que, respecto a su implementación anterior, ahora da la opción de elegir la cantidad que se desea enviar. El segundo botón implementa la nueva extensión del contrato que permite enviar tokens de reputación a los revisores, tal como se explica en el punto “5.6 Reputación (RPT) para Revisores”. Y el último envía un Award al revisor en forma de token no fungible, pudiendo elegir entre cuatro tipos distintos: “Gold”, “Silver”, “Bronce” e “Easy”. Decidimos categorizar de esta manera los Awards, ya que consideramos que era la manera más fácil de diferenciarlos. A su vez, cada uno de ellos

incrementaba el coste de enviarlo en función de su tipo. Más tarde, descartamos la categoría “Easy” porque consideramos que no seguía la escala de los tres primeros y podría dar lugar a confusión.

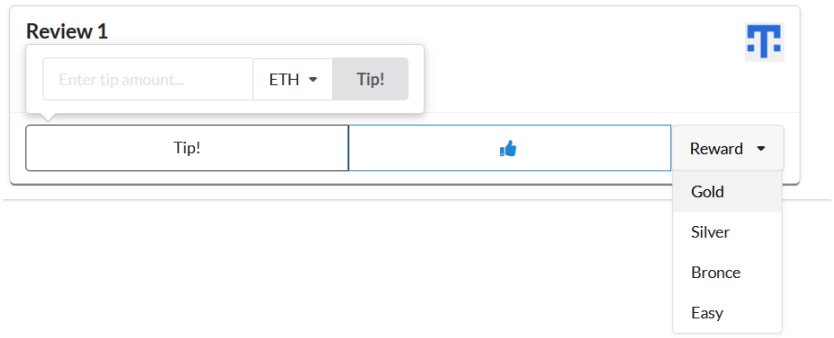


Figura 4.6: Botones de recompensas

Tras conectar todos los botones con las funciones de los contratos y comprobar su correcto funcionamiento, decidimos realizar una nueva página para la visualización de las estadísticas de las recompensas. La nueva vista se encarga de recoger y ordenar los datos de los revisores seguidos de la reputación y los premios recibidos de una manera esquematizada.

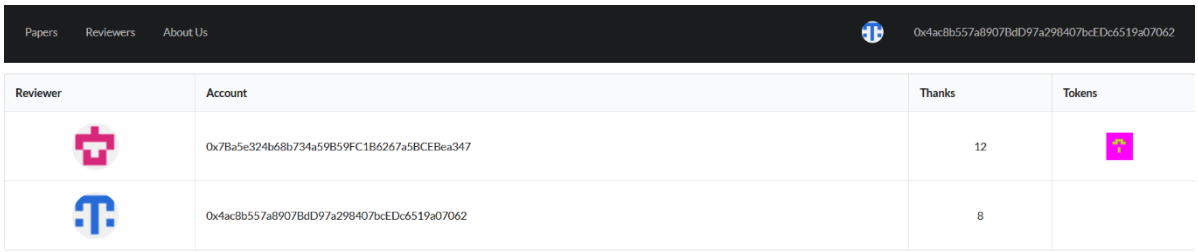


Figura 4.7: Vista tabla de revisores

Como se ha mencionado antes, una de las nuevas características que trajo el nuevo contrato fue la implementación de un nuevo tipo de recompensas, los Awards. En este momento dedicamos parte de nuestros esfuerzos a investigar en mayor profundidad el ERC-721 donde dimos con un curso en YouTube en el que se mostraba como crear CriptoColeccionables utilizando este estándar[2]. En él, se mostraba como utilizar los códigos de colores hexadecimales para generar NFTs, por lo que decidimos aplicar algo similar en nuestro trabajo. Reutilizando la librería de Identicon que habíamos implementado para visualizar los avatares de los revisores, generamos la representación visual de los Awards.

[2] https://www.youtube.com/watch?v=YPbgjPPC1d0

Figura 4.8: Tipos de Awards

Utilizando este segundo desarrollo como entorno de pruebas, hemos comprendido en mayor profundidad como construir una aplicación en React, desde el desarrollo de componentes y el manejo de los estados, a decidir cuales de ellos se están renderizando en función de la URL por medio de React Router. También hemos aprendido a interactuar con los contratos de la blockchain utilizando Web3.js. Por otro lado, nos ha servido para seleccionar las herramientas que íbamos a utilizar. Además, algunas partes del código han sido posteriormente reutilizadas en el desarrollo del proyecto final gracias a la reusabilidad de compontes que ofrece React.

4.4 Pruebas con Solidity

Para poder entender mejor la interacción que debía hacer Web3 con los contratos, decidimos que la mejor opción era aprender cómo funcionaba Solidity, aprendiendo este lenguaje y haciendo nosotros mismos algunos contratos de prueba.

Al ser una tecnología que ninguno conocíamos de antemano, ponernos a trabajar directamente con Solidity nos facilitó entender cómo hacer de una manera correcta todas las conexiones que requiere la aplicación para trabajar en armonía con la blockchain.

Para poder adquirir conocimientos de este lenguaje y la manera en la que trabaja, los miembros del grupo decidimos matricularnos en la asignatura de Introducción a la Tecnología Blockchain y Smart Contracts (TBC) que, por primera vez, se imparte en la Facultad de Informática.

Esto nos permitió que la toma de contacto fuera más completa e inmersiva, ya que teníamos entregas semanales de prácticas que nos ayudaron a comprender nuevos conceptos de este lenguaje de programación.

4.5 Primeras implementaciones con Solidity

Tanto por nuestra cuenta como gracias a la asignatura de TBC, nos vimos practicando con nuestros propios contratos de Solidity, haciendo pruebas de distintos estándares (como el ERC-721 y ERC-20) y escribiendo código para distintas aplicaciones.

Hicimos prácticas de temas muy diversos, como un Smart Contract para aprender cómo funciona el uso de arrays y mappings en Solidity que consistía en una hucha multiusuario permitiendo almacenar ahorros en un array que más tarde puede ser recorrido para ver el total de Ethereum almacenado, así como ver la aportación individual de cada usuario.

Hasta contratos más completos y complejos, como un sistema de creación de personajes para un videojuego, que genera un Token No Fungible (NFT) basado en una versión simplificada del estándar ERC-721.

Este proyecto consiste en un contrato que almacena dos estructuras, una de Armas (Weapons) y otra de Personajes (Characters). Cada personaje puede tener varias armas ya que el Smart Contract tiene distintas funciones que permiten interactuar con el mismo. Con ellas, podemos desde crear un personaje o un arma nueva, hasta incrementar la potencia de fuego de todas las armas de un personaje concreto. También se hace uso de las funciones típicas del estándar ERC-721 como pueden ser comprobar el balance de un jugador, habilitar la transferencia de propiedad de un token o "aprobar" direcciones de otros jugadores para que puedan interactuar con los tokens propios de un jugador en su nombre.

Vernos involucrados en proyectos como estos nos facilitó entender como funcionaban los contratos que íbamos a utilizar en la versión final de la aplicación, también nos habilitó la posibilidad de realizar algunas modificaciones en los mismos, como se explica en el punto 5.8.

4.6 Conclusiones de la Experimentación

Esta etapa de investigación y pruebas iniciales nos ha permitido darnos cuenta de con qué requisitos debíamos contar en la versión final de nuestro proyecto. Nos ha servido como proceso de aprendizaje e investigación, dotándonos de la capacidad de finalmente poder desarrollar RevLounge, la versión final de nuestro proyecto. Gracias a la realización de estos pequeños prototipos, aprendimos como desenvolvernos de manera fructífera con estos nuevos entornos y lenguajes, tomando ventaja a la hora de desarrollar el producto final ya que nos permitió conocer:

- Cómo interactúan React y JavaScript para poder desarrollar el proyecto.
- La manera en la que trabajan los contratos de Solidity para poder conectarlos con Web3 y usarlos en el frontend.
- Cómo utilizar el entorno de trabajo, ya que tener que trabajar con Ganache y Metamask era algo completamente nuevo para nosotros.

Capítulo 5. RevLounge: plataforma de recompensas para revisores

5.1 Especificación

RevLounge es una plataforma orientada a dar recompensas a revisores. Para ello, nos encontramos con una plataforma principal que permite registrar un perfil de revisor en el que se pueden importar las revisiones que tiene ese usuario para, más adelante, poder ser recompensadas con propinas, tokens de reputación y NFTs. La aplicación muestra una tabla con los usuarios registrados en el sistema y sus recompensas. También permite visualizar perfiles de usuarios concretos que muestran toda la información de un revisor y que los revisores importen más revisiones en cualquier momento. Implementa un apartado con las preguntas más frecuentes para solucionar las dudas que se puedan encontrar en el proceso de uso de la aplicación.

Por otro lado, como se ve con más detalle en el punto “6.1.3 Paquete de npm”, se ha creado un módulo que permite ser ejecutado desde páginas externas para implementar el sistema de recompensas. Para ello recibe un identificador de revisión y, en el caso de que esté registrada en la plataforma, habilita la opción de enviar recompensas a la revisión, mostrando las que ya se han enviado previamente por otros usuarios.

Cualquier revisor que cuente con una cartera de Ethereum puede registrarse e importar sus revisiones. Una vez han sido añadidas, cualquier usuario de internet que también tenga una cuenta con fondos, podrá utilizar el módulo de recompensas con el que podrá realizar una de las tres acciones anteriormente nombradas. Si la elección del usuario es dar una propina, el importe llegará íntegramente a la cuenta del revisor. En caso de elegir enviar un token de reputación, la reputación del creador de la revisión se verá incrementada. Como última opción, el usuario podrá dar una recompensa única a la revisión, pudiendo elegir entre tres tipos distintos (oro, plata y bronce). En función del premio elegido el revisor recibirá un token no fungible, representado por un Identicon con un patrón único del tipo seleccionado.

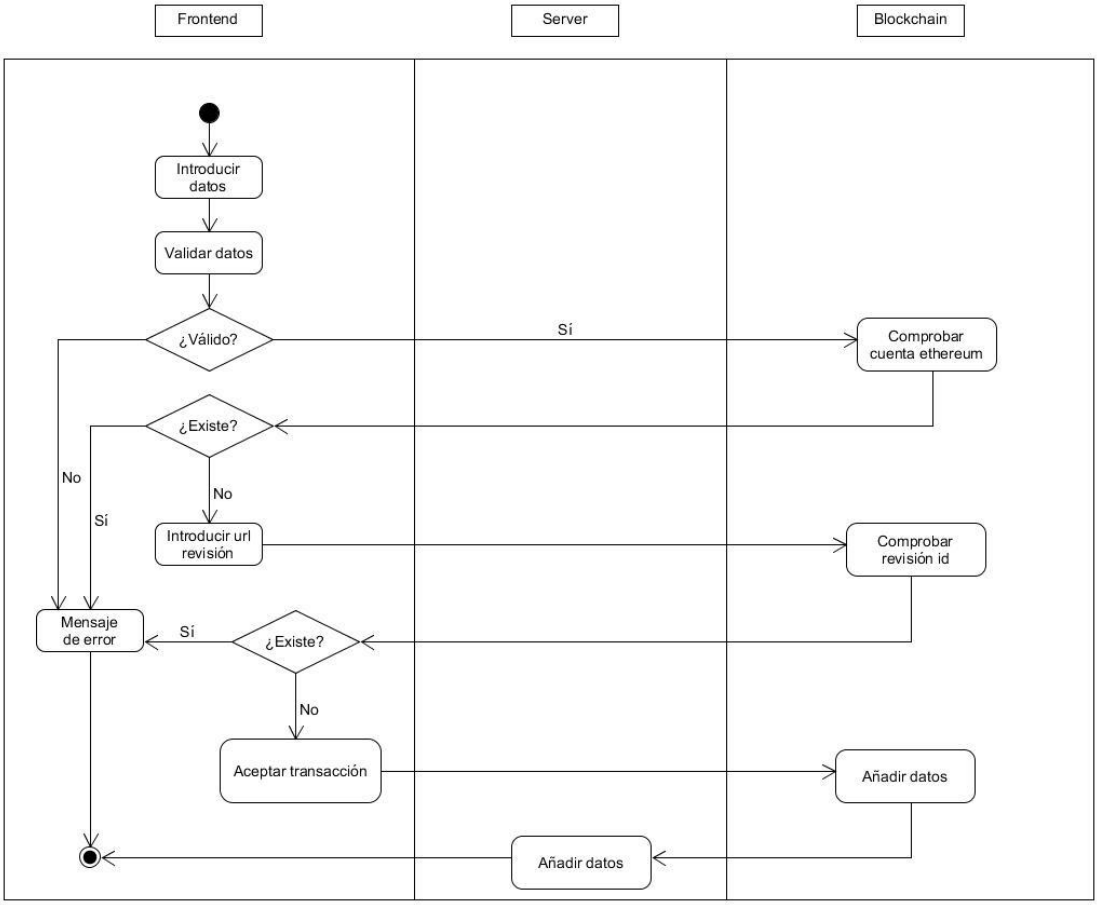
Todas las recompensas recibidas, así como la reputación se mostrarán en ambos componentes. La página principal mostrará una lista de revisores, así como un resumen de las recompensas que han recibido.

5.2 Requisitos Funcionales

Los requisitos funcionales se identifican con el código “RF-P” si pertenecen al componente principal o con el código “RF-M” si pertenecen al módulo externo.

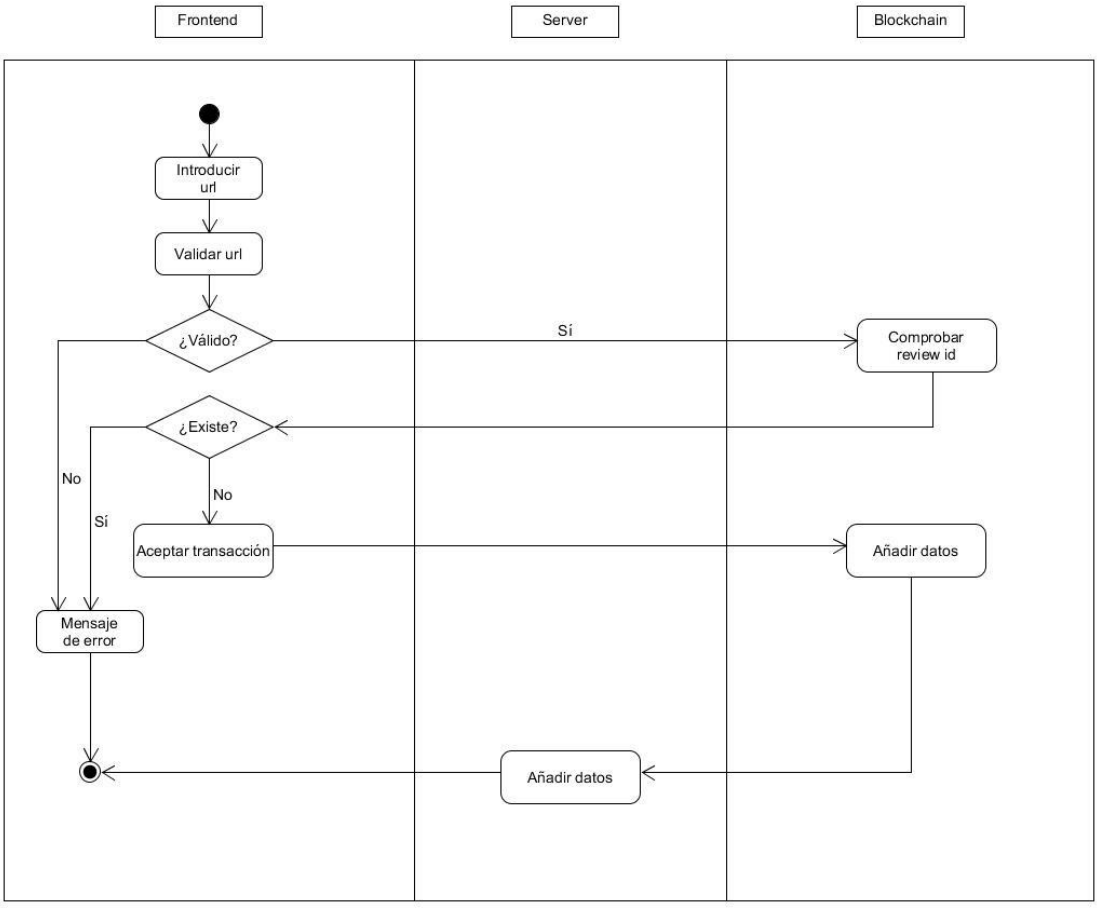
- RF-P-1 Registrar revisor:

Descripción	Añadir el revisor y una revisión al sistema. Los datos se guardarán en la cadena de bloques y en el servidor.
Entrada	Nombre, apellido, cuenta de Ethereum, email, dirección, ORCID, id de revisión
Salida	-
Necesita	Cartera de Ethereum, MetaMask, Ether.
Precondición	Los datos son válidos y cumplen el formato requerido. No existe un revisor con la misma cuenta de Ethereum. La revisión no ha sido registrada previamente.
Postcondición	Se registra el usuario y la revisión en la cadena de bloques, así como en la base de datos.
Postcondición Error	Si el usuario o su revisión ya han sido registrados previamente aparecerá un mensaje de error.



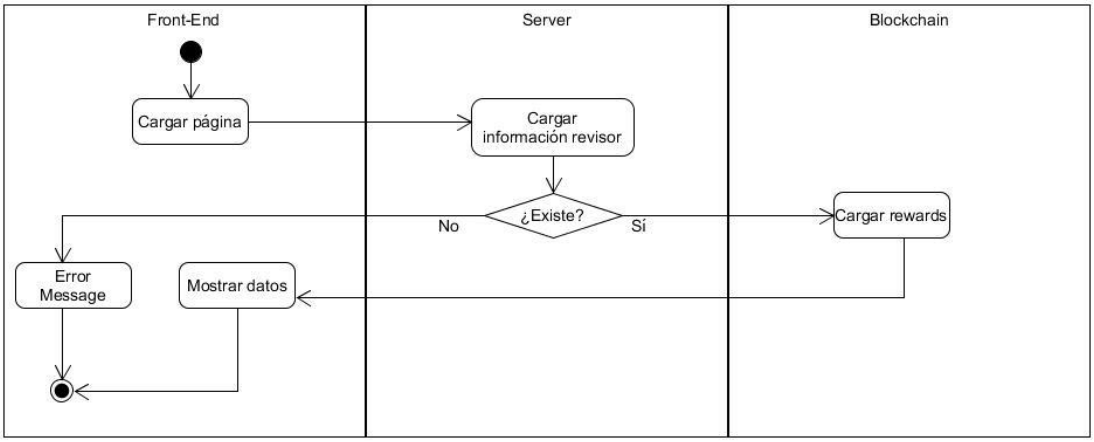
- RF-P-2 Importar revisión

Descripción	Añadir una revisión a un revisor ya registrado. Para ello se utilizará el link (URL) de la revisión como identificador único.
Entrada	Link de la revisión
Salida	-
Necesita	Cartera de Ethereum, MetaMask, Ether.
Precondición	El usuario que importa los datos ha sido previamente registrado en el sistema. La URL es válida y cumple el formato requerido. La revisión no ha sido registrada previamente.
Postcondición	Se registra el usuario y la revisión en la cadena de bloques, así como en la base de datos.
Postcondición Error	Si la revisión ya ha sido registrada previamente o la URL no cumple el formato requerido aparecerá un mensaje de error.



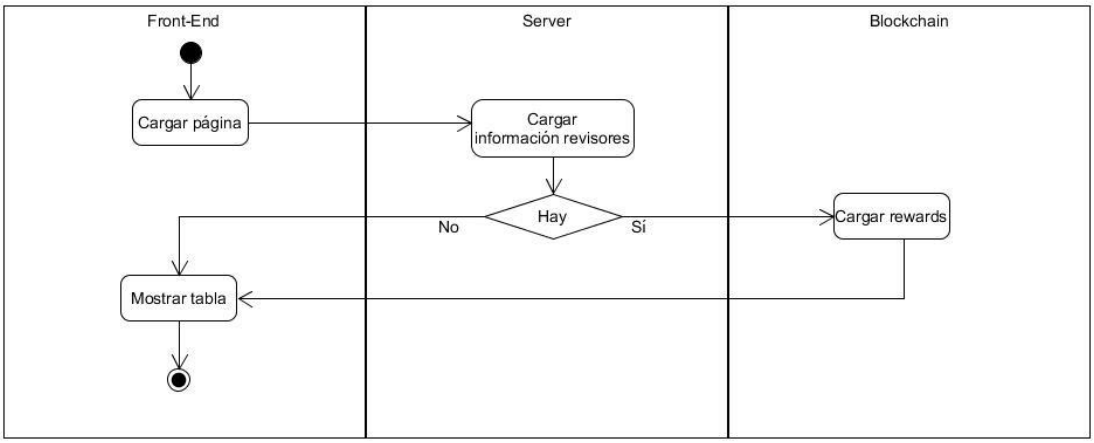
- RF-P-3 Mostrar perfil revisor

Descripción	Acceder al perfil de un revisor donde se muestra su reputación junto a las revisiones importadas con los respectivos premios que ha recibido cada una de ellas.
Entrada	-
Salida	-
Necesita	Cuenta de Ethereum, MetaMask
Precondición	Ser un perfil registrado
Postcondición	Se mostrarán los datos que el revisor uso en el registro y el resumen de las recompensas recibidas. Seguido aparecerá la lista de revisiones y finalmente se mostrarán los NFT recibidos separados por categorías.
Postcondición Error	Una página de error aparecerá avisando de que el perfil no existe.



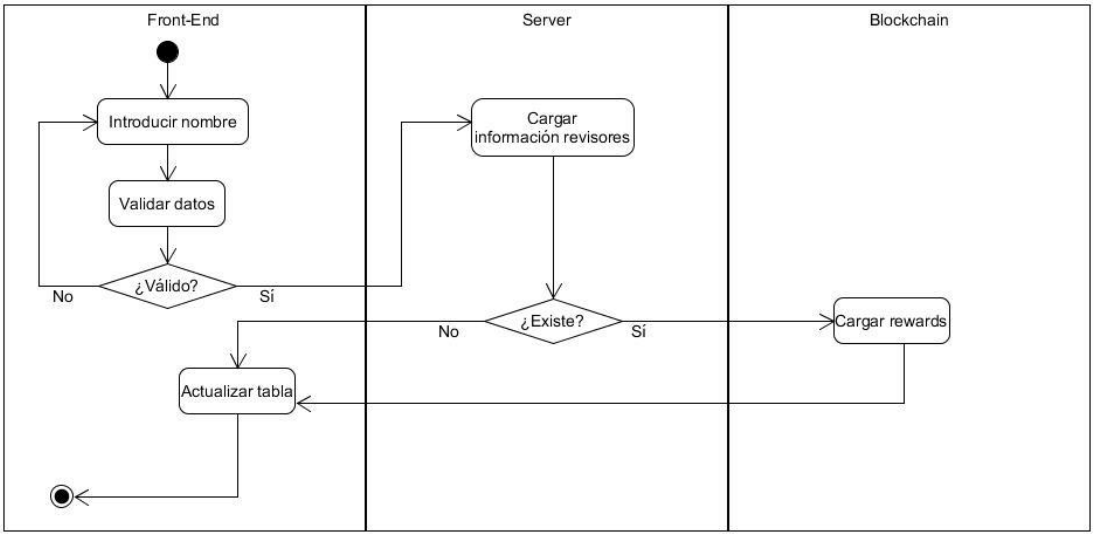
- RF-P-4 Mostrar resumen de los revisores

Descripción	Cuando un usuario accede a la web, encuentra una lista de revisores con un resumen que muestra su reputación y premios totales recibidos.
Entrada	-
Salida	-
Necesita	Cuenta de Ethereum, MetaMask
Precondición	-
Postcondición	Se muestra una tabla, en cada columna aparecen nombre y apellidos, acompañados de la reputación total y los awards totales de cada uno de los usuarios registrados en la plataforma.
Postcondición Error	Si la consulta al contrato o la base de datos han fallado un mensaje de error aparece en lugar de la tabla.



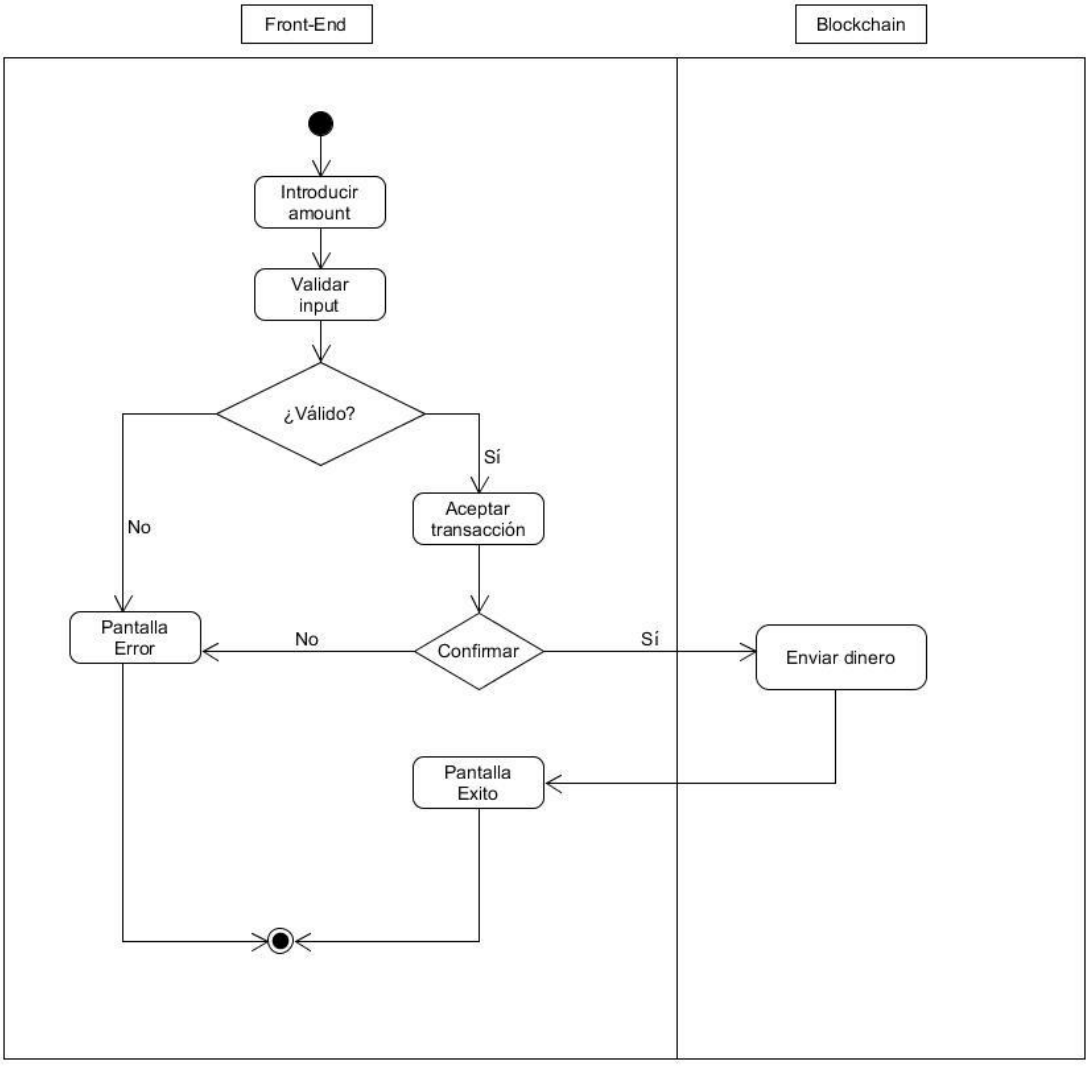
- RF-P-5 Buscar un revisor

Descripción	Buscar en la lista de revisores por nombre o apellido.
Entrada	Nombre, apellido o ambos.
Salida	Se actualiza la tabla con los resultados que coincidan con la búsqueda.
Necesita	Cuenta de Ethereum, MetaMask
Precondición	-
Postcondición	Si el nombre y apellidos coinciden con el de uno o varios revisores registrados en el sistema, la lista se actualizará mostrando únicamente las coincidencias.
Postcondición Error	En caso de que no haya coincidencias la lista aparecerá vacía.



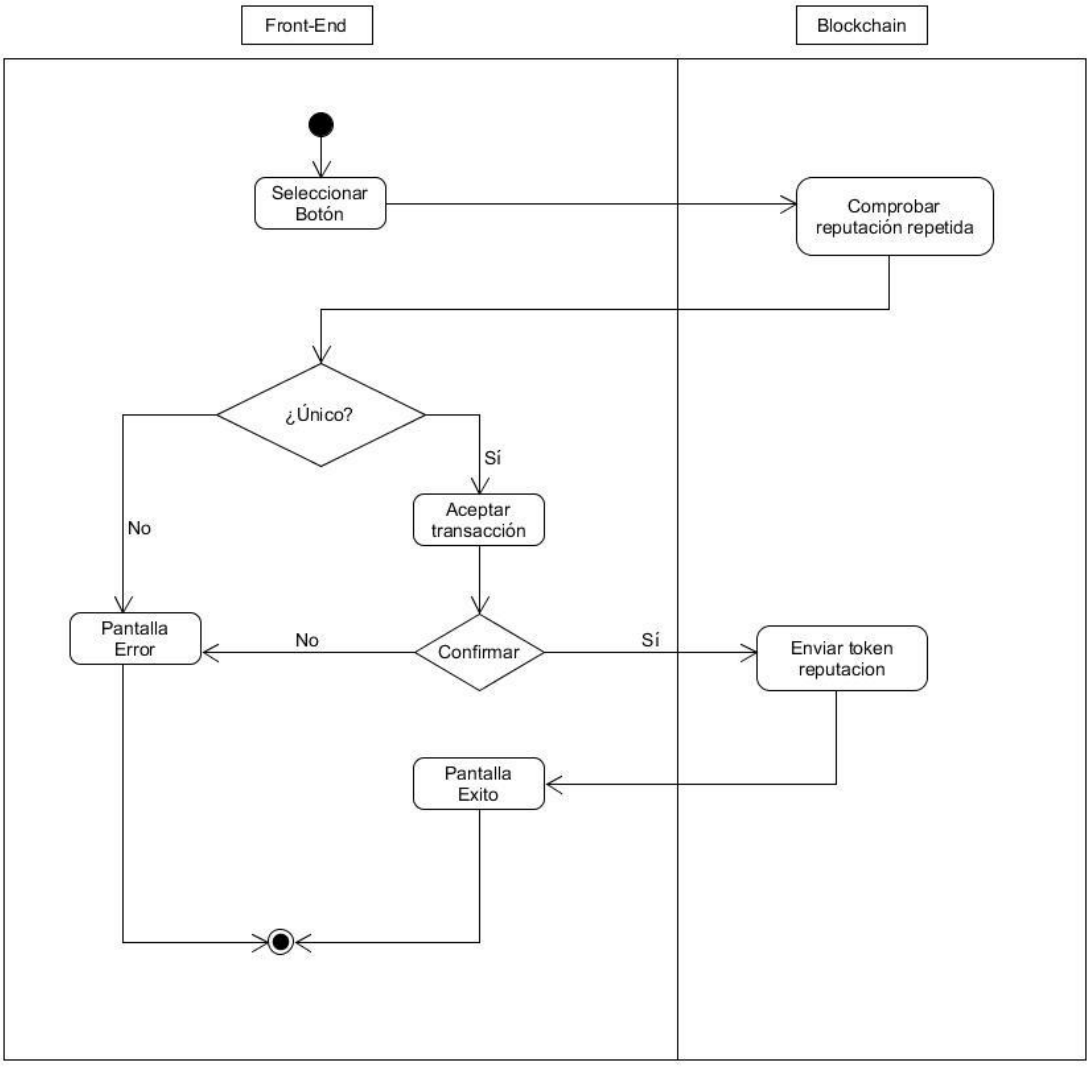
- RF-M-1 Dar una propina

Descripción	Enviar dinero de una cartera de Ethereum a la cuenta del creador de una revisión.
Entrada	Cantidad de Ether
Salida	-
Necesita	Cartera de Ethereum, MetaMask, Ether.
Precondición	Se ha incluido una cantidad de Ether para ser enviada. El dueño de la revisión, así como la misma están dados de alta en la plataforma. No ser el dueño de la revisión.
Postcondición	La cantidad de Ether indicada se envía de la cuenta del donante a la del revisor.
Postcondición Error	La transacción ha fallado, un mensaje de error aparece en la pantalla del usuario.



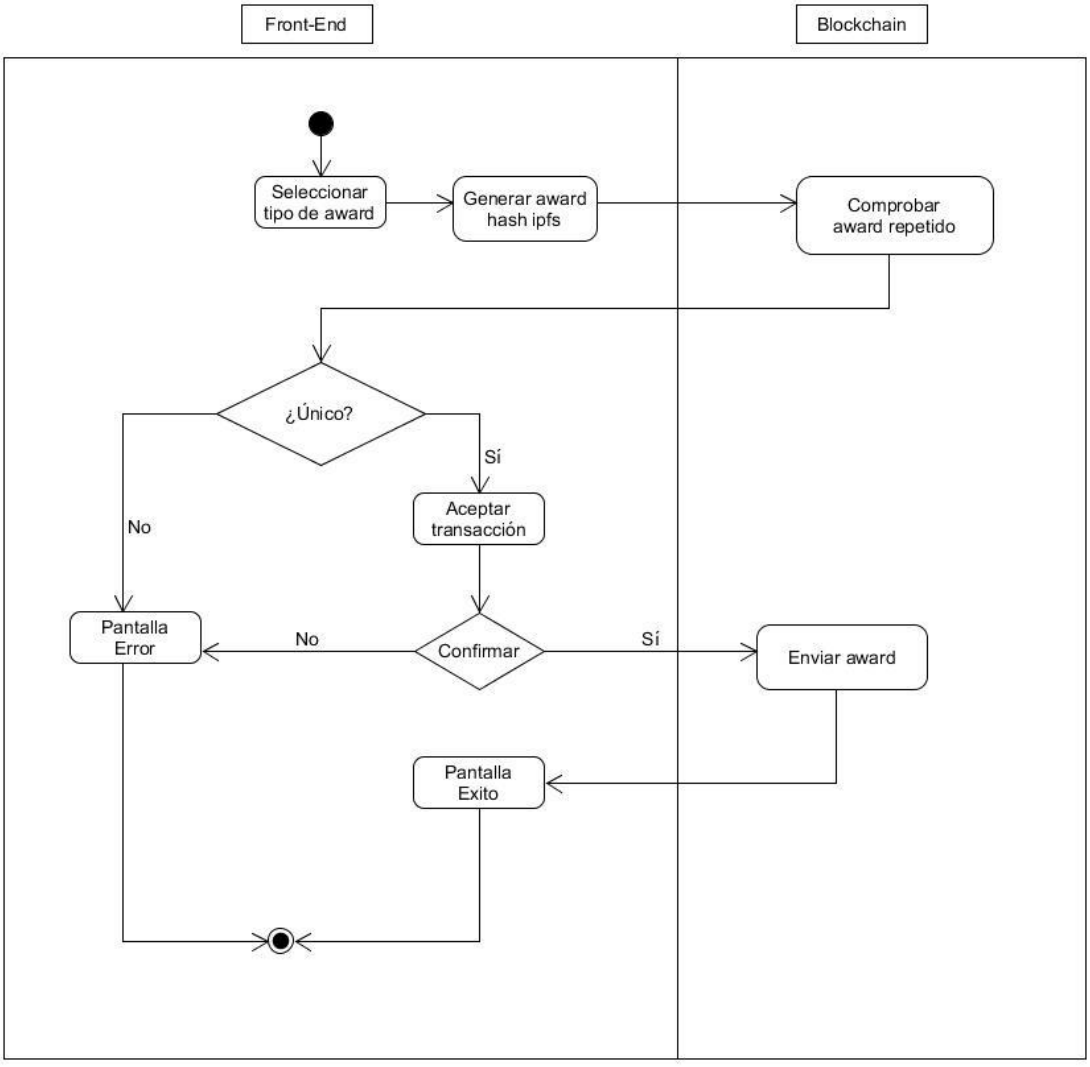
- RF-M-2 Dar Reputación

Descripción	Enviar un token de reputación a la cuenta del creador de una revisión.
Entrada	-
Salida	Un token de reputación intransferible
Necesita	Cartera de Ethereum, MetaMask, Ether.
Precondición	El dueño de la revisión, así como la misma están dados de alta en la plataforma. No ser el dueño de la revisión. No haber dado reputación a esa revisión anteriormente.
Postcondición	El token de reputación se envía a la cartera del revisor y su contador de reputación aumenta en 1.
Postcondición Error	La transacción ha fallado, un mensaje de error aparece en la pantalla del usuario.



- RF-M-3 Dar un Reward

Descripción	Enviar un NFT único a la cuenta del creador de una revisión.
Entrada	Tipo del NFT (Gold, Silver, Bronze)
Salida	Un NFT único intransferible
Necesita	Cartera de Ethereum, MetaMask, Ether.
Precondición	El dueño de la revisión, así como la misma están dados de alta en la plataforma. No ser el dueño de la revisión. No haber dado una recompensa del mismo tipo a esa revisión anteriormente.
Postcondición	El NFT se envía a la cartera del revisor y sus contadores de awards aumentan en 1, dependiendo del tipo de award que haya sido recibido. El expositor de awards del perfil del revisor recompensado mostrara el NFT
Postcondición Error	La transacción ha fallado, un mensaje de error aparece en la pantalla del usuario.



5.3 Requisitos no Funcionales

- Coste:

Uno de los principales problemas de almacenar datos en la blockchain es el coste implícito que hay en las transacciones. Esta tarifa depende en parte de la manera en la que se ha escrito el contrato inteligente pero también en la cantidad y tamaño de información que se quiere almacenar. Pare reducir el coste hemos decidido guardar parte de la información en la base de datos.

- Usabilidad:

Es muy importante que el usuario pueda aprender a utilizar la interfaz en poco tiempo. Para ello hemos añadido las funcionalidades de una manera visual y clara, acompañando los componentes de descripciones explicativas que ayuden al usuario a entender la aplicación. Como para utilizar la plataforma es necesario contar con una cartera de Ethereum y una extensión de navegador, hemos creado guías explicando como instalarlas. Además, hemos añadido un apartado de FAQ en el que explicamos todas las funcionalidades que ofrecemos.

5.4 Tipos de Usuario

En una aplicación de recompensas para revisores distinguimos fácilmente entre dos tipos principales de usuarios. Los revisores y los usuarios que las otorgan. Los primeros tendrán que registrar sus revisiones en el sistema para poder recibir recompensas por las mismas, mientras que los usuarios que deseen premiarlas no deberán estar previamente registrados en el sistema. Además, hemos considerado un tercer tipo de usuario, que es el que puede acceder a la página con el objetivo de ver la lista de ver la lista de revisores para contactar con uno de ellos.

- Revisor:

Son usuarios que han realizados revisiones de artículos en cualquier otra plataforma. Podrán registrarse e incluir los enlaces de sus revisiones, desde el componente principal, para que puedan ser recompensadas. Un revisor puede premiar las revisiones de otros, pero nunca las suyas propias.

- Donante:

Son usuarios que no tienen por qué ser revisores, pueden ser cualquier usuario de internet, esto quiere decir que pueden estar registrados en la plataforma, pero no es un requisito indispensable. Entran a la aplicación por medio de una llamada al módulo externo desde una plataforma que integra revisiones con el objetivo de realizar una donación en Ether, dar un agradecimiento al revisor u otorgar una recompensa a las revisiones que hayan sido importadas al sistema.

- Visitante:

Son usuarios que pueden estar registrados, aunque no obligatoriamente, y su actividad en la aplicación es meramente informativa. Acceden con el cometido de ver la tabla de revisores, un perfil de un revisor en concreto o contactar con un revisor.

5.5 Dinero para Revisores

La primera opción de recompensa que se le presenta al usuario es la de realizar una aportación económica al creador de la revisión. Esta función está desarrollada en los contratos que utiliza la aplicación sin utilizar ningún tipo de token o premio, sino realizando una transferencia de Ether directamente de una cuenta a otra.

Al seleccionar esta opción, se le presenta una entrada de texto en el que puede elegir el importe que desea donar al autor de la revisión. De esta forma, y tras confirmar la transacción con MetaMask, el dinero se envía directamente al revisor que ha escrito esa revisión en concreto.

Puesto que las donaciones son algo personal entre individuos, decidimos que el importe de estas, así como el número de transacciones recibidas, no se muestre en ningún punto de la plataforma. Es una forma directa y sencilla de mostrar agradecimiento o apoyo a un revisor por su labor.

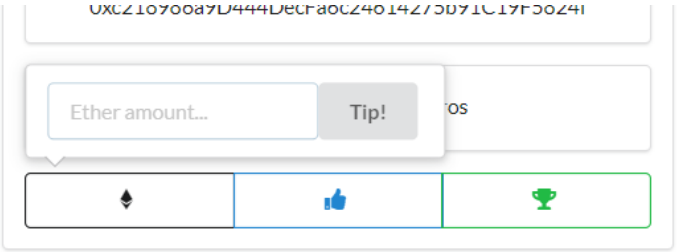


Figura 5.1: Entrada del importe a transferir

5.6 Reputación (RPT) para Revisores

La segunda opción con la que cuenta nuestra aplicación es la de agradecer al creador de la revisión. En este caso se utiliza el token de reputación que el contrato implementa basándose en el ERC-20. De esta forma un usuario puede mostrar su agradecimiento a un revisor sin necesidad de utilizar una cantidad concreta de Ether.

Cuando un usuario selecciona esta opción tiene que aceptar la transacción en Metamask y pagar los gastos de ejecución (gas) correspondientes, en este caso no se puede seleccionar la cantidad de Ether a gastar, sino que es un número previamente establecido y generado automáticamente por los contratos en función del precio que hay que pagar a los mineros.

Este token de reputación RPT sirve para agradecer y valorar a un revisor en concreto por su trabajo, es por esto por lo que en la pantalla principal encontramos una columna

de cada revisor dedicada exclusivamente a mostrar el número total de tokens RPT que poseen acompañado de un icono de un “thumbs-up”. Asimismo, dentro del perfil de cada uno, también se muestra el nivel de reputación que ostentan, debajo de la foto de perfil.

Es una forma clara de mostrar agradecimiento por el trabajo de un revisor en concreto y ayudar a destacarlo en la plataforma.

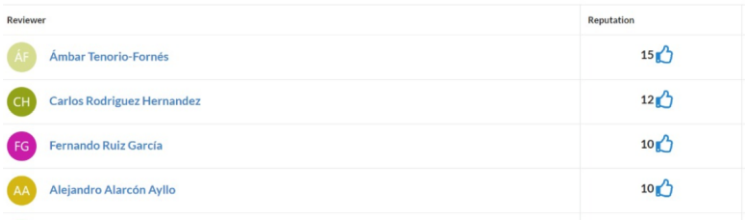


Figura 5.2: Representación de las distintas recompensas en la tabla de revisores

5.7 Awards (NFTs) para Revisores

La última forma de agradecer a los revisores es mediante el envío de una imagen única que representa los Awards. Como se menciona en la segunda implementación del capítulo 4, las imágenes se generan utilizando la librería de identicon.js y los premios pueden ser de una de las siguientes tres categorías: “Gold”, “Silver” y “Bronze”.

Los Identicon consisten en una imagen única creada a partir de un hash, cadena hexadecimal de quince o más caracteres y hasta un límite de treinta y dos. Nunca habrá dos imágenes iguales, de esta forma se puede conseguir una huella digital única a partir de datos, de forma que no se generará otra imagen igual a no ser que los datos de entrada sean exactamente los mismos. Para generar este hash utilizamos los datos relacionados con la revisión. Las direcciones de las cuentas de del revisor y el donante, la categoría de premio escogida y por último el identificador de la revisión. De esta manera nunca existirá un diseño repetido cuando se cree puesto que, por restricción del contrato, un mismo usuario no puede dar dos premios iguales a un mismo revisor y por lo tanto el hash nunca se repetirá.

Debido a que la longitud de los datos supera el límite de treinta y dos símbolos, se ha decidido convertirlos a un hash MD5[3] y crear la representación gráfica del Award con ello. Finalmente, para darle la distinción única de cada categoría de premio, se ha ajustado el fondo de cada imagen a los colores oro, plata y bronce. Esto se hizo gracias a la opción de la librería identicon.js que permite elegir los colores del fondo y del patrón.

[3] MD5 (Message-Digest Algorithm 5): algoritmo de reducción criptográfico de 128 bits. Se representa mediante un número de 32 caracteres hexadecimales.

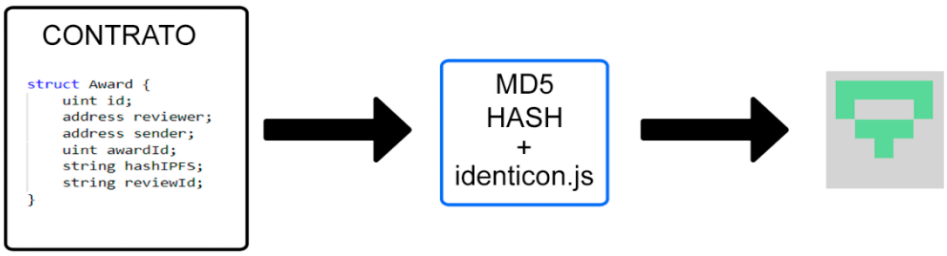


Figura 5.3: Creación de un award

Las tres categorías simbolizan un nivel diferente de apreciación al trabajo del revisor, se diferencian entre ellos en la forma en la que se representan, variando el fondo adoptando el color del material que les da nombre.

Cuando un usuario selecciona la categoría y se ha generado el Identicon con los datos, se realiza una conversión a base64 de la imagen, de esta forma el contenido no se almacena como archivo, sino que se representa con una cadena de caracteres. Finalmente se sube a IPFS haciendo uso del nodo de Infura, tecnología de la que hemos hablado en el capítulo de "Tecnologías".

```
function uploadIPFS(award, type, giveAward) {
    var buf = Buffer.from(award, 'base64');
    var awardHash = "https://ipfs.infura.io/ipfs/"
    ipfs.add(buf, (error, result) => {
        awardHash = awardHash.concat(result[0].hash)
        giveAward(type, awardHash);
        if (error) {
            console.error(error)
            return
        }
    })
}
```

Figura 5.4: Función uploadIPFS

El resultado de este proceso genera otro hash que representa la dirección de IPFS donde se almacena nuestra imagen, esta información junto a los datos de la revisión se utiliza en la llamada al contrato para que pueda emitir un evento a la cadena de bloques donde quede registrado el token no fungible.

Cabe mencionar que una vez el NFT es emitido, tal y como su nombre indica, no se puede modificar ninguno de sus parámetros.

Estos tokens adoptan un papel muy importante en nuestra aplicación, ya que es la representación más visual entre los distintos tipos de recompensas. Podemos encontrarlos separados por categorías dentro de los perfiles de los revisores, así como en el módulo de recompensas. También aparece recogida la cantidad de Awards que cada revisor ha recibido en la página principal.

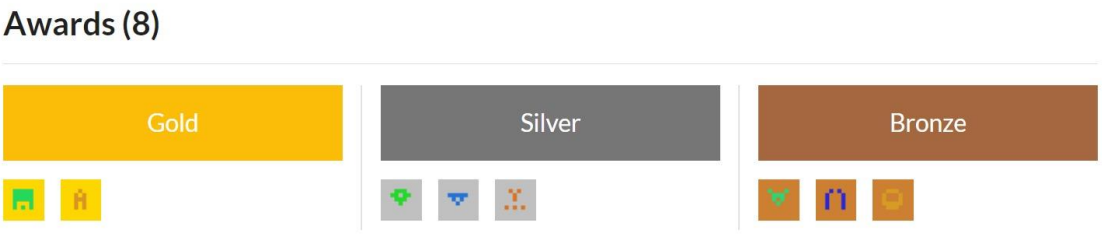


Figura 5.5: Awards en el perfil del revisor

Figura 5.6: Reputación y resumen de awards en el perfil del revisor

5.8 Modificaciones del Contrato Inteligente

Puesto que nuestra aplicación ha tomado como base unos contratos, orientados a artículos científicos y revisiones, ya existentes, con los distintos estándares (ERC-721 y ERC-20) implementados, no hemos tenido que realizar grandes modificaciones a los mismos. A continuación, aparece el enlace al repositorio que contiene el código de los contratos originales.

https://github.com/DecentralizedScience/Rewards/tree/mvp/contracts

No obstante, para poder enfocarlos a los requisitos que se buscaban, se tuvo que reescribir alguna función en concreto y tomar alguna decisión de diseño para que nuestra plataforma fuera exactamente como la queríamos desarrollar.

La primera decisión que se tomo fue la de implementar un artículo científico o paper maestro. Tal y como estaban escritos los contratos era necesario que cada revisión dependiera de un paper concreto y perteneciera a un autor en específico, en nuestro caso esta opción no era necesaria ya que no se trata con datos de autores de papers sino con revisores exclusivamente.

Es por esto por lo que, al desplegar el contrato a la cadena de bloques, es obligatorio que se genere un artículo del que van a depender todas las revisiones, y que el mismo pertenezca a un autor, que en el caso de nuestro proyecto se decidió que sea el Deployer, es decir, la persona encargada de lanzar el contrato a la blockchain.

```
async loadBlockchainData() {
  const web3 = window.web3;
  try {
    const accounts = await web3.eth.getAccounts();
    this.setState({ account: accounts[0] });
    // Get the contract instance.
    const networkId = await web3.eth.net.getId();
    const deployedNetwork = RewardsContract.networks[networkId];
    const instance = new web3.eth.Contract(
      RewardsContract.abi,
      deployedNetwork.address
    );
    const paperCount = await instance.methods.getPaperCount().call();
    const reviewersCount = await instance.methods.getReviewersCount().call();
    if (paperCount < 1) {

      await instance.methods
        .createPaper("paper")
        .send({ from: this.state.account })
        .once("receipt", (receipt) => {
        });
    }
```

Figura 5.7: Comprobación inicial del "paper maestro"

Por esto mismo, al lanzar la aplicación por primera vez, se ejecuta la función “loadBlockchainData” que comprueba que exista un único paper en la cadena de bloques y, de no ser así, utiliza Web3.js para llamar al contrato a crearlo, y lanzar así la petición de Metamask a la espera de ser aceptada, problema que pudimos solventar directamente en el Front-End.

La siguiente decisión que tomamos fue modificar la manera de crear, emitir y mintear, es decir, de “acuñar”, el NFT o “Award”. Partíamos de la base de que los tokens almacenaban el identificador del paper al que pertenecen, las direcciones del emisor y del receptor, y su propio identificador, generado con un hash de los tres primeros datos junto con el tipo de recompensa: Gold, Silver o Bronze, como ya se ha explicado con anterioridad.

```
struct Award {
    uint id;
    address reviewer;
    address sender;
    uint awardId;
}
```

```
function giveAward(uint _id, address payable _reviewer, uint _awardId ) public payable{
    require(_id < papers.length);
    require(_awardId < 4);
    require(bytes(papers[_id].reviews[_reviewer]).length != 0);
    string memory _hash = string(abi.encode(_id, _reviewer, msg.sender, _awardId));
    require(!hasGivenAward[_hash]);
    awardsToken.mint(_reviewer, _hash);
    awards[_reviewer].push(_hash);

    emit AwardGiven(_id,  _reviewer, msg.sender, _awardId);
}
```

Figura 5.8: Award y su finción give originales

Como se quería poder representar de manera gráfica los tokens, para que éstos almacenaran la información de su representación, y como todas las revisiones iban a depender del mismo paper, lo que ocasiona que solo se pudiera dar una recompensa a un revisor una vez de cada tipo, se decidió modificar la estructura de los tokens, así como la función “giveAward”.

De esta manera, los “Award” también almacenan el identificador de la revisión a la que pertenecen y la dirección de Infura/IPFS donde se encuentra su representación gráfica.

Ahora el identificador propio del Award es un hash de los datos anteriores más los dos datos nombrados anteriormente. Gracias a esta implementación ahora los usuarios pueden dar un NFT de cada tipo una vez a cada revisión de un revisor concreto.

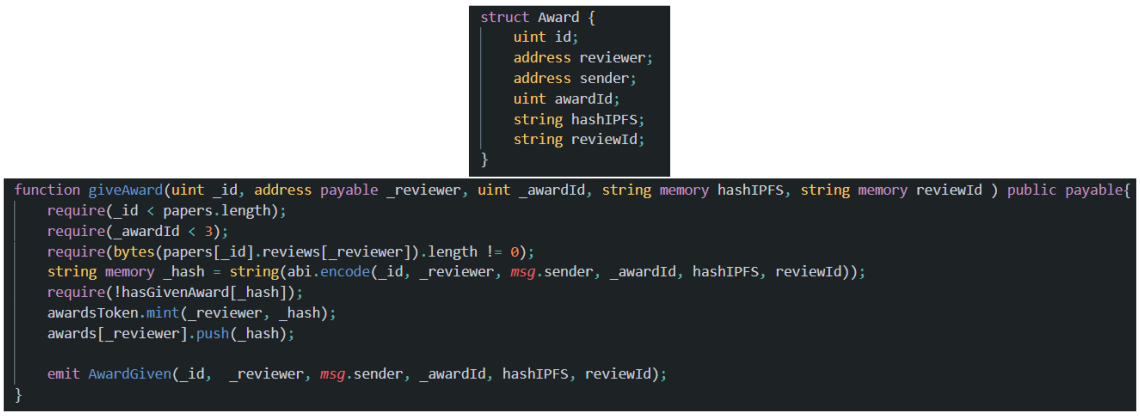


Figura 5.9: Award y su función give modificados

Por último, decidimos crear una función que nos permitiera encontrar a quérevisor pertenece un identificador de revisión concreto, ya que esta información facilita algunas peticiones desde el frontend a los contratos, para ello creamos un mapping que relaciona los strings de los identificadores con las direcciones de sus revisores, y posteriormente la función “getReviewerbyReview” que nos devuelve este dato.

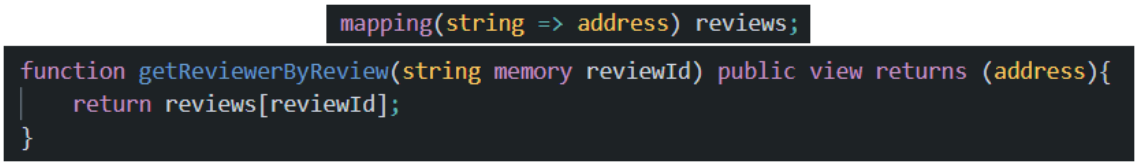


Figura 5.10: Mapping y getter de revisores-identificadores de revisiones

5.9 Casos de Uso

Figura 5.11: Diagrama de Casos de Uso

5.10 Especificación del Diseño

Tras el desarrollo de los prototipos, las primeras reuniones del equipo estuvieron muy orientadas al diseño del proyecto final. Estos prototipos ofrecieron una idea más clara de cuál era el objetivo, por ello decidimos crear unos bocetos y posteriormente unos mockups que sirvieran de guía durante todo el proceso de desarrollo.

Se comenzó creando unos bocetos a mano con la herramienta Notability[72] para reunir de una manera rápida todas las ideas que queríamos incluir. De esta forma, conseguimos en poco tiempo un diseño que serviría de punto de partida.

En esta primera fase, los bocetos recogían únicamente la página web principal y el módulo del botón de una manera muy sencilla. La página web constaba de una única vista donde se recogían todos los revisores, así como su reputación y awards.

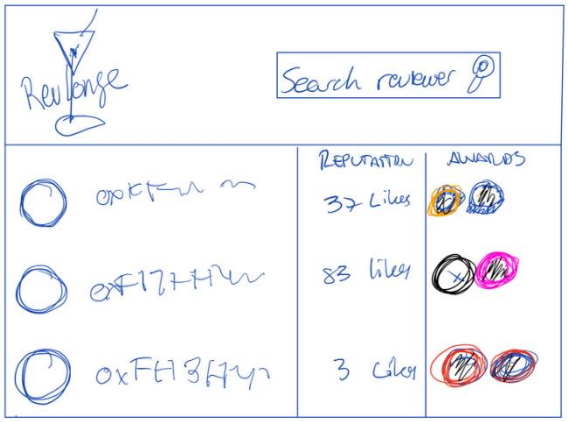


Figura 5.12: Primer boceto de la aplicación

En el caso del botón, se trata de una interfaz básica con el logo y debajo la información relacionada con las cuentas entre las que se va a realizar la transacción. Además, también encontramos los botones para dar las diferentes recompensas al revisor. Este boceto muestra también las ventanas en caso de éxito o error que verían los usuarios.

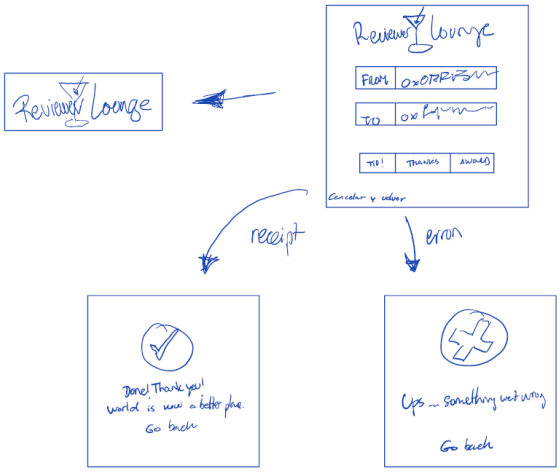


Figura 5.13: Primer boceto del botón

Una vez que los primeros bocetos estaban creados, se llevó a cabo una reunión de todo el equipo para validar los bocetos. Tras tener el visto bueno, se quería ampliar el alcance de la aplicación. Para ello, se realizó un brainstorming y así acordar tanto las funcionalidades que incluiría el proyecto como las metodologías para llevarlas a cabo.

Con el objetivo de realizar una nueva iteración sobre los bocetos, se validaron estas ideas para empezar a desarrollar unos nuevos bocetos mucho más concretos y que aportarían una idea mucho más clara de cara al desarrollo de la plataforma.

Tanto la página web como el módulo del botón se incluyen nuevas características. En el caso de la página web, no incluye muchos cambios, únicamente se añade un nuevo botón para que un usuario pueda ir a su perfil.

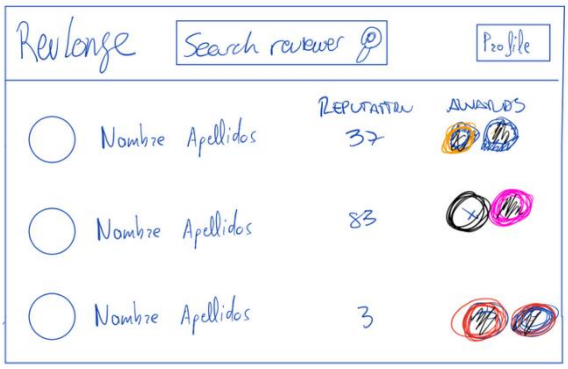


Figura 5.14: Segundo boceto de la aplicación

Por otro lado, en el caso del botón, contiene mucha más información en relación con el revisor al que se quiere recompensar. Además de ver la información de la transacción y los botones de los tres tipos de recompensa como en el boceto de la fase anterior, se añaden tanto datos del revisor, como su nivel de reputación, los rewards que ha recibido y el número de revisiones que ha realizado.

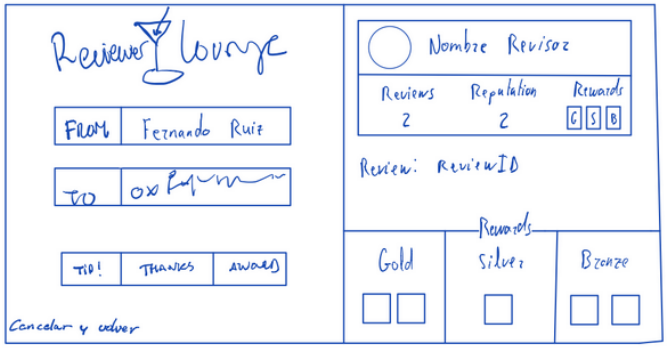


Figura 5.15: Segundo boceto del botón

En esta fase se crearon además unos bocetos para los rewards, basados en el logo de la aplicación y que tendrían un diseño diferente dependiendo del tipo que fuera, de manera que la franja externa es del color del tipo que representa.

Figura 5.16: Boceto del perfil de revisor

Finalmente, aunque los bocetos eran ya bastante concisos, se pusieron en común nuevas ideas para mejorarlos. Con ellas, empezamos a desarrollar los mockups de las vistas de la aplicación con la herramienta Balsamiq Wireframes[73].

El primer mockup es el de la página principal de la aplicación, donde se podrán ver todos los revisores con su nivel de reputación y awards. Como novedad se incluye una nueva columna para facilitar el contacto con los revisores.

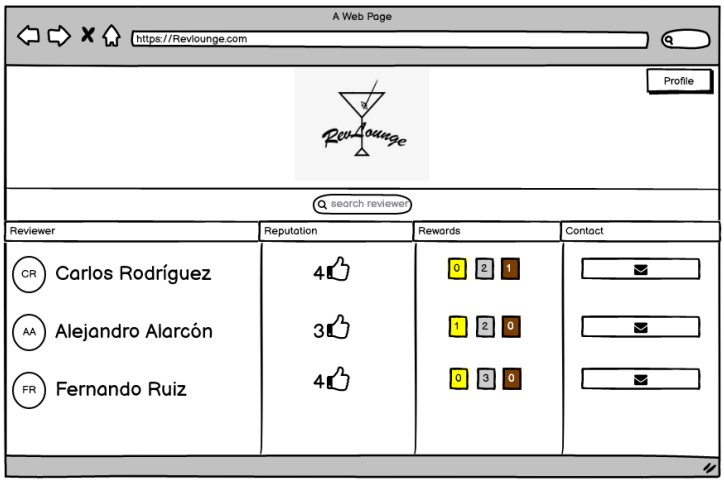


Figura 5.17: Boceto en Balsamiq de la tabla de revisores

El segundo mockup que desarrollamos se correspondía con el módulo del botón. En este caso, no hubo grandes cambios, únicamente se añadió un botón para resolver dudas relacionadas con los elementos del propio módulo.

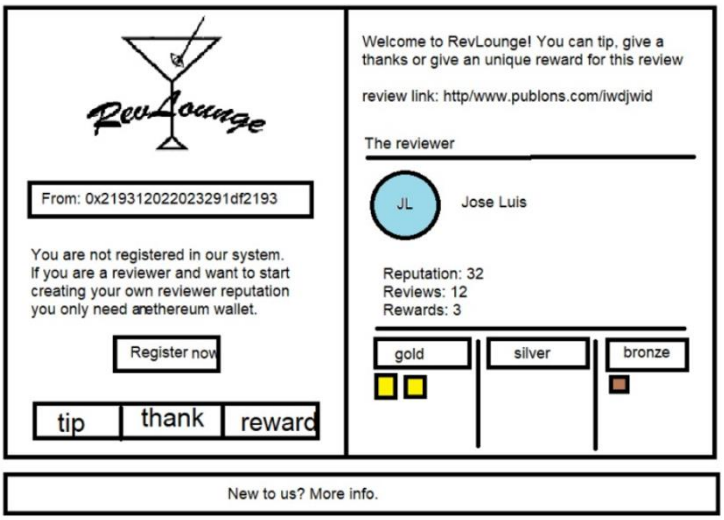


Figura 5.18: Boceto del popup del botón

El mockup que más cambios recibió con respecto a los bocetos previamente creados fue el de la página del perfil. Tras la reunión del equipo, se decidió crear una página mucho más desarrollada y que expusiera mejor la información acerca del revisor ya que apenas había diferencias entre la página de inicio y la de perfil en cuento a la información mostrada. Por ello, se decidió mantener el mismo patrón de diseño que en la página principal, con el logo en la parte central del banner superior y debajo toda la información de la página. En primer lugar, encontramos una tarjeta que incluye las recompensas de reputación y los tipos de rewards que tiene. A su derecha tenemos la información personal del revisor y una tarjeta que incluye su información profesional.

En la parte central encontraremos todas las revisiones que ha importado el revisor con su contenido y descripción.

Por último, en la parte de abajo encontramos todos los rewards del revisor clasificados por tipo.

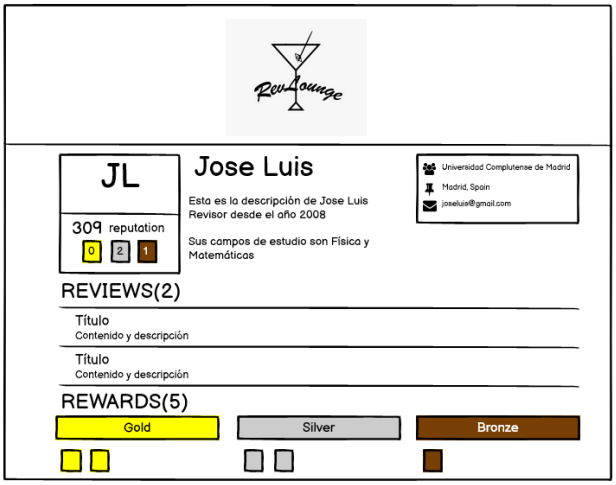


Figura 5.19: Boceto en Balsamiq del perfil de revisor

Capítulo 6. Implementación

RevLounge interactúa con las diferentes tecnologías que hemos mencionado previamente. El proyecto cuenta con varias partes:

- Aplicación web en React que los usuarios pueden visitar para hacer las funciones de visualización y registro.
- Un módulo en React que permite otorgar recompensas a los revisores.
- Un paquete npm que permite acceder al módulo de recompensas desde otras plataformas.
- Una base de datos en Mongo que permite manejar y almacenar la información que no se guarda en los contratos.

Todas estas tecnologías interaccionan entre ellas, y con los contratos de Solidity, para dar forma a la plataforma como se puede ver en el siguiente diagrama:

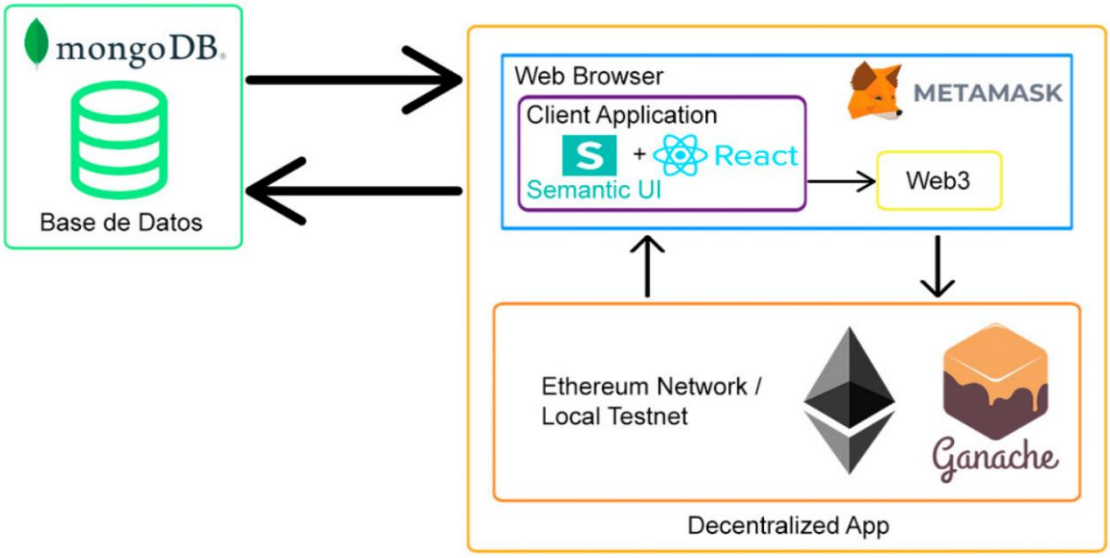


Figura 6.1: Estructura de la aplicación

6.1 Aplicación Web en React

Para poder entender mejor el funcionamiento de la aplicación y cómo ha sido desarrollada en React, se van a introducir unos conceptos a los que se harán referencia durante este capítulo. Uno de los elementos más importantes de las aplicaciones hechas en React son los componentes. Estos consisten en partes de la interfaz creados independientemente unos de otros con el fin de reutilizarlos y crear una interfaz mucho más compleja. Toda aplicación comienza desde el componente App, del cual descuelgan los componentes hijos formando un árbol. Para entenderlo más fácilmente imaginemos que queremos crear una aplicación como YouTube. Podemos dividir la página en distintos componentes. La "NavBar" o barra de navegación, la barra lateral o "SideBar" con accesos directos a distintas categorías y la "Feed" dónde aparecen los videos recomendados.

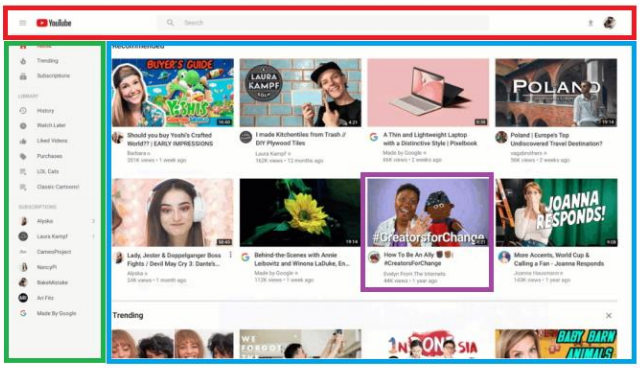


Figura 6.2: Ejemplo de una posible división por componentes en YouTube

De esta manera crearíamos un árbol de componentes que comenzaría con App.js, seguido del resto de elementos. La "Feed" por ejemplo, puede contener el componente Video, el cual se puede reutilizar en otras páginas o incluso en otra aplicación. Es decir, creamos los elementos independientemente para luego crear una interfaz más compleja.

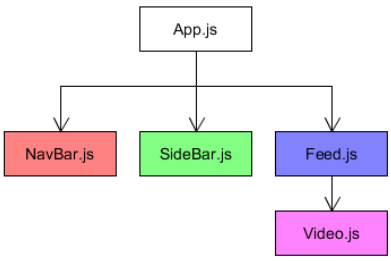


Figura 6.3: Ejemplo de un posible árbol de componentes de YouTube

Otros dos conceptos de los que se harán mención más adelante son los estados y las props. Los estados almacenan la información que se va a mostrar en la interfaz cuando se renderice el componente y además pueden ser modificados. Por otro lado, los props son inmutables y sirven para enviar información de un componente padre a uno hijo.

A continuación, se muestra el árbol de componentes de la Aplicación Web:

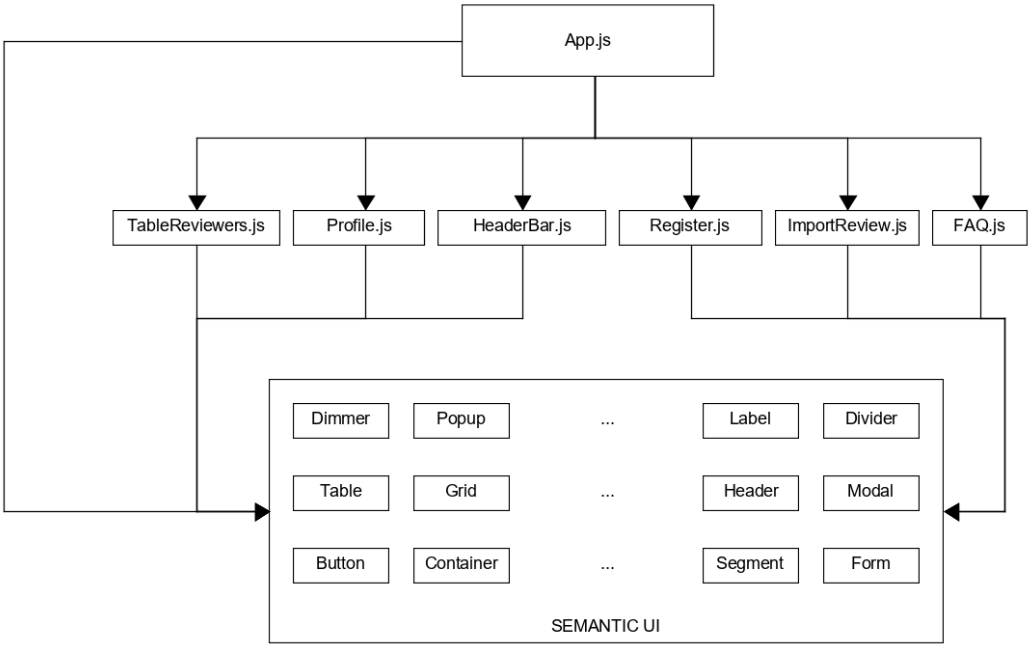


Figura 6.4: Árbol de Componentes de la Aplicación Web

- App: es el componente raíz del proyecto. Utiliza la API de mongo para cargar la información de la base de datos y Web3.js para obtener los datos de la blockchain, que serán guardados en los estados del componente. De la base de datos se obtiene la información relacionada con el perfil de los revisores registrados mientras que de la blockchain se obtienen la reputación y premios obtenidos por cada revisión. Además, se encarga del enrutamiento mediante React Router. Esta librería permite seleccionar los componentes independientes que se renderizan en la interfaz y enviar los props que cada uno de ellos necesita.
- TableReviewers**:** crea una tabla de los revisores registrados y muestra su nombre con un resumen de sus recompensas.
- HeaderBar**:** implementa una barra superior con accesos a distintas páginas.
- Register**:** implementa el formulario de registro, así como la ventana para importar la primera revisión.
- ImportReview**:** crea la página para introducir la URL de la revisión que se quiere importar.
- Profile**:** implementa una página que contiene los datos del revisor, la lista de sus revisiones importadas y la información de sus recompensas.
- FAQ**:** crea la página con las preguntas más frecuentes al usar la aplicación.

Todos los componentes arriba mencionados importan elementos del framework Semantic UI. Se han utilizado desde los elementos más básicos como Button, Input, Icon o Label hasta elementos estructurales como Grids, Table o Menu. También hemos diseñado los

paneles de ayuda con elementos como Dimmer o Popup. El uso de esta UI ha permitido conseguir consistencia en el diseño de la aplicación.

Vistas y ejemplos de uso

- Visualizar tabla de revisores

Primeramente, cuando un usuario entra en la plataforma, lo primero que visualiza es la página principal, donde se encuentra una lista de los revisores registrados en formato de tabla.

Debajo del banner superior hay un buscador que permite filtrar los resultados de la tabla por nombre.

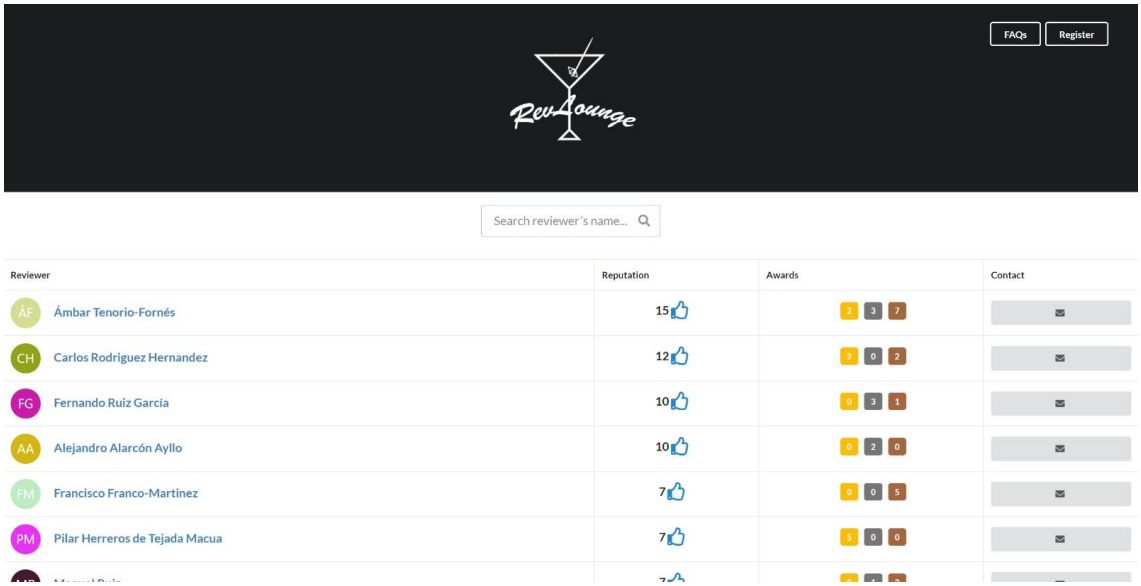


Figura 6.5: Página principal de la aplicación

Como podemos observar, en la tabla se visualizan los datos principales de cada revisor, esto es, su foto de perfil generada automáticamente, su nombre y apellidos, la cantidad de tokens de reputación RPT que tienen, la cantidad total de NFT agrupados por tipos que poseen y un botón de contacto enlazado al correo electrónico, datos que recibe por props directamente el componente.

```
<TableReviewers thanks={this.state.thanks} golds={this.state.golds} silvers={this.state.silvers} bronzes={this.state.bronzes} />
```

Figura 6.6: Ejemplo de componente recibiendo props

- Registrar revisor

Si un usuario decide crearse un perfil, simplemente debe darle al botón de “Register” de la esquina superior derecha del encabezado de la página, será redirigido a una página que contiene un formulario de registro.

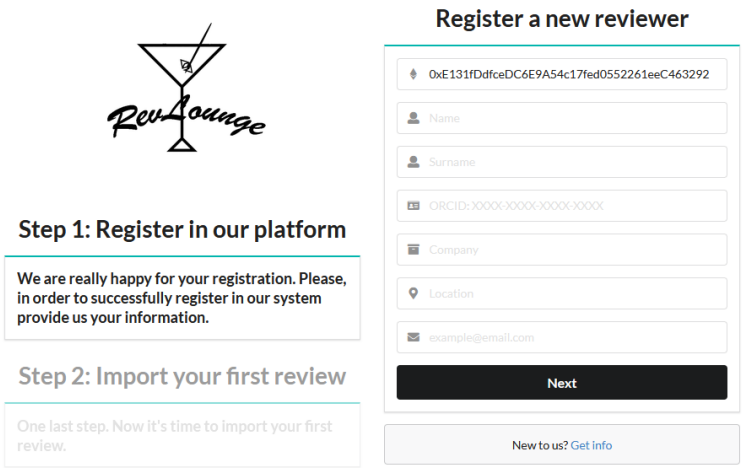


Figura 6.7: Primer paso de registro de usuario

Primeramente, la dirección de su cartera se toma automáticamente de la cuenta que se encuentra abierta en Metamask y no se puede modificar de forma manual, al tratarse de un requisito obligatorio, en caso de que el usuario no tenga la extensión instalada, se le presenta un mensaje para informarle. A continuación, se le piden los datos que serán almacenados en Mongo con su información personal.

Esta información se pasa por un validador para comprobar que es rellena por completo, y que el ORCID concuerda con la longitud real, así como que el correo electrónico tenga el formato correcto.

Después de hacer click en el botón “Next”, se pide al usuario que introduzca la URL de la primera revisión que desea registrar en el sistema, debido a la forma en la que están estructurados los contratos, es obligatorio que al registrar un revisor en la blockchain también se registre una revisión a su nombre.

En el caso de que haya alguna duda a la hora de encontrar la URL de la revisión, en el panel lateral izquierdo se puede encontrar una breve explicación general y un botón que redirige a las preguntas más frecuentes, donde se explica detalladamente donde encontrar el enlace a la revisión en ORCID y en F1000Research.[74]

Cuando el usuario está registrado, en la parte superior derecha de la pantalla, el botón de “Register” pasa a ser “Profile” para poder remitirle directamente a su perfil personal.

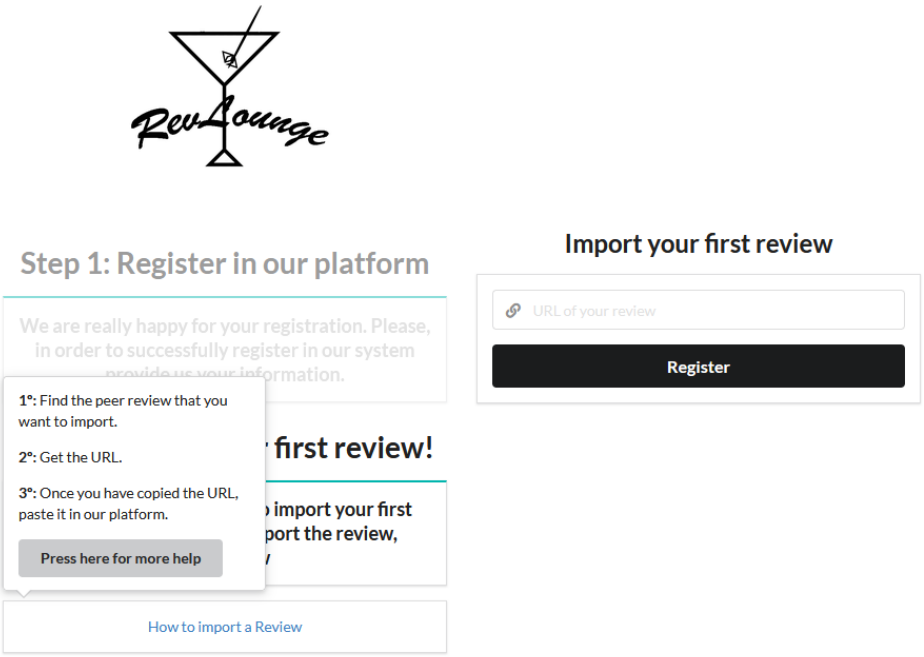


Figura 6.8: Segundo paso de registro de usuario

A la hora de registrar en los contratos y en Mongo los datos que el usuario acaba de introducir, primero se hace una llamada al contrato que comprueba que no exista un usuario con esa cuenta o con ese identificador de revisión, es entonces cuando Metamask muestra una notificación solicitando aceptar la transacción y, solo cuando esta solicitud es aceptada y registrada de forma correcta, es cuando se graban los datos en la blockchain, para posteriormente llamar a la API de Mongo para almacenar toda la información de registro del revisor.

- Ayuda y documentación

Sería ideal que nuestro sistema no necesitase de ninguna ayuda o documentación para ser usado. No obstante, hemos añadido una sección de FAQs fácilmente accesible desde el encabezado de la aplicación, así como en los lugares que el usuario podría necesitar ayuda. En la sección de preguntas añadimos la lista de pasos concretos que el usuario tiene que seguir para solucionar el problema.

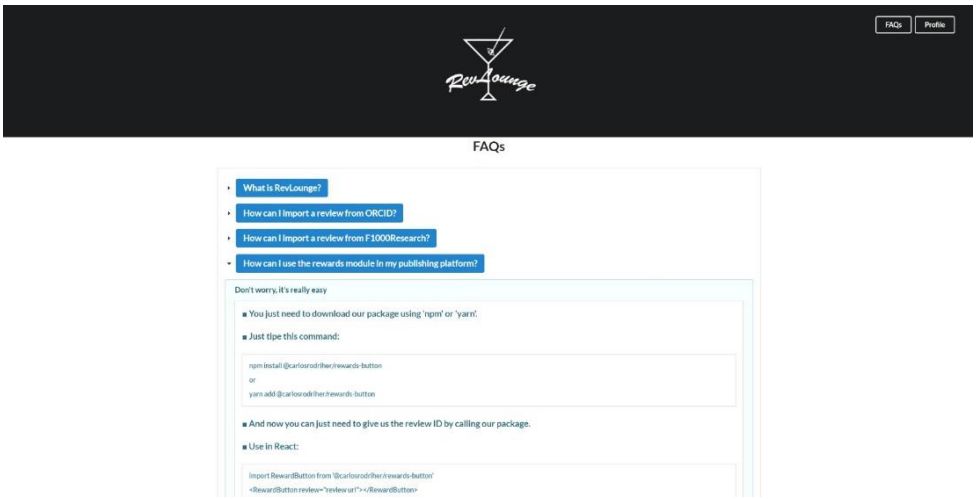
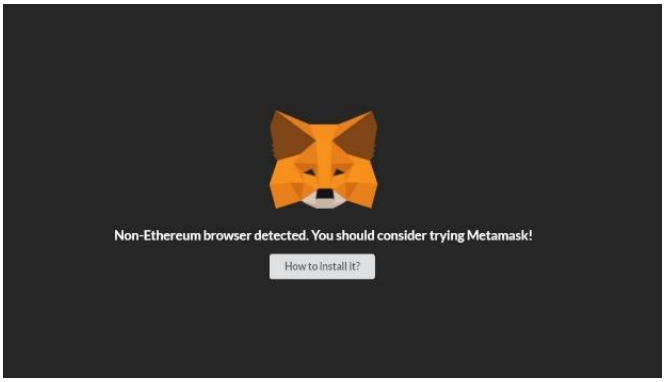


Figura 6.9: Página de las FAQs

Para encontrar la respuesta a las preguntas que aparecen, el usuario simplemente debe pulsar sobre ellas y se desplegará la respuesta.

Como nuestra aplicación requiere MetaMask, en caso de no tenerlo, se ha creado una guía que explica como instalar la extensión de navegador. De la misma manera, esta ayuda se ha implementado en el módulo de recompensas.



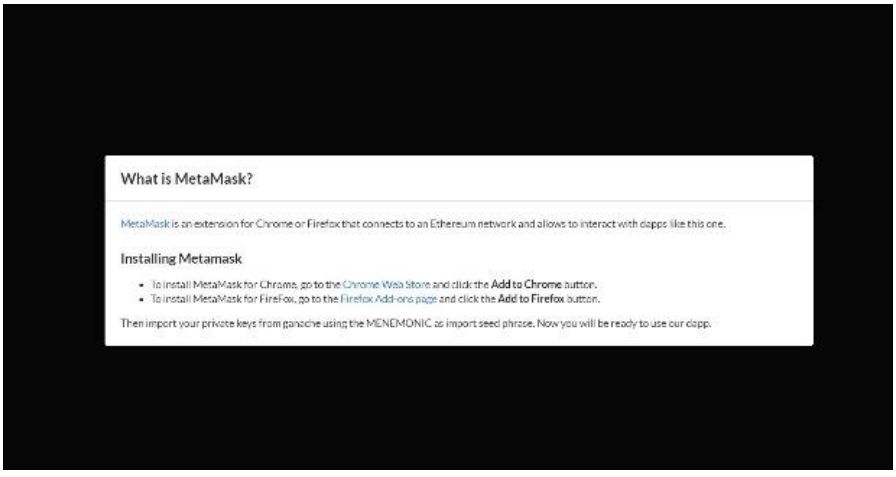


Figura 6.10: Guía de como instalar MetaMask

- Ver el perfil de un Revisor

Si, desde la pantalla principal, un usuario decide acceder al perfil de un revisor seleccionando su nombre, se abrirá la página correspondiente al identificador del perfil seleccionado.

En ella podemos encontrar reflejada toda la información que se tiene, tanto en la blockchain como en la base de datos, del revisor en cuestión. En la parte izquierda encontramos su foto de perfil generada con las iniciales de su nombre y apellido, junto con los datos que se mostraban en la tabla, el número de tokens de reputación RPT que posee y un contador total de cuantas recompensas NFT de cada tipo tiene.

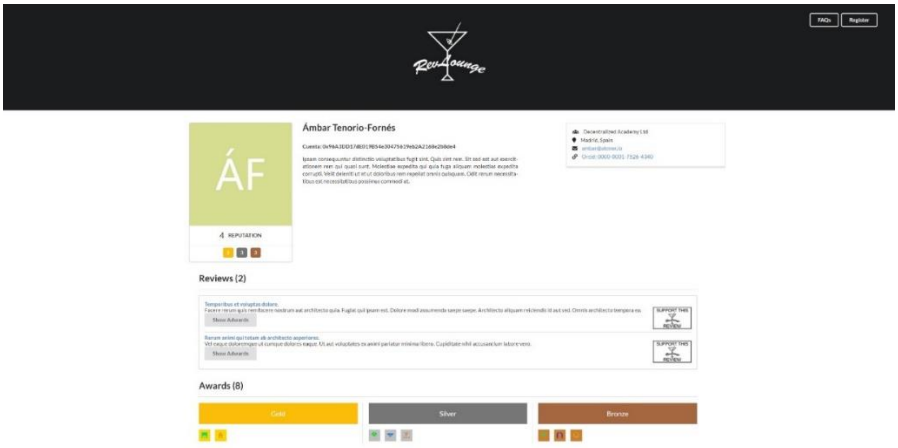


Figura 6.11: Perfil del revisor

En la parte derecha de la pantalla encontramos la información personal del perfil, la organización a la que pertenece, su localización, su correo electrónico y su número de ORCID enlazado a la página de su perfil en dicha plataforma.

A continuación, encontramos una lista con todas las revisiones que el usuario ha subido a la plataforma, acompañadas de los awards en concreto que ha recibido cada revisión representados de forma gráfica por su Identicon, el título enlazado a la URL de la misma y su descripción. A la derecha de cada revisión encontramos el botón que abre el módulo popup que puede ser implementado en páginas externas. De esto hablaremos más adelante en el apartado 6.1.2.

En la parte inferior de la página de perfil, encontramos todos los awards totales que ha recibido el revisor, con su representación gráfica. Cada uno de ellos está organizado por su tipo y contiene un “hover”, es decir, cuando se pasa el cursor por encima muestra la información de quién lo ha mandado, a quien se ha mandado y cuál es la revisión a la que pertenece.

Todos estos datos, como ya hemos comentado anteriormente, son cargados de la base de datos y de los contratos de Solidity, utilizando el identificador del revisor.

- Importar Revisión

En el caso de que un usuario esté visitando su propio perfil, se coteja la dirección de la cuenta de Metamask con la dirección de la cuenta asociada al perfil seleccionado para mostrar un botón de “Add a new review”.

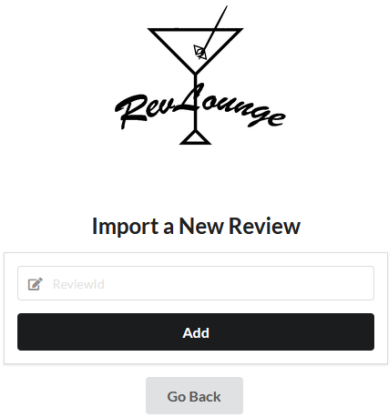


Figura 6.12: Insertar nueva revisión

Cuando el usuario acciona este botón, se le redirige a una página que permite agregar una revisión nueva a su cuenta, para ello se comprueba que la persona que está accediendo es la propietaria de la cuenta a la que se está añadiendo una revisión. Se comprueba también que esa revisión no exista ya en la blockchain haciendo uso del mapping que describimos en la sección 5.3.1 “Cambios en el contrato”. En caso de que todas las condiciones sean correctas, se llama a la función del contrato que se encarga de registrar la nueva revisión y, si la transacción de MetaMask es correcta, se procede a hacer lo mismo en la base de datos.

6.2 Módulo de Recompensas

A continuación, se muestra el árbol de componentes del módulo de recompensas:

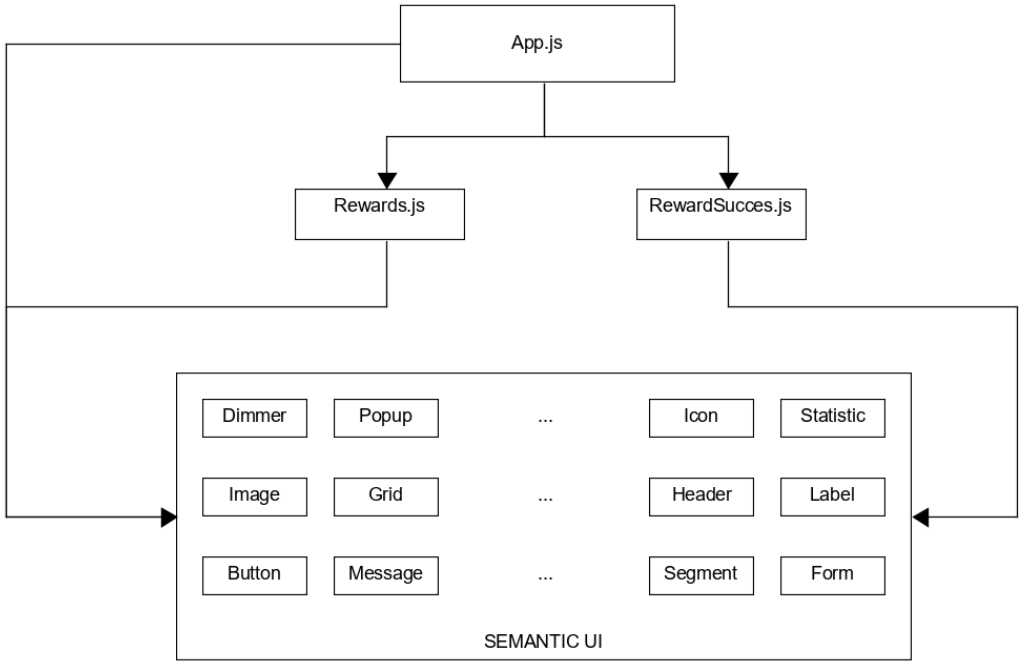


Figura 6.13: Árbol de Componentes del Módulo de Recompensas

Para poder dar una recompensa a una revisión, se tendrá que acceder al módulo de recompensas. Para ello, y como se explica en el punto 6.1.3, se ha creado un botón que cualquier página de revisiones puede añadir en su interfaz. El botón se sitúa al lado de una revisión y si se presiona, se abre un popup como el que muestra la figura 6.14.

El módulo consiste en una aplicación de React, conectada a la misma base de datos que la aplicación web principal de este proyecto. Además, utiliza la librería de Web3.js para realizar las operaciones y consultas en la blockchain. Cuando se abre el popup, Web3.js realiza una consulta en la blockchain y si la revisión existe, se cargan los datos relacionados con la revisión y el revisor realizando la consulta en la base de datos.

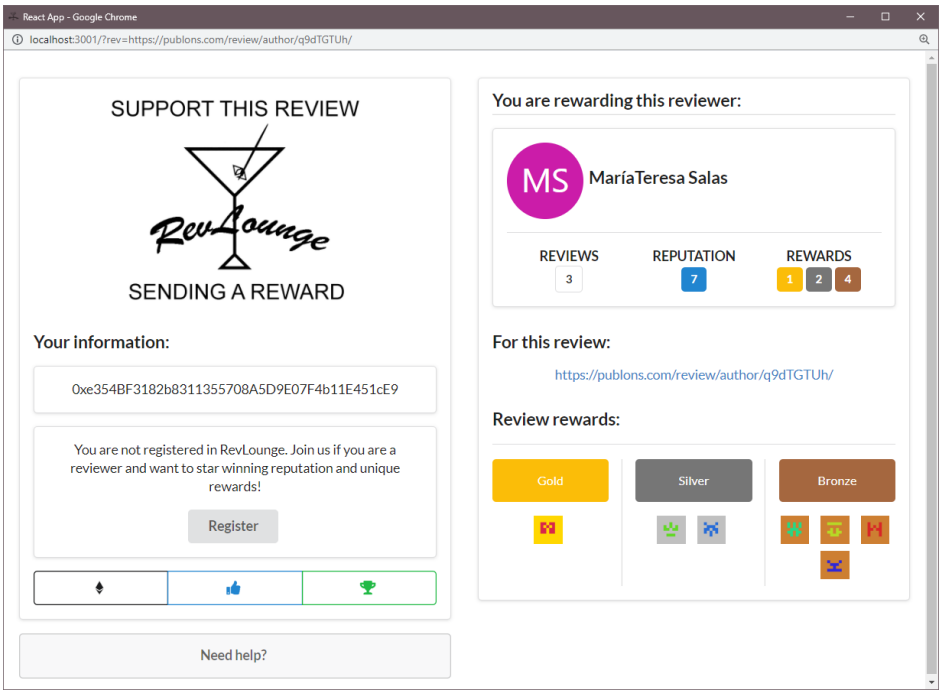


Figura 6.14 Vista principal del módulo de recompensas

En el popup encontramos, por un lado, la dirección de la cuenta del usuario que ha interactuado con el botón y, si este se encuentra registrado en nuestra plataforma, se muestra también su nombre y foto de perfil.

Por otro lado, se muestra la información principal del revisor al que pertenece esa revisión y los awards que ya ha recibido anteriormente esa revisión en concreto, de haber recibido alguno.

Debajo de la información del usuario que ha ejecutado la llamada al módulo, se encuentran las tres opciones principales de rewards con las que cuenta la plataforma y que ya han sido descritas anteriormente: mandar una propina, dar un token de reputación o enviar un award NFT.

Cuando se interacciona con una de las opciones, se llama a las funciones de los contratos y la acción queda debidamente registrada en la cadena de bloques. Para ello, primero salta la notificación de Metamask, a la espera de aceptar la transacción.

Las diferentes acciones que se pueden realizar con este popup son:

- Dar una Recompensa de tipo Award

En este caso, el usuario seleccionará el botón “Award” y se abrirá un popup mostrando los tipos de Award que hay para que el usuario elija el que desee otorgar.

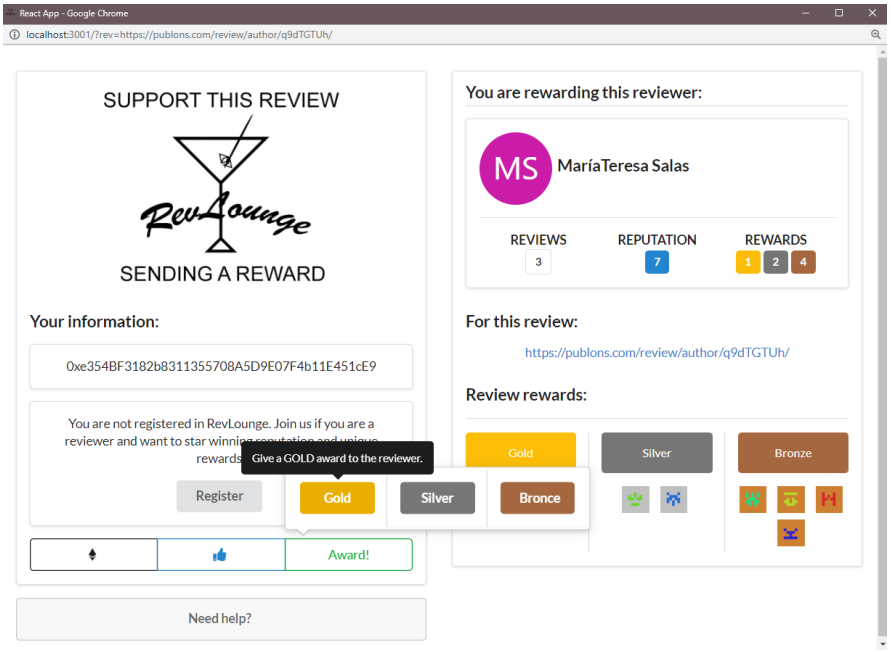


Figura 6.15: Dar una Recompensa de Tipo Award

- Dar una Recompensa en Ether

En este segundo caso, el usuario seleccionará el botón con el logo de Ethereum, el primero por la izquierda lo cual abrirá una entrada de texto en el que el usuario puede indicar la cantidad en ether que aportar al revisor.

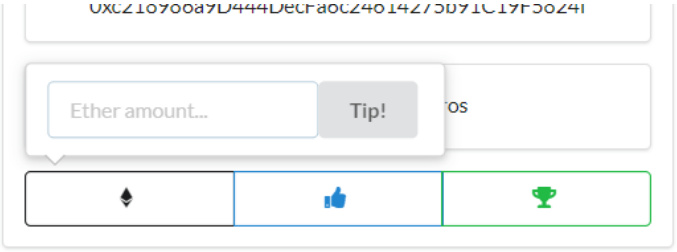


Figura 6.16: Dar Recompensa de Tipo Tip

- Dar una Recompensa de tipo Reputación

En el caso de querer dar una recompensa de reputación, el usuario tiene que seleccionar el botón situado en medio, el cual hará saltar la notificación de Metamask a la espera de ser aceptada para realizar la transacción.

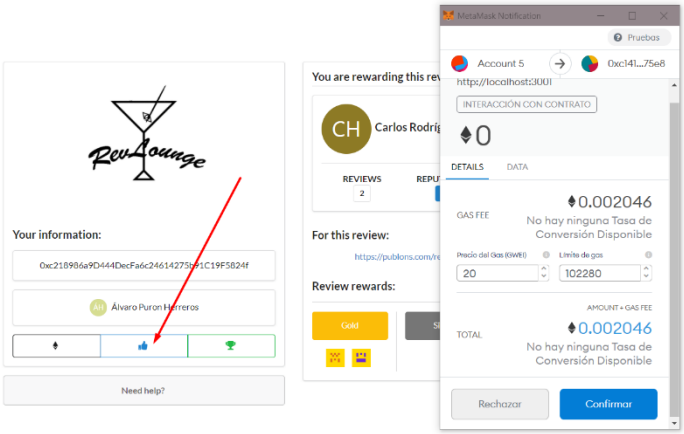


Figura 6.17: Dar Recompensa de Tipo Reputación

Tras haber dado una de estas recompensas, independientemente del tipo, el popup informará al usuario de que todo ha salido bien de la siguiente manera:

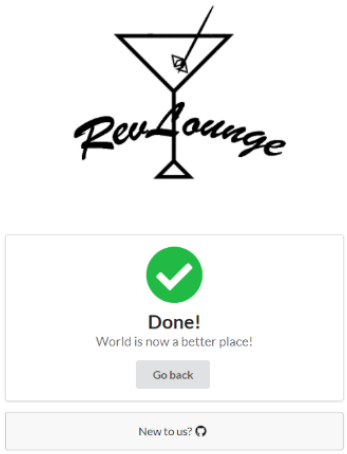


Figura 6.18: Mensaje de éxito al enviar una recompensa

6.3 Paquete de NPM

El gestor de paquetes de node (NPM) es una plataforma que permite descargar y compartir paquetes o módulos escritos en JavaScript. Los paquetes publicados pueden ser descargados y combinados con otros para crear aplicaciones o incluso otros paquetes.

En nuestro caso, se ha creado y publicado un paquete que consiste en una imagen que funciona como si fuera un botón. Si se presiona el botón, se abre un popup que contiene el módulo de recompensas explicado en el punto anterior 6.1.2. No obstante, el módulo no forma parte del paquete, es decir, tiene que haber sido lanzado en un servidor y el

botón se encarga de redirigir a la dirección donde se ha desplegado. De esta manera, cualquier usuario con una plataforma de revisión por pares puede descargarse el paquete de NPM e implementarlo en su aplicación como en la figura 6.19.

Figura 6.19: Posible implementación del paquete en F1000Research

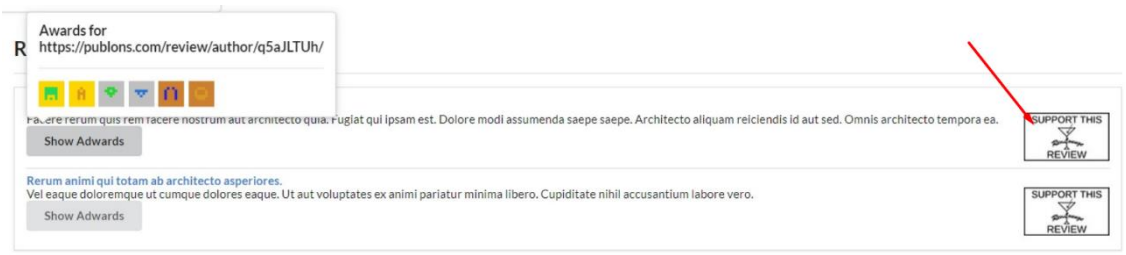


Figura 6.20: Botón implementado en el perfil de un Revisor

Para que el popup se abra con éxito, el botón tiene que recibir como parámetro el identificador de la revisión que ha sido registrada en la plataforma explicada en el 6.1.1. Si la revisión no ha sido registrada, el popup mostrara un mensaje de error.

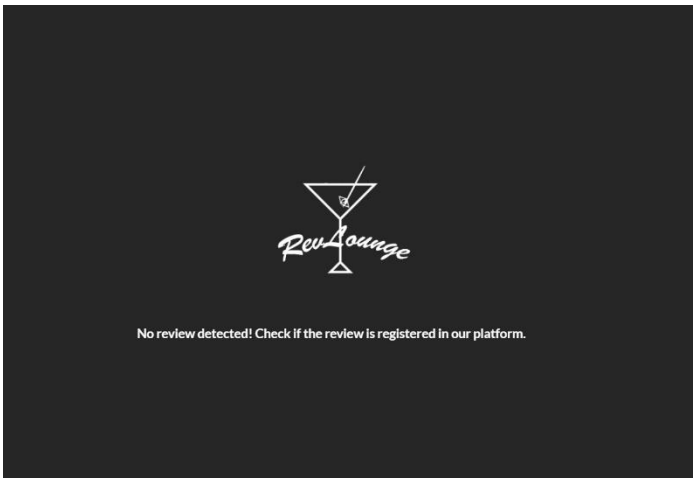


Figura 6.21: Mensaje de error en caso de no encontrar la revisión en el sistema

Enlace al paquete en la plataforma de NPM:

https://www.npmjs.com/package/@carlosrodriher/reward-button

6.4 Base de Datos

Decidimos implementar una base de datos en Mongo para almacenar toda aquella información de los revisores de la que carecen los contratos, así como aquella que nos era útil para construir el frontend de nuestro producto final.

Estos datos incluyen un identificador único y correlativo interno de revisor, el nombre y los apellidos del revisor, la dirección de su cuenta de MetaMask, una descripción de perfil, su identificador de ORCID[3], la compañía a la que pertenecen, la ciudad en la que residen, su correo electrónico de contacto y un array de revisiones que almacenan el identificador de cada una, un título y una breve descripción.

Este identificador de revisor coincide con el identificador de los contratos para que la integración con los mismos sea rápida y funcional.

Como no contamos con una plataforma que, dado un identificador de revisión, haga un “scrapping”, es decir, que tome los datos automáticamente de las APIs de las páginas de revisiones de las que provienen los identificadores, tanto el título como la descripción de estas son datos generados aleatoriamente por la librería de Javascript “faker.js[75]”. Esto nos permite tener datos que mostrar con fines explicativos de cómo se verían las revisiones de haber utilizado alguna herramienta como Bloxberg[76], que se encarga de recoger toda la información real de una revisión utilizando su identificador único y de almacenarla en la blockchain.

También hicimos uso de esta librería para generar datos aleatorios en la descripción del usuario.

```
{
    name: { type: String, required: true },
    surname: { type: String, required: true },
    summary: {type : String, required: true},
    orcid: {type : String, required: true},
    account: {type : String, required: true},
    company: {type : String, required: true},
    location: {type : String, required: true},
    email: {type : String, required: true},
    reviews:[{
        reviewId: String,
        title: String,
        description : String,
    }],
}
```

Figura 6.22: Schema de la base de datos

Para implementar la base de datos en nuestra aplicación, tenemos un directorio donde se crea el servidor de esta, dentro contamos con los archivos que permiten su correcto funcionamiento.

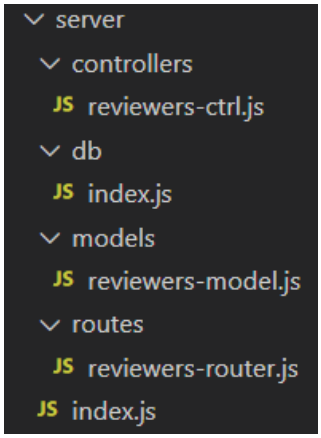


Figura 6.23: Esquema del servidor de Mongo

Utilizando el MERN stack, explicado en el Punto 3.4, se ha implementado la base de datos. El index de la raíz se encarga de usar la librería “express.js”, que sirve de ayuda para crear la API de la que más adelante hablaremos, controllers es donde se crean las distintas operaciones que se pueden llamar desde la API, db almacena la información de la base de datos, models contiene el schema que va a utilizar Mongo y, finalmente, Routes hace las conexiones de las funciones del controlador con “express.js” para que sean accesibles desde la API y, por ende, desde servicios externos.

Finalmente disponemos de una carpeta API que se encarga de hacer todas las conexiones al servidor, y que se puede utilizar desde cualquier proyecto.

Como ya hemos descrito anteriormente, la base de datos junto con los contratos nos permite que la aplicación funcione de una manera correcta.

Capítulo 7. Experimentación y Resultados

7.1 Evaluación

Para finalizar el proyecto hemos realizado una fase de experimentación con usuarios. Para ello hemos seleccionado a un grupo de 5 personas, incluyendo mujeres y hombres con el fin de valorar la usabilidad y accesibilidad de la aplicación, así como valorar el estado actual del paradigma de las revisiones. Debido a la situación en la que se encontraba la COVID-19 al realizar las entrevistas, la experimentación se ha realizado de manera remota vía Google Meet y utilizando Team Viewer para realizar las pruebas y permitir el control de la aplicación. La duración media fue de 1 hora por persona.

Las 5 personas contactadas realizan o han realizado revisiones de artículos en distintas áreas de conocimiento:

- Revisor 1 – mujer, edad 55-60, doctorada, profesora de psicobiología, Universidad Complutense de Madrid.
- Revisor 2 – hombre, edad 25-30, doctorando en Ingeniería Mecánica, Universidad Politécnica de Madrid.
- Revisor 3 – hombre, edad 50-55, Médico de familia.
- Revisor 4 – hombre, edad 30-35, doctorado, profesor en la Facultad de Informática, Universidad Complutense de Madrid.
- Revisor 5 – hombre, edad 35-40, doctorado, profesor en la Escuela Técnica Superior de Ingenieros Industriales, Universidad Politécnica de Madrid.

La experimentación se dividió en 3 pasos, entrevista inicial, testing con el usuario y una encuesta final. Esta última se ve recogida en el Capítulo 9 “Conclusiones y Trabajo Futuro”.

7.2 Entrevistas

Realizamos las entrevistas con el objetivo de conocer en mayor profundidad la opinión de los entrevistados acerca de recompensar el trabajo de los revisores, quién debería dar estas recompensas y si un sistema de reputación podría mejorar la calidad de las revisiones. Además, nos interesaba conocer sus conocimientos acerca de las plataformas de revisión online y sobre las tecnologías descentralizadas como blockchain y Ethereum.

Las entrevistas eran estructuradas divididas en tres grupos principales. El primer grupo de preguntas tenía el objetivo de conocer la relación de los entrevistados con los artículos científicos y sobre todo ver si han realizado alguna revisión. El siguiente grupo, consistía en una serie de preguntas para conocer su opinión acerca del proceso de las revisiones

por pares y algunas formas de aportar una mejora a la situación de los revisores. Por ejemplo, se preguntó acerca de si creían que el proceso de revisión actual es ágil y de qué manera se podría mejorar o si de si consideraban que un sistema de reputación podría mejorar este proceso. Por último, el tercer grupo estaba enfocado en saber si los entrevistados tenían algún conocimiento previo de las tecnologías que se han utilizado en este proyecto. Se preguntó por la tecnología Blockchain y por la plataforma Ethereum, además de los tokens y las aplicaciones descentralizadas.

7.2.1 Conclusiones

- Todos los usuarios afirmaron realizar pocas revisiones debido a que no reciben ninguna compensación por ello.
- Todos coincidieron en que un sistema de recompensas podría ser beneficioso.
- Todos los usuarios coincidían que un sistema de reputación de revisores ayudaría a mejorar la calidad y honestidad de las revisiones.
- Dos de los cinco mencionaron que deberían ser las editoriales las que se encarguen de recompensar a los revisores y no los demás usuarios.
- La mayoría de los usuarios no sabe dónde se almacenan sus revisiones.
- Dos de los cinco usuarios conoce las plataformas donde se almacenan sus revisiones, pero nunca acceden a ellas.
- La mayoría de los usuarios desconocían el funcionamiento de las tecnologías descentralizas.
- La mayoría de los usuarios desconocían la existencia del ecosistema Ethereum y los NFTs.

7.3 Testing

Realizamos una prueba con los usuarios para observar la experiencia de los revisores al interactuar con la aplicación. Con este fin, preparamos una serie de tareas para que las realizaran en tiempo real mientras se analizaba su interacción con la interfaz. Además, también se recogieron sus comentarios mientras se movían por la aplicación para conseguir unas conclusiones objetivas.

7.3.1 Objetivos

Los objetivos que se persiguen se diferencian en dos categorías, por un lado, métricas cuantitativas y, por otro lado, métricas cualitativas.

Las métricas cuantitativas que se analizar son:

- El usuario completa la tarea sin ayuda.

- El usuario necesita pistas para poder continuar con el flujo de las tareas.
- ¿Cuánto tiempo tarda el usuario en completar la tarea?
- ¿Durante cuánto tiempo el usuario ha permanecido bloqueado?
- ¿Dónde ha tenido problemas el usuario?
- ¿En qué puntos el usuario espera un resultado diferente?

En cuanto a las métricas cualitativas:

- ¿Cómo se siente el usuario al desarrollar la tarea?
- ¿Cómo se siente el usuario mientras se encuentra bloqueado?
- ¿Qué nivel de dificultad encuentra el usuario en la tarea?
- ¿Siente el usuario que la tarea está completada al llegar al final de la ejecución?
- ¿Entiende el usuario todos los elementos visuales de la aplicación?

7.3.2 Conclusiones

El principal resultado de la prueba a los diferentes revisores es que en la totalidad de los casos han podido completar las tareas sin ningún tipo de inconveniente. Aunque los revisores que están menos acostumbrados a manejar sus revisiones a través de plataformas digitales necesitaron ayuda por nuestra parte en algún punto concreto de la prueba.

Finalmente, sacamos las siguientes conclusiones:

- La aplicación es una idea novedosa y cuenta con una interfaz muy sencilla.
- Aconsejable añadir más información de tu perfil de investigador.
- La funcionalidad de importar una revisión manualmente es muy útil.
- El logo no tiene relación con el objetivo de la aplicación.
- Las recompensas son sencillas de entender.
- Sería aconsejable añadir información de ayuda explicando los tipos de awards.
- Hacer más comprensible la representación de los awards.
- La interfaz es muy rápida.

Tras realizar el test, queríamos conocer más a fondo la opinión de ellos acerca de la experiencia que habían tenido usando la aplicación, así como de su opinión sobre la misma y conocer si tenían alguna sugerencia de mejora, por lo que les solicitamos cumplimentar una encuesta.

Para realizarla, decidimos utilizar una escala Likert del 1 al 5, expresando el 1 “Totalmente en desacuerdo” y 5 “Totalmente de acuerdo”.

Pregunta	Media
Para poder utilizar la aplicación necesito aprender cosas nuevas que desconocía	3.2

Creo que necesitaría ayuda para poder usar esta aplicación	1.4
Creo que la plataforma en sí alberga funciones interesantes y bien integradas	4.8
Creo que la mayoría de las personas podrán usar la aplicación con facilidad	4.2
El sistema es intuitivo	5
El sistema de registro no me supuso un problema	4.4
Me resultó fácil dar algún tipo de recompensa a una revisión	5
¿Usarías esta plataforma si estuviera disponible?	4.2
¿Preferirías utilizar una plataforma igual si no tuviera que utilizar Blockchain?	3
Como revisor estaría dispuesto a pagar por poder incluir sus revisiones en la plataforma	1

Figura 7.1: Tabla de resultados de las encuestas

También, se recibieron varios comentarios acerca de la aplicación.

Por un lado, nos comunicaron que resultaría útil que los perfiles de revisores albergaran la información de áreas de conocimientos y palabras clave para poder incluirlas en el buscador de la lista de revisores.

Por otro lado, dos encuestados nos comentaron que de momento no estarían dispuestos a pagar por incluir sus revisiones en la plataforma, aunque sí que estarían dispuestos en caso de que las editoriales gratificaran previamente a los revisores, asumiendo el coste de importar los datos a la blockchain.

Finalmente, a modo de resumen, podríamos identificar tres conclusiones principales con la conclusión de esta fase.

La primera es que todos los usuarios encuestados han reportado que un sistema de recompensas para los revisores es una idea muy acertada.

En segundo lugar, los usuarios que no conocían el concepto blockchain previamente, han mostrado mucho interés en esta tecnología y opinan que nuestra propuesta tiene mucho potencial.

Por último, concluimos que pese al gran interés mostrado por los usuarios y las buenas opiniones que hemos recibido por nuestra plataforma, los usuarios en general, pero sobre todo los revisores de artículos científicos no están preparados para utilizar las cuentas de

Ethereum ya que el proceso de instalación e iniciación con ellas es poco intuitivo y puede resultar algo tedioso.

Capítulo 8. Contribuciones al proyecto

8.1 Trabajo colaborativo

Tras una primera reunión con los directores del proyecto, nos encomendaron el objetivo de desarrollar un frontend con React, tecnología novedosa que utilizaríamos para conectarlo con los smart contracts de la blockchain. Estos contratos han sido desarrollados por un proyecto de TFG en paralelo al nuestro y han sido utilizados como backend de nuestra aplicación.[77] Como ninguno de los miembros del equipo teníamos conocimientos previos en estas tecnologías, así como en el diseño web, decidimos enfocar nuestros primeros esfuerzos en la investigación de React. Elegimos como punto de partida el tutorial que la propia librería ofrece en su página web[78]. Este ejemplo lo realizamos de manera individual para más tarde poner en común los problemas a los que nos habíamos enfrentado, así como las conclusiones que habíamos obtenido. El primer desarrollo consistió en un "tic-tac-toe" que enseñaba los principios fundamentales para poder trabajar con esta tecnología.

Una vez entendimos en mayor profundidad como construir aplicaciones en React, decidimos dar un salto y comenzamos a investigar el paradigma Blockchain. En este momento empezamos a informarnos sobre Ethereum y como había dado lugar a la creación de contratos inteligentes y las DApps. Finalmente dimos con un curso en el que se trataba en profundidad como desarrollar una aplicación descentralizada sin ningún conocimiento previo. En el curso aprendimos desde la programación de contratos en Solidity hasta la creación de un FrontEnd con React, pasando por su integración con Web3.js. Este desarrollo fue realmente útil para aprender algunas de las tecnologías que posteriormente tendríamos que utilizar para nuestra DApp, como la suite de Truffle o la extensión de navegador MetaMask. De nuevo, esta parte la realizamos los miembros del equipo individualmente con el objetivo de obtener un mayor conocimiento de las herramientas y sus posibilidades.

Después de finalizar el prototipo anteriormente mencionado, decidimos desarrollar a modo esquemático un proyecto utilizando los contratos que la aplicación final implementa. Con este objetivo, nos reunimos nuevamente para realizar los bocetos de como queríamos que fuera nuestra aplicación, basándonos en el proyecto anteriormente realizado y dividirnos el trabajo. En un principio la distribución del trabajo era de manera individual pero debido a nuestra falta de experiencia y a que nuestros contratos pertenecen a otro trabajo que se encontraba en desarrollo al mismo tiempo que el nuestro, se añadía la dificultad de adaptar el FrontEnd a los constantes cambios que recibían los smart contracts. Es por ello por lo que algunas partes tuvieron que ser reimplementadas y en muchos casos tuvimos que realizar peer programming, es decir, programar en simultaneo utilizando una pantalla.

Durante estas sesiones los tres miembros realizamos la página de revisiones. Desde el diseño de la página al registro de los datos en la blockchain.

En el desarrollo de la plataforma final, realizamos distintas iteraciones de diseño en las que cada uno participamos con nuestros bocetos para ponerlos en común y decidir el diseño final de la aplicación. Además, algunas de las partes del desarrollo fueron realizadas realizando peer programming.

8.2 Trabajo individual

8.2.1 Alejandro Alarcón Ayllo

Cuando empezamos el proyecto, no tenía conocimientos previos en ninguno de los lenguajes utilizados en el desarrollo de este proyecto. Es por ello qué los primeros pasos que di fueron de investigación acerca de la tecnología React. Antes de empezar a investigar acerca de la interacción de React con Solidity, me di cuenta de que era necesario que le dedicará algo más de tiempo a este primer lenguaje, ya que no me había quedado del todo claro cómo funcionaba e íbamos a utilizarla durante todo el año, por lo que era necesario saber manejarla con soltura.

Para ello me formé viendo algunos tutoriales de primeros pasos utilizando React en YouTube[79] y preguntando las dudas que tenía a mis compañeros.

Posteriormente, investigué acerca de la tecnología de los Smart Contracts, Solidity. Personalmente, sí que tenía algo de conocimiento acerca de la tecnología blockchain, ya que hace unos años participé en una comunidad de inversores y me resultó realmente interesante el concepto. Fue entonces cuando decidí informarme acerca de ella.

Sabiendo que iba a realizar el TFG sobre tecnologías blockchain, tomé la decisión de matricularme, junto con mis compañeros, a la asignatura Introducción a la Tecnología Blockchain y Smart Contracts, que se impartía por primera vez en la Facultad. No obstante, como no empezaba hasta el segundo cuatrimestre, en esta primera parte del proyecto me informé por mi cuenta sobre las bases y como crear contratos sencillos.

Una vez ya sabía cómo interaccionaban los contratos de Solidity con el frontend de React, me encargué de conectar el backend del prototipo inicial a uno de los tres botones del mismo. También me encargué del desarrollo de la página del listado de revisores, utilizando Web3.js para cargar su información.

En el proyecto final, en un principio decidimos reimplementar los contratos con los que ya partíamos para eliminar la dependencia que tenían de los papers y que solamente interactuaran con los revisores, también quisimos agregar al ERC-721 la extensión de metadatos, entre Fernando y yo hicimos modificaciones a los contratos para poder ajustarlo a nuestras necesidades pero, según íbamos creando funciones, nos dimos cuenta que el ERC-721 con el que partíamos había sido modificado, e implementar la extensión y eliminar la dependencia suponía reescribir todos los contratos al completo, por lo que,

tras una reunión de equipo, descartamos la idea ya que no era la competencia real de nuestro proyecto y nos iba a llevar bastante tiempo. Al final, realizamos una serie de modificaciones explicadas en el punto '5.3.1 Modificaciones del contrato'.

Como decidimos implementar una base de datos al proyecto para poder tratar con los datos que no contábamos en los contratos, y yo había cursado la asignatura Bases de Datos NoSQL, investigué cual era la mejor opción para implementar en el proyecto. Tras hacer un curso que implementara un sistema de bases de datos[80], le presenté a mi grupo la decisión de adoptar MERN como sistema de almacenamiento de datos. La implementación y adaptación la realicé junto a Fernando, hasta dejar el servidor funcional, adaptado a nuestras necesidades y con una API que pudiéramos llamar donde la necesitáramos.

También me encargué de que la tabla de revisores hiciera las llamadas correctas a los contratos y a la base de datos, para mostrar los datos de todos los usuarios registrados, su reputación, y los awards que tienen.

Asimismo, me encargué de que la página de perfil funcionara correctamente, recogiendo todos los datos necesarios y mostrándolos en sus componentes. También implementé el sistema de validaciones en el registro de usuarios e importación de revisiones.

Participé también en la preparación de las encuestas y el guion de la entrevista que le hicimos a los usuarios que testearon nuestra aplicación.

Al final hice varias iteraciones al código del programa corrigiendo algunas partes del código que eran mejorables e implementando algunas funcionalidades visuales.

Del desarrollo de la memoria, me encargué de redactar los siguientes puntos:

- Capítulo 1: Antecedentes
- Capítulo 2: 'Plataformas de Revisiones por Pares', 'Tecnología Blockchain', 'Ethereum', 'Altcoins' y 'Contratos ERC y Tokens No Fungibles (NFT)'
- Capítulo 3: 'Trello', 'Enfoque de Software Libre' y 'Tecnologías' (Git, Ganache, Metamask, Web3, ReactJS, MongoDB)
- Capítulo 4
- Capítulo 5: 'Implementación' y 'Modificaciones del Contrato'

8.2.2 Carlos Rodríguez Hernández

Al igual que mis compañeros comencé investigando acerca de la tecnología React. Debido a mis conocimientos previos en JavaScript, gracias a la asignatura “Desarrollo de Videojuegos mediante tecnologías Web” que había curso el año anterior, no me fue muy complicado entender la sintaxis de esta nueva biblioteca. Sin embargo, no tenía un conocimiento previo de Blockchain más allá de haber oído hablar de él y en concreto ninguno acerca de la plataforma Ethereum, así que decidí investigar un poco más acerca del tema.

En el desarrollo de Rewards MVP, me encargue de realizar la primera conexión de los contratos con la interfaz de usuario utilizando todo lo que había aprendido realizando la Red Social en Blockchain. Las primeras funcionalidades que implementé incluían generar nuevos papers, mostrarlos y la opción de darles una propina.

Tras recibir los nuevos cambios en los contratos, se me planteó el problema de generar más de una página en el Frontend. No me fue difícil dar con React Router, el cual nos permitió implementar nuevas páginas en el proyecto y en concreto la página de papers.

Con la integración del nuevo tipo de awards, me dediqué a la investigación del ERC-721 y para ello realicé un curso en YouTube qué más tarde me sirvió como idea para realizar la representación visual de los awards, tal y como se desarrolla en punto 5.7.

Más tarde quise investigar acerca de cómo funcionaban los NFTs reales y dónde se almacenan la información de los mismos. Tras ver cómo trabajan algunas webs acabé por descubrir IPFS, tecnología que aprendí a utilizar realizando otro curso. De esta manera en el desarrollo de la aplicación final me encargué de la subida de las imágenes a IPFS. Una vez terminado, comente con mis compañeros la posibilidad de modificar el contrato para de esta manera almacenar el hash que se generaba.

Tras haber realizado los bocetos de la plataforma final, me encargué de realizar una primera maquetación de la aplicación en React. Creé las plantillas de la pantalla principal, el perfil del revisor y el formulario de registro e importación de revisiones, que posteriormente fueron modificadas por el resto de los miembros en función de las necesidades que se fueron planteando.

Después de que mis compañeros implementarán la base de datos y realizarán las modificaciones pertinentes en el contrato, me encargué de la interacción con la red de Ethereum. El registro de los revisores y sus revisiones, así como rescatar los Rewards que han recibido.

Respecto al Popup de recompensas realicé su maquetación y desarrollo. Esto incluye realizar la plantilla inicial y las mejoras en su diseño. También me encargue de conectar los botones de recompensas con los contratos, así como la representación del resumen de los revisores, tanto de la blockchain como de la base de datos. Finalmente me encargué de la validación de los datos que el módulo maneja, así como de los mensajes de ayuda.

Una vez se finalizó la mayor parte del desarrollo me encargué de investigar cómo se publicaban paquetes en npm. Esta parte fue realmente tediosa puesto que tuve que intentar

muchas maneras de realizar la tarea hasta hallar con una que funcionase y se ajustase a lo que queríamos hacer. Además, realice su desarrollo y publicación.

En la fase de testing me encargue de la búsqueda de revisores para participar en las entrevistas y realice varias de ella. Adjunto mi aportación a la memoria por capítulos:

Participación en la memoria en los distintos capítulos:

- Capítulo 1: en conjunto con el resto del equipo.
- Capítulo 3: IPFS
- Capítulo 4: en conjunto con el resto del equipo.
- Capítulo 5: Especificación, Requisitos, Tipos de Usuario, Casos de Uso, Implementación en conjunto con el resto del equipo.
- Capítulo 6: en conjunto con el resto del equipo.
- Capítulo 7: en conjunto con el resto del equipo.

8.2.3 Fernando Ruiz García

En primer lugar, he realizado bastante trabajo de investigación. Durante la primera fase del proyecto, empecé informándome de las diferentes tecnologías que tendríamos que usar en el proyecto. Como punto de partida, empecé con React y Solidity, aunque nuestro trabajo se iba a desarrollar principalmente en la primera. Cuando fui profundizando en ellas, descubrí el concepto de las DApps y comencé a ver que, para utilizarlas correctamente, tendría que aprender a utilizar las herramientas de Ganache y Metamask.

En mi caso, todos los conceptos eran nuevos y nunca había trabajado con ninguna de estas tecnologías por lo que esta primera fase me fue muy útil para mí y me permitió poder empezar a implementar las diferentes partes del trabajo posterior.

En este momento, empezamos a desarrollar el Rewards MVP en el que inicialmente me encargué de investigar diferentes herramientas para la representación de avatares de los usuarios para poder utilizar en nuestro proyecto como los JDenticon[81] aunque fueron descartadas. Finalmente encontré los Identicon [82] y vi que se ajustaban a lo que estábamos buscando por lo que decidí que los implementaríamos.

El prototipo además permitía dar a los revisores tres tipos diferentes de recompensas, cada una por medio de un botón diferente. Aquí me encargué de hacer las conexiones de los contratos a uno de los botones. Para realizar esta conexión previamente tuve que aprender a cómo utilizar Web3.js[83]. Finalmente, realicé algunos cambios estéticos para pulir la interfaz.

Una vez terminado el desarrollo de este proyecto, empecé una nueva fase de investigación. Aunque inicialmente me había informado acerca de en qué consistía Solidity, y para que se utilizaba, realmente no había aprendido a utilizarlo más allá de algunos contratos sencillos de algún curso y que implementé en la IDE de Remix[84]. Por ello, en esta fase comencé a profundizar más en este lenguaje. Además, conté con la

ayuda de la asignatura de Introducción a la Tecnología Blockchain y los Smart Contracts en la facultad.

Coincidiendo con esto, en el equipo llegamos a la conclusión de que los contratos necesitaban algunos cambios, ya que tal y como estaban hechos, no se ajustaban a las necesidades que requería nuestra aplicación. Por ello, junto a mi compañero Alejandro, realicé los cambios en los contratos que creíamos pertinentes, creando un contrato nuevo con la base del que ya existía. Dentro del equipo, vimos que la cantidad de trabajo que requerían las modificaciones era muy grande y pertenecía al backend, por lo que decidimos no utilizarlo y centrarnos en el frontend.

Tras esto realicé una investigación de la extensión de metadatos URI para el estándar ERC-721 para de esta forma poder manejar los metadatos de los NFT, aunque requería mucho trabajo en el backend por lo que desechamos la idea.

Para la base de datos en MongoDB, tras la investigación de Alejandro, hicimos tanto la implementación como la adaptación a la plataforma de manera conjunta. Además, estuve investigando algún software que nos ayudara a manejarla en un primer momento, cuando no teníamos todavía desarrollado un frontend. Encontré dos herramientas que nos podrían ayudar mucho, la primera era Postman[85] y la segunda, Robo 3T[86].

En el desarrollo de la aplicación final, me he encargado principalmente del sistema de registro y el de importación de revisiones. Realicé las modificaciones en los formularios, creación de los elementos de ayuda para los usuarios y su conexión con la base de datos.

Posteriormente, me dediqué a hacer la funcionalidad de importar una revisión. De nuevo, me encargué de conectarla con la base de datos por medio de la API.

Por último, en la página web, creé la página de FAQs y realicé las diferentes preguntas con la ayuda de Alejandro.

Una vez se había acabado el desarrollo de la plataforma, quedaba la fase de testing con usuarios, en la que me encargué de crear el script para las entrevistas.

Finalmente, en la parte de la memoria, aunque se está realizando de forma colaborativa, me he encargado de redactar:

- Capítulo 1: Motivaciones, Objetivos y Alcance.
- Capítulo 2: Revisiones por pares, Tipos de Arquitectura, Bitcoin y sus orígenes.
- Capítulo 3: Introducción a las Metodologías Ágiles, SCRUM, Plan de trabajo, tecnologías.
- Capítulo 5: Dinero para revisores, Reputación (RPT) para revisores, Rewards (NFT) para revisores, Especificación del diseño, Implementación.
- Capítulo 6: Evaluación.
- Capítulo 7: Conclusiones, Trabajo Futuro.

Capítulo 9. Conclusiones y Trabajo Futuro

9.1 Conclusiones

Este trabajo ha supuesto la primera vez en la que todos los integrantes del equipo nos hemos tenido que enfrentar a un proyecto de estas dimensiones. En muchas partes de su desarrollo nos hemos encontrado con diferentes desafíos que hemos superado gracias a la cooperación entre nosotros. La principal conclusión que hemos obtenido a nivel personal es que ha sido una experiencia que nos ha hecho crecer en muchos aspectos, además de ser muy enriquecedora al permitirnos descubrir y trabajar con las distintas tecnologías tratadas en el proyecto que, resultan ser muy punteras y cada vez más extendidas.

Por otro lado, al principio del proyecto se nos plantearon una serie de objetivos para intentar cumplir durante todo el proceso. El principal objetivo que teníamos era desarrollar un frontend en React haciendo uso del ecosistema Ethereum, tal y como se ha expuesto en los capítulos cuatro, cinco y seis, podemos afirmar que se ha conseguido cumplir con las expectativas.

Asimismo, teníamos los objetivos de aprender a utilizar las tecnologías blockchain, Ethereum y las aplicaciones descentralizadas. Este objetivo también se ha logrado como ya se ha expuesto en los capítulos mencionados anteriormente.

Durante los últimos años, el uso de las herramientas que utilizan la tecnología blockchain ha ido en aumento. Cada vez es más habitual encontrar empresas y proyectos que implementan este tipo de tecnologías. Este creciente uso, ha hecho que hoy en día estas herramientas representen una alternativa real al uso de las tecnologías centralizadas.

Por otro lado, durante el desarrollo de este proyecto hemos podido observar las ventajas que React aporta en la creación de aplicaciones web. La declaración de componentes y la facilidad de comunicarse con la blockchain, gracias a librerías como Web3.js, hace que el uso de React sea popular en el desarrollo de aplicaciones descentralizadas.

Tras haber realizado las experimentaciones, hemos concluido que el sistema de recompensas podría ayudar a incentivar a los revisores a realizar su actividad con mejor desempeño. Esto se debe a la motivación que genera recibir recompensas a cambio de su esfuerzo. Asimismo, este sistema ha resultado ser rápido e intuitivo de utilizar.

Sin embargo, se ha encontrado que los NFTs no son lo suficientemente representativos para que se entienda su significado, por esto se debería diseñar unos tokens orientados a la revisión por pares, que faciliten su entendimiento e incentiven su uso.

Finalmente, la implementación del paquete NPM permite que otras plataformas de revisión por pares puedan incluir el sistema de recompensas en sus aplicaciones. De esta manera se facilita la integración de nuestro sistema en dichas plataformas.

Por todo esto, concluimos que el desarrollo de este proyecto utilizando estas tecnologías es viable, pero personalmente creemos que, si se implementase utilizando un sistema alternativo a la red blockchain tendría una mayor acogida por parte de los usuarios.

Sin embargo, es posible que en un futuro en el que la tecnología blockchain esté más extendida y normalizada, será más sencillo el acceso a este tipo de aplicaciones.

9.2 Trabajo Futuro

Tras haber acabado el desarrollo del proyecto y la fase de experimentación, existen una serie de funcionalidades y cambios que se podrían llevar a cabo en diferentes partes de la aplicación actual:

- Se podría mejorar el modo de importación de las revisiones a nuestra aplicación, utilizando la herramienta de Bloxberg la cual permitiría importar las revisiones de manera automática.
- Realizar un sistema de avisos que informe a los revisores de las recompensas que han recibido.
- En vez de utilizar los Identicon para representar los Awards, mejorar su diseño utilizando tokens diseñados específicamente para este concepto.
- Llevar a cabo un rediseño en el contrato para que se base con exactitud al estándar ERC 721.
- Añadir la posibilidad de buscar a los revisores por sus campos de conocimiento y palabras clave.
- Desplegar el proyecto en una testnet. Actualmente se ha desarrollado utilizando Ganache. Herramienta que, como se ha explicado anteriormente, permite crear una blockchain local y experimentar rápida y fácilmente con ella. Una testnet, es una copia casi idéntica de una blockchain pública (mainnet), pero que trabaja con criptodivisas sin valor real y cualquiera puede acceder a la información. Gracias a esto, permite la experimentación con unas condiciones muy similares a las de la mainnet.

Chapter 9. Conclusions and Future Work

9.1 Conclusions

This work has been the first time that all the members of the team have had to face a project of this size. In many parts of its development we have encountered different challenges that we have overcome thanks to the cooperation between us. The main conclusion we have drawn on a personal level is that it has been an experience that has made us grow in many aspects, as well as being very enriching as it has allowed us to discover and work with the different technologies dealt with in the project, which, as it turns out, are very cutting-edge and increasingly widespread.

On the other hand, at the beginning of the project we were given a series of objectives to try to achieve throughout the process. The main objective we had was to develop a React frontend using the Ethereum ecosystem, and as we have seen in chapters four and five, we can affirm that we have managed to meet our expectations.

We also had the objective of learning how to use blockchain technologies, Ethereum and decentralized applications. This objective has also been achieved as already stated in the chapters mentioned above.

Over the last few years, the use of tools using blockchain technology has been increasing. It is increasingly common to find companies and projects that implement this type of technology. This growing use has meant that today these tools represent a real alternative to the use of centralized technologies.

On the other hand, during the development of this project we have been able to observe the advantages that React brings to the creation of web applications. The declaration of components and the ease of communication with the blockchain, thanks to libraries such as Web3.js, makes the use of React popular in the development of decentralized applications.

After having carried out the experimentations, we have concluded that the reward system could help to incentivise reviewers to perform better. This is due to the motivation to receive rewards in return for their effort. The system has also been found to be quick and intuitive to use.

However, it has been found that the NFTs are not sufficiently representative for their meaning to be understood, which is why tokens oriented to peer review should be designed to facilitate their understanding and encourage their use.

Finally, the implementation of the NPM package allows other peer review platforms to implement the reward system in their applications. This facilitates the integration of our system into these platforms.

For all these reasons, we conclude that the development of this project using these technologies is feasible, but we personally believe that if it were implemented using an alternative system to the blockchain network, it would be more widely accepted by users.

However, it is possible that in a future in which blockchain technology is more widespread and standardised, it will be easier to access this type of application.

9.2 Future Work

After having finished the project development and the phase explained in chapter 6, there are a number of functionalities and changes that could be made to different parts of the current application:

- The way revisions are imported into our application could be improved by using the Bloxberg tool which would allow revisions to be imported automatically.
- Set up a notification system that informs reviewers of the rewards they have received.
- Instead of using the Identicons to represent the Awards, improve their design by using tokens specifically designed for this concept.
- Carry out a redesign of the contract to be accurately based on the ERC 721 standard.
- Add the possibility to search for reviewers by their fields of expertise and keywords.
- Deploy the project in a testnet. Currently it has been developed using Ganache. A tool that, as explained above, allows you to create a local blockchain and experiment quickly and easily with it. A testnet is an almost identical copy of a public blockchain (mainnet), but it works with cryptocurrencies with no real value and anyone can access the information. Thanks to this, it allows experimentation with conditions very similar to those of the mainnet.

Bibliografía

[1] M. Tamayo y Tamayo, *El Proceso de la Investigación Científica*. Limusa, México: Editorial Limusa S.A, 2004.

[2] J. González Álvarez, "EL MÉTODO CIENTÍFICO NACE CON LA EDAD MODERNA."

[3] "ORCID." https://orcid.org/ (accessed May 21, 2021).

[4] "Publons." https://publons.com/about/home/ (accessed May 21, 2021).

[5] "¿Qué es un token y cómo funciona?" https://es.cointelegraph.com/explained/what-is-a-token-and-how-does-it-work (accessed Jun. 09, 2021).

[6] R. Kamath, "Food Traceability on Blockchain: Walmart's Pork and Mango Pilots with IBM," *The Journal of the British Blockchain Association*, vol. 1, no. 1, pp. 1–12, Jul. 2018, doi: 10.31585/jbba-1-1-(10)2018.

[7] N. Hackius and M. Petersen, "Blockchain in Logistics and Supply Chain: Trick or Treat?," *Reinforced Plastics*, vol. 9783745043, no. April, p. 23, 2017, doi: 10.15480/882.1444.

[8] "TradeLens | Digitizing Global Supply Chains." https://www.tradelens.com/ (accessed May 23, 2021).

[9] "20 Empresas que están implementando la tecnología blockchain." https://101blockchains.com/es/empresas-implementando-blockchain/ (accessed May 23, 2021).

[10] G. J. Katuwal, S. Pandey, M. Hennessey, and B. Lamichhane, "Applications of Blockchain in Healthcare: Current Landscape & Challenges," *arXiv*, Dec. 2018, Accessed: May 23, 2021. [Online]. Available: http://arxiv.org/abs/1812.02776

[11] A. Ferrer-Sapena and E.-A. Sánchez-Pérez, "Aplicaciones de la tecnología blockchain en la documentación científica: situación actual y perspectivas Applications of blockchain technology in scientific documentation: current situation and perspectives," pp. 1699–2407, 2019, doi: 10.3145/epi.2019.mar.10.

[12] M. Ladrón De Guevara Cervera, J. Hincapié, J. Jackman, O. Herrera, C. Vinicio, and C. Uribe, "Revisión por pares: ¿Qué es y para qué sirve? Peer Review: what it's and what it's for?," *Barranquilla (Col.)*, vol. 24, no. 2, pp. 258–272, 2008.

[13] "Vista de La revisión por pares ('peer review') en las revistas científicas: un proceso que requiere intervención."

https://revistasum.umanizales.edu.co/ojs/index.php/tempuspsi/article/view/3410/5973 (accessed May 21, 2021).

[14] "F1000 | Services for Researchers, Institutions & Research Funders." https://f1000.com/ (accessed May 21, 2021).

[15] "PeerJ." https://peerj.com/ (accessed May 21, 2021).

[16] "How it works? - Peereviewers.com." https://peereview.jimdofree.com/ (accessed May 21, 2021).

[17] "Bitcoin - Dinero P2P de código abierto." https://bitcoin.org/es/ (accessed May 21, 2021).

[18] "¿Conoces la historia de las criptomonedas?" https://www.daviescoin.io/es/blog/conoces-la-historia-de-las-criptomonedas (accessed May 21, 2021).

[19] U. W. Chohan, "A History of Bitcoin," 2017.

[20] "Minería Bitcoin ¿Cómo se crea un bloque? | Bit2Me Academy." https://academy.bit2me.com/mineria-bitcoin-como-se-crea-un-bloque/ (accessed Jun. 03, 2021).

[21] S. Nakamoto, "Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer."

[22] "VENTAJAS Y DESVENTAJAS. Blockchain | Federico Camargo." https://camargo.life/blockchain-ventajas-y-desventajas/ (accessed Jun. 12, 2021).

[23] "Cambridge Bitcoin Electricity Consumption Index (CBECI)." https://cbeci.org/ (accessed May 21, 2021).

[24] "ethereum/serpent." https://github.com/ethereum/serpent (accessed May 21, 2021).

[25] "Solidity — documentación de Solidity." https://solidity-es.readthedocs.io/es/latest/index.html (accessed May 21, 2021).

[26] "Decentralized Applications: The good, the bad, and why should enterprises care?" https://valid.network/post/decentralized-applications-the-good-the-bad-and-why-should-enterprises-care (accessed Jun. 12, 2021).

[27] "Global Payment Solutions - Instant Processing | Ripple." https://ripple.com/ (accessed May 21, 2021).

[28] D. Schwartz, N. Youngs, and A. Britto, "The Ripple Protocol Consensus Algorithm." Accessed: May 21, 2021. [Online]. Available: https://arxiv.org/abs/1802.07242

[29] “¿Qué es Ripple? Todo lo que necesitas saber.” https://es.cointelegraph.com/ripple-101/what-is-ripple (accessed May 21, 2021).

[30] “Lumens - Stellar.” https://www.stellar.org/lumens?locale=es (accessed May 21, 2021).

[31] “¿Qué es Stellar (XLM)? | Bit2Me Academy.” https://academy.bit2me.com/que-es-stellar-xml-lumens-criptomoneda/ (accessed May 21, 2021).

[32] “Cardano | Home.” https://cardano.org/ (accessed May 21, 2021).

[33] “¿Qué es Cardano (ADA)? | Bit2Me Academy.” https://academy.bit2me.com/que-es-cardano-ada/ (accessed May 22, 2021).

[34] “ERC Token Standards - EthHub.” https://docs.ethhub.io/built-on-ethereum/erc-token-standards/what-are-erc-tokens/ (accessed May 22, 2021).

[35] “ERC | Ethereum Improvement Proposals.” https://eips.ethereum.org/erc (accessed May 22, 2021).

[36] “EIP-20: ERC-20 Token Standard.” https://eips.ethereum.org/EIPS/eip-20 (accessed May 22, 2021).

[37] “EIP-721: ERC-721 Non-Fungible Token Standard.” https://eips.ethereum.org/EIPS/eip-721 (accessed May 22, 2021).

[38] “Beeple (b. 1981), EVERYDAYS: THE FIRST 5000 DAYS | Christie’s.” https://onlineonly.christies.com/s/beeple-first-5000-days/beeple-b-1981-1/112924 (accessed May 22, 2021).

[39] “CryptoKitties | Collect and breed digital cats!” https://www.cryptokitties.co/ (accessed May 22, 2021).

[40] U. W. Chohan, “Non-Fungible Tokens: Blockchains, Scarcity, and Value,” 2021.

[41] “CryptoSushis | NFT collectibles | Limited Cryptoart.” https://cryptosushis.com/ (accessed May 22, 2021).

[42] “OpenSea: Buy NFTs, Crypto Collectibles, CryptoKitties, Decentraland, and more on Ethereum.” https://opensea.io/ (accessed May 22, 2021).

[43] “AtomicHub - Interface for the EOSIO AtomicAssets NFT standard.” https://eos.atomichub.io/ (accessed May 22, 2021).

[44] “EIP-1155: ERC-1155 Multi Token Standard.” https://eips.ethereum.org/EIPS/eip-1155 (accessed May 22, 2021).

[45] “Proyecto de ‘token semi-fungible’ se lanza en Binance Smart Chain.” https://es.cointelegraph.com/news/semi-fungible-token-project-launches-on-binance-smart-chain (accessed May 22, 2021).

[46] A. N. Cadavid, J. Daniel Fernández Martínez, and J. Morales Vélez, “Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development.”

[47] M. Trigas Gallego, “Metodología Scrum.”

[48] H. A. Johnson, “Trello,” *Journal of the Medical Library Association*, 2017, doi: 10.5195/jmla.2016.49.

[49] J. Roche, “Kanban vs. Scrum.” https://www2.deloitte.com/es/es/pages/technology/articles/kanban-vs-scrum.html (accessed May 19, 2021).

[50] R. M. Stallman, “Software libre para una sociedad libre,” 2004.

[51] “Ventajas y desventajas del Software libre para su implementación en la escuela cubana. - Informática Jurídica.” http://www.informatica-juridica.com/trabajos/ventajas-y-desventajas-del-software-libre-para-su-implementacion-en-la-escuela-cubana/ (accessed Jun. 12, 2021).

[52] C. Juárez, M. Gómez Herrera, W. Guadalupe Torres Sánchez, and S. México, “Software libre vs software propietario Ventajas y desventajas,” 2006.

[53] “GitHub Desktop Documentation - GitHub Docs.” https://docs.github.com/en/desktop (accessed May 20, 2021).

[54] “What is Ethereum? — Ethereum Homestead 0.1 documentation.” https://ethdocs.org/en/latest/introduction/what-is-ethereum.html (accessed May 20, 2021).

[55] “Ganache | Truffle Suite.” https://www.trufflesuite.com/ganache (accessed May 20, 2021).

[56] K. Bhosale, K. Akbarabbas, J. Deepak, and A. Sankhe, “Blockchain based Secure Data Storage,” *International Research Journal of Engineering and Technology*, vol. 5058, 2008, Accessed: May 20, 2021. [Online]. Available: www.irjet.net

[57] “React – Una biblioteca de JavaScript para construir interfaces de usuario.” https://es.reactjs.org/ (accessed May 20, 2021).

[58] “Introduction - Semantic UI React.” https://react.semantic-ui.com/ (accessed May 20, 2021).

[59] “CryptoZombies | Lección 6 Capítulo 1.” https://cryptozombies.io/es/lesson/6/chapter/1 (accessed May 20, 2021).

[60] “JSON-RPC 2.0 Specification.” https://www.jsonrpc.org/specification (accessed May 20, 2021).

[61] “MERN Stack: Qué es y qué ventajas ofrece | OpenWebinars.” https://openwebinars.net/blog/mern-stack-que-es-y-que-ventajas-ofrece/ (accessed May 20, 2021).

[62] “La base de datos líder del mercado para aplicaciones modernas | MongoDB.” https://www.mongodb.com/es (accessed May 20, 2021).

[63] “Express - Infraestructura de aplicaciones web Node.js.” https://expressjs.com/es/ (accessed May 20, 2021).

[64] “IPFS Powers the Distributed Web.” https://ipfs.io/ (accessed May 20, 2021).

[65] “Ethereum API | IPFS API & Gateway | ETH Nodes as a Service | Infura.” https://infura.io/ (accessed May 20, 2021).

[66] B. Munster, “People’s Expensive NFTs Keep Vanishing. This Is Why.” https://www.vice.com/en/article/pkdj79/peoples-expensive-nfts-keep-vanishing-this-is-why (accessed Jun. 13, 2021).

[67] Y. Napora and M. Wagner, “Storing NFTs on IPFS.” https://blog.ipfs.io/2021-04-05-storing-nfts-on-ipfs/ (accessed Jun. 13, 2021).

[68] “Metadata Standards.” https://docs.opensea.io/docs/metadata-standards (accessed Jun. 13, 2021).

[69] “Blockchain Tutorial For Developers: Step-By-Step Guide (Ethereum, Solidity, Web3.js) - YouTube.” https://www.youtube.com/watch?v=nvw27RCTaEw&t=692s (accessed Jun. 09, 2021).

[70] S. Lord, “stewartlord/identicon.js: GitHub-style identicons as PNGs or SVGs in JS.” https://github.com/stewartlord/identicon.js/tree/master (accessed May 23, 2021).

[71] “Rewards/contracts at mvp · DecentralizedScience/Rewards · GitHub.” https://github.com/DecentralizedScience/Rewards/tree/mvp/contracts (accessed Jun. 03, 2021).

[72] “Notability by Ginger Labs.” https://www.gingerlabs.com/ (accessed May 18, 2021).

[73] “Balsamiq Wireframes - Industry Standard Low-Fidelity Wireframing Software | Balsamiq.” https://balsamiq.com/wireframes/

[74] “F1000Research | Open Access Publishing Platform | Beyond a Research Journal.” https://f1000research.com/ (accessed May 18, 2021).

[75] “Marak/faker.js: generate massive amounts of realistic fake data in Node.js and the browser.” https://github.com/marak/Faker.js/ (accessed May 21, 2021).

[76] “Blockchain Infrastructure for Scientific Research - Bloxberg.” https://bloxberg.org/ (accessed May 18, 2021).

[77] “Rewards/contracts at mvp · DecentralizedScience/Rewards · GitHub.” https://github.com/DecentralizedScience/Rewards/tree/mvp/contracts (accessed Jun. 03, 2021).

[78] “Tutorial: Introducción a React – React.” https://es.reactjs.org/tutorial/tutorial.html (accessed May 21, 2021).

[79] V. Robles, “Curso React JS en Español Tutorial de React.js desde cero Instalación, Componentes, JSX y más,” 2019. https://www.youtube.com/watch?v=vAvCcjSAGDY (accessed May 21, 2021).

[80] “How to create your first MERN (MongoDB, Express JS, React JS and Node JS) Stack | by Sam Barros | The Startup | Medium.” https://medium.com/swlh/how-to-create-your-first-mern-mongodb-express-js-react-js-and-node-js-stack-7e8b20463e66 (accessed May 21, 2021).

[81] “Jdenticon - Open source identicon generator.” https://jdenticon.com/#quick-start-node-js (accessed May 22, 2021).

[82] “stewartlord/identicon.js: GitHub-style identicons as PNGs or SVGs in JS.” https://github.com/stewartlord/identicon.js/tree/master (accessed May 21, 2021).

[83] “web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation.” https://web3js.readthedocs.io/en/v1.3.4/ (accessed May 21, 2021).

[84] “Remix - Ethereum IDE.” https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.1+commit.df193b15.js (accessed May 22, 2021).

[85] “Download Postman | Try Postman for Free.” https://www.postman.com/downloads/ (accessed May 21, 2021).

[86] “Robo 3T | Free, open-source MongoDB GUI (formerly Robomongo).” https://robomongo.org/ (accessed May 21, 2021).

Apéndice A

Descarga y ejecución del código

El desarrollo de este proyecto se ha realizado íntegramente en Windows 10.

Lista de dependencias:

- Node
- Truffle suite
- Ganache or ganache-cli
- MetaMask
- Yarn
- MongoDB

Instalación:

Node

En primer lugar, es necesario tener instalado node.js y npm. Para instalarlo puede utilizar el siguiente enlace:

https://nodejs.org/es/

Truffle

Una vez instalado npm, utilice la consola para instalar truffle:

npm install -g truffle

Ganache

Todo el trabajo ha sido desarrollado con la versión de escritorio de Ganache, no obstante, si se prefiere, se puede emplear la versión de consola. Para descargar la versión de escritorio utilice este enlace:

https://www.trufflesuite.com/ganache

Si se prefiere la versión de consola:

npm install -g ganache-cli

MetaMask

Es necesario instalar la extensión de MetaMask para poder interactuar con la aplicación en el navegador. Los navegadores compatibles son Chrome, Firefox, Edge y Brave. Utilice el siguiente enlace para elegir su versión:

https://metamask.io/download.html

Una vez instalada la extensión, importe sus claves privadas de Ganache siguiendo los pasos que se indican en MetaMask. Ahora debemos conectarnos a la red de pruebas de Ganache. Para ello, despliegue el menú donde pone "Main Network" y seleccione "Custom RPC". En la casilla de "New RPC URL" introduce:

http://127.0.0.1:7545

En el ID de Cadena: 1337. Y escoja el nombre que más le guste. Para más información puede vistar este enlace: https://www.trufflesuite.com/docs/truffle/getting-started/truffle-with-metamask.

Yarn

Hemos utilizado yarn como gestor de paquetes, en un principio no sería necesario y con npm se podría realizar la misma función. Sin embargo, recomendamos el uso de yarn para evitar posibles problemas en la instalación de los módulos.

npm install --global yarn

MongoDB

Por último, es necesario tener instalado Mongo, para ello utilice:

https://www.mongodb.com/try/download/community

Una vez este corriendo MongoDB, descargue el código o clone el proyecto de github desde este enlace:

https://github.com/RevLounge/Dapp

Después de instalar las dependencias y descargar el código:

Desde la terminal de Visual Studio Code realice *yarn install* desde las siguientes carpetas:

- app
- popup

Desde /app, realice la migración de los contratos, ejecute:

truffle migrate

Es necesario copiar los json de los contratos de app/src/contracts -> popup/src/contracts

Desde app/server escriba en la terminal (si este comando le da problemas, una posible solución es tener yarn y npm en el path):

nodemon

Ya está todo listo para arrancar la aplicación, lance las aplicaciones en este orden:

- En localhost:3000, desde la carpeta de app -> yarn start
- En localhost:3001, desde la carpeta de popup -> yarn start