Recompensas en Blockchain: dinero, reputación y agradecimiento para revisores

Blockchain Rewards: Money, Badges and Reputation for peer reviewers

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA DE SOFTWARE
CURSO 2020–2021

Pablo Agudo Brun

Daniel Fidalgo Panera

Directores

Samer Hassan Collado

Ámbar Tenorio Fornés

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Madrid, Junio de 2021

Agradecimientos

Queremos agradecer a los directores del proyecto Samer Hassan Collado y Ámbar Tenorio Fornés su ayuda y apoyo durante el proceso de desarrollo de este proyecto. Y también a todos los desarrolladores de Software Libre y Código Abierto ya que sin su trabajo este proyecto no hubiera sido posible.

Pablo Agudo Brun

Lo imposible es el fantasma de los tímidos y el refugio de los cobardes.

Estas líneas son de las más importantes de mi vida y por fin llegó el momento de escribirlas.

Quiero empezar agradeciendo a la universidad Complutense de Madrid y a los profesores que me han formado estos duros pero inolvidables años.

Ahora me gustaría continuar dicendo lo mucho que he crecido gracias a la educación que mis padres me han impartido. Los valores de esfuerzo, sufrimiento y superación que tengo son gracias a ellos. Gracias también a mi compañera de vida que desde hace tantos años me apoya, y sin duda es para mi un pilar fundamental. Gracias abuelos porque sé y me hacéis saber lo mucho que significa para vosotros que haya conseguido esto. Por último y con especial cariño, gracias abuela María, t'estime.

No me olvido de mis compañeros y amigos de la universidad, han sido toda una motivación, y sobre todo gracias a mi más que compañero, mi amigo Daniel que me acompaña haciendo este trabajo. ¡Lo conseguimos!.

Daniel Fidalgo Panera

Llevamos un mundo nuevo en nuestros corazones.

Quiero agradecer a todas las personas que me han apoyado durante esta etapa, sobre todo a mi familia, en especial a mi madre y a mi hermana, por todo el apoyo recibido, y a mi pareja, por estar siempre ahí cuando más la he necesitado. También a mis amigos, sin los cuales el mundo sería muchísimo más aburrido. Este TFG se ha realizado durante una pandemia y bajo unas condiciones que distan mucho de las ideales, y en las que se han ido personas a las que amamos, aún así hemos peleado y estoy orgulloso de lo que hemos conseguido. Durante mi estancia en la universidad he ampliado mis horizontes, así como mi conocimiento, y he tenido la oportunidad de crecer como persona. Este proyecto no hubiera sido posible sin la constancia y el trabajo de mi compañero de carrera y de TFG, gracias Pablo.

Resumen

Hoy en día para publicar un artículo académico es necesario seguir una serie de trámites. Uno de estos es la revisión del artículo. Las personas encargadas de esta tarea no reciben ningún tipo de visibilidad ni compensación por el trabajo realizado. Para corregir esta situación, en este proyecto, se ha implementado un sistema de recompensas descentralizado, utilizando blockchain para recompensar a los revisores de artículos de investigación. Estas recompensas se dan en forma de transferencia de criptomonedas ETH, en forma de reputación o como premios digitales no transferibles basados en NFTs (*Non-fungible tokens*).

Al utilizar la blockchain de Ethereum nos aprovechamos de su ecosistema para desarrollar una plataforma descentralizada, transparente y sin intermediarios.

El sistema engloba una web desarrollada con HTML, CSS, JavaScript y el framework React, y a esta web se conecta una serie de Smart Contracts desarrollados sobre la red de Ethereum con Solidity.

Para la realización de este proyecto ha sido necesario llevar a cabo experimentaciones sobre el desarrollo de un Smart Contract, la implementación y modificación de tokens en Ethereum, el despliegue en una blockchain y el coste que supondría interactuar con el Smart Contract.

Como trabajo futuro, se ha considerado principalmente añadir soporte para poder subir los artículos a la aplicación mediante IPFS, desplegar los Smart Contracts en la red de Ethereum, implementar pruebas en el front-end mediante Selenium o Cypress y desplegar la aplicación en un servidor.

Palabras clave: Blockchain, Smart contract, Ethereum, ERC20, ERC721, NFTs.

Abstract

Nowadays, in order to publish an academic article, it is necessary to follow a series of steps. One of these is the review of the article. The people in charge of this task do not receive any kind of visibility or compensation for the work done. To correct this situation, in this project, a decentralized reward system has been implemented, using blockchain to reward reviewers of research articles. These rewards are given in the form of ETH cryptocurrency transfer, in the form of reputation or as non-transferable digital rewards based on NFTs (non-fungible tokens).

Using Ethereum's blockchain we take advantage of its ecosystem to develop a decentralized, transparent and unmediated platform.

The system encompasses a website developed with HTML, CSS, JavaScript and the React framework, and to this website is connected a series of Smart Contracts developed over the Ethereum network with Solidity.

For the realization of this project it has been necessary to carry out experiments on the development of a Smart Contract, the implementation and modification of tokens in Ethereum, the deployment on a blockchain and the cost of interacting with the Smart Contract.

As future work, we have mainly considered adding support to be able to upload items to the application via IPFS, deploy Smart Contracts on the Ethereum network, implement front-end testing via Selenium or Cypress and deploy the application on a server.

Key words: Blockchain, Smart contract, Ethereum, ERC20, ERC721, NFTs.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, antes de publicarse, los artículos académicos pasan por un proceso de revisión, donde otros expertos, los revisores, evalúan la calidad y validez de dichos documentos. Sin embargo, las tareas realizadas por estos revisores no se ven compensadas económica ni profesionalmente, puesto que los artículos individualizados que han revisado no quedan plasmados en su currículum.

El proceso queda centralizado por la revista publicadora dejando sin visibilidad el trabajo realizado por los revisores. Esta parte del proceso de publicación es poco transparente y en ocasiones no se cumplen las condiciones idóneas.

Para solventar la problemática alrededor de la centralización se ha planteado como alternativa el uso de blockchain puesto que permite la implementación de un sistema descentralizado. La tecnología blockchain facilita la implementación de sistemas económicos y de reputación online descentralizados, es decir, no controlados o controlables por ninguna entidad central, a la vez que es altamente transparente y libre de intermediarios. Esto ofrece nuevas posibilidades para recompensar a los revisores por su trabajo.

1.2. Objeto de la Investigación

El objetivo de este proyecto es ofrecer una alternativa basada en Ethereum al proceso actual de revisiones, de la manera más descentralizada y transparente posible. Generando un sistema de reputación el cual permita recompensar, agradecer y donar a estos revisores por cada revisión que realicen, de esta manera se ve recompensado su trabajo. A la vez abstrayendo la complejidad de la implementación subyacente para que cualquier persona que no tenga conocimientos de blockchain pueda utilizar la aplicación.

1.3. Estructura de la memoria

Este trabajo está compuesto por un total de nueve capítulos, a continuación se definen brevemente el contenido de cada uno:

- Capítulo 1 Introducción. Se define la motivación del proyecto y el objeto de la investigación.
- Capítulo 2 Introduction. Traducción a inglés del Capítulo 1.
- Capítulo 3 Estado del arte. Presenta el contexto tecnológico y el estado del arte.
- Capítulo 4 Metodologías y tecnologías. Trata las metodologías de desarrollo y las herramientas usadas en el proyecto.
- Capítulo 5 Resultados. Se describe cómo se han obtenido los resultados finales del proyecto.
- Capítulo 6 Experimentación. Se tratan los diferentes experimentos y estudios relacionados con la generación y despliegue de los Smart Contracts.
- Capítulo 7 Aportaciones Individuales. Se indican las contribuciones individuales de cada componente del equipo.
- Capítulo 8 Conclusiones y Trabajo Futuro. Se muestran las conclusiones del trabajo realizado y líneas futuras de investigación.
- Capítulo 9 Conclusions and Future Work. Traducción a inglés del Capítulo 8.

1.4. Repositorio

El código del proyecto es Software libre y está disponible públicamente en Github, `https://github.com/DecentralizedScience/Rewards`.

Capítulo 2

Introduction

2.1. Motivation

Currently, before being published, academic articles go through a review process, where other experts, the reviewers, evaluate the quality and validity of these documents. However, the tasks performed by these reviewers are not compensated financially or professionally, since the individual articles they have reviewed are not included in their curriculum vitae.

The process is centralized by the publishing journal, leaving the work performed by the reviewers without visibility. This part of the publication process is not very transparent and sometimes the ideal conditions are not met.

To solve the problem of centralization, the use of blockchain has been proposed as an alternative, since it allows the implementation of a decentralized system. Blockchain technology facilitates the implementation of decentralized economic and online reputation systems, i.e. not controlled or controllable by any central entity, while being highly transparent and free of intermediaries. This offers new possibilities for rewarding reviewers for their work.

2.2. Object of the Investigation

The aim of this project is to offer an Ethereum-based alternative to the current review process, in the most decentralised and transparent way possible. By generating a reputation system which allows these reviewers to be rewarded, thanked and donated for each review they perform, thus rewarding their work. At the same time abstracting the complexity of the underlying implementation so that anyone with no knowledge of blockchain can use the application.

2.3. Proyect Scructure

This proyect is made up of a total of nine chapters, the content of each one is briefly defined below:

- Chapter 1. The motivation of the project and the object of the research are defined.
- Chapter 2. English translation of Chapter 1.
- Chapter 3. Presents the technological context and the state of the art.
- Chapter 4. Covers development methodologies and tools used in the project.
- Chapter 5. It describes how the final results of the project have been obtained.
- Chapter 6. The different experiments and studies related to the generation and deployment of Smart Contracts are discussed.
- Chapter 7. The individual contributions of each team member are indicated.
- Chapter 8. The conclusions of the work carried out and future lines of research are shown.
- Chapter 9. English translation of Chapter 8.

2.4. Repository

The project code is Free Software and it is publicly available on Github, `https://github.com/DecentralizedScience/Rewards`.

Capítulo 3

Estado del arte

3.1. Estado del arte

Es difícil encontrar plataformas descentralizadas para la gestión del proceso de revisión de artículos científicos, ya que las revistas en las que se suelen publicar se oponen firmemente a cualquier cambio que vea amenazado su férreo control. Aún así lentamente la situación ha ido cambiando y hay algunas portales como eLife, F1000Research, Royal Society Open Science, que han empezado a hacer este proceso más transparente, aunque el objetivo final de los autores es que se publiquen las revisiones junto con el artículo publicado [1]. Cabe destacar la plataforma Publons [2], donde se recogen métricas de autores y revisores para potenciar y reconocer el trabajo de los mismos. Frente a las plataformas centralizadas hay alternativas, entre las que destacan:

- Principia [3] es un framewrok que intenta solucionar esta situación creando un mercado para autores y revisores, donde estos últimos se vean recompensados según la calidad de su trabajo, para ello proponen un sistema de reputación que no dependa de terceras partes. En esta plataforma los autores hacen pujas para obtener revisiones [4].

Current peer-review system	Principia peer-review system	Consequences
Referees review (mostly) for free	Referees review and are remunerated	Power and money more balanced between researchers and publishers
Referees have often no incentives for quality reviews	Referees remuneration depends on quality reviews	Referees are motivated to write good reviews
New journals are difficult and expensive to start	New journals are easy and free to start	More genuine competition between journals. Journals get naturally more specialised
Hard to bootstrap a new journal's reputation	A new journal inherits the reputation of its founders	Journals not achieving their mission, or high quality, will die out automatically
Open access journals desirable, but often very expensive	All journals are open access and very cost-effective	Dramatic reduction of costs to access research output ad higher impact of research content

Figura 3.1: Principia comparation table [3]

- ScienceRoot es otra plataforma muy similar a Principia donde los usuarios también pueden recompensar y en cierto modo ser patrones de los investigadores. ScienceRoot propone una nueva forma de realizar el proceso de investigación y revisión, donde cada paso está registrado y es reproducible por otros investigadores. Así mismo, se podría almacenar los datos, publicándolos cuando los autores crean conveniente, parecido a como plataformas de control de versiones como GitHub o Gitlab funcionan. Además proponen la creación de un token llamado Science Token(ST) que serviría para sufragar en incentivar las interacciones dentro de la plataforma [5].

Problem	Solution
Researchers don't receive any money from their published work	Users will be able to reward them
Negative results are not published	Regardless of the outcome, scientific results may be published and stored
Concerns about the reproducibility of published results	Traceable "Blockchainified" repositories
Long waiting times until articles get reviewed and not enough reviewers	Process will be sped up since reviewers will get paid
Divided platforms for different purposes	One ecosystem containing all the platforms a scientist needs
Dispersed funding organizations	Centralized platform containing all necessary informations

Figura 3.2: Scienceroot comparation table [6]

- Decentralized Science [7] busca ofrecer un repositorio público de revisiones abiertas [8] y una red de reputación para revisores. En esta plataforma los revisores obtendrían recompensas y reconocimiento por su labor, y los autores podrían encontrar los mejores revisores dependiendo de su reputación [9, 10].

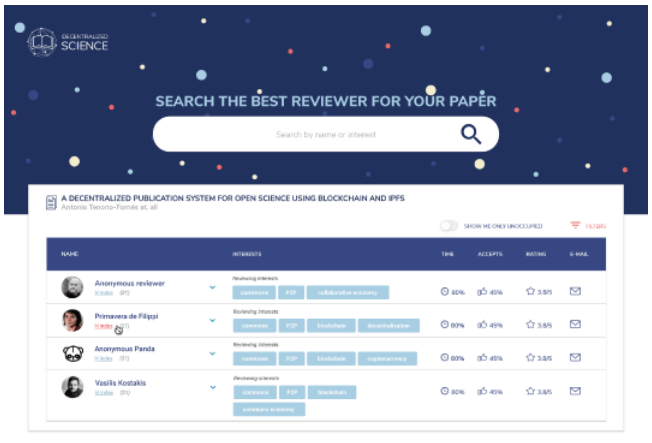

Figura 3.3: Decentralized Science [7]

3.2. Contexto Tecnológico

3.2.1. Blockchain

Satoshi Nakamoto revolucionó internet cuando publicó en 2008 un artículo titulado *"Bitcoin: A Peer-to-Peer Electronic Cash System"* [11], en él detalla un nuevo sistema para procesar transacciones financieras sin depender de una autoridad central. Este sistema se basa en una red p2p utilizando *Proof of Work* para recoger todas las transacciones que se realizan en un registro público. Este registro se llama Blockchain o cadena de bloques, y al estar replicado en todos los nodos está descentralizado y además es resistente a bloqueos y censura, ya que mientras exista al menos un nodo el sistema sigue funcionando. *Proof of Work* es un sistema de consenso que se basa en la obtención de un hash que comience con un número determinado de ceros, la obtención del hash es una tarea muy intensa que requiere mucha potencia de cálculo, por lo que hay personas (los mineros) que tienen hardware dedicado a esta tarea. Obtienen beneficio a través de las comisiones de las transacciones y de nuevos bitcoins generados en cada bloque. El número de ceros aumenta progresivamente según aumenta la capacidad de minado global (*Hashrate*) para mantener una generación de bloques constante. Los bloques son inmutables y su estructura se basa en árboles de Merkle, donde cada bloque contiene el hash del anterior, y al igual que en Git [12], al modificar un bloque se alteran todos los posteriores, lo que otorga resistencia a la modificación por actores con malas intenciones. La convención establecida es que la cadena más larga es la válida. En la siguiente imagen se puede ver como una transacción modifica el estado anterior de la blockchain para generar un estado nuevo.

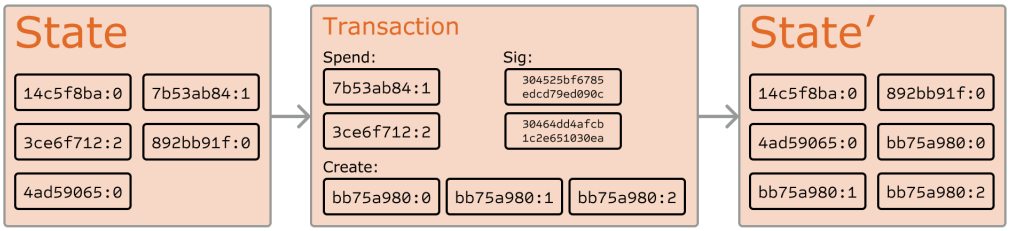

Figura 3.4: State Transition [13]

3.2.2. Smart Contracts

Los Smart Contracts o Contratos Inteligentes [14] son un conjunto de instrucciones software que implementan de forma digital las reglas y acciones de un contrato o acuerdo entre dos o más partes. Las máquinas expendedoras son el primer ejemplo, ya que permiten adquirir un bien o servicio sin mediación de intermediarios. El contrato inteligente fue propuesto por primera vez por el criptógrafo Nick Szabo en 1994, sólo cinco años después de la creación de la *World Wide Web*. Según la definición de Szabo [15], cuando se activa una condición preprogramada, el contrato inteligente ejecutará las condiciones contractuales correspondientes. La tecnología Blockchain nos proporciona un sistema descentralizado, resistente a la manipulación y altamente fiable en el que los contratos inteligentes son muy útiles.

3.2.3. Blockchain 1.0

Partiendo de Bitcoin, la tecnología blockchain se aplicaba inicialmente a criptomonedas, facilitando formas de pago anónimas, seguras y descentralizadas. Las criptomonedas se popularizaron con Bitcoin pero se basan en el antecedente de Ecash creado por David Chaum en 1983 [16], que utilizaba firmas digitales ciegas para garantizar la seguridad de las transacciones

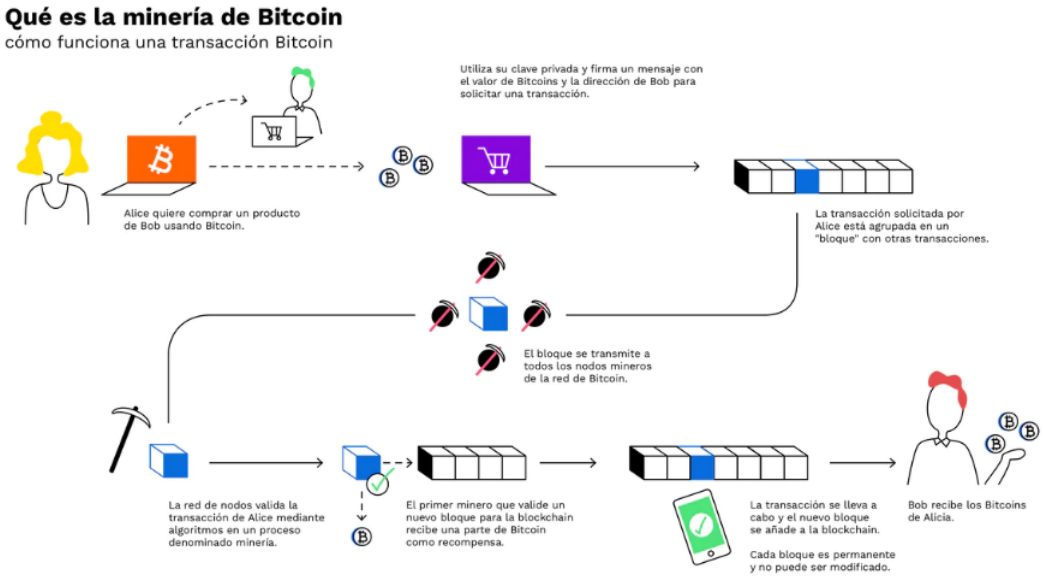

Figura 3.5: Bitcoin Flow [17]

3.2.4. Blockchain 2.0

Ethereum lideró una revolución en la tecnología Blockchain en forma de Smart Contracts. Ethereum fue concebido por Vitalik Buterin en 2013 y fue lanzado en 2015 después de una campaña de *crowdfunding* el año anterior. Gavin Wood escribió el Yellow Paper [18] de Ethereum donde se detalla como funciona la Máquina Virtual [19], sobre la que se ejecuta el código de los Smart Contracts. Estos Contratos inteligentes permiten crear Organizaciones Descentralizadas Autónomas o DAOs [20, 21], estas organizaciones funcionan de forma totalmente descentralizada sobre la red de Ethereum sin estar influidas por un poder centralizado, y donde las reglas se definen en los propios contratos. Un ejemplo de este tipo de organizaciones lo encontramos en MakerDAO, compuesta por los propietarios del token MKR(Un token es "una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas". Extraído del libro "The business blockchain"[22]), donde ellos mismos pueden votar propuestas para cambiar las reglas por las cuales se regula DAI [23], una moneda estable vinculada al precio del Dólar Estadounidense.

3.2.5. Blockchain 3.0 y futuro

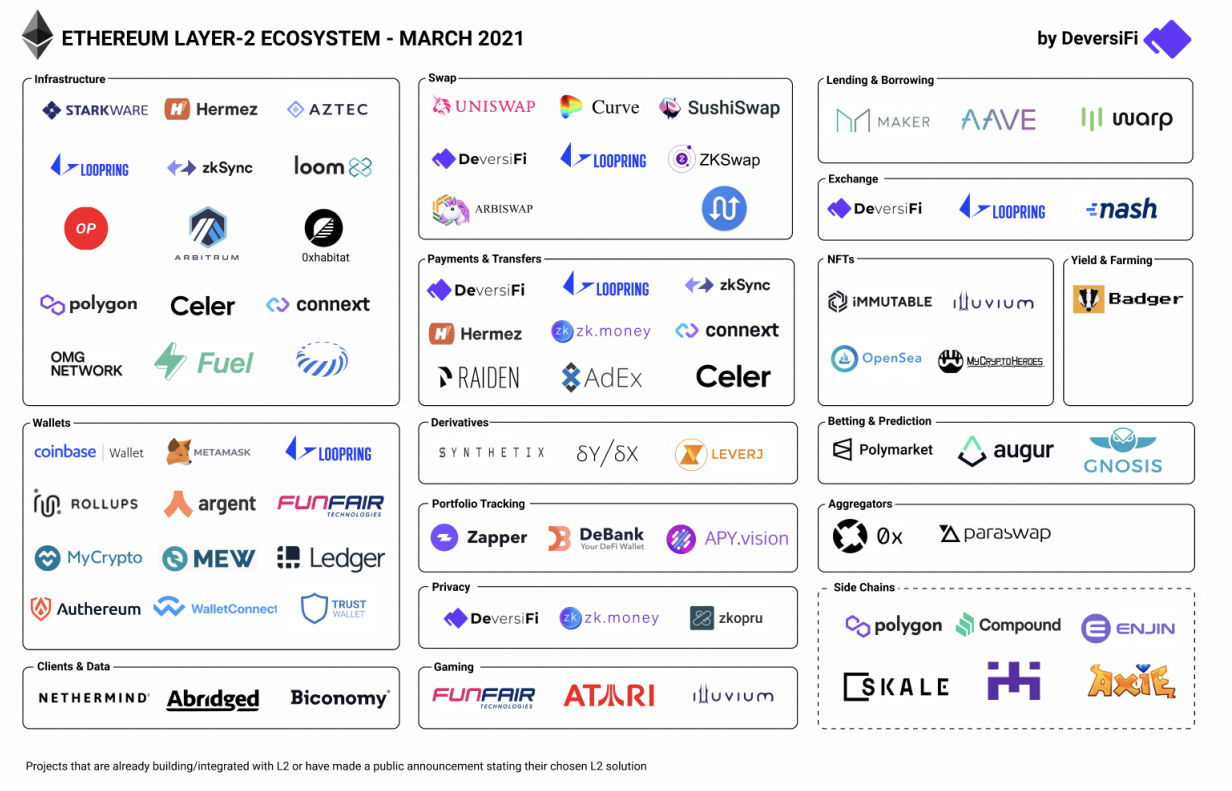

Figura 3.6: Layer 2 Ecosystem

Después de el nacimiento de Ethereum han surgido nuevas plataformas[24] que utilizan todo el potencial blockchain, en 2014 Neo [25] fue fundada por Da HongFei y Erik Zhang, además de ofrecer una plataforma para la ejecución de contratos inteligentes, dispone de un sistema de gestión de identidades digitales basadas en el estándar X.509 [26]. Posteriormente ha habido una explosión de plataformas descentralizadas, desde IOTA, una cryptomoneda que no está basada en blockchain sino que utiliza Tangle [27], un gráfico acíclico dirigido (DAG) para almacenar las transacciones. IOTA se utiliza para desarrollar en el ecosistema de internet de las cosas(IoT) sin necesidad de comisiones ni mineros, ya que cuando ejecutas una transacción validas otras dos, por que escala bien. El problema de escalabilidad dentro de blockhain, y sobre todo en Ethereum, es uno de los mas importantes y complejos. El trilema de la escalabilidad nos da a elegir dos opciones entre seguridad, descentralización y escalabilidad, siempre en detrimento de la opción no elegida.

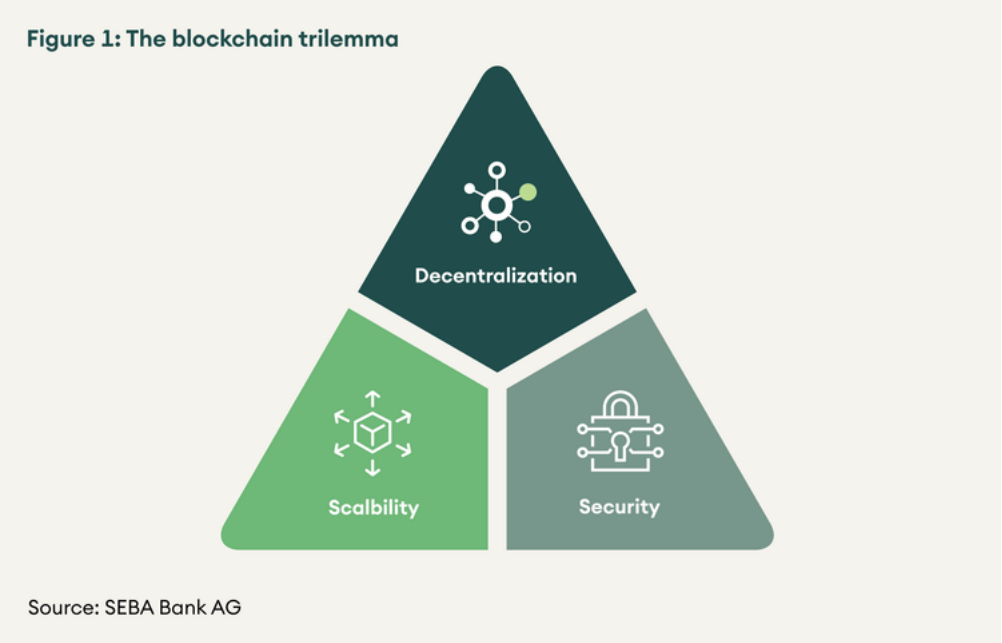

Figura 3.7: Scalability Trilema

Plasma es una solución de escalado de capa 2, como los *rollups* [28, 29], que fue propuesta originalmente por Joseph Poon y Vitalik Buterin en *Plasma: Scalable Autonomous Smart Contracts.* [30] Es un marco de trabajo para construir aplicaciones escalables utilizado por plataformas como Polygon(Matic) [31]. La capa 2 se basa en utilizar árboles de Merkle para poder crear un número ilimitado de cadenas hijas asociadas a la red principal, en la cual se pueden ejecutar transaciones y Smart Contracts sin congestionar la red principal [32].

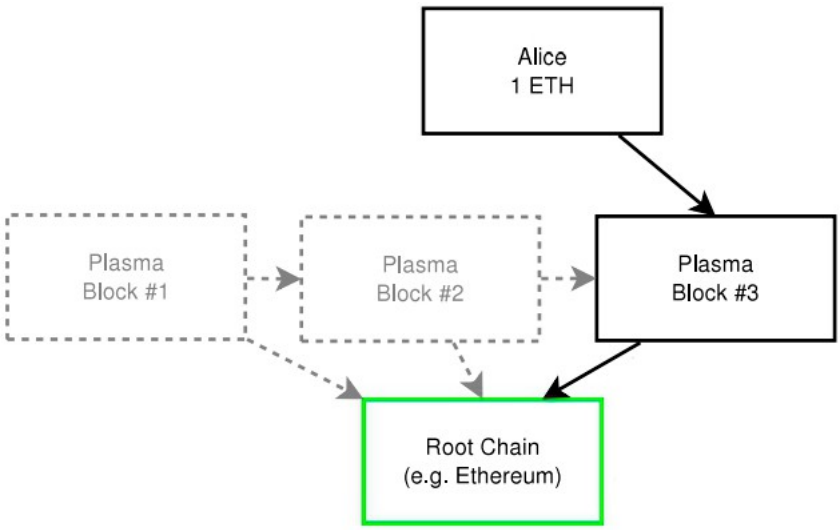

Figura 3.8: Layer 2 Scaling

Dentro del apartado de escalabilidad cabe destacar la transición en proceso a la versión 2.0 de Ethereum donde se introducen hasta 64 *Shards* o Particiones. Estas particiones son blockchains que deberían descongestionar la red principal o *Beacon Chain*, que a la vez migraría a un nuevo mecanismo de consenso basado en *Proof of Stake* (PoS). PoS se basa en que las personas que tienen ETH (la criptomoneda de la red de Ethereum) validen las transacciones de la red, en vez de los mineros en la red 1.0 que se basa en PoW. Esta transición se completaría en algún momento de 2021 o 2022.

Por último tenemos alternativas como Cosmos [33], Cardano [34] o Polkadot [35], esta última tienen un concepto parecido al *sharding* donde existen *parachains* que comparten estado con la cadena principal y se subastan por periodos de tiempo limitados. Otra de las ventajas que a nuestro juicio hacen destacar a Polakdot es que permite desarrollar Smart Contracts en cualquier lenguaje de programación gracias a la utilización de WebAssembly [36] y también Substrate [37], el framework que ofrecen para construir tu propia blockchain. Creemos que el futuro de blockchain es muy prometedor y cada día surgen plataformas o aplicaciones nuevas, lo cual contribuirá a un uso generalizado de estas tecnologías y su adopción institucional, que aumentan la transparencia, privacidad y seguridad en comparación a la estructura centralizada actual.

Capítulo 4

Metodologías y tecnologías

En este capítulo se describen las herramientas que hemos utilizado para construir este proyecto, destacando de forma sustancial la utilización de Software Libre y de Código Abierto, además de la metodología seguida.

4.1. Metodologías

4.1.1. Scrumban

Se ha seguido la metodología Scrumban [38] durante la duración de todo el proyecto. Scrumban es una metodología ágil [39] que agrupa concepto de Scrum y Kanban, el trabajo se organiza en iteraciones rápidas y se visualiza el estado en una tabla de Kanban, para la gestión de la tabla se ha utilizado Trello, donde las tareas pueden tener los estados: *ToDo*, *In Progress*, *Blocked*, *In Review* y *Done*. También hemos utilizado Confluence como única fuente de la verdad, donde se encuentra documentación, bibliografía o tutoriales, además de notas y apuntes de reuniones. Los límites de trabajo en curso o WIP facilitan que el equipo esté enfocado ya que limitan que una persona no esté trabajando en más de una tarea a la vez. En Scrumban no hay ningún líder que asigne tareas al equipo, si no que cada miembro elige las tareas que desarrolla de la lista de ToDo. En nuestro contexto creemos que Scrumban es mas apropiado que Scrum puro ya que tiene menos burocracia y las características de Scrumban nos han permitido trabajar a mayor velocidad.

Integración Continua

Dentro de las metodologías ágiles con contextos de alta incertidumbre, donde abunda la refactorización y la iteración de código, se da un escenario donde es más que probable que surjan errores. Por ello se ha utilizado Integración continua en el proyecto, con el objetivo de ejecutar un conjunto de pruebas cada vez que se genera código nuevo y así poder controlar los errores que puedan surgir rápidamente. Para la implementación de la IC se utilizó GitHub Actions dentro del repositorio del proyecto. Con el objetivo de utilizar GitHub Actions fue necesario definir la configuración de flujo de trabajo para acciones de GitHub en el archivo *testcontracts.yml*. Dentro de este archivo se definieron los trabajos que se realizarían y las dependencias que hacían falta para ejecutar estos trabajos. En la siguiente imagen se muestran los trabajos que se ejecutaban al subir nuevo código al repositorio del proyecto y su resultado final.

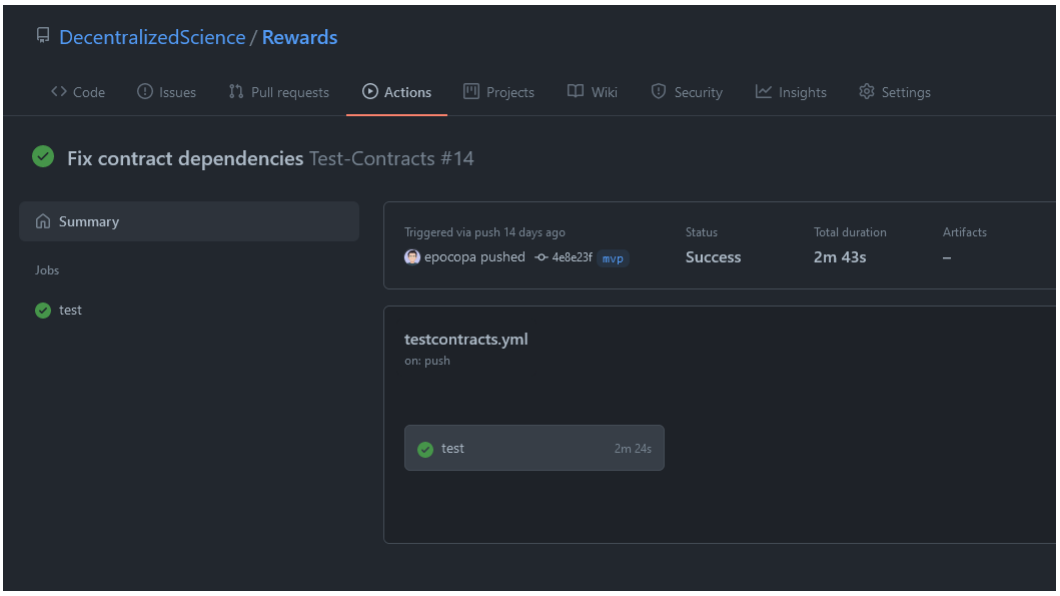

Figura 4.1: Github Actions workflow

4.2. Herramientas

En este capítulo se trataran todas las herramientas y tecnologías usadas durante el desarrollo del proyecto, así como las diferentes fases que se siguieron en este desarrollo y qué se hizo en cada una de estas.

4.2.1. Solidity

Un Smart Contract o contrato inteligente es un programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos registrados entre dos o más partes [40]. Para programar estos contratos inteligentes se ha usado el lenguaje de programación de Solidity. En estos Smart Contracts es donde reside la lógica inmutable de la aplicación. Esta lógica se despliega en la blockchain y se ejecuta sobra la Máquina Virtual de Ethereum(EVM).

Dentro del proyecto el código fuente de Solidity es compilado y desplegado en un nodo de la red de Ethereum mediante un *access-point* a través de la librería de Web3.

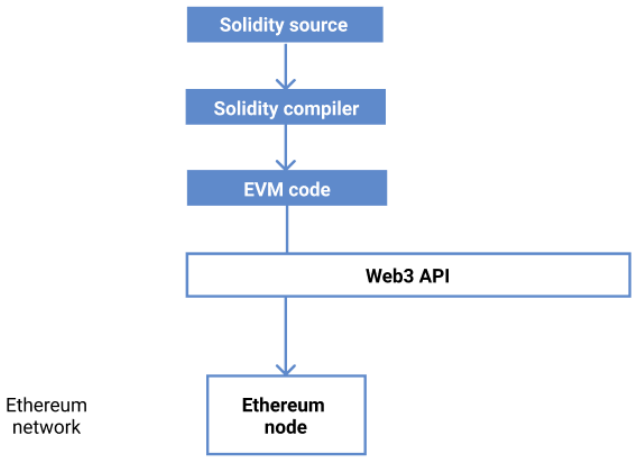

Figura 4.2: Solidity

4.2.2. Truffle

Truffle es el entorno de desarrollo más popular para Ethereum. Se ha valorado el uso de otras alternativas como Hardhat pero la documentación sobre Truffle es mayor y tiene un ecosistema más maduro. Truffle se encarga de la gestión del ciclo de vida de los contratos inteligentes, como las migraciones(scripts de despliegue que definen las acciones a ejecutar) o el testing, ya que incluye el framework Mocha [41] y la librería Chai [42] para aserciones además de incluir una blockchain integrada y soporte a otras redes [43]. Dentro de la suite Truffle se encuentra Ganache, esta herramienta permite ejecutar una blockchain personal de Ethereum, que se puede usar para la ejecución de pruebas. Al permitir depurar las transacciones de forma individual, y ver el estado de la blockchain en un momentos determinados nos ha facilitado enormemente el desarrollo y la solución de errores.

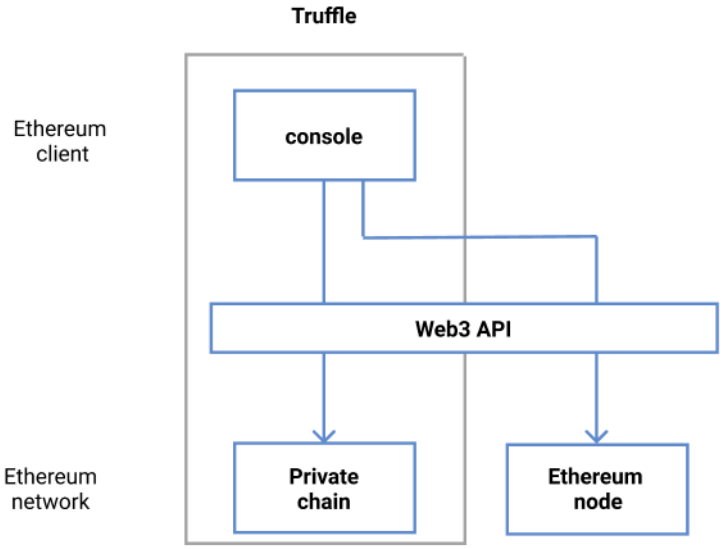

Figura 4.3: Truffle

4.2.3. Remix IDE

Remix es un entorno de desarrollo de código abierto que permite desarrollar, implementar y administrar Smart Contracts para la blockchain de Ethereum. Cuenta también con una función de depuración para comprobar el resultado de cada operación lanzada desde el Smart Contract. Se ha utilizado en el proyecto para prototipar funcionalidades y para poder interactuar con el contrato desplegado en local de forma sencilla utilizando una GUI.

Remix usa la librería de Web3 para conectarse desde el cliente a la red de Ethereum e interactuar con los contratos.

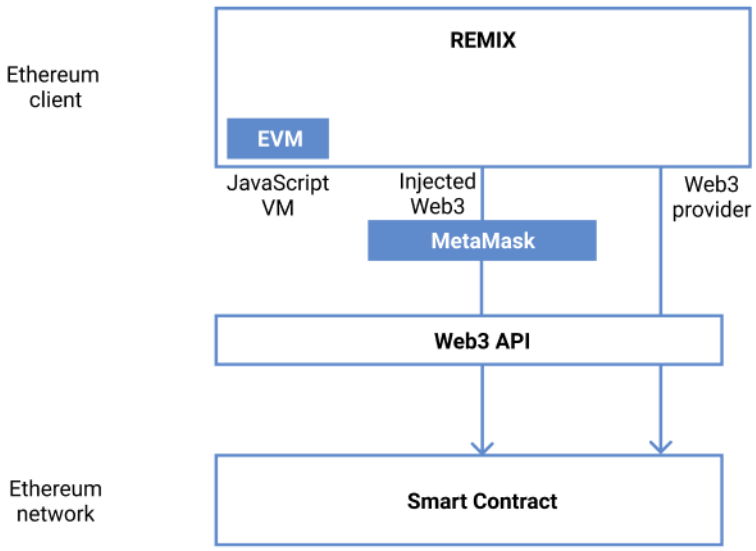

Figura 4.4: Remix

4.2.4. Web3

Web3.js es una colección de bibliotecas que permite interactuar con un nodo Ethereum local o remoto utilizando conexiones HTTP o IPC [44].

4.2.5. Metamask

Metamask es una extensión para el navegador que funciona como cartera de criptomonedas y permite a los usuarios interactuar con aplicaciones que utilizan blockchain de forma sencilla [45].

4.2.6. React

React es una biblioteca de JavaScript para crear interfaces de usuario con el fin de desarrollar SPAs. Es declarativo, lo que permite que usando React que sea más sencillo crear interfaces interactivas. Está basado en componentes encapsulados, donde cada uno de estos componentes gestiona su propio estado. Por otro lado con la composición de estas componentes se pueden llegar a generar interfaces de usuario complejas.

Junto con React [46] se ha usado el framework Ant Design [47] para la creación de las interfaces de usuario. Ant Design es un marco de desarrollo que ayuda a crear diseños bonitos y con capacidad de respuesta utilizando un HTML amigable.

Para la gestión del estado se ha utilizado Drizzle, un herramienta basada en el store de Redux [48] para manejar los datos del contrato y su interacción desde el front-end.

Capítulo 5

Desarrollo

En este capítulo se aborda el proceso de desarrollo que se ha seguido durante la realización de este proyecto.

5.1. Implementación

Hay dos partes bien diferenciadas en el trabajo realizado, por una parte están los contratos inteligentes donde se encuentra el grueso de la funcionalidad y por otra parte el front-end que se conecta a la blockchain para interactuar con los contratos.

Dentro de la metodología Scrumban fijamos las tareas a realizar en un tablero de Kanban en Trello. Después de estimar la cantidad de trabajo que supone cada historia de usuario y de dividir las historias de usuario en tareas, se recogieron todas ellas en el tablero de Trello ordenadas por prioridad. En esta herramienta el equipo llevó el seguimiento del trabajo realizado y controló el estado en el que se encontraba cada tarea. Una vez creado el Backlog se ordenaron las tareas por prioridad y se seleccionaron las más importantes para crear el Producto Mínimo Viable (MVP). Para posteriormente iterar e ir añadiendo funcionalidad hasta llegar al producto completo.

El comienzo del proyecto vino marcado por un periodo de investigación y estudio, donde las tareas que se abordaron fueron de investigación sobre las diferentes tecnologías usadas en el desarrollo del proyecto. En el apar-

tado de Smart Contract, se realizaron una serie de tutoriales introductorios a Solidity, lenguaje de programación usado para escribir Smart Contracts, además de Mocha y Chai para el testeo de las funcionalidades implementadas. Respecto a las tareas de React se realizaron tutoriales introductorios de React y otras librerías relacionadas con el front-end de la aplicación, como Redux para la gestión del flujo de la información, React Router para la gestión de rutas y Ant Design para la generación de la interfaz de usuario dentro de la aplicación. Para la generación de la interfaz gráfica el equipo creó una serie de *mockups* de la aplicación con la herramienta online *Figma* [49]. Estos *mockups* han sufrido cambios debido al feedback recibido en el marco de la metodología Scrum, donde hemos tratado cada reunión con los directores como una demo, con la posterior realización de reuniones de retrospectiva. Dentro de la investigación sobre las herramientas a utilizar y el ecosistema blockchain el equipo acudió a un meetup "*Tips de prototipado rápido de Dapps con Ethereum y React*" donde se repasaron mediante un enfoque práctico, flujos de trabajo habituales y de competencia profesional dirigidos al prototipado rápido de Dapps utilizando las tecnologías de Ethereum, Javascript React, así como servicios y librerías asociados [50].

5.2. Mockups

En este apartado se muestran los *mockups* generados los cuales sirvieron durante el desarrollo de la aplicación para implementar la interfaz gráfica de usuario. Todas las vistas contienen un espacio común de navegación que permite al usuario moverse entre las diferentes partes de la aplicación. Otra característica común que contienen todas las vistas es la cuenta asociada con la que se operará dentro de la aplicación. Estos elementos se pueden observar en todos los *mockups* que se muestran a continuación.

El *mockup* de la pantalla de inicio incluye un formulario en la parte superior de la pantalla el cual permite crear un nuevo artículo. A continuación de este formulario se presentan en forma de cajas un listado de todos los artículos existentes con un encabezado que contiene el título y el autor del artículo, y un cuerpo con una pre visualización de este artículo. A través de un botón se puede acceder a la vista del artículo en detalle.

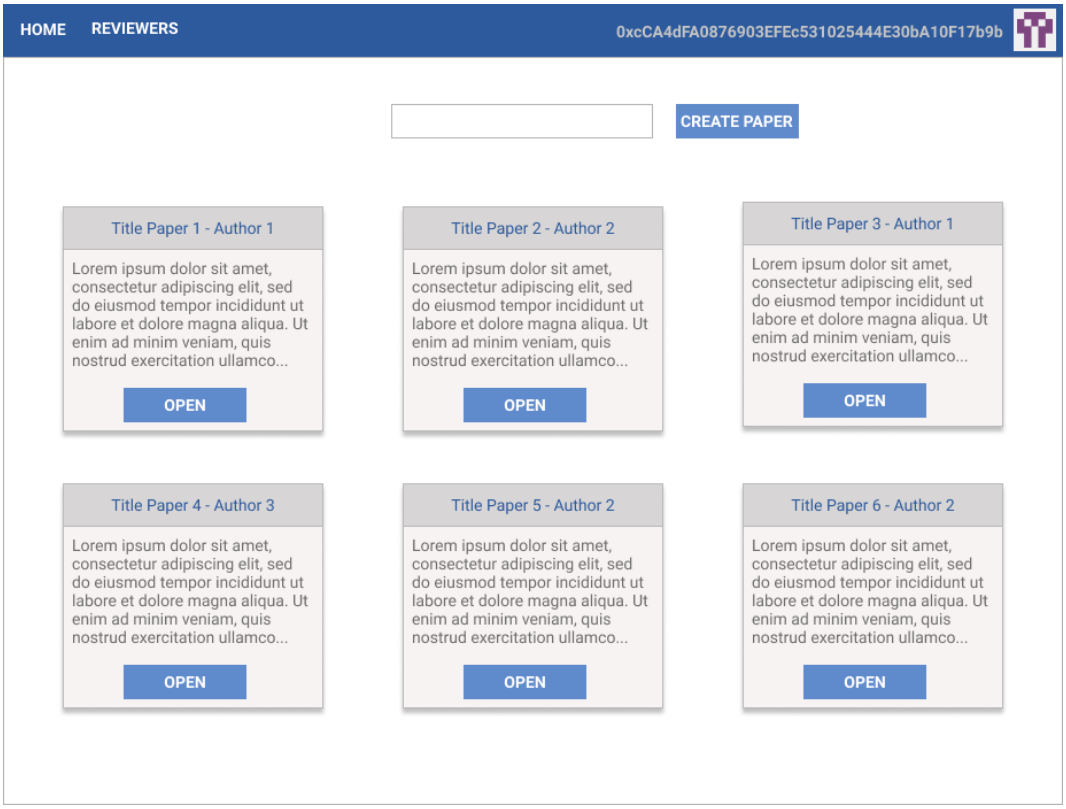

Figura 5.1: Mockup Home

Al abrir uno de los artículos de la pantalla de inicio se accede a la vista en detalle de este artículo. Para la creación de esta pantalla se utilizó el siguiente *mockup* como base en el cual se muestra en el apartado izquierdo de la pantalla el contenido del artículo con el título y e autor en la parte superior. Por otro lado en la parte derecha de la pantalla se muestra un listado de los revisores de este artículo. Por cada revisor se muestran tres botones, uno para enviar una propina, otro para dar reputación y el último para dar un trofeo o premio.

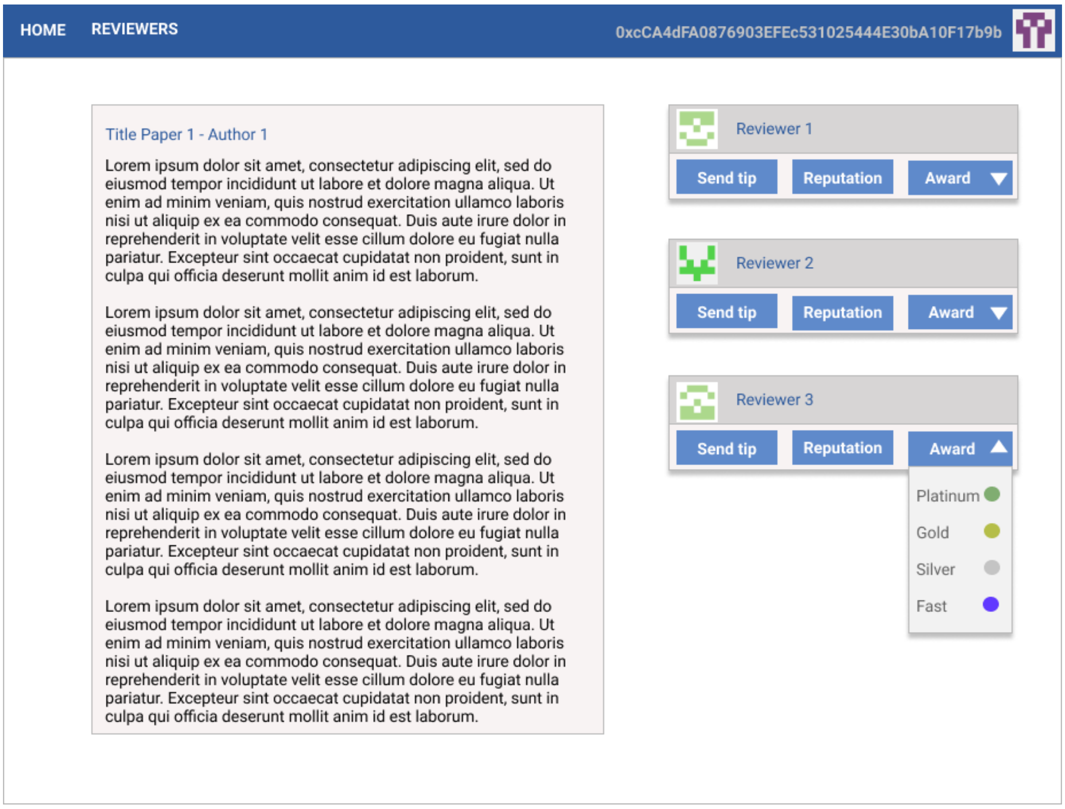

Figura 5.2: Mockup Paper

El último *mockup* sirvió para establecer un concepto de cómo se mostraría la vista de revisores. Esta vista se compondría por una tabla donde se mostrarían todos los revisores con una serie de datos por cada uno de ellos.

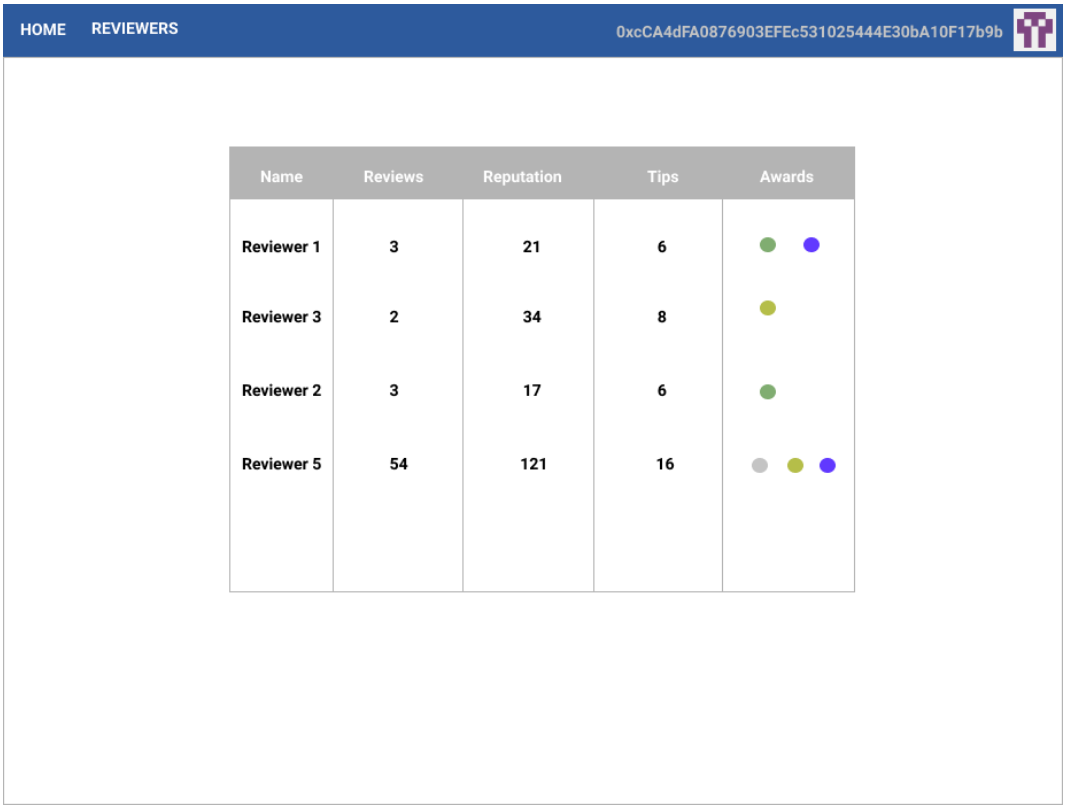

Figura 5.3: Mockup Reviewers

5.3. MVP

Durante la creación del MVP de la aplicación el equipo implementó el siguiente conjunto de tareas.

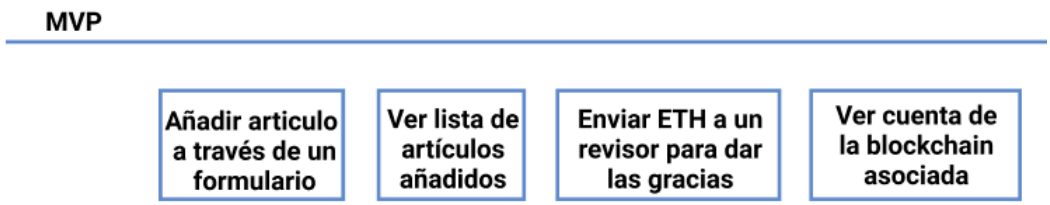

Figura 5.4: MVP

Para realizar estas tareas se empezó con la generación y pruebas del Smart Contract de Rewards. Este Smart Contract, en una primera versión, permitía

enviar una propina de 1 ETH a un revisor. Para completar la generación del MVP también fue necesario implementar parte de la interfaz gráfica de la aplicación. Para comprobar el correcto funcionamiento de la aplicación en este punto, era necesario realizar una transacción. Para ello se lanzaba la aplicación y la blockchain de Ganache en local y se pulsaba sobre el botón de enviar propina para iniciar la transacción. En la siguiente imagen se pude ver la interfaz de Ganache, que ejecuta una blockchain en local. Se lanza con 10 cuentas por defecto con 100 ETH cada una, para poder hacer pruebas o desplegar contratos.

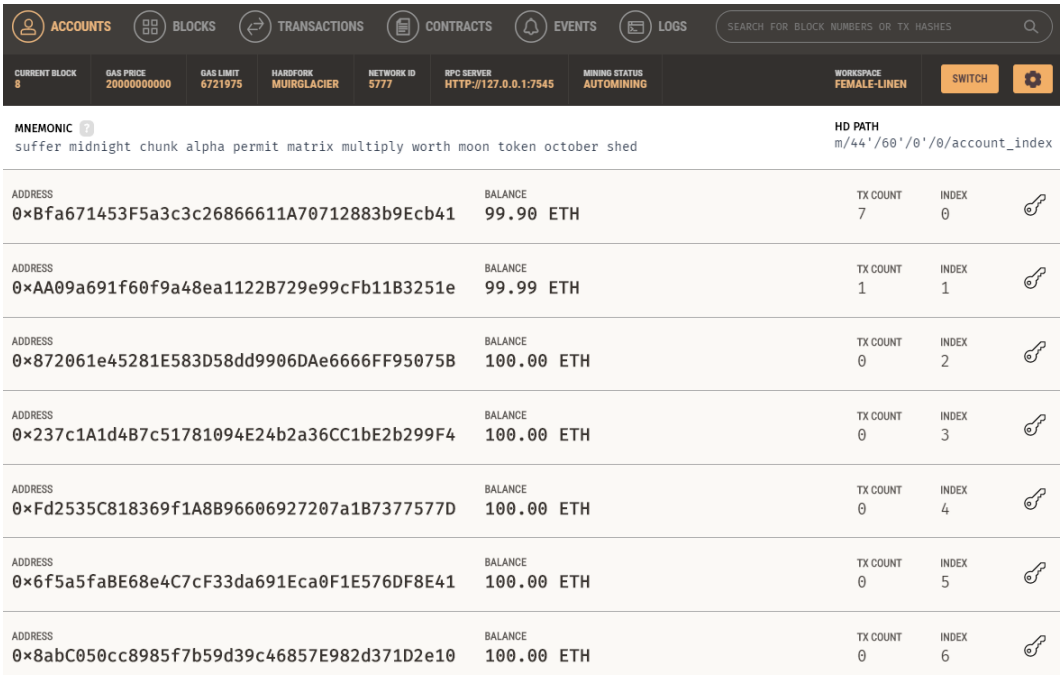

Figura 5.5: Ganache

Una vez pulsado el botón se abre desde el navegador la cartera de Metamask, cuya configuración se había hecho previamente. En este paso el usuario autoriza que se ejecute la transacción que moverá 1 ETH de su cartera, a la cartera del revisor.

Figura 5.6: Metamask Transaction

Al finalizar la transacción se podía comprobar desde la blockchain de Ganache cómo se había generado un movimiento de ETH entre diferentes cuentas.

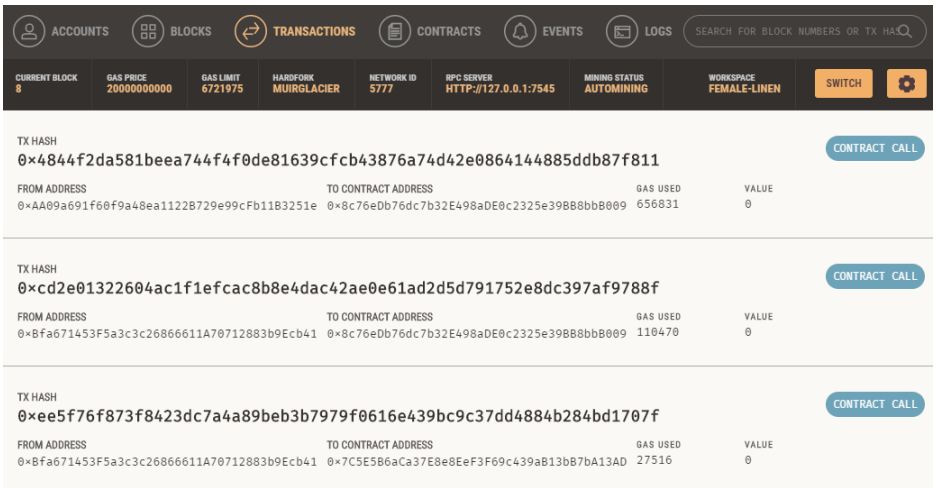

Figura 5.7: Ganache Transactions

5.4. Release

Una vez generado el MVP de la aplicación el siguiente hito del equipo fue la investigación sobre tokens de Ethereum. Antes de proseguir definir que los ERC en Ethereum son documentos técnicos utilizados por los desarrolladores de contratos inteligentes en Ethereum. Definen un conjunto de reglas necesarias para implementar tokens para el ecosistema Ethereum. Estos documentos suelen ser creados por desarrolladores e incluyen información sobre especificaciones de protocolo y descripciones de contratos. Antes de convertirse en un estándar, un ERC debe ser revisado, comentado y aceptado por la comunidad a través de un EIP (propuesta de mejora de Ethereum) [51]. Se pueden consultar los ERCs existentes en *https://eips.ethereum.org/erc*. Siguiendo con el desarrollo, los tokens a investigar fueron el ERC20 [52] y el ERC721 [53]. Para esta tarea el equipo comenzó con el estudio de la librería de Smart Contracts de OpenZeppelin [54], la cual incluía una documentación muy completa e implementaciones sobre los ERCs a investigar. OpenZeppelin es un proyecto en el que se han desarrollado diversas herramientas para facilitar el uso de aplicaciones descentralizadas y la creación de Smart Contracts. Siguiendo con los tokens, el equipo decidió utilizar el estándar ERC20 para la implementación del sistema de reputación. Por otro lado después de realizar la investigación del token ERC721 el equipo eligió este para usarlo en el sistema de recompensas ya que este token tenía como característica ser no fungible, es decir, que cada token era único y por ende las recompensas otorgadas también.

Después de la investigación sobre los tokens, se realizaron una serie de modificaciones sobre las implementaciones de los estándares de OpenZeppelin para adaptarlos a las necesidades del proyecto. Las modificaciones realizadas en los tokens ERC20 y ERC721 fueron hacer de estos tokens no transferibles, ya que en los sistemas de reputación y recompensas que se querían implementar en el proyecto no se permite hacer transferible la reputación o las recompensas que un revisor pudiera obtener. Estas modificaciones podrían presentarse como nuevos ERCs y formalizarlos en estándares.

Además para el ERC20 se suprimieron los decimales, ya que la reputación es indivisible, también se modificó el suministro de tokens para que sea infinito. En el ERC721 se eliminaron dependencias de IERC721Receiver, IERC721Metadata, IERC721Enumerable y ERC165, ya que no eran necesarias para nuestra implementación, y se modificó el identificador para que sea un hash formado por el usurario que da la recompensa, el revisor que

la recibe, el tipo de recompensa, y el artículo al cual pertenece la revisión. De esta forma nos aseguramos que las recompensas sean únicas y no puedan duplicarse.

Al realizar las modificaciones se decidió nombrar los contratos de ERC20 como *ReputationToken* y de ERC721 como *AwardsToken*. Con estas modificaciones se consiguieron implementar los sistemas de reputación y recompensas acordes a las especificaciones que se marcaron en el proyecto. Para comprobar el correcto funcionamiento de las implementaciones realizadas se utilizó el IDE de Remix el cual contiene una función de depuración para los Smart Contracts.

El desarrollo de las modificaciones sobre los tokens anteriores se vio facilitado y acelerado gracias al uso de Integración Continua por la detección temprana de errores.

Durante todo el periodo de desarrollo del proyecto, el equipo fue recopilando documentación la cual se usó posteriormente para la generación de esta memoria. En la generación de la documentación, el equipo determinó que a medida que se fuera desarrollando esta documentación se fuera revisando y actualizando. Esto se hizo con la finalidad de mejorar posteriormente los distintos apartados de la memoria y corregir posibles erratas que fueran surgiendo.

Capítulo 6

Resultados

Para la generación del mapa de historias de usuario el equipo se hizo la siguiente pregunta, ¿qué harán los usuarios con el sistema?. A partir de esta pregunta se pensaron en las distintas actividades que se realizarían dentro de la aplicación y se ordenaron de izquierda a derecha creando así el flujo del usuario. Cada una de estas actividades resultaría ser una historia de usuario Épica [55], es decir, una historia de usuario con mucha funcionalidad. Estas funcionalidades se dividieron en tareas debajo de cada una y se ordenaron por prioridad.Para generar este documento se utilizó la misma herramienta que para los mockups: Figma.El mapa de historias de usuario resultante fue el siguiente.

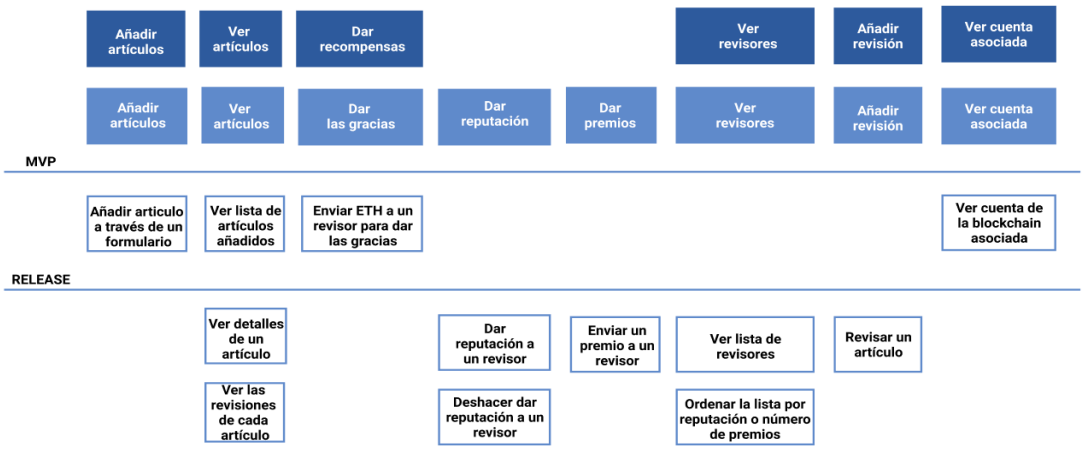

Figura 6.1: User Story Map

La arquitectura que se implementó en el proyecto fue finalmente la que se muestra en la siguiente imagen. Para la creación del Smart contract se usó el framework de Truffle junto con Solidity para la programación de este y Chai y Mocha para las realizar pruebas. Para la parte del front-end se usó React junto con Ant Design como librería de componentes, la API de Web3 para conectarse con el Smart Contract y Drizzle para la gestión de estado. Por otro lado las pruebas se realizaron mediante la blockchain que proporcionaba Ganache de manera local.

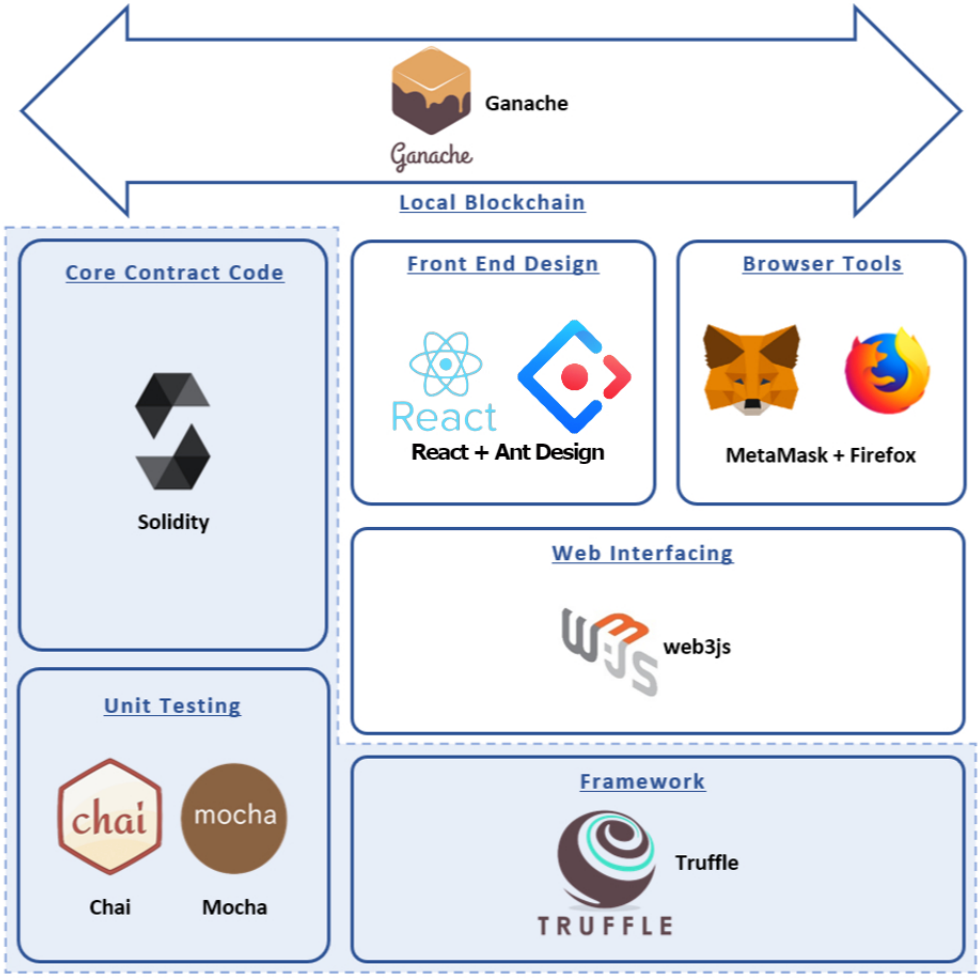

Figura 6.2: Software Architecture [56]

6.1. MVP

Tras terminar el desarrollo del Producto Mínimo Viable el estado del producto era el siguiente:

- El apartado del front-end la web se conectaba de forma exitosa con la blockchain, y consistía una pantalla con la lista de artículos donde se mostraba un artículo de prueba junto a un formulario para añadir nuevos artículos. Cada artículo disponía de un botón donde poder enviar ETH al revisor, al no estar finalizada la estructura de los datos

dentro del Contrato principal, se daba la limitación de un solo revisor por artículo. En la barra de navegación se mostraba la información de la cuenta seleccionada en ese momento en MetaMask.

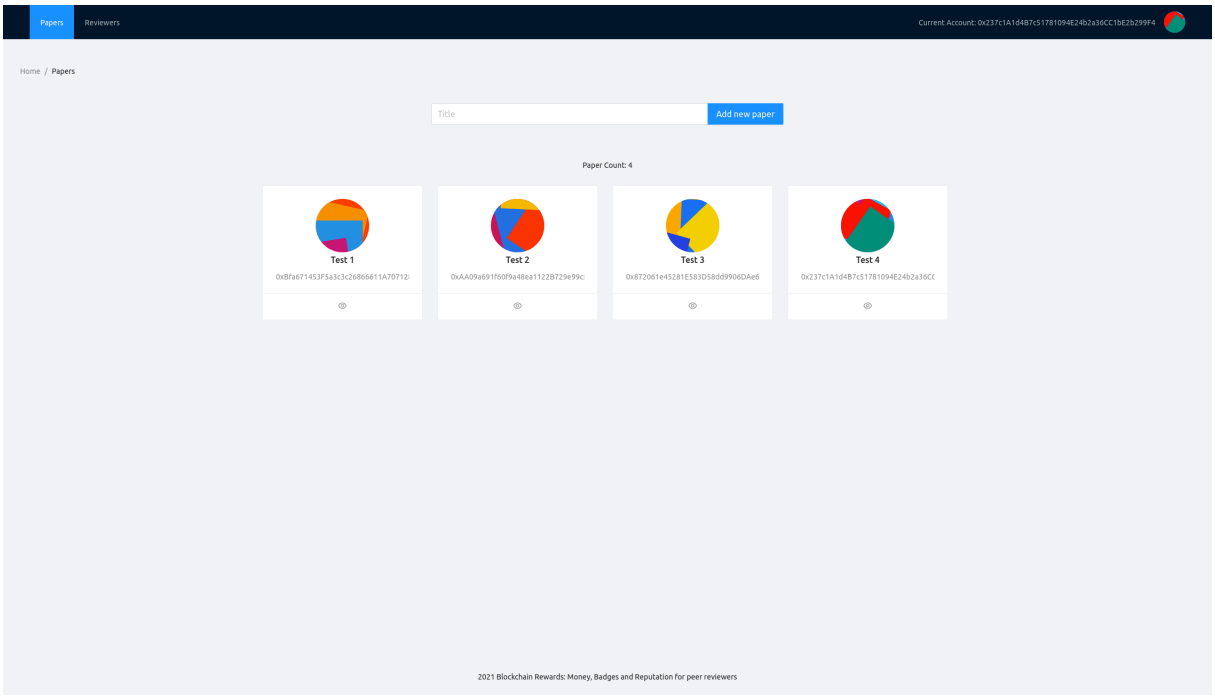

Figura 6.3: Paper List

- La parte del Smart Contract consistía en las estructuras de datos para almacenar la información de los artículos y revisores en la blockchain, además de las funciones para crear un artículo y enviar ETH como agradecimiento junto con los test correspondientes y la definición de los eventos emitidos por estas funciones.

La pantalla creada consistía en una tarjeta con la información más relevante del revisor, y un botón con la función de dar una propina al revisor.

6.2. Release

Tras terminar todas las iteraciones sobre el Producto las funcionalidades desarrolladas consisten en:

- Añadir nuevo artículo de investigación a través de un formulario.
- Ver lista de artículos añadidos.
- Enviar ETH a un revisor para dar las gracias.
- Ver cuenta de blockchain asociada al usuario actual.
- Ver detalles de un artículo.
- Ver las revisiones de un artículo.
- Dar reputación a un revisor.
- Deshacer dar reputación a un revisor.
- Enviar un premio a un revisor.
- Ver lista de revisores.
- Ordenar la lista por reputación o número de premios.
- Revisar un artículo.

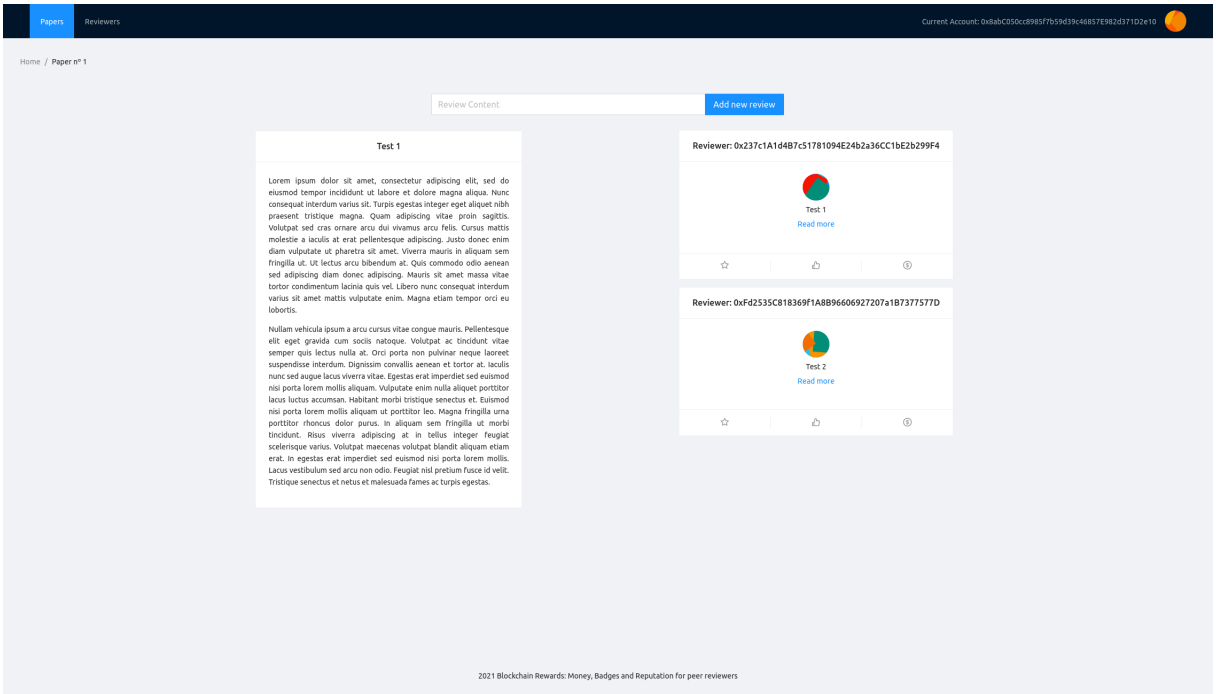

Figura 6.4: Paper Details

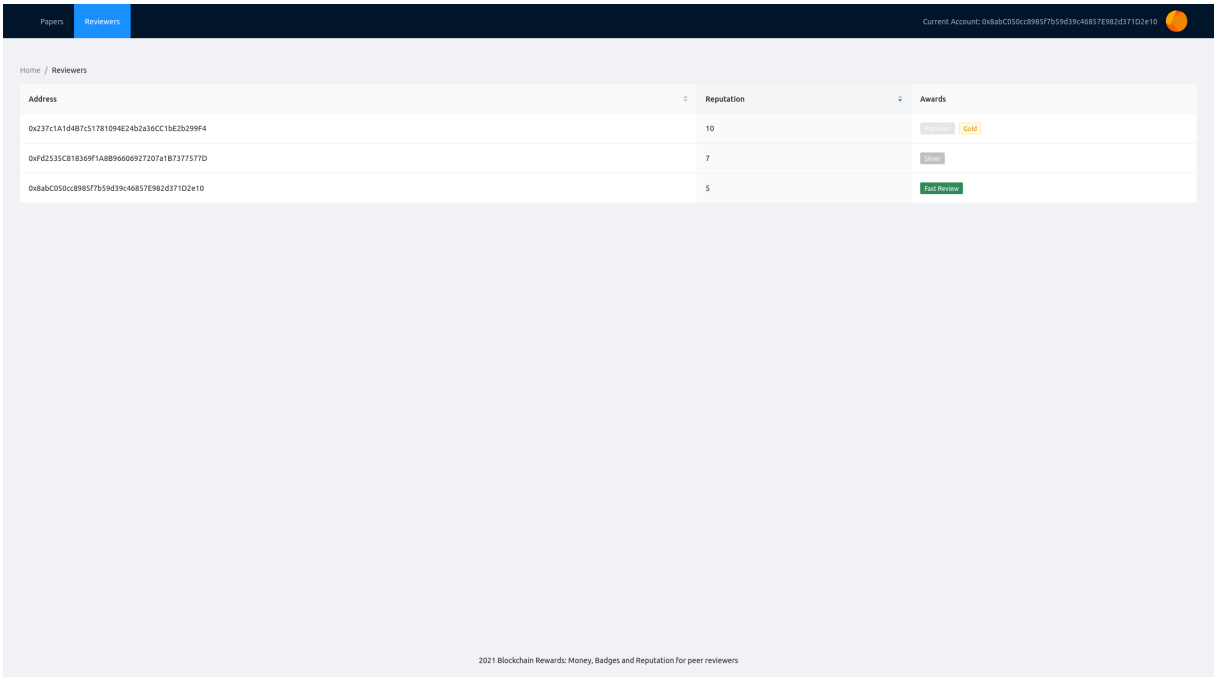

Figura 6.5: Reviewer list

Respecto al desarrollo de la interfaz gráfica de usuario el equipo debatió sobre qué framework usar. Todos los frameworks que se mencionan a continuación tenían compatibilidad con React lo que era un requisito indispensable. Entre los candidatos estaban Ant Desing, Material UI y Semantic UI. Finalmente el equipo optó por elegir Ant Desing como framework a utilizar en el proyecto por su popularidad, su documentación clara e intuitiva y su facilidad de aprendizaje.

Para desarrollar el resto de la interfaz gráfica, el equipo se apoyó en los *mockups* previamente generados y validados. Estos *mockups* se muestran en apartados anteriores.

Finalmente tras la generación de las diferentes pantallas que componían la aplicación se realizó la conexión entre el Smart Contract y la interfaz de usuario mediante Web3. Para ello se usaba la librería de Web3 para conectarse a un nodo, en este caso local, a través de llamadas RPC (*Remote Procedure Call*). De esta manera se hacía posible llamar a las distintas funciones del Smart Contract a través del objeto *eth* de Web3.

Capítulo 7

Experimentación

En este apartado se tratarán los diferentes experimentos y estudios relacionados los *Smart Contracts* desarrollados donde se analiza el impacto económico de la utilización de Ethereum, a través del cálculo de costes en Gas y del despliegue de los mismos en una red de prueba.

7.1. Gas en Ethereum

Para entender el Gas en Ethereum es necesario hablar previamente de las transacciones. Una transacción de Ethereum se puede definir como la ejecución de una instrucción, como podría ser transferir Ethereum de una cuenta a otra. Una transacción requiere una tarifa y a su vez se deben minar para validarlas. En este punto es donde interviene el Gas. El Gas es la unidad que mide la cantidad de esfuerzo computacional requerido para ejecutar operaciones específicas en la red de Ethereum. Es una referencia al cálculo requerido para procesar la transacción por parte de un minero, o dicho de otra manera, es el precio que los usuarios deben pagar por este cálculo. Las tarifas de gas se pagan en ETH, que es la moneda nativa de Ethereum. Los precios del gas se indican en Gwei y la equivalencia es 1 Gwei igual a 0,000000001 Ether [57].

El gas en Ethereum se puede dividir en tres conceptos: el costo en unidades de gas, el precio del gas y el límite de gas [58].

- El **costo de gas** se entiende como las unidades de gas que son necesarias para hacer cada operación y ya están definidos en la documentación de Ethereum.
- El **precio del gas**, ya explicado anteriormente y sobre el que se puede pagar un mayor o menor precio de gas por transacción para obtener así prioridad y ser procesada antes por los mineros.
- El **límite de gas** es la cantidad máxima de gas que se está dispuesto a pagar para realizar una transacción. Esta cantidad suele y debe ser mayor que la cantidad real de gas que requiere una cierta transacción ya que en caso contrario la transacción fallará debido a un límite de gas demasiado bajo y la transacción quedará en un estado bloqueado. Para establecer el límite de gas necesario *Ether Gas Station* [59] recomienda establecer este límite en 21000 aunque no existe un límite fijo preestablecido.

Con los factores explicados anteriormente se puede concluir que la existencia del gas en Ethereum tiene los siguientes usos.

- Se incentiva a que los mineros inviertan su tiempo y energía en la ejecución de transacciones a cambio del pago de unas tasas de gas.
- Ayudan a mantener la seguridad la red de Ethereum. Por cada transacción se necesita una tarifa, esto evita el posible spam a la red.
- Resuelve el problema del *halting problem* [60] gracias al límite de gas. El *halting problem* o problema de la detención consiste de manera resumida en saber si un programa arbitrario con una entrada establecida se ejecutará y terminará o si continuará funcionando para siempre. En Ethereum este problema no es replicable ya que en una transacción con código malicioso o incorrecta el gas se acabará agotando y la ejecución finalizará.

Para calcular el coste de una transacción en el *Smart contract* se usó la calculadora de *ETH Gas Station* [59]. Los datos que se aportaron fueron el uso de gas que se requiere en una transacción, que a día de hoy es de 60000, el límite del gas establecido en 21000 por defecto en *Ether Gas Station* y el precio del Gas. Para este último dato se optó por elegir el precio medio del Gas que a día de hoy equivale a 94 Gwei. Los datos resultados fueron los siguientes.

% of last 200 blocks accepting this gas price	93.8271604938
Transactions At or Above in Current Txpool	166
Mean Time to Confirm (Blocks)	2
Mean Time to Confirm (Seconds)	33
Transaction fee (ETH)	0.00564
Transaction fee ($)	**19.1478**$

Cuadro 7.1: Transaction cost

De estos resultados se pudo observar que el precio de las tarifas de transacción de son bastante elevadas. Para estudiar el precio de las tarifas de transacción se acudió a *blockchair.com* de donde se extrajo la siguiente gráfica que representa la evolución que han tenido en el tiempo las tarifas de Ethereum.

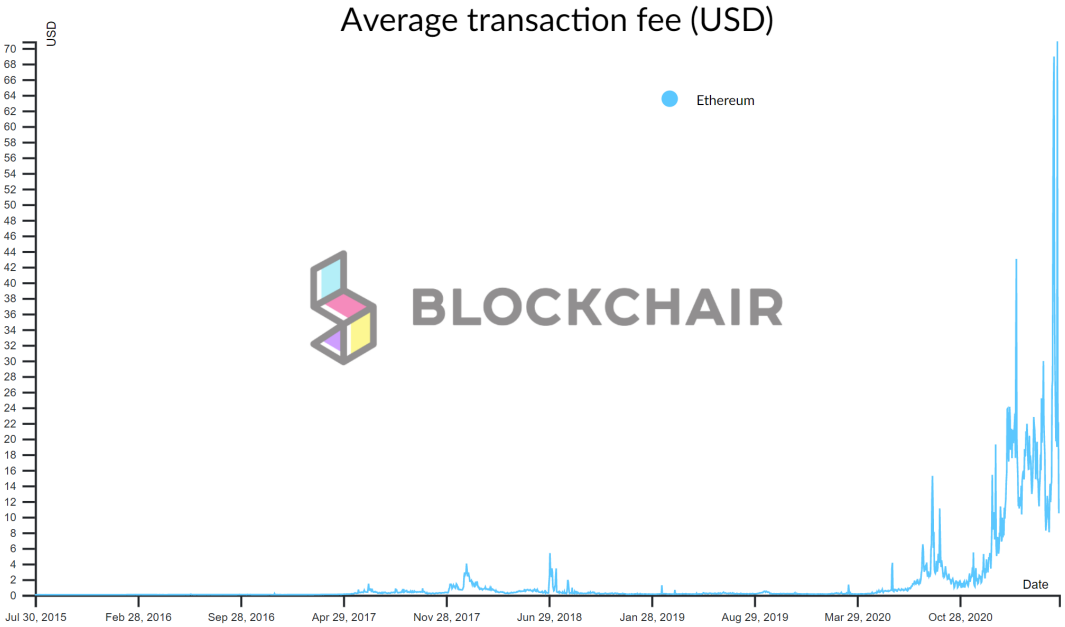

Figura 7.1: Average transaction fee (USD)

Como se puede observar, las tasas por transacción en Ethereum se han disparado a principios de este año 2021. Para entender este crecimiento se debe retroceder hasta 2017 donde hubo un gran auge de ICO (*Initial Coin Offering*), debido a la gran cantidad de tokens que se crearon en la cadena de bloques de Ethereum. Estos tokens fueron posteriormente vendidos a través

de sitios web centralizados. A partir de entonces, durante los siguientes años el espacio ICO se fue regulando, lo que provocó que los nuevos intercambios pasaran por procedimientos KYC (*Know Your Customer*) cada vez más engorrosos. Esta situación provocó la aparición de una solución en forma de intercambios descentralizados (DEX), donde cualquier individuo sería capaz de realizar intercambios sin la necesidad de pasar por un proceso de verificación. Un DEX o un intercambio descentralizado, consiste en una plataforma de intercambio de activos digitales. En un DEX se hace uso del principio de descentralización, lo que quiere decir que no es necesaria la participación de una entidad central para realizar este intercambio. Esto permite el comercio *peer-to-peer* o entre pares y soluciona el problema de que una entidad central pueda manipular los precios o que se realicen operaciones fraudulentas [61].

Con la aparición de *DeFi* o Finanzas Descentralizadas el volumen de intercambios descentralizados aumentó drásticamente pasando de 540.026.286$ en 2017 a 426.503.124.056$ en 2021. Para la obtención de estos datos se utilizó la página *duneanalytics.com*, que sirve para realizar análisis sobre datos de la blockchain de Ethereum. En esta página se lanzó la siguiente consulta, que pide los datos sobre el volumen de intercambio en *DEXes* o Plataformas de Intercambio Descentralizadas en los últimos 5 años.

```
SELECT date_trunc('year', block_time),
SUM (usd_amount) AS usd_volume
FROM dex.trades t
WHERE block_time  >= date_trunc('year', now()) - interval '5 years'
GROUP BY 1;
```

El resultado de la consulta se puede observar en esta gráfica.

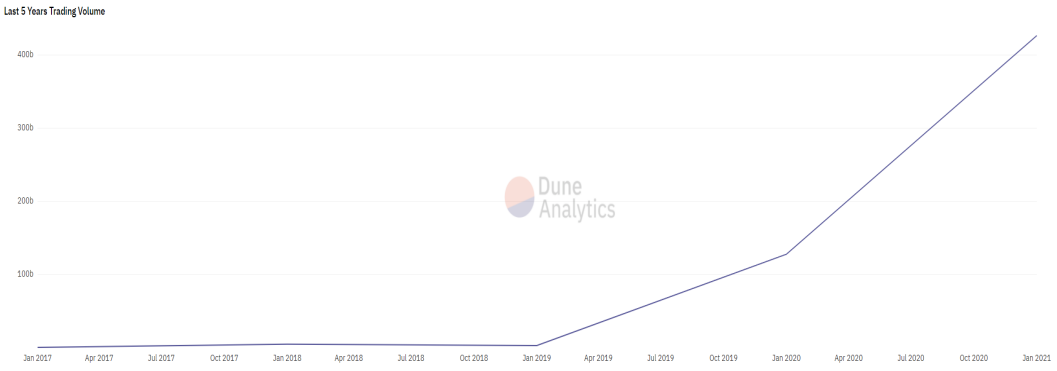

Figura 7.2: Trading volume in the last 5 years

Como vemos, todos estos factores, junto con la volatilidad del precio de los tokens, han contribuido a que el coste económico de desplegar contratos e interactuar con la blockchain sean procesos caros y de coste impredecible. Para evitar estos altos costes transaccionales existen varias alternativas. Se podría utilizar la blockchain de Bloxberg, que se caracteriza por proporcionar a los científicos servicios descentralizados para sus investigaciones, y al utilizar el *Faucet* se podría afrontar el pago de las transacciones. Un *Faucet* es una web donde puedes obtener generalmente pequeñas cantidades de una criptomoneda de forma gratuita. Se analizará en el apartado 6.3. Por otro lado, se podrían utilizar plataformas basadas en la capa 2 como Polygon, las *parachains* de Polkadot o directamente Ethereum 2.0, con el objetivo de que las interacciones con la blockchain sean más económicas. Estas formas de escalado se detallan en el apartado 3.1.5.

7.2. Despliegue de un Smart Contract

Una vez que el Smart Contract fue generado se desplegó en la red local de Ethereum. Para hacer el despliegue fue necesario previamente generar los archivos de configuración *1_initial_migrations.js* y *2_deploy_contracts.js* que nos sirvieron para hacer el despliegue de los Smart contracts. El archivo *1_initial_migrations.js* es requerido por Truffle para realizar una primera migración del contrato *Migrations.sol*, definido ya por Truffle, y el cual se usa para mantener un registro de las migraciones que se han realizado en la red. En el archivo *2_deploy_contracts.js*, se especifican todos los Smart Contracts a desplegar.

Desde una terminal se compilaron los Smart Contracts con el comando *truffle compile* el cual generó los artefactos en formato *JSON* de los Smart Contracts dentro del proyecto. Dentro de este paso es importante la configuración de la versión de solc en el archivo *truffle-config.js*. Para este proyecto se usó la versión 0.8.1 de solc. Y también es necesario seleccionar la red, se ha usado Ganache pero también se puede levantar una blockchain con *truffle develop*.

Tras compilar los Smart Contracts se pasó a ejecutar la migración con el comando *truffle migrate*. Para realizar la migración se conecta a la blockchain local de Ganache.

Existe una migración inicial requerida por Truffle para habilitar la funcionalidad de migraciones. El resultado de la migración de los Smart Contracts queda reflejado en la siguiente imagen.

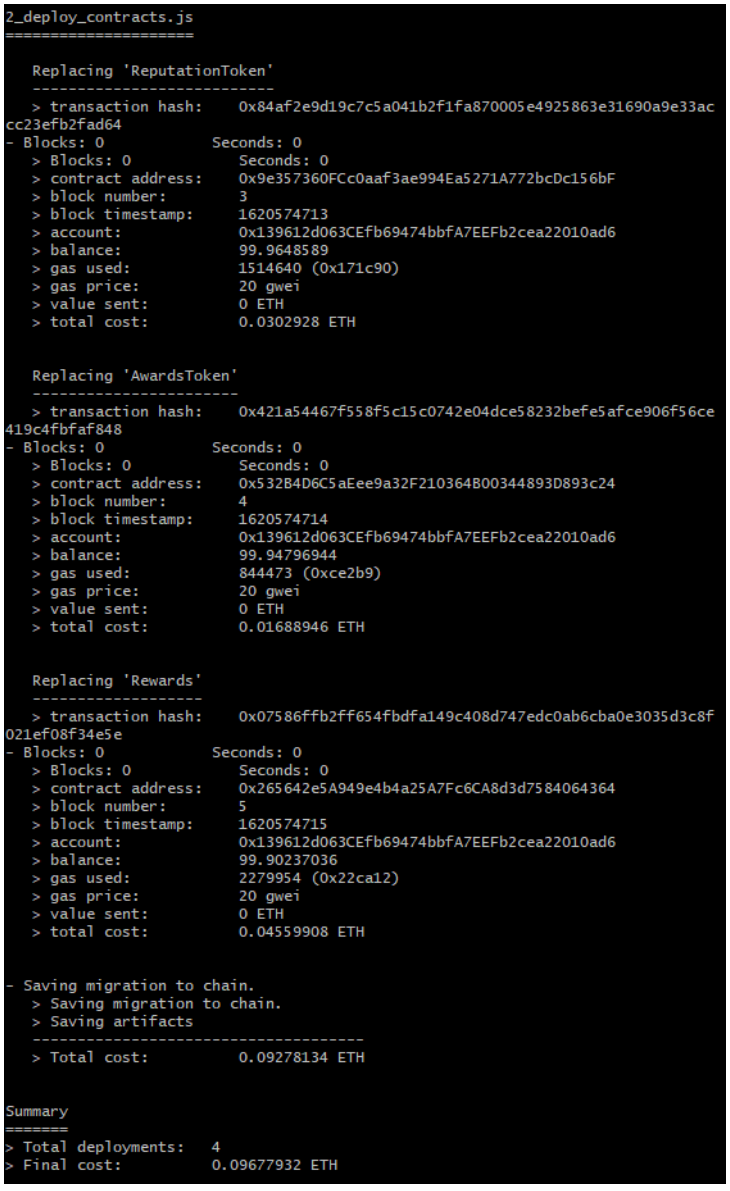

Figura 7.3: Smart Contracts migration

Como se puede observar en la imagen, todo despliegue conlleva un coste de ETH ya que existe un cálculo computacional que debe de ser cubierto. El

coste final del despliegue es de 0.09677932 ETH, lo que equivale a 213.3465€. Este coste contiene el ETH que fue necesario para el despliegue de todos los Smart Contract, incluyendo todas las migraciones.

7.2.1. Coste de interactuar con los Smart Contracts desarrollados

En esta sección se analizan los costes de interactuar con la funcionalidad que se ha desarrollado dentro de los Smart Contracts y lo que tendría que pagar un usuario. En el Yellow Paper, Gavin Wood detalla la tabla de tarifas. Dentro de esta tabla nos centramos en $G_{transaction}$, $G_{txdatanonzero}$, $G_{txdatazero}$.

APPENDIX G. FEE SCHEDULE

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for an EXTCODESIZE operation.
$G_{extcodehash}$	700	Amount of gas to pay for an EXTCODEHASH operation.
$G_{balance}$	700	Amount of gas to pay for a BALANCE operation.
G_{sload}	800	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
G_{txcreate}	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	16	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
G_{logdata}	8	Paid for each byte in a LOG operation's data.
G_{logtopic}	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	20	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

Figura 7.4: Costs table

Vemos que:

- 21000 gas se paga por cada transacción.
- 68 gas se paga por cada byte distinto de cero de datos o código por cada transacción.
- 4 gas se paga por cada byte igual a cero de datos o código por cada transacción.

Tras el despliegue de los Smart Contracts se usó la aplicación de Remix para interactuar con ellos y analizamos los costes que tendrían las diversas operaciones que los componen. El resultado de estos costes se muestra en la siguiente tabla.

Operation	**Transaction cost**		**Execution cost**	
	gas	euros	gas	euros
Create Paper	111142	0,2189	88974	0.2272
Add Reviewer with a Review	163064	0.4164	139168	0.3554
Give Reputation	163064	0.4164	139168	0.3554
Undo Give Reputation	26701	0.0682	30593	0.0781
Send Tip to Reviewer	34764	0.0888	11956	0.0305
Give Award	183544	0.4687	160544	0.4099

Cuadro 7.2: Operations costs

7.3. Bloxberg

Para el despliegue del Smart Contract se hizo una investigación sobre la blockchain global de Bloxberg. Esta cadena de bloques se caracteriza por ser segura y estar creada por un consorcio de organizaciones dedicadas a la investigación científica. Bloxberg pretende promover la primera red científica descentralizada para proporcionar a los científicos servicios descentralizados de manera global. También pretende fomentar la colaboración entre la comunidad científica mundial ofreciendo a los investigadores un servicio robusto y autónomo [62].

Las principales características por las que el equipo decidió hacer una investigación sobre *Bloxberg* fueron:

- La **certificación de datos de investigación** permite que los investigadores aprovechen la cadena de bloques de *Bloxberg* para crear una huella del trabajo sobre la investigación realizada. También permite generar un certificado que demuestra cuándo se realizaron la carga de los datos en un momento determinado. Estos datos no tienen por que hacerse públicos.
- El **grifo de *Bloxberg*** proporciona bergs (la moneda de *Bloxberg*) entre aquellas entidades que quieran construir en *Bloxberg* o utilizar funcionalidades de las aplicaciones. Estos tokens sirven para sufragar los despliegues de Smart Contracts o para interactiar con las aplicaciones ya desplegadas.

Tras esta investigación se vio como una posibilidad el despliegue de los Smart Contracts dentro de la blockchain de *Bloxberg*. Al estar realizando un proyecto de investigación se podrían usar los recursos que se ponían a nuestra disposición. Con estos recursos el equipo sería capaz de registrar mediante una huella temporal el trabajo de investigación realizado y hacer pruebas sin limitaciones gracias al grifo de begs que proporcionan los miembros del consorcio y así solventar el costo de las transacciones que se mencionan en los apartados anteriores.

Capítulo 8

Aportaciones Individuales

Las aportaciones individuales son iguales a grandes rasgos ya que hemos trabajado siempre de forma conjunta y por igual en todas las partes del producto, incluida esta memoria.

8.1. Pablo Agudo Brun

En el desarrollo de este proyecto he realizado las siguientes aportaciones:

- Investigación sobre los fundamentos de Blockchain
- Asistencia al meetup "*Tips de prototipado rápido de Dapps con Ethereum y React*" donde se repasaron mediante un enfoque práctico, flujos de trabajo habituales y de competencia profesional dirigidos al prototipado rápido de Dapps utilizando las tecnologías de Ethereum, Javascript React, así como servicios y librerías asociados
- Investigación sobre Truffle, Ganache y Drizzle así como la realización de los tutoriales oficiales.
- Realización del mapa de historias de usuario.
- Aprendizaje de Javascript.
- Aprendizaje de React para construir la interfaz de usuario.

- Aprendizaje de Solidity para la generación de los Smart Contracts.
- Aprendizaje de Mocha y Chai para el testeo de los Smart Contracts.
- Aprendizaje de Redux.
- Aprendizaje de React-Router.
- Aprendizaje de Integración Continua y concretamente Github actions.
- Aprendizaje del framework de Semantic UI.
- Aprendizaje del framework de Ant Design.
- Investigación sobre Web3 e integración para conectar el front-end y los contratos.
- Realización de tutoriales introductorios a Solidity.
- Realización de tutoriales introductorios a React.
- Estudio de la herramienta Remix para la depuración de los Smart Contracts.
- Investigación sobre el token ERC20.
- Investigación sobre tokens no fungibles.
- Investigación sobre el token ERC721.
- Modificaciones de los ERCs para hacerlos no transferibles.
- Implementación del Smart Contract *Rewards.sol*.
- Integración y adaptación de los ERCs de OpenZeppelin
- Testing de los Smart Contracts.
- Investigación del despliegue de un Smart Contract.
- Investigación sobre Bloxberg.
- Investigaciones sobre el del Gas en Ethereum.
- Investigación sobre las tasas de transacciones en Ethereum.
- Realización de prototipo con TypeScript.

- Realización de curso sobre LaTeXpara la redacción de la memoria.
- Realización de tablas estadísticas a través de la página web de *Dune analytics*
- Redacción de la memoria.
- Traducción de los apartados en inglés de la memoria.
- Elaboración de los *mockups*.

8.2. Daniel Fidalgo Panera

Mis aportaciones a este proyecto han sido las siguientes:

- Investigación sobre los fundamentos de Blockchain
- Asistencia al meetup "*Tips de prototipado rápido de Dapps con Ethereum y React*" donde se repasaron mediante un enfoque práctico, flujos de trabajo habituales y de competencia profesional dirigidos al prototipado rápido de Dapps utilizando las tecnologías de Ethereum, Javascript React, así como servicios y librerías asociados
- Investigación sobre Truffle, Ganache y Drizzle así como la realización de los tutoriales oficiales.
- Realización del mapa de historias de usuario.
- Aprendizaje de Javascript.
- Aprendizaje de React para construir la interfaz de usuario.
- Aprendizaje de Solidity para la generación de los Smart Contracts.
- Aprendizaje de Mocha y Chai para el testeo de los Smart Contracts.
- Aprendizaje de Redux.
- Aprendizaje de React-Router.
- Aprendizaje de Integración Continua y concretamente Github actions.
- Aprendizaje del framework de Semantic UI.

- Aprendizaje del framework de Ant Design.
- Investigación sobre Web3 e integración para conectar el front-end y los contratos.
- Realización de tutoriales introductorios a Solidity.
- Realización de tutoriales introductorios a React.
- Estudio de la herramienta Remix para la depuración de los Smart Contracts.
- Investigación sobre el token ERC20.
- Investigación sobre tokens no fungibles.
- Investigación sobre el token ERC721.
- Modificaciones de los ERCs para hacerlos no transferibles.
- Implementación del Smart Contract *Rewards.sol*.
- Integración y adaptación de los ERCs de OpenZeppelin
- Testing de los Smart Contracts.
- Investigación del despliegue de un Smart Contract.
- Investigación sobre Bloxberg.
- Investigaciones sobre el del Gas en Ethereum.
- Investigación sobre las tasas de transacciones en Ethereum.
- Realización de prototipo con TypeScript.
- Realización de curso sobre LaTeXpara la redacción de la memoria.
- Realización de tablas estadísticas a través de la página web de *Dune analytics*
- Redacción de la memoria.
- Traducción de los apartados en inglés de la memoria.
- Elaboración de los *mockups*.

8.3. Otras aportaciones

Carlos Rodríguez Hernández hizo un getter de una línea en uno de los contratos.

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Conclusiones

Llegados a este punto podemos echar una vista atrás y decir que se han cumplido los objetivos marcados en el desarrollo del proyecto, además de aquellos que fueron añadiéndose por el camino.

El objetivo de este trabajo consistía en explorar las alternativas descentralizadas para gestionar recompensas de revisores. Creemos que hemos avanzado bastante explorando las opciones que nos ha ofrecido la tecnología blockchain y esperamos que pueda servir de referencia para investigaciones relacionadas o como base para proyectos futuros, ya que en cierto modo, de esto trata el software libre.

En este proyecto hemos realizado una intensa labor de experimentación respecto al mundo que rodea a los Smart Contracts. Durante este proceso hemos aprendido muchísimo, hemos digerido cantidades ingentes de documentación y hemos iterado sobre numerosos errores.

En esta labor hemos sido capaces de aprender cómo se despliega un Smart Contract, como funciona el concepto de Gas en Ethereum y como se ejecuta la blokchain a bajo nivel. También hemos investigado el posible despliegue en Bloxberg. Todo este trabajo nos ha aportado nuevos conocimientos sobre este mundo. Después de estas experimentaciones, hemos aprendido lo siguiente: sobre el precio del Gas, el papel de las alternativas de escalado dentro de Ethereum, aunque la mayor esperanza de descongestión es Ethereum 2.0.

Existe beneficio al utilizar Bloxberg, aunque no sea una plataforma muy popular, y concluimos que la optimización durante el diseño y desarrollo de Smart Contracts es una parte clave para reducir el coste de despliegue.

Sobre las tecnologías usadas podemos decir que muestran un gran potencial. Creemos que Ethereum va a seguir creciendo y que mas proyectos van a hacer uso de esta plataforma en el el futuro. En el caso de Solidity. al estar en una fase beta nos obligó a realizar varias iteraciones durante el desarrollo de los Smart Contracts según incrementábamos la versión del compilador utilizada. Respecto a React como framework para construir la interfaz de usuario, podemos decir que es un sistema robusto, con un aprendizaje más intuitivo que nos ha permitido crear interfaces declarativas de forma dinámica. Gracias a su sistema de componentes hemos sido capaces de encapsular las diferentes funcionalidades implementadas. También destacar las librerías que nos han facilitado el desarrollo de la aplicación, como es el caso de Drizzle.

Para concluir, podemos afirmar que este trabajo asienta unas bases para crear un sistema descentralizado, que tenga como objetivo dar visibilidad y reconocimiento a los revisores de artículos científicos y con ello contribuir al cambio de la situación actual referente a revisión de artículos científicos.

9.2. Trabajo Futuro

Con los resultados obtenidos en este proyecto, se ha considerado como posible trabajo futuro las siguientes tareas:

- Añadir soporte para poder subir y almacenar los artículos directamente en la aplicación utilizando IPFS, ya que el coste de almacenaje en la blockchain no es factible ni escalable ya que es excesivamente costoso [63].
- Integrar la aplicación con un proveedor de identidades para que los revisores puedan importar sus datos e historial mas fácilmente.
- Añadir sistema de filtrado en la tabla de revisores para permitir realizar búsquedas en base a unos parámetros.
- Añadir sistema de filtrado en la vista de artículos para permitir realizar búsquedas en base a unos parámetros.

- Optimizar los Smart Contracts aprovechando las funcionalidades de Solidity para reducir el uso de Gas.
- Desplegar los contratos en la red principal de Ethereum.
- Desplegar el front-end en un servidor para poder acceder a la aplicación desde cualquier lugar o dispositivo.
- Añadir funcionalidad para soportar múltiples revisiones por cada revisor.
- Implementar test automatizados de interfaz en el front-end con Selenium o Cypress.

Capítulo 10

Conclusions and Future Work

10.1. Conclusions

At this point we can look back and say that the objectives set in the development of the project have been met, in addition to those that were added along the way.

The objective of this proyect was to explore decentralized alternatives for managing reviewer rewards. We believe that we have made good progress exploring the options offered by blockchain technology and we hope that it can serve as a reference for related research or as a basis for future projects, since in a way, this is what free software is all about.

In this project we have done an intense work of experimentation regarding the world surrounding Smart Contracts. During this process we have learned a lot, we have digested huge amounts of documentation and we have iterated on numerous errors. In this work we have been able to learn how Smart contracts are deployed, how the concept of Gas works in Ethereum and how the blokchain is executed at a low level. We have also investigated the potential deployment on Bloxberg. All this work has given us new insights into this world. After these experimentations, we have learned the following: about Gas pricing, the role of scaling alternatives within Ethereum, although the biggest hope for decongestion is Ethereum 2.0. There is benefit in using Bloxberg, although it is not a very popular platform, and we concluded that optimization during the design and development of Smart Contracts is a key part to reduce the cost of deployment.

About the technologies used we can say that they show great potential. We believe that Ethereum will continue to grow and that more projects will make use of this platform in the future. In the case of Solidity, being in a beta phase forced us to make several iterations during the development of Smart Contracts as we increased the version of the compiler used. Regarding React as a framework to build the user interface, we can say that it is a robust system, with a more intuitive learning process that has allowed us to create declarative interfaces dynamically. Thanks to its component system we have been able to encapsulate the different functionalities implemented. We also highlight the libraries that have facilitated the development of the application, as is the case of drizzle.

10.2. Future Work

- Add support for uploading and storing articles directly in the application using IPFS, due to the cost of storage on the blockchain is neither feasible nor scalable because it is prohibitively expensive.
- Integrate the application with an identity provider so that reviewers can import their data and history more easily.
- Add a filtering system in the reviewers table to allow searches based on certain parameters.
- Add filtering system in the article view to allow searches based on parameters.
- Optimize Smart Contracts taking advantage of Solidity functionalities to reduce Gas usage.
- Deploy contracts on the Ethereum mainnet.
- Deploy the front-end on a server to be able to access the application from any location or device.
- Add functionality to support multiple reviews per reviewer.
- Implement automated front-end interface testing with Selenium or Cypress.

Bibliografia

[1] Jessica K Polka y col. *Publish peer reviews*. 2018.

[2] *Publons*. URL: https://publons.com/about/mission.

[3] *Principia Network*. URL: http://www.principia.network.

[4] Andrea Mambrini y col. "PRINCIPIA: a Decentralized Peer-Review Ecosystem". En: *arXiv preprint arXiv:2008.09011* (2020).

[5] Vlad Günther y Alexandru Chirita. "Scienceroot". En: (2018). URL: https://www.scienceroot.com/wp-content/uploads/2020/11/whitepaper.pdf.

[6] *Scienceroot*. URL: https://www.scienceroot.com.

[7] *Decentralized Science*. URL: decentralized.science.

[8] Tony Ross-Hellauer. "What is open peer review? A systematic review". En: *F1000Research* 6 (2017).

[9] Ambar Tenorio Fornés y col. "A decentralized publication system for open science using blockchain and ipfs". En: (2018).

[10] Ambar Tenorio-Fornés y col. "Towards a decentralized process for scientific publication and peer review using blockchain and IPFS". En: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. 2019.

[11] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". En: (2008). URL: www.bitcoin.org.

[12] URL: https://mirrors.edge.kernel.org/pub/software/scm/git/docs/user-manual.html#trust.

[13] Vitalik Buterin. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform". En: (2013). URL: ethereum.org/whitepaper.

[14] Alexander Savelyev. "Contract Law 2.0: «Smart» Contracts As the Beginning of the End of Classic Contract Law". En: (2016).

[15] Nick Szabo. "Formalizing and Securing Relationships on Public Networks". En: *First Monday* 2 (1997). URL: https://firstmonday.org/ojs/index.php/fm/article/view/548.

[16] David Chaum. "Blind Signatures for Untraceable Payments". En: (1983).

[17] *BTC Flow*. URL: https://www.bitpanda.com/academy/es/lecciones/que-es-la-mineria-de-bitcoin-y-como-funciona-la-mineria.

[18] Gavin Wood y col. "Ethereum: A secure decentralised generalised transaction ledger". En: *Ethereum project yellow paper* (2014).

[19] *EVM*. URL: https://ethereum.org/en/developers/docs/evm.

[20] Christoph Jentzsch. "Decentralized autonomous organization to automate governance". En: *White paper, November* (2016).

[21] Youssef El Faqir, Javier Arroyo y Samer Hassan. "An overview of decentralized autonomous organizations on the blockchain". En: *Proceedings of the 16th International Symposium on Open Collaboration*. 2020, págs. 1-8.

[22] William Mougayar. *The business blockchain: promise, practice, and application of the next Internet technology*. John Wiley & Sons, 2016.

[23] Maker Team. "The dai stablecoin system". En: *URl: https://makerdao.com/whitepaper/DaiDec17WP. pdf* (2017).

[24] Damiano Di Francesco Maesa y Paolo Mori. "Blockchain 3.0 applications survey". En: *Journal of Parallel and Distributed Computing* 138 (2020), págs. 99-114.

[25] "Neo: A distributed network for the Smart Economy". En: (2014). URL: docs.neo.org/docs/en-us/basic/whitepaper.html.

[26] Sharon Boeyen y col. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. 2008. URL: https://rfc-editor.org/rfc/rfc5280.txt.

[27] Serguei Popov. "The tangle". En: *White paper* 1 (2018), pág. 3.

[28] *ZK-Rollups*. URL: https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-rollups.

[29] *Optimistic Rollups*. URL: https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/optimistic_rollups.

[30] Joseph Poon y Vitalik Buterin. "Plasma: Scalable autonomous smart contracts". En: *White paper* (2017), págs. 1-47.

[31] Sandeep Nailwal y Milhailo Bjelic Jaynti Kanani Anurag Arjun. "Polygon. Ethereum's Internet of Blockchains". En: (febrero de 2021). URL: https://polygon.technology/lightpaper-polygon.pdf.

[32] *Layer 2 Scaling*. URL: https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/plasma.

[33] Ethan Buchman Jae Kwon. "Cosmos: A Network of Distributed Ledgers". En: (2016). URL: https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md.

[34] *Cardano*. URL: https://cardano.org.

[35] Gavin Wood. "Polkadot: Vision for a heterogeneous multi-chain framework". En: *White Paper* (2016).

[36] *Webassembly*. URL: webassembly.org.

[37] *Substrate*. URL: substrate.io.

[38] Ajay Reddy. *The Scrumban [r] evolution: getting the most out of Agile, Scrum, and lean Kanban*. Addison-Wesley Professional, 2015.

[39] Martin Fowler, Jim Highsmith y col. "The agile manifesto". En: *Software Development* 9.8 (2001), págs. 28-35.

[40] *Smart Contract*. URL: https://www.ibm.com/topics/smart-contracts.

[41] *Mocha*. URL: mochajs.org.

[42] *Chai*. URL: chaijs.com.

[43] *Truffle Suite*. URL: https://www.trufflesuite.com/truffle.

[44] Wei-Meng Lee. "Using the web3.js APIs". En: *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript*. Berkeley, CA: Apress, 2019, págs. 169-198.

[45] *Metamask*. URL: metamask.io.

[46] *React*. URL: https://es.reactjs.org/.

[47] *Ant Design*. URL: https://ant.design/.

[48] *Redux*. URL: https://es.redux.js.org/.

[49] *Figma*. URL: figma.com.

[50] Jorge V. "Tips de prototipado rápido de Dapps con Ethereum y React". En: (15 de octubre de 2020). URL: https://www.meetup.com/es-ES/Ethereum-Spain/events/273948330.

[51] *ERC*. URL: https://docs.ethhub.io/built-on-ethereum/erc-token-standards/what-are-erc-tokens/.

[52] Fabian Vogelsteller y Vitalik Buterin. "ERC-20 token standard". En: *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland* (2015).

[53] William Entriken y col. "Erc-721 non-fungible token standard". En: *Ethereum Foundation* (2018).

[54] *OpenZeppelin*. URL: https://docs.openzeppelin.com/openzeppelin.

[55] URL: https://www.scrummanager.net/bok/index.php?title=Epic.

[56] *Dapp architecture*. URL: https://www.geeksforgeeks.org/creating-dapps-using-the-truffle-framework.

[57] Kevin Ziechmann. "Gas and fees". En: (30 de marzo de 2021). URL: https://ethereum.org/en/developers/docs/gas.

[58] Anil Donmez y Alexander Karaivanov. "Transaction Fee Economics in the Ethereum Blockchain". En: (2021).

[59] *Eth Gas Station*. URL: https://ethgasstation.info.

[60] Leslie Burkholder. "The halting problem". En: *ACM SIGACT News* 18.3 (1987), págs. 48-60.

[61] *DEX*. URL: https://economia3.com/que-es-dex-solucion-intercambio-descentralizado/.

[62] Dr. Sandra Vengadasalam y James Lawton Friederike Kleinfercher. "Bloxberg. The Trusted Research Infrastructure". En: (febrero de 2020). URL: https://bloxberg.org/wp-content/uploads/2020/02/bloxberg_whitepaper_1.1.pdf.

[63] Juan Benet. "Ipfs-content addressed, versioned, p2p file system". En: *arXiv preprint arXiv:1407.3561* (2014).