Organización Descentralizada Autónoma para certificación de cursos online usando blockchain

Noelia Calderón Mateo
Daniel Valverde Menasalvas

UNIVERSIDAD COMPLUTENSE
MADRID

Trabajo de Fin de Grado
Grado en Ingeniería Informática

Dirigido por
Samer Hassan Collado
María Cruz Valiente Blázquez

Decentralized Autonomous Organization for the certification of online courses using blockchain

Madrid, 2020-2021

Organización Descentralizada Autónoma para certificación de cursos online usando blockchain

Memoria que se presenta para el Trabajo de Fin de Grado

Noelia Calderón Mateo
Daniel Valverde Menasalvas

Dirigido por

Samer Hassan Collado
María Cruz Valiente Blázquez

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, 2021

Agradecimientos

Queremos agradecer a los tutores de este trabajo, Samer Hassan Collado y María Cruz Valiente Blázquez, la gran cantidad de esfuerzo y dedicación que han puesto en él. Han estado constantemente disponibles para solventar nuestras dudas y ayudarnos a dar lo mejor de nosotros mismos, lo que sin duda ha quedado reflejado en el trabajo.

También agradecer a Marta Ranz González por su colaboración en el trabajo como tercera investigadora y desarrolladora. A toda la comunidad de desarrolladores de Ethereum y Aragon por hacer este proyecto posible y sentar las bases de una tecnología que posee un gran potencial. Finalmente, a todas las personas que han participado en la evaluación de este proyecto o que han compartido sus comentarios durante su desarrollo para ayudarnos a mejorar la experiencia de usuario.

Muchas gracias a todos.

Resumen

El desarrollo de este proyecto está enfocado en la investigación y experimentación de tecnología blockchain para afrontar los desafíos que presenta la economía colaborativa a través del enfoque de las Organizaciones Autónomas Descentralizadas (DAOs).

Para conseguir este objetivo, se realiza la implementación de una DAO llamada dAcademy. dAcademy es una red descentralizada de cursos que permite a los usuarios actuar como profesores y/o alumnos mejorando sus aptitudes y habilidades o compartiéndolas con el resto de la comunidad. La descentralización de esta aplicación permite que cualquiera pueda verificar la validez de cada certificado de finalización de curso de forma inmediata. Además, será posible obtener una representación del valor del curso a través del agregado de las valoraciones de la comunidad. De esta manera, es la propia comunidad la que crea los cursos, la que los consume y la que respalda su valor.

Esta aplicación se desarrolla sobre la plataforma Aragon que facilita el desarrollo y la creación de DAOs. Para la implementación del back-end, Aragon se fundamenta en el desarrollo de contratos inteligentes (smart contracts) en Ethereum, haciendo uso del lenguaje de programación Solidity. Respecto a la interfaz de usuario, Aragon está integrado con el conocido framework React de JavaScript, que permite un desarrollo versátil y accesible.

Palabras clave: Aragon, Colony, DAOstack, Ethereum, Bitcoin, Blockchain, DAO, Smart Contract, React, Solidity.

Abstract

This memory thoroughly relates how we experimented with blockchain technology with the aim of solving some of the many problems that colaborative economy presents nowadays using the approach of Decentralized Autonomous Organizations (DAOs).

With that purpose, we created a DAO called dAcademy. dAcademy is a decentralized comunity made of courses that grants users the opportunity to take part as students and/or instructors. Therefore, users can improve their knowledge and skills, or share them with the rest of the community. Decentralization is key in this application as it permits that anyone can verify a course certificate at any moment. Also, courses will be valued regarding the oppinions that users have of them, so every user will be able to find the course that best fits his needs. This way, an autonomous community is created where the community itself creates, consumes and backs the courses.

The application is developed using the Aragon framework which main purpose is to help creating and developing DAOs. Aragon uses Ethereum smart contracts coded in Solidity to take care of the back-end of the applications. The front-end is integrated with React, a popular JavaScript framework that makes the process of creating user interfaces easier.

Key words: Aragon, Colony, DAOstack, Ethereum, Bitcoin, Blockchain, DAO, Smart Contract, React, Solidity.

Índice general

Página

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El desarrollo de Internet posibilita a las personas conectarse rápidamente y sin intermediarios lo que permite colaborar unos con otros prestando servicios y obteniendo beneficios para ambos, dependiendo de las necesidades de cada uno. Esta actividad se conoce como economía colaborativa [9].

La economía colaborativa incluye todas aquellas funciones que realizan un intercambio de bienes y/o servicios pactado entre dos partes. Es un modelo centrado en la colaboración y la ayuda mutua que permite una gestión más óptima de los recursos, que de otra manera quedarían en desuso.

A través de la economía colaborativa, se puede conseguir un ecosistema basado en el compromiso, solidaridad y la generación de conocimiento explícito e implícito por parte de todos los participantes. Además, favorece un mayor ahorro económico y desarrollo sostenible, facilita la cooperación social y optimiza la asignación de recursos.

A pesar del crecimiento de las plataformas de este tipo como Airbnb [10], Uber [11], Ulule [12] o Vibbo [13] entre otras, aún existen limitaciones para conseguir un ecosistema totalmente descentralizado basado en la colaboración. Para ello, debemos eliminar definitivamente las entidades centralizadas que aprovechan los datos personales para poder beneficiarse de ellos. Por el contrario, se ha de conseguir una comunidad donde el poder, la toma de decisiones y los beneficios se repartan equitativamente entre todos sus miembros.

Durante los últimos años, las tecnologías están adaptando los modelos de negocio tradicionales a otros más enfocados a las necesidades del consumidor. Algo que sin el desarrollo y evolución de plataformas que ofrecen los mercados online, junto con las tecnologías de las redes P2P, no hubiese sido posible. Gracias a ello, se promueve un ecosistema en el cual hacer transacciones e intercambiar bienes resulta tan sencillo como proporcionar algunos datos y dar unos pocos clics.

Estos modelos y formas de gobernanza se han impulsado aún más con la aparición de tecnologías descentralizadas, como por ejemplo *blockchain*. Sobre todo tras el nacimiento de Ethereum [14], que amplía la posibilidad de creación de Organizaciones Autónomas Descentralizadas (DAO), organizaciones dirigidas por reglas codificadas mediante la implementación de contratos inteligentes (*smart contracts* en inglés). Tanto la gestión financiera de la DAO como las reglas que la gobiernan, están gestionadas por la blockchain.

Una DAO puede definirse como un organismo carente de jerarquías y de carácter democrático que habilita a los individuos y comunidades a intercambiar valor en un ecosistema dado y tomar decisiones conjuntas que afecten a la comunidad gracias a los mecanismos de gobernanza que establezca. Ethereum ha logrado un entorno de desarrollo de DAOs que promueve un modelo de organización con un sistema de toma de decisiones descentralizado, en el cuál las interacciones de los miembros es regulada a través de código.

Para facilitar el desarrollo de DAOs, se ha acrecentado la aparición de plataformas que facilitan la gestión y proporcionan herramientas para crear este tipo de aplicaciones. Entre ellas, destacamos Aragon, DAOstack, DAOhaus y Colony.

1.1. Motivación

Los conceptos de colaboración y compartición de recursos deben estar presentes cuando hablamos de economía colaborativa, pero la economía colaborativa que conocemos actualmente, aprovecha las ventajas de la tecnología y de la hiperconectividad, para fundar mercados bajo demanda en los que los usuarios en realidad no están colaborando. Al conocer el alcance de la tecnología blockchain y las posibilidades que ya ofrece siendo aún tan inmadura e innovadora, es interesante desarrollar una DAO con un propósito aplicable, que puedan usar los usuarios y que habilite lo que realmente entendemos como economía colaborativa descentralizada. Un ecosistema centrado en un propósito común que puede tratarse de un producto o un servicio y donde las comunidades tienen el papel fundamental.

Este tipo de proyectos no se está difundiendo lo suficiente como para darse a conocer por todo el mundo. Es por ello importante, promover su desarrollo y extender su conocimiento al resto de desarrolladores fuera de la comunidad y a todas las partes interesadas en la economía colaborativa.

Por otro lado, contamos con las plataformas como Aragon, DAOstack, DAOhaus y Colony, que están orientadas a facilitar la creación de DAOs. Por este motivo, uno de los objetivos del proyecto se centra en valorar qué ayuda ofrece cada uno de estos *frameworks*. Para ello, se investigará cómo se lleva a cabo el desarrollo de una DAO en cada uno de ellos y finalmente, se creará una aplicación a través de Aragon puesto que, actualmente, es la plataforma que ofrece más posibilidades de desarrollo.

A pesar de que poco a poco se va facilitando el desarrollo de las DAOs, fuera de la comunidad de blockchain sigue extendida la creencia de que el potencial de la tecnología está completamente limitado al ámbito monetario. Sin embargo, esto se encuentra lejos de la realidad ya que *blockchain* nos ofrece un amplio marco de posibilidades abarcando ámbitos tan diversos como la política o el derecho. Con el desarrollo de este trabajo, pretendemos demostrar como a través de las DAOs se pueden favorecer otros tipos de uso de esta tecnología más allá del ámbito financiero.

Para ello, nos vamos a centrar en la problemática que presenta la formación de cursos online. Creemos que a través de una DAO se puede mejorar la accesibilidad al mercado educativo y facilitar la comunicación entre el profesor y el alumno. Además, se pretende equilibrar los intereses entre ambas partes y aportar veracidad a la validez de los títulos o certificados obtenidos. A través de la DAO, se podrá obtener un reconocimiento global de las certificaciones de cada usuario, así como obtener una reputación acorde a la calidad

de las enseñanzas que se imparten.

1.2. Objetivos

El objetivo principal de este trabajo de fin de grado es desarrollar una aplicación descentralizada que se distinga del ámbito financiero haciendo uso de la tecnología blockchain, que ayude al crecimiento de ecosistemas de economía colaborativa.

Para conseguir el objetivo principal se definen los siguiente subobjetivos:

1. Entender el funcionamiento de la tecnología *blockchain* y el concepto de DAO.
2. Implementar la lógica de negocio a través de *smart contracts* haciendo uso del lenguaje de programación Solidity.
3. Explorar las posibilidades de las DAOs fuera del ámbito financiero y fomentar su desarrollo.
4. Comparar las diferentes plataformas existentes para el desarrollo de DAOs.
5. Utilizar el framework de Aragon para implementar una DAO y comprobar la validez de la misma.

Para dar una solución al problema de formación de cursos que presentamos en el apartado anterior, implementamos dAcademy. Pretendemos ofrecer a los usuarios una red donde pueden ampliar sus aptitudes y conocimientos realizando cursos que ofrece la comunidad. Añadiendo el ingrediente de la descentralización, conseguimos llegar a todo el mundo, dando lugar a una validación descentralizada de los títulos y asegurando la autenticidad de estos. Además, conseguimos completa transparencia en la valoración de los cursos y de sus ponentes. Por otro lado, para asegurarnos de que los intereses de los alumnos y profesores estén alineados, creamos los incentivos necesarios mediante el uso de tokens y smart contracts.

Introduction

Internet developments allow people from all over the world to be permanently interconnected without intermediaries. In this way, people collaborate giving and receiving goods and services looking to fulfill their needs and desires. This activity is commonly known as collaborative economy.

Collaborative economy includes all exchanges of goods and services in a free agreement between two parts. It is a model based in social cooperation that permits a better distribution of the resources.

Although some platforms like Airbnb, Uber, Ulule or Vibbo are growing a lot, there are still limitations to obtain completely decentralized communities. Centralized entities are still very widely extended, making use of people personal data to obtain private benefit. The aim of collaborative economy is to achieve communities where power and decision making are distributed in a decentralized way between their members.

In the recent years, some companies are using technology to decentralize their business models. Online markets together with P2P networks are the technologies that have made it possible. Currently, doing online transactions is something anyone can do with just a few clicks.

This governance models have evolved even more with the development of decentralized technologies such as *blockchain*. Using Ethereum[14], it is now possible to create organizations guided by rules coded in *smart contracts*. This kind of organizations are known as Decentralized Autonomous Organitations (DAOs). The rules coded in the smart contracts can be consulted by anyone, and thoroughly detail what can and cannot be done inside each DAO. Thus, a DAO can be described as a democratic organism without any hierarchy that enable individuals to share services and goods and take community decision following the established governance mechanisms.

There are some frameworks that provide tools to ease the development of DAOs. Some of the most notorious ones are Aragon, DAOstack, DAOhaus and Colony.

Motivation

Collaboration and resource sharing are essential characteristics of collaborative economy. However, technology developments and hyperconnectivity are currently use to create on demand markets where users are not really collaborating. Knowing the possibilities that blockchain technology bring us, and the fact that it is a very innovative technology, we decide to develop a DAO with the purpose of solving some real users' needs and create

an environment where collaborative economy can take place. An ecosystem focused on a common purpose where the community have the main role.

This kind of projects are still not known by most people. Therefore, it is a good time to enhance their development and try to reach as many people as possible with them.

On the other hand, we have platforms like Aragon, DAOstack, DAOhaus and Colony that are oriented to ease the creation of DAOs. This is why one of the main purposes of this project will be to evaluate each of those frameworks. In order to do that, we will try to create a DAO in each of those platforms and, finally, we will develop a DAO with all the functionalities needed in Aragon, as it is the platform that currently gives more development possibilities.

Although DAOs development is getting easier with the time, people outside of blockchain community already think that blockchain capabilities are restricted to monetary means. However, this is far from true as blockchain gives us a huge world of possibilities including fields like politics of law. With the development of this project, we aim to demonstrate, using the possibilities of a DAO, that blockchain can mean much more than just a monetary progress.

The field we decided to work with is online courses. We think that, using the possibilities that a DAO brings to us, online education can be made more accessible and communication between teachers and students can be improved. We also aspire to align the interests of both parties and assure the veracity of each course certificate. The DAO will naturally create a community recognition for each course taking into account the opinions of the users that take the course.

Objectives

The main objective of this final essay is to develop a decentralized blockchain application outside of the financial world that helps to produce an environment where collaborative economy takes place. In order to achieve that main purpose, we define the following subobjectives:

1. Understand how *blockchain* technology works and the DAO concept.
2. Implement business logic thought *smart contracts* using Solidity coding language.
3. Explore DAOs possibilities outside the financial world and enhance their growing.
4. Compare different DAOs frameworks.
5. Use Aragon framework to implement a DAO and validate it.

Capítulo 2

Estado del arte

2.1. Blockchain y Bitcoin

Estamos en el inicio de una nueva revolución tecnológica que aspira a ser también financiera, social y política. Una revolución que empezó con una moneda alternativa llamada Bitcoin que era emitida y respaldada no por una autoridad centralizada, si no por un consenso entre individuos interconectados. Su verdadera originalidad residía en que los individuos no tenían que confiar los unos en los otros y, por la forma en la que estaba diseñada, cualquier intento de defraudar el sistema sería rechazado. A día de hoy, la capitalización del mercado de las criptomonedas es de 1,5 billones de dólares, lo que indica que ya es ampliamente usado como reserva de valor.

2.1.1. ¿Qué es Bitcoin?

Bitcoin (BTC) es una moneda digital y un sistema de pagos en línea en el que se usan técnicas de encriptación para regular la generación monetaria y verificar las transferencias, operando de forma independiente a un banco central [15]. Bitcoin fue lanzado el 9 de enero de 2009 por una personalidad anónima que utilizaba el pseudónimo de Satoshi Nakamoto. El concepto inicial y todos los detalles pueden consultarse en un artículo llamado "Bitcoin: Una forma de pagos electrónicos de igual a igual"[16]. Los pagos usando la moneda virtual descentralizada, quedan reflejados en un registro público almacenado en numerosos (potencialmente todos) ordenadores de usuarios de Bitcoin, accesible en cualquier momento a través de Internet. Bitcoin es la primera y más grande criptomoneda descentralizada y el primer activo digital real (tiene valor propio, no es un derecho de cobro de otro activo).

Desde su creación han surgido cientos de criptomonedas alternativas (*altcoins*) como Litecoin o Dogecoin, sin embargo, Bitcoin a día de hoy tiene el 45 por ciento (ver figura 2.1) de la tasa de capitalización del mercado de las criptomonedas siendo el estándar por defecto. Bitcoin es una red pseudónima, no anónima, ya que se usan claves de dirección públicas (cadenas de 27 a 32 caracteres alfanuméricos que hacen una función similar a la dirección de correo electrónico), en vez de información personal, para enviar y recibir bitcoins (BTC) y dejar las transacciones grabadas.

	Nombre	Símbol...	Precio (USD)	Cap. mercado
	Bitcoin	BTC	56.858,3	1,06T $
	Ethereum	ETH	3.553,81	411,80B $
	Binance Coin	BNB	635,43	97,41B $
	Dogecoin	DOGE	0,605530	77,61B $
	Ripple	XRP	1,67727	76,27B $
	Tether	USDT	1,0009	53,56B $
	Cardano	ADA	1,621816	51,65B $
	Polkadot	DOT	41,41690	38,28B $
	Bitcoin Cash	BCH	1.508,00	28,46B $
	Litecoin	LTC	345,391	23,20B $

Figura 2.1: Capitalización de mercado de las principales criptomonedas a 6 de mayo de 2021 [1].

Las unidades de Bitcoin se crean como compensación por trabajo computacional, conocido como *minería*, en el que los usuarios ofrecen la potencia computacional de sus ordenadores para verificar transacciones y dejarlas grabadas en el registro público. Gran cantidad de individuos o compañías se dedican a minar a cambio de comisiones de transacción y BTC de nueva creación. Además, se pueden intercambiar otras monedas *fiat* (euro, dolar, etc.) o productos y servicios, para obtener bitcoins. Los usuarios pueden enviar y recibir bitcoins de forma electrónica usando *monederos digitales* desde su ordenador, móvil o aplicación web.

2.1.2. ¿Qué es blockchain?

Blockchain (cadena de bloques) es el registro público de todas las transacciones de Bitcoin que han sido ejecutadas desde su creación [15]. Cuando un usuario de la red decide hacer una transacción, se lo comunica a los demás y estos anotan la transacción en sus registros dentro de un bloque, aunque la transacción aún no está en la red blockchain de forma definitiva. Cuando más transacciones se van solicitando, llega el momento en el que el bloque se llena y se debe *validar* o **sellar**, proceso llevado a cabo por los mineros. La minería consiste en una competición de los mineros por solucionar una serie de complejos cálculos que requiere de tiempo y, cada vez más, electricidad. Cuando algún minero consigue solventar el problema, este consigue una recompensa en forma de criptomoneda de la red. Además, el bloque queda registrado de forma permanente en la cadena de bloques y no puede ser modificado por nadie (ver figura 2.2). Los bloques son añadidos de forma lineal en orden cronológico a la red. Cada ordenador en una red *blockchain* que valida transacciones (también llamados nodos), tiene una copia de todas las transacciones que se han realizado desde la creación de esta. El hecho de que el registro sea público hace sencillo realizar consultas desde un explorador de la cadena de bloques [17].

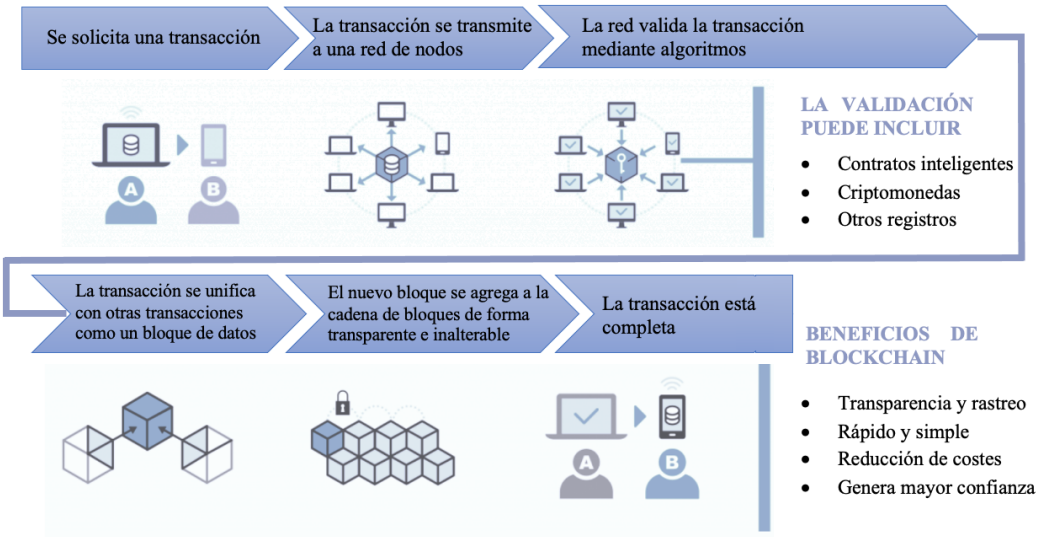

Figura 2.2: Funcionamiento de la tecnología *blockchain* [2].

La característica más fascinante de *blockchain*,es que verifica todas las transacciones que se efectúan sin necesidad de que los usuarios confíen unos en los otros, ni tampoco en terceras partes, como puede ser un banco u otras instituciones. Esto dota al proceso de completa transparencia y da al usuario soberanía total sobre sus pertenencias en la red. De esta manera, esta tecnología permite intercambiar criptomonedas, cerrar contratos financieros e incluso intercambiar cualquier otro tipo de bienes ya sean físicos o digitales. Además, puede no ser solo utilizada para transacciones sino también para mantener un registro completamente exhaustivo de cualquier conjunto de activos, desde propiedades físicas a bienes intangibles como votos, ideas, intencionalidades, registros médicos, etc.

2.1.3. Monedas, smart contracts y aplicaciones más allá de los mercados financieros

La primera aplicación basada en *blockchain* que nos viene a la cabeza es la monetaria, ya que permite a sus usuarios protegerse frente a la manipulación monetaria de los emisores (bancos centrales) de monedas centralizadas como el euro o el dólar estadounidense. Sin embargo, el potencial del uso de *blockchain* no es solo económico, también llega a ámbitos políticos, sociales y científicos [18] [19] [20]. Por ejemplo, a la hora de combatir regímenes políticos autoritarios, *blockchain* puede servir para realizar funciones que antes necesitaban de una administración jurisdiccional centralizada. Por ejemplo, en casos como el de WikiLeaks (donde el gobierno prohibió a los procesadores de pagos aceptar donaciones en la polémica situación de Edward Snowden) y también para organizaciones internacionales, como los grupos de estándares de Internet ICANN y DNS. A parte de estas aplicaciones donde se permite que el interés público sobrepase las estructuras de poder gubernamentales, otras industrias y colectivos sociales pueden ser librados de situaciones de hiper regulación sujetos a estructuras jerárquicas y a la influencia gubernamental de grupos de presión, permitiendo la creación de nuevos modelos de negocio descentralizados. A pesar de las regulaciones promovidas por los grupos de presión institucionales, nuevos modelos de economía parcialmente descentralizada, como Airbnb y Uber, han surgido y

se han mantenido firmes ante los ataques legales de los grupos de presión afectados.

Además de los beneficios políticos y económicos, la coordinación, los registros y la irrevocabilidad de las transacciones son características que pueden llegar a ser verdaderamente fundamentales en el progreso de nuestras sociedades. En este sentido, *blockchain* puede servir como un registro público de nuestras sociedades, donde quede guardada toda la información referente a documentos, eventos, identidades y bienes [21]. En este sistema, la propiedad se podría convertir en *propiedad inteligente* [22], lo que significa que cada bien material tendría un identificador único en *blockchain* permitiendo rastrearlo, controlarlo e intercambiarlo. Es decir, todos los bienes tanto digitales como físicos, podrían estar registrados y ser intercambiados a través de *blockchain*.

Podemos ver un ejemplo de esta utilidad en el mundo de la propiedad intelectual. La industria emergente del arte digital ofrece servicios para registrar de forma privada el contenido exacto de un bien digital (archivos, imágenes, registro médico, programa, etc.) en *blockchain*. Lo que nos lleva a pensar que podría reemplazar todos los sistemas de gestión de propiedad intelectual. El funcionamiento consiste en ejecutar un algoritmo sobre el archivo en cuestión, que será comprimido en un código único de 64 caracteres (llamado *hash*). Esto se puede hacer sin importar el tamaño del archivo el cual será transformado mediante un proceso que no podrá deshacerse. El código *hash* se guarda en la red *blockchain* junto a la fecha, sirviendo como prueba de que el archivo existió en ese momento. El código se podrá volver a generar a partir del archivo, confirmando que el contenido original no ha sido modificado. Este mecanismo podría ser un punto de inflexión para la coordinación a gran escala de nuestras sociedades ya que la economía está cada vez más dirigida por la creación de ideas.

Podemos relacionar este último tema con la reciente popularización de los tokens no fungibles (del ingles non-fungible token, NFT). Últimamente, numerosos artistas y personajes influyentes (Logan Paul, Ozuna, Banksy, Lil Pump, Cristiano Ronaldo, Elon Musk, etc.) se han aprovechado de ello para la creación y comercialización de ediciones limitadas de productos digitales.

2.1.4. Hacia la adopción generalizada: Confianza, accesibilidad y facilidad de uso

Uno puede pensar que Bitcoin y *blockchain* difícilmente llegarán a ser de uso generalizado debido a que se sustentan en una base complejos conceptos técnicos que no están al alcance del público general. Sin embargo, lo mismo pasaba con Internet y ahora la gran mayoría de la gente lo considera una parte indispensable de su vida. La barrera técnica no es en realidad una barrera ya que, de la misma manera que puedes utilizar Internet sin tener conocimientos previos de como funcionan los protocolos TCP/IP, se puede utilizar *blockchain* sin tener que conocer en profundidad los detalles técnicos de una dirección pública alfanumérica. Simplemente es cuestión de que vayan surgiendo aplicaciones con una interfaz sencilla y confiable, adecuada para el público general. Es verdad que al tratarse de cuestiones relacionadas con el dinero, los usuarios necesitarán construir una fuerte confianza en los productos antes de usarlos, pero ya hemos presenciado un proceso similar con el fenómeno de las compras por Internet.

Actualmente, ya contamos con aplicaciones de compra e intercambio de criptomonedas (del inglés *exchanges*) con buenas interfaces de usuario y gran reputación (ver figura 2.3).

Haciendo uso de una aplicación monedero como Metamask [23], prácticamente cualquier persona puede introducirse en el mundo de blockchain de forma fácil y segura. Además, la propia página oficial de Bitcoin ofrece información sobre los exchanges disponibles en cada región e incluso un test para elegir qué monedero es mejor dado un perfil de usuario [24].

#	Nombre	Puntaje del exchange	Volumen (24h)	Liquidez prom.	Visitas semanales	# Mercados	# Monedas
1	Binance	9.9	$46,961,259,016 ▼ 30.73%	662	27,657,536	1230	361
2	Huobi Global	9.2	$15,809,853,501 ▼ 37.29%	612	1,342,816	971	332
3	Coinbase Pro	8.9	$7,320,540,308 ▼ 29.48%	497	3,577,326	207	64
4	Kraken	8.4	$2,830,572,939 ▼ 36.93%	497	2,938,774	285	62
5	KuCoin	8.3	$2,514,239,672 ▼ 24.79%	450	1,381,926	760	333

Figura 2.3: Exchanges más populares. Datos del 15 de mayo de 2021 [3].

Las aplicaciones que usan blockchain como solución monetaria parecen casi listas para un uso masivo, pero las potenciales aplicaciones de *blockchain* van mucho más allá. Por ejemplo, podríamos llegar a presenciar servicios de notaría virtuales mucho más rápidos y sencillos que los actuales para registrar propiedad intelectual, contratos, testamentos y documentos similares. Por supuesto, habrá situaciones en las que la interacción humana siga siendo preferible a un simple argumento de eficiencia, pero eso no quita que tengamos un abanico enorme de posibilidades por delante.

Desde luego, si la industria de *blockchain* llega a extenderse ampliamente por la sociedad lo hará pasando por varias fases, de la misma manera que sucedió con Internet. Hasta la fecha, *blockchain* es usado por personas especialmente preocupadas por la teoría monetaria y la ideología política. Los próximos pasos probablemente pasen por servir de solución a problemas mayores de grupo concretos de personas, como la censura gubernamental en Internet o las litigaciones de propiedad intelectual. En cualquier caso, y aunque aún quede mucho camino por delante, es muy probable que *blockchain* sea uno de los grandes protagonistas en los avances tecnológicos y sociales de los próximos años.

2.2. Ethereum

Ethereum es comúnmente definido como 'el ordenador global', pero ¿qué significa esto en realidad?. Ethereum es una infraestructura descentralizada de computación de código libre que ejecuta programas llamados *contratos inteligentes* [25] (del ingles *smart contract*). Utiliza tecnología *blockchain* para sincronizar y registrar los cambios de estado del sistema, junto a una criptomoneda llamada *ether* (ETH) utilizada para medir y restringir los recursos necesarios para las ejecuciones. La plataforma de Ethereum permite a los desarrolladores construir potentes aplicaciones descentralizadas. A la vez que proporciona alta disponibilidad, fácil auditoría, transparencia y neutralidad, también ayuda a

combatir la censura y a reducir riesgos mediante la eliminación de intermediarios.

A diferencia de Bitcoin, el propósito principal de Ethereum no es ser un método de pago digital. Aunque la existencia del *ether* es fundamental para la existencia de Ethereum, está diseñado como una moneda utilizada para pagar por el uso de la plataforma Ethereum como un ordenador global. Ethereum está construido con la intención de ser una red programable de propósito general, que lance una máquina virtual capaz de ejecutar código sin límites de complejidad. Mientras que el lenguaje de Bitcoin es intencionadamente limitado para hacer evaluaciones booleanas de transacciones, el lenguaje de Ethereum es *Turing complete*, es decir, que cualquier programa, independientemente de su complejidad, puede ser ejecutado en Ethereum. Los usuarios pagan tasas (denominadas *gas*) a cambio de la capacidad de computación necesaria para ejecutar sus smart contracts. Además, estas tasas también garantizan que un programa no se ejecutará de manera indefinida.

2.2.1. El surgimiento de Ethereum

En los inicios de *blockchain*, los desarrolladores empezaron a buscar aplicaciones más allá de las puramente monetarias. Se encontraron con el el dilema de si intentar construir sobre Bitcoin sorteando sus restricciones o directamente crear una nueva red *blockchain*. Para proyectos que buscaran mayor libertad y flexibilidad una nueva red *blockchain* era la única opción, pero eso suponía una gran cantidad de trabajo.

En diciembre de 2013, Vitali Buterin, un joven programador y entusiasta de Bitcoin publicó un artículo [26] en el que proponía la idea principal de Ethereum: una red *blockchain* de propósito general y Turing complete. La idea era que, al usar Ethereum, los desarrolladores pudieran programar sus aplicaciones sin tener que implementar los mecanismos subyacentes de *blockchain*. Ethereum era un proyecto destinado a abstraer a los desarrolladores de todos esos detalles y proporcionarles un entorno de programación seguro y determinista para aplicaciones descentralizadas usando *blockchain*. Los fundadores de Ethereum trabajaron durante años hasta que, el 30 de julio de 2015 [27], el primer bloque de Ethereum fue minado. El ordenador global empezó a funcionar.

2.2.2. Cultura de desarrollo en Ethereum

Ethereum no solo marcó un punto de inflexión a nivel tecnológico y de objetivos, si no que también se diferenció claramente en su cultura de desarrollo. El progreso en Bitcoin está guiado por principios conservadores, es decir, antes de dar cada paso se aseguran de que este sea compatible con los sistemas actualmente vigentes. En contraposición a esto, en Ethereum se implementan cambios siempre que estos sean pertinentes, aunque esto signifique invalidar asunciones previas, romper compatibilidades o forzar a los clientes a actualizarse. Este tipo de cultura está caracterizada por la rápida evolución y la voluntad de desplegar mejoras con vistas a futuro, aún a costa de generar incompatibilidades. Esto ha generado que se hayan creado numerosas bifurcaciones del proyecto (del inglés *fork*) desde su creación a la actualidad (ver figura 2.4).

Uno de los grandes retos de los desarrolladores de Ethereum, es la inherente contradicción de desplegar código en un sistema supuestamente inmutable que a la vez es una plataforma en constante evolución. No puedes simplemente actualizar tus smart contracts, debes

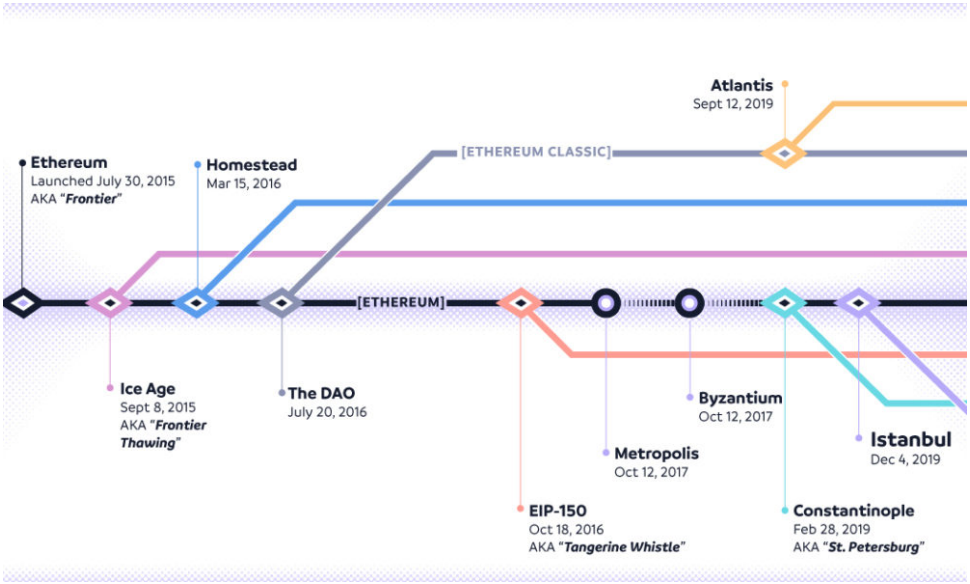

Figura 2.4: Principales forks de Ethereum hasta el año 2019 [4].

estar preparado para crear nuevos, migrar usuarios, aplicaciones, fondos y empezar de nuevo.

Irónicamente, esto también significa que el objetivo de construir sistemas autónomos descentralizados aún no es del todo realizable. La autonomía requiere una mayor estabilidad en la plataforma, la cual seguramente se consiga con el paso de los años. Para que la plataforma evolucione, has de estar preparado para desechar y reiniciar tus smart contracts, lo que significa que mantendrás un cierto grado de control sobre ellos.

Todo esto tiene un lado positivo, y es que Ethereum se está moviendo muy rápido. Es un entorno en el que debes estar preparado para cambios y actualizaciones constantes. Llegará el momento en el que el desarrollo de la plataforma se ralentice y las interfaces se vuelvan estáticas, pero hasta entonces, la innovación será el principio fundamental de Ethereum.

2.3. DAOs

Desde que Bitcoin eliminó a los intermediarios en las transacciones financieras, la comunidad de *blockchain* se ha interesado por un modelo de gestión que establezca una empresa u organización que funcione de manera transparente y sin la necesidad de una estructura jerárquica y que además, se gobierne y tome decisiones por sí misma. Después de profundizar en el funcionamiento de *blockchain* y las posibilidades que nos ofrece en los puntos anteriores, nos vienen a la mente ejemplos de aplicaciones (una plataforma que otorgue visibilidad a artistas emergentes, una comunidad de noticias veraces, etc.) que haciendo uso de esta tecnología, pueden tomar sus características para ser descentralizadas, transparentes y colaborativas.

Una Organización Autónoma Descentralizada (*Decentralized Autonomous Organization, DAO*) [28] es una entidad distribuida, alojada en Internet que existe de manera autónoma pero que requiere de la participación de los humanos para que interactúen en base a un acuerdo predefinido. No tiene ánimo de lucro, aunque los integrantes pueden ganar dinero

participando en su ecosistema.

Inicialmente, Bitcoin fue considerado la primera DAO por su funcionamiento autónomo y la coordinación a través de un protocolo de consenso. Las DAOs están totalmente controladas por programas informáticos llamados *smart contracts* que definen las reglas de la organización y se ejecutan en *blockchain* siendo, de esta manera, transparentes y descentralizados. Aunque los *smart contracts* son posibles gracias a la creación de Bitcoin, puesto que es necesario que haya transacciones programables sobre un sistema financiero que las reconozca, el uso de los mismos ha sido más accesible a los usuarios gracias a la plataforma de Ethereum.

Estas nuevas organizaciones presentan diferencias notables con respecto a las organizaciones tradicionales, que suelen estar estructuradas con una jerarquía vertical conformada por distintas capas de poder. La información se comunica hacia la parte superior de la jerarquía donde se realiza la toma de decisiones. En cambio, en una DAO no existe tal jerarquía, las reglas son completamente transparentes y las decisiones se toman conjuntamente mediante la votación de todos los participantes, añadiendo más influencia en función de la cantidad de tokens que posean.

2.3.1. Entonces... ¿Qué es una DAO?

El concepto de DAO aún no está lo suficientemente claro, no existe una definición exacta que podemos tomar para entender en definitiva en qué consiste, debido al proceso de cambio constante en el que se encuentra. El concepto y definición de DAO ha ido evolucionando a lo largo de los años [29]. Es por ello que consideramos tan necesario seguir desarrollando e investigando este tipo de sistema basado de la tecnología blockchain, que contienen un gran potencial pero que aún no se conocen lo suficiente. Sin embargo, numerosos artículos describen las características que estas deben tener para ser consideradas organizaciones autónomas descentralizadas.

Tomando todas las definiciones y características obtenidas de las referencias anteriores, entendemos como DAO a un sistema que representa una organización alojada en Internet, que ejecuta sus funciones de manera autónoma sin depender de una administración central [30]. Implementa un conjunto de reglas automáticas y públicas descritas en un *smart contract*, que dictan el funcionamiento que debe seguir. Hace uso de la tecnología *blockchain* para ofrecer transparencia, inmutabilidad, autonomía y seguridad a los usuarios que participan en ella.

Sus principales ventajas residen en reducir la toma de decisiones de forma centralizada en las que el poder se centra en unas pocas personas, disminuir costes burocráticos y de gestión e implantar un mecanismo de seguridad regulado mediante código.

En la figura 2.5 podemos apreciar las diferencias entre aplicaciones centralizadas y descentralizadas.

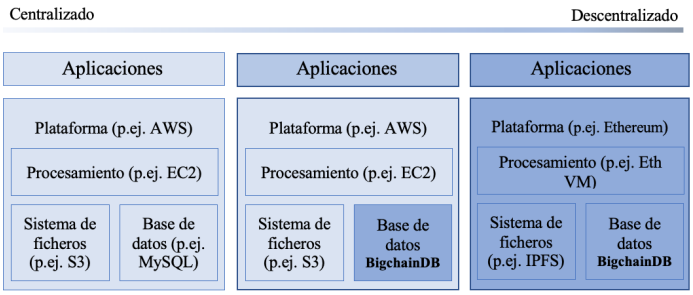

Figura 2.5: Diferencias entre estructuras centralizadas y descentralizas [5].

2.3.2. The DAO, la primera Organización Autónoma Descentralizada

Un ejemplo bastante conocido es The DAO [31], una organización que tenía como objetivo construir la DAO más grande jamás conocida usando el poder de Ethereum. Se lanzó el 30 de abril de 2016 como una organización autónoma descentralizada siendo la primera DAO de gran escala sobre Ethereum. La campaña de venta de participaciones fue todo un éxito en el cual se recaudaron millones de dólares a los pocos días de su lanzamiento, consiguiendo alcanzar un récord en colecta de micro-mecenazgo (crowdfunding). Siendo un fondo de capital de riesgo, tenía como objetivo proporcionar un modelo comercial para organizar empresas comerciales y sin fines de lucro con características *blockchain*. Un fondo de inversión completamente descentralizado y transparente que cualquiera podía ver y auditar.

Ofrecía a los usuarios decidir qué proyectos financiar, sin la necesidad de intermediarios para llevar a cabo el proceso. Un usuario podía proponer un proyecto y buscar la financiación de la comunidad. Mediante un proceso de votación entre los poseedores de tokens y siguiendo un consenso de más del 20 %, los fondos eran destinados al solicitante.

A pesar del éxito que estaba teniendo, la alegría duró poco. Un mes después de su lanzamiento, algunos usuarios señalaron una serie de vulnerabilidades advirtiendo a los inversores. Informaban de problemas relacionados con las llamadas recursivas. En junio, The DAO sufrió un ataque que explotaba varias vulnerabilidades entre las que se encontraba aquella de la que tanto se había hablado. Se comprometieron millones de dólares en Ether. El atacante se aprovechó de una función de *split* para transferir fondos a una pequeña DAO que pasado un periodo de 28 días, podría retirar.

Después de varios debates sobre cómo llegar a una solución tras el ataque, la red de Ethereum se decantó por realizar un *fork* para poder recuperar los fondos de los propietarios originales. Algunos usuarios se opusieron a esta bifurcación y continuaron usando la rama que ahora se conoce como Ethereum Classic.

2.3.3. Ejemplos de DAOs

MolochDAO

MolochDAO [32] es una organización autónoma descentralizada, creada con el objetivo de encontrar una solución a los problemas de incentivos y coordinación en la financiación

de protocolos de código abierto. Generalmente todas las partes interesadas se benefician del proceso pero, sin embargo, nadie quiere proporcionar la financiación necesaria.

Los miembros contribuyen con una propuesta beneficiando a la comunidad y con una cantidad de *ether* (no se utiliza un token propio), que les ofrece acciones con derecho a voto proporcionales a esa cantidad. Cualquier miembro puede proponer una subvención que se someterá a una votación para ser aprobada en consenso. Desde la página de MolochDAO [33] se pueden consultar los proyectos financiados gracias a esta DAO, por ejemplo Tornado Cash, una herramienta para realizar transacciones privadas. Para realizar esta financiación, cada usuario contribuye con una cantidad en función del porcentaje de participación que posea. Si finalmente se aprueba una propuesta, el solicitante en cuestión puede retirar los fondos, pero el resto de miembros debe hacerse cargo de la carga financiera del usuario que abandona. Esto se hace para evitar votos o subvenciones polémicas o que no son viables.

Esta DAO tiene dos *smart contracts* que gestionan su funcionamiento. El primero se encarga de la gestión de los miembros, votación y presentación de propuestas. El segundo administra los activos de la comunidad.

MakerDAO

MakerDAO [34] un proyecto de finanzas descentralizadas (DeFi) de código abierto, basado en la cadena de bloques de Etherum que tiene como principal objetivo ofrecer una moneda imparcial, estable, descentralizada y virtual que pueda usar cualquier persona, en cualquier lugar y en cualquier momento, según sus necesidades. Esta moneda se llama DAI y su valor es de 1 USD.

Una cantidad significativa de personas, no tiene acceso a un banco actualmente y DAI puede ayudar a esas personas a gestionar sus finanzas con solo acceso a Internet.

El proyecto es gestionado por usuarios que son propietarios del token de gobernanza [35] del protocolo Maker (MKR). Estos miembros se encargan, mediante el sistema de gobernanza y votación, de garantizar la estabilidad, transparencia y eficiencia de DAI. El poder de decisión del votante va en función de la cantidad de MKR con el que participe.

Por otro lado, cuentan con el Protocolo Maker, una aplicación financiera descentralizada y significativa que permite a los usuarios crear DAI. Un usuario tiene la posibilidad de bloquear una cantidad determinada de ETH para generar ganancias de DAI. Es decir, cuanto más ETH bloquee, más DAI ganará por el tiempo que permanezca bloqueado. Una vez generado, se puede usar de la misma manera que cualquier otra criptomoneda.

Los usuarios también pueden pedir prestado DAI de hasta el 66 % del valor de su garantía o generar intereses con la tasa de ahorro de DAI (DSR) (ver figura 2.6).

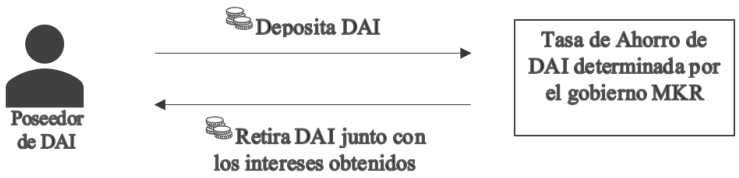

Figura 2.6: Un usuario deposita DAI para generar intereses gracias a la tasa de ahorro y después de un tiempo puede recoger sus ganancias.

Decentraland

Decentraland [36] se describe como el primer mundo completamente descentralizado que es gestionado a través de *smart contracts*, que conceden a los usuarios el poder de decidir mediante votación cómo funciona el mundo. Se permite crear tu propio mundo, personalizar tu personaje y comprar productos, con la moneda MANA a través de tu monedero virtual, respaldados por Ethereum. Los miembros deciden cómo llevar a cabo la organización, qué tipo de artículos se permiten, moderación de contenido, políticas, subastas, etc.

Las votaciones se llevan a cabo en la interfaz de gobierno de la DAO (ver figura 2.7), impulsada por Aragon [37].

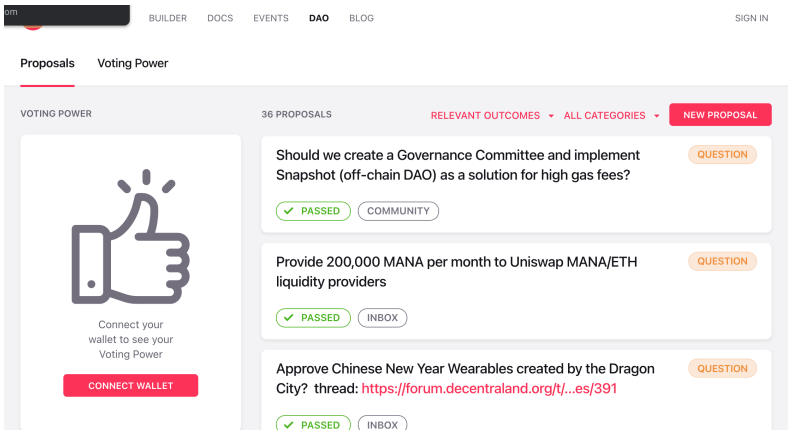

Figura 2.7: Interfaz de gobernanza de Decentraland DAO.

2.4. Frameworks

La idea inicial del proyecto era experimentar con diferentes frameworks e implementar nuestra aplicación en varios de ellos para poder compararlos. Sin embargo, como explicamos detenidamente a continuación, DaoStack y Colony generaron tantos problemas que, finalmente, se decidió abandonarlos y solo implementar la aplicación en Aragon. A continuación, se realizará una pequeña introducción de estas tres plataformas que además

de ser DAO, tienen el propósito de fomentar la creación y desarrollo de otras DAOs, sirviendo de entorno de trabajo.

2.4.1. DaoStack

DaoStack [38] es una plataforma para construir y ejecutar DAOs. Definen una DAO como organizaciones ejecutadas en redes P2P que animan a grupos de personas a tomar decisiones no jerárquicas sobre recursos compartidos. A diferencia de Aragon, ofrece menos posibilidades de personalización.

Su objetivo principal es respaldar el proceso de creación de una DAO con herramientas para la toma de decisiones descentralizada, interfaces de usuario o gestión de la organización.

Cuenta con la siguiente arquitectura (ver figura 2.8):

- **Infra**: Una biblioteca que contiene los aspectos básicos del protocolo. Un smart contract para gestionar los procesos de votación y cómo se distribuyen.
- **Arc**: Una biblioteca que sostiene la construcción de una DAO. Algunos de los componentes básicos son la cuenta, los tokens, complementos, restricciones y el control de acceso.
- **Subgraph**: Hace uso del protocolo The Graph para indexar los eventos producidos en *blockchain* y dar acceso a los datos en las aplicaciones a través de consultas GraphQL.
- **Arc.js**: Una biblioteca que facilita el acceso a los smart contracts y al subgrafo.
- **Dapps**: Aplicaciones descentralizadas. Por ejemplo, Common y Alchemy. Esta última proporciona a los usuarios formas de implementar nuevas DAO o interactuar y examinar aquellos que ya existen.

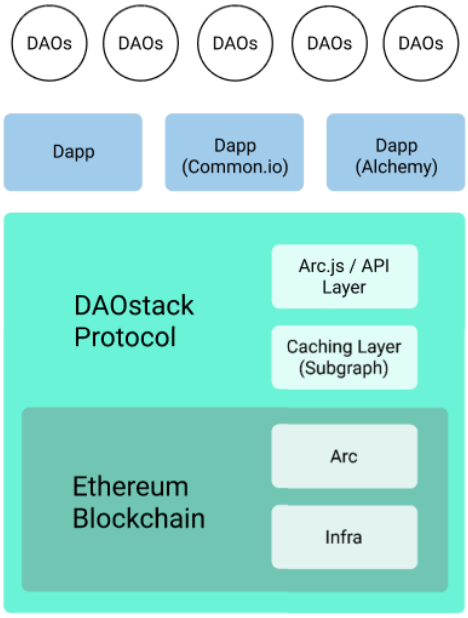

Figura 2.8: Arquitectura de DAOStack.

Cuenta con un mecanismo llamado Consenso Holográfico que consiste en aproximar de manera eficiente y confiable, las decisiones de grupo que tienen un número pequeño de participantes. La implementación de este mecanismo se encuentra en el protocolo Génesis, que funciona como una máquina de votación con predicciones permitiendo aprobar propuestas de alta confianza sin requerir una participación numerosa.

El consenso holográfico pone un precio para las propuestas que debe pagarse con la moneda GEN, para que de esta manera pueda considerarse la propuesta. Después, los predictores apuestan tokens sobre si la propuesta se aprobará o no. Si la predicción fue un acierto recibirán una recompensa y en caso contrario, perderán la cantidad apostada.

2.4.2. Colony

Colony [39] es una infraestructura descentralizada, construida sobre Ethereum, para organizaciones nativas de Internet con infraestructura imparcial y confiable. Se definen como un marco potente, práctico, flexible y fácil de usar.

Tiene un potencial enorme que proporciona a las organizaciones y comunidades, un marco de desarrollo general para crear y definir sus funciones esenciales. Colony ayuda a las comunidades a organizarse en términos de propiedad, estructura, autoridad, gestión financiera, etc. Facilita la organización para que más personas puedan colaborar y administrar fondos compartidos sin necesidad de confiar entre sí.

A diferencia de Aragon y DAOstack, que son impulsados por el voto, Colony hace uso de un proceso asíncrono de toma continua de decisiones financieras, permitiendo que cada parte de la organización funcione de manera autómata. Estas DAOs se dividen en dominios o tareas que los miembros pueden realizar para ganar influencia. Es decir, se basa en un sistema meritocrático en el que los miembros deben trabajar para la organización, para obtener más prestigio y facilitar el desarrollo colaborativo al trabajar por un objetivo común.

La reputación que pueden obtener los participantes es un valor que señala el mérito de las contribuciones realizadas, plasmando el trabajo aportado a la organización. No solo se obtiene reputación realizando tareas, sino también participando en votaciones o creando colonias nuevas.

Podemos distinguir dos partes importantes:

- **colonyJS**: Una biblioteca de JavaScript con una interfaz simple que permite la integración de la capa de aplicación con los *smart contracts* de colonyNetwork.
- **colonyNetwork**: Infraestructura sobre la que se ejecutan y son gestionadas todas las colonias.

Existe una colonia llamada Meta Colony que se encarga del proceso de minado en la red y de gestionar los parámetros necesarios para el correcto funcionamiento de ella, incluyendo las votaciones.

Colony quiere hacer frente a uno de los principales problemas en la red: el coste de las tarifas de gas en Ethereum para la creación de nuevas colonias o comunidades. Es por ello que la funcionalidad de crear nuevas colonias se encuentra deshabilitada actualmente, hasta que se desarrolle la nueva versión de la plataforma.

2.4.3. Aragon

Aragon es una de las plataformas DAO más importantes hasta el momento, fundada en 2016 por los emprendedores de Internet Luis Cuende y Jorge Izquierdo. La definición que Aragon ofrece para una DAO es que son organizaciones en línea flexibles, globales y sin censura. Aragon proporciona aplicaciones y herramientas que facilitan la creación y gestión de nuevas DAOs, sin necesidad de intermediarios. Contribuye a la gestión de estas comunidades para aprovechar todo el potencial de blockchain y los *smart contracts* desplegados en la red de Ethereum.

Aragon facilita una plantilla para crear una DAO, la cual puedes personalizar y adaptar a un propósito específico. Por otro lado, se pueden instalar o eliminar aplicaciones que poseen diferentes funcionalides, por ejemplo, almacenar fondos, ofrecer sistemas de toma de decisiones, etc. También concede la posibilidad de establecer un token propio para su uso en la organización y permite otorgar permisos a los miembros y a las aplicaciones para conectarlos de forma segura. El creador de la DAO obtiene permisos de administrador que puede transferir a la aplicación de votación para que la DAO se administre mediante una gobernanza descentralizada más democrática.

La plataforma cuenta con distintos módulos:

- **Aragon Govern**: Es un software para crear y gobernar organizaciones que se basa en el concepto optimista, es decir, asumir que los actores están actuando racionalmente y con buenas intenciones.
- **Aragon Vocdoni**: Es un sistema de votación sin confianza, totalmente anónimo, que garantiza la disponibilidad de datos y un protocolo de comunicación destinado a combatir la censura. Esta tecnología también es la base para una cadena de bloques específica de votación de capa dos llamada Vochain.
- **Aragon Protocol and Agreements**: Permiten que las personas tengan más libertad para tomar decisiones y actuar, al tiempo que garantizan que esas acciones puedan ser moderadas de manera justa por el resto de la comunidad. Si los miembros abusan de sus poderes de alguna manera y la disputa no se puede resolver por otros medios, el protocolo ayudará a la decisión de si deben ser sancionados o no.
- **Aragon Connect**: Es un conjunto de herramientas que permite crear una organización con Aragon de forma más flexible integrando aplicaciones y sitios web.

Aragon pone a disposición de los desarrolladores una gran cantidad de recursos para la creación de DAOs [40]. Dedicaremos las siguientes subsecciones a comentar aquellas con las que hemos tenido más contacto durante el desarrollo del proyecto.

AragonCLI

AragonCLI es la interfaz de linea de comandos de Aragon usada para crear, desarrollar y configurar DAOs y aplicaciones de Aragon. Está integrada con Node.js y git. Es la encargada de generar la plantilla de nuestro proyecto, incluyendo todas las dependencias necesarias para asegurar un correcto funcionamiento (ver figura 2.9).

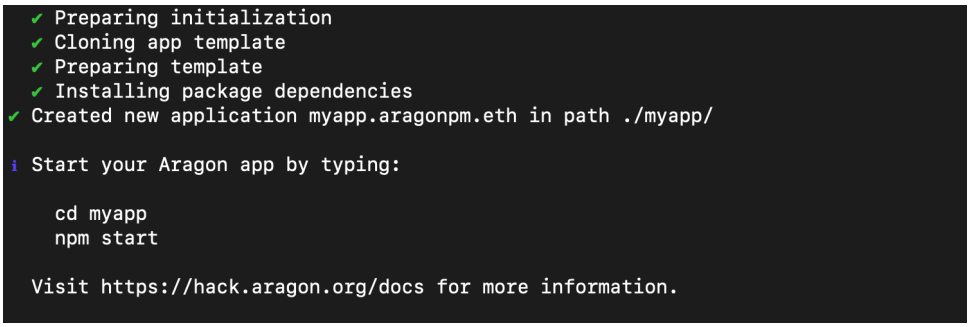

Figura 2.9: Creación de la plantilla de React de Aragon usando AragonCLI.

AragonAPI

AragonAPI es el conjunto de APIs y especificaciones usadas para interactuar con el *back-end* de Aragon y gestionar las transacciones y el estado del contrato sin depender de un servicio centralizado. Está implementada en JavaScript y a continuación, se muestran algunas de las funciones que desempeña:

- Conectar los smart contracts con el *front-end*.
- Interactuar con el núcleo de Aragon y sus aplicaciones a través de web3.js.
- Acceder al estado de la aplicación con almacenamiento en caché del lado del cliente.
- Crear descripciones fáciles de leer en las transacciones de los contratos inteligentes.

AragonUI

AragonUI es un conjunto de herramientas de interfaz de usuario nativo de Aragon, diseñado específicamente para aplicaciones descentralizadas. Sigue el estilo del cliente de Aragon y es útil a la hora de integrar tus aplicaciones con su ecosistema (ver figura 2.10). Aunque su uso no es obligatorio, debido a su comodidad y sus elegantes diseños hemos hecho amplio uso de él en nuestros desarrollos.

2.5. Blockchain en educación

Como veremos más adelante (ver sección 5.1), nuestro proyecto se basará en una aplicación de blockchain en el campo educativo. Es por eso que consideramos pertinente una sección donde hagamos un repaso de las principales ventajas de blockchain en este sector [41] mencionando algunas iniciativas que han surgido en los últimos años.

2.5.1. Verificación de títulos

El registro inmutable de blockchain refleja de forma precisa los eventos transcurridos en tiempo real. Esto es verdaderamente útil a la hora de verificar títulos, compartir resultados académicos y hacer que los alumnos sean honestos al hablar de sus progresos. Iniciativas en esta linea pueden conseguir una mayor transparencia en el sector educativo que termine beneficiando tanto a profesores como alumnos.

Algunos ejemplos de proyectos que ya están haciendo posibles estos avances son:

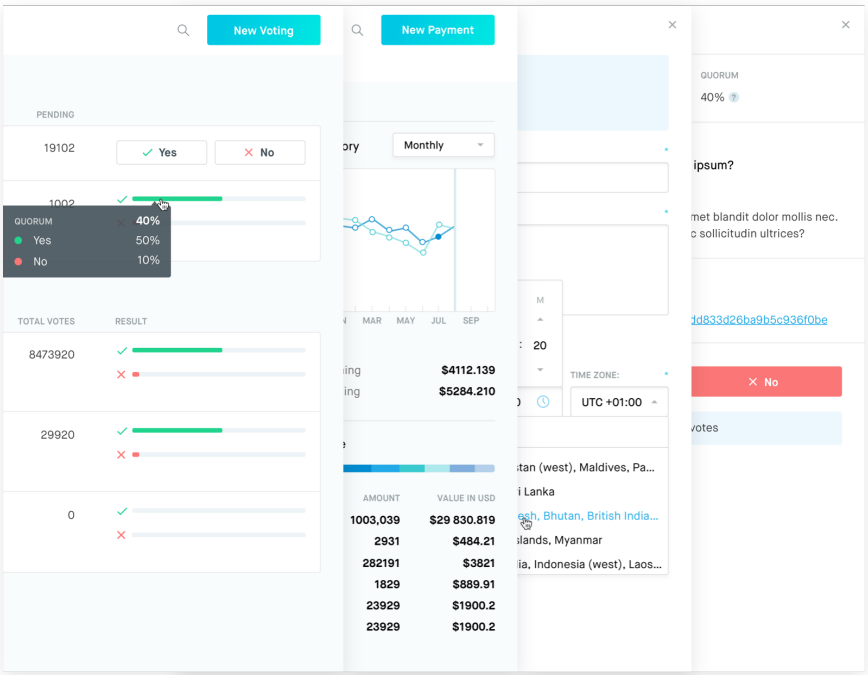

Figura 2.10: Ejemplo de algunos de los componentes proporcionados por aragonUI.

- **Blockcerts** [42]: Proyecto de software libre desarrollado por MIT Media Lab y Hylan Credentials que permite emitir y verificar certificados respaldados con tecnología blockchain. Su sistema permite guardar notas, resultados académicos y diplomas que quedarán registrados de forma inmutable en su red blockchain.
- **APPI** [43]: Aplicación donde puedes crear un perfil y agregar los datos de tu currículum académico. Utiliza la tecnología blockchain para verificar la información y, finalmente, la deja guarda de forma permanente. También permite registrar datos como el registro médico, historial delictivo e información financiera.
- **Sony Global Education** [44]: Aplicación desarrollada por Sony e IBM para permitir a las diferentes instituciones añadir logros e información académica de estudiantes para poder generar un registro académico irrefutable en la blockchain.

2.5.2. Redes educativas

Los smart contracts permiten fijar claramente las responsabilidades de profesores y alumnos durante el proceso educativo. De esta manera los contenidos de cada asignatura, las fechas de entrega y el periodo lectivo pueden quedar minuciosamente estipulados en estos. Además, también pueden ser usados como medio para sentar las condiciones de los préstamos académicos.

Por otro lado, los tokens se han convertido en una parte fundamental de la tecnología blockchain. Estos permitirán a los profesores incentivar a los alumnos a ser más activos en los cursos o completar ciertas tareas ofreciendo recompensas a cambio.

Algunos ejemplos de proyectos que crean redes educativas aprovechando las ventajas de

la tecnología blockchain son:

- **ODEM** [45]: Es un mercado descentralizado de productos y servicios educativos. Usando la tecnología blockchain, el proyecto conecta a estudiantes con cursos y contenido educativo ofrecido por profesores. El historial de profesores y alumnos queda guardado en la blockchain de forma que los alumnos pueden elegir el tipo de profesor que más se ajuste a sus preferencias y viceversa.
- **Blockchain Education Network** [46]: Junta a alumnos y profesores de todo el mundo alrededor de contenido centrado en blockchain y criptomonedas. Impulsado inicialmente por alumnos del MIT y la Universidad de Michigan, ha promovido la formación de proyectos que, actualmente, tienen un valor de dos billones de dólares.
- **BitDegree** [47]: Combina registros de blockchain e incentivos con tokens en una plataforma de educación online centrada en la tecnología. Ofrece cursos de desarrollo con diferentes tecnologías e incentiva a los alumnos mediante becas en tokens por completar cursos u obtener ciertos logros.

Capítulo 3

Metodología y tecnología utilizada

Para trabajar en el complejo entorno de las Dapps en Ethereum, existen numerosas herramientas y tecnologías que facilitan el trabajo al desarrollador. Antes de comenzar a hablar de nuestro proyecto, enumeraremos el proceso de trabajo empleado para el desarrollo y las herramientas que hemos utilizado realizando un breve comentario de cada una de ellas.

3.1. Metodología

3.1.1. Etapas de trabajo

Respecto a la planificación y las etapas del trabajo, sabíamos que nos enfrentábamos a un proyecto complejo en el que primero, tendríamos que superar una gran barrera de aprendizaje. Por este motivo, empezamos a trabajar en él a principios de septiembre, para contar con todo el curso por delante. El primer cuatrimestre lo dedicamos a familiarizarnos con las diferentes tecnologías que emplearíamos en el proyecto, realizando primero una investigación teórica, que más tarde pondríamos en práctica con pequeños desarrollos.

Al inicio del segundo cuatrimestre y una vez ya familiarizados con las tecnologías, comenzamos directamente con el trabajo relacionado con nuestra aplicación. Dedicamos una semana a decidir el tipo de aplicación que queríamos desarrollar y otras dos a decidir las funcionalidades que esta tendría y cómo orientaríamos la interfaz de usuario. Finalmente, a mediados de marzo comenzamos con el desarrollo de la aplicación que estaba planificado para durar aproximadamente un periodo de dos meses y finalizar a mediados de mayo (ver tabla 3.1).

Etapa I: Aprendizaje		
Fase de Investigación	2020-09-01 2020-10-31	Exploración de las tecnologías implicadas, obteniendo conocimientos acerca de cómo es su funcionamiento, con reuniones periódicas para aclarar y asentar los conceptos importantes.
Etapa II: Primeros desarrollos		
Lotería	2020-11-01 2020-11-30	Implementación de una aplicación que gestiona el funcionamiento de una lotería con Solidity y React.
Task Manager	2020-12-01 2020-12-31	Implementación de una aplicación para la gestión de tareas usando el framework de Aragon.
Pruebas en los frameworks	2020-01-01 2020-01-31	Investigación del desarrollo de DAOs en los frameworks DAOstack, Aragon y Colony.
Etapa III: dAcademy		
Brainstorming y selección de la DAO	2021-02-21 2021-02-28	Brainstorming entre los participantes del grupo para determinar las mejores propuestas para la DAO a realizar.
Diseño	2021-03-01 2021-03-14	Planificación del desarrollo y del diseño mediante mockups.
Implementación	2021-03-15 2021-05-15	Implementación de la DAO sobre Aragon.

Tabla 3.1: Fases de trabajo en el proyecto.

3.1.2. Software libre

La tecnología blockchain promueve la distribución de aplicaciones descentralizadas por lo que todo el software de este trabajo es libre.

El código de la aplicación se puede encontrar en el siguiente repositorio de Github:

`https://github.com/P2PModels/Courses-Network`

Posee una licencia *GNU General Public License v3.0* por lo que los usuarios podrán acceder al código bajo los permisos y condiciones que ofrece esta licencia.

3.1.3. Método de desarrollo

Para el desarrollo del proyecto se han utilizado metodologías ágiles basadas en SCRUM (ver figura 3.1), aunque adaptadas a la situación, necesidades y características del trabajo, aportando de esta manera flexibilidad en cada etapa del desarrollo. Se ha dividido el proyecto en distintas etapas que deben desarrollarse en un periodo de tiempo determinado por el equipo. Además, se han llevado a cabo reuniones periódicas para definir los objetivos, solventar problemas y revisar cada avance en el proyecto.

Por otro lado, se han definido listas de tareas a realizar en cada reunión. Estas listas han estado al alcance de todos los miembros del equipo para repartir cada una de las tareas y obtener un correcto desarrollo.

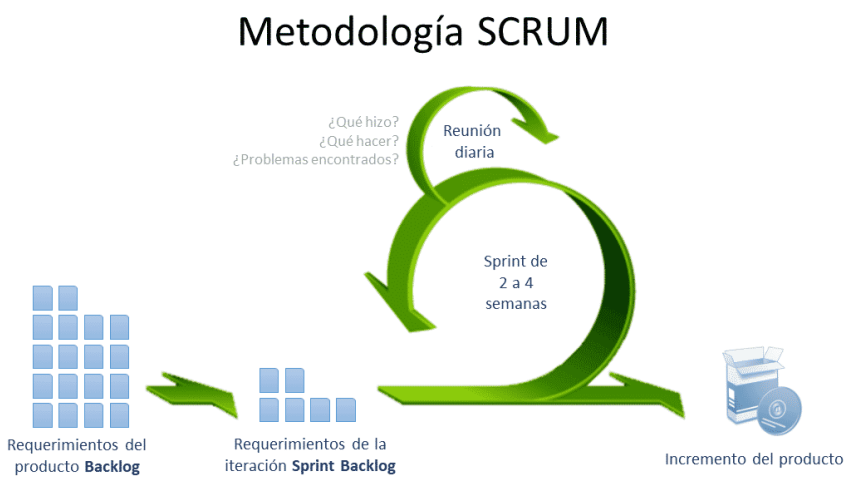

Figura 3.1: Gráfico explicativo del funcionamiento de la metodología SCRUM [6].

Se ha utilizado Github para el control de versiones y Telegram/WhatsApp y Google Meet para la comunicación de los miembros del equipo.

3.1.4. Método para la obtención de ideas: Brainstorming

Para escoger que tipo de aplicación íbamos a desarrollar, decidimos hacer una sesión de *brainstorming*. Mecánica empleada en trabajos grupales para facilitar la aparición de nuevas ideas sobre un tema o problema determinado [48]. Para aprovechar el máximo potencial de este método, se deben seguir las siguentes reglas:

- **Suspender el juicio**: Evitar toda crítica durante la sesión y reservar la evaluación para después.
- **Pensar libremente**: Las imposibles son bien recibidas. Permitiéndonos pensar fuera de los límites de lo normal o convencional puede dar lugar a soluciones innovadoras.
- **La cantidad es importante**: Hay dos razones por las cuales debemos intentar conseguir el máximo número de ideas posibles. Como es lógico las propuestas obvias y habituales son las primeras aparecer, por lo que es probable que las primeras 20 o 25 no sean relevantes. La segunda es que, con una mayor cantidad de temas propuestos, dispondremos de más posibilidades para escoger, adaptar o combinar la idea definitiva.
- **El efecto multiplicador**: Se busca la combinación y la mejora de las ideas para dar lugar a propuestas más innovadoras y adaptadas al problema en cuestión.

3.2. Tecnologías utilizadas

Es reconocido que los proyectos para desarrollar aplicaciones web, suelen involucrar varias tecnologías muy diferentes entre sí, ya que implican el desarrollo de dos sistemas muy diferentes como son el back-end y el front-end así como la necesitad de asegurar una correcta comunicación entre ellos. Este proyecto cuenta, además de la coordinación del equipo, de utilizar blockchain como tecnología de back-end. Lo que ha provocado la

necesidad de utilizar gran cantidad de aplicaciones y tecnologías durante el desarrollo del proyecto, como mostramos en la tabla 3.2.

Solidity	Lenguaje de escritura de smart contracts.
JavaScript	Escritura de código que da funcionalidad.
React	Crear interfaz de usuario.
CSS	Estilo de la interfaz.
AragonCLI	Creación y desarrollo de DAOs.
AragonOS	Marco para la creación de DAOs con smart contracts.
AragonAPI	APIs y especificaciones para interactuar con AragonOS.
AragonUI	Componentes de interfaz de usuario para aplicaciones descentralizadas.
Remix IDE	Para la creación y depuración de smart contracts.
Visual Studio Code	Desarrollo del proyecto.
Overleaf (LateX)	Escritura de la memoria.
Google Docs	Escritura de documentos de planificación y anotaciones.
Github	Control de versiones.
Balsamiq	Desarrollo de mockups.

Tabla 3.2: Tecnologías utilizadas para el desarrollo del proyecto.

3.2.1. Backend con Solidity y Remix

Una de las partes esenciales para construir nuestra DAO, es la implementación de los smart contracts. Para su desarrollo hacemos uso del lenguaje Solidity [49], un lenguaje de alto nivel similar a JavaScript y enfocado específicamente a la máquina virtual de Ethereum (EVM) que es el entorno donde se ejecutan los smart contracts.

Comparte los tipos (*int*, *string*, *boolean*, *struct*, etc.) y estructuras de control (*if*, *else*, *while*, *for*, etc.) comunes en otros lenguajes. Incluye además el tipo *address*, para definir direcciones de cuentas de Ethereum o *mapping*, una variable bastante importante a la hora de almacenar datos.

Por otro lado, debemos mencionar los modificadores de visibilidad (*private*, *public*, etc.) o de función (*payable*, *view*, etc.). *Payable* permite a una función recibir y enviar fondos siendo una propiedad importante a la hora de transferir ETH de una cuenta a otra. Para ello, existen funciones como *transfer()* que nos permite enviar una cantidad determinada a una dirección.

Existen dos tipos de cuentas en Ethereum:

- **Cuentas externas** controladas por un par de claves pública-privada. Estas cuentas están ligadas a los monederos digitales (*wallets*) que pertenecen a usuarios concretos y están determinadas por la clave pública.
- **Cuentas contrato** son controladas por el código y se definen al crear un contrato.

Las transacciones son mensajes que se envían de una cuenta a otra y sirven tanto como mandar ETH, como para crear contratos. Cada transacción necesita una cantidad de gas para ejecutarse, que establece el creador de la misma. Si después de la ejecución sobra parte del gas, este se reembolsará.

El lenguaje cuenta con un conjunto de instrucciones que permiten llamar a otros contratos y configurar las transacciones determinando el gas, las direcciones, etc.

Para crear los *smart contract* sin necesidad de instalaciones adiccionales, se ha hecho uso de Remix, un entorno de desarrollo (del inglés Integrated Development Environment, IDE) basado en un navegador que integra el compilador y un contexto en tiempo de ejecución para este lenguaje, sin los componentes orientados al servidor (ver figura 3.2).

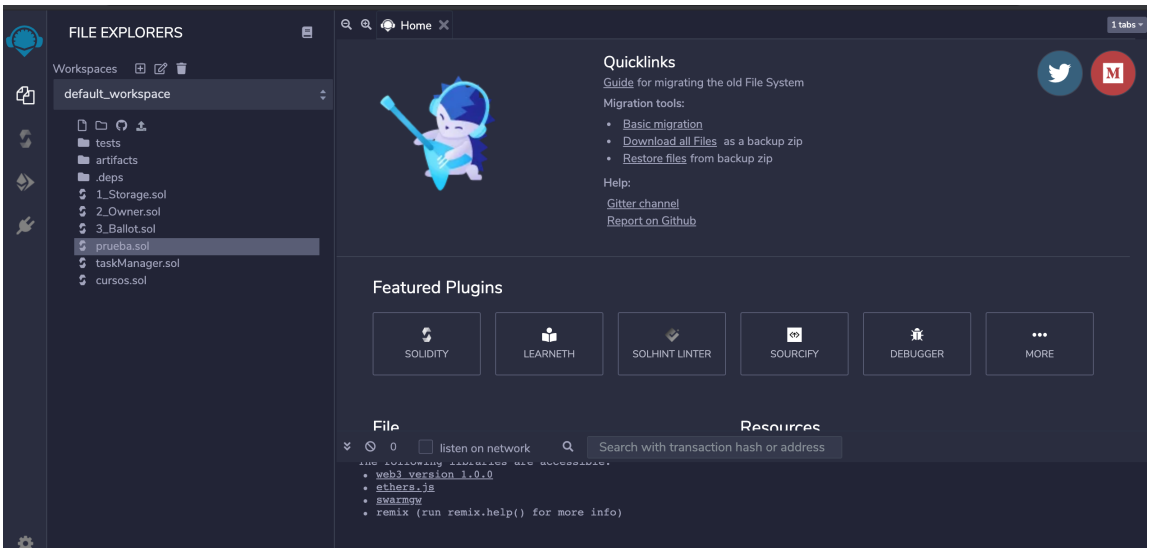

Figura 3.2: Entorno de desarrollo Remix (IDE).

Remix nos permite seleccionar el compilador y la versión de EVM que deseamos para depurar nuestro smart contract y otros ajustes de compilación. También otorga la posibilidad de desplegarlo y servir de interfaz para interactuar con la funcionalidad implementada.

3.2.2. Cartera virtual Metamask

Usamos el término *monedero virtual* (*wallet* en inglés) para referirnos a una aplicación de software que se encarga de administrar las cuentas externas de Ethereum. Contiene las claves de la cuenta y permite crear y difundir transacciones en su nombre. Existen muchos tipos de monederos (de escritorio, de móvil, web, etc) y la elección de la misma depende del uso que le queramos dar.

En nuestro caso usamos Metamask, un monedero que se instala como una extensión en el navegador (ver figura 3.3). Su uso es bastante sencillo y es apropiada para realizar pruebas por su facilidad a la hora de conectarse a distintos nodos de Ethereum (ver figura 3.4):

- **Main Ethereum Network**: Red principal de Ethereum con ETH real.
- **Ropsten Test Network**: Red de prueba pública en el que ETH no tiene valor.
- **Kovan Test Network**: Red de prueba pública que usa el protocolo de consenso Aura. El ETH tampoco tiene valor en esta red.

- **Rinkeby Test Network**: Red de prueba pública que usa el protocolo de consenso Clique. El ETH de esta red no tiene valor.
- **Localhost 8545**: El nodo se ejecuta en el mismo equipo que el navegador. Puede ser público o privado.
- **RPC personalizado**: Puede conectarse a cualquier nodo con una llamada a procedimiento remoto compatible con RPC. Puede conectarse a una cadena de bloques pública o privada.

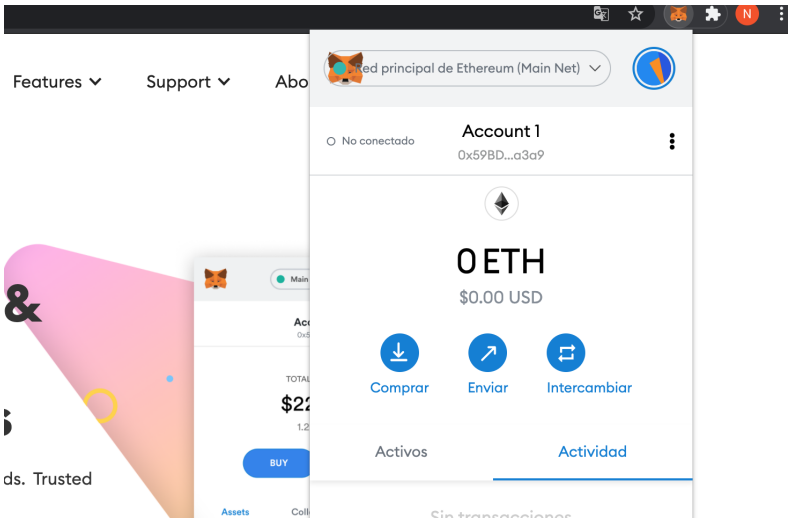

Figura 3.3: Extensión de Metamask en el explorador Google Chrome.

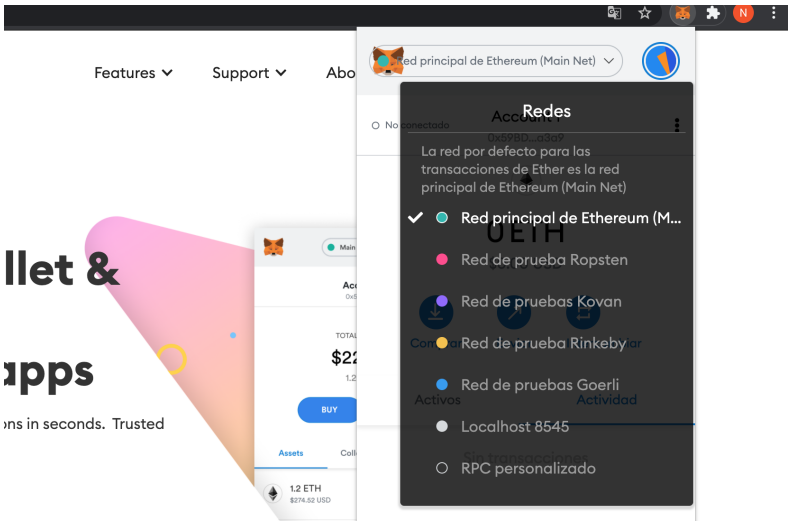

Figura 3.4: Distintas redes de Ethereum.

3.2.3. Red de pruebas: Rinkeby y Localhost 8545

Para el desarrollo de este trabajo, hemos hecho uso de la red de prueba de Rinkeby. En ella podemos obtener ETH para realizar las transacciones requeridas en nuestra DAO, simulando su funcionamiento como si se ejecutase en la red real.

Por otro lado, también se ha hecho uso de la red local. Al desplegar el contrato en local, se generan una cuentas con sus respectivas claves pública-privadas (ver figura 3.5) que importándolas en Metamask, permiten interactuar con el contrato.

```
main      | Starting Aragon app development...
main      | App name: app
main      | App id: 0x740e14bd8efcf3fa51e40db0c746650c88690730bd094e78c8981ba6d61981b1
main      | Accounts mnemonic "explain tackle mirror kit van hammer degree position ginger unfair soup bonus"
main      | Account 0 private key 0xa8a54b2d8197bc0b19bb8a084031be71835580a01e70a45a13babd16c9bc1563
main      |           public key 0xb4124cEB3451635DAcedd11767f004d8a28c6eE7
main      | Account 1 private key 0xce8e3bda3b44269c147747a373646393b1504bfcbb73fc9564f5d753d8116608
main      |           public key 0x8401Eb5ff34cc943f096A32EF3d5113FEbE8D4Eb
main      | Account 2 private key 0x8716d2701596f51aa39d061a685d5ae5ec946eb2c7adb059d29024b5bb3b02c8
main      |           public key 0x306469457266CBBe7c0505e8Aad358622235e768
main      | Account 3 private key 0x62d7bb725787d84b059eb4950f6eea060d898183250ca3ea673a36b8e113018f
main      |           public key 0xd873F6DC68e3057e4B7da74c6b304d0eF0B484C7
main      | Account 4 private key 0x705df2ae707e25fa37ca84461ac6eb83eb4921b653e98fdc594b60bea1bb4e52
main      |           public key 0xDcC5dD922fb1D0fd0c450a0636a8cE827521f0eD
main      | Account 5 private key 0x6b12b45143fc6c7721d0ffbb9811905e773868376501fd1f46c64bf34ae29991
main      |           public key 0x27E9727FD9b8CdDdd0854F56712AD9DF647FaB74
main      | Account 6 private key 0x33f3f34569f997abb165d6967895d963a2b15ec609efcec844e65b60ee8340c7
main      |           public key 0x9766D2e7FFde358AD0A40BB87c4B88D9FAC3F4dd
main      | Account 7 private key 0x5a013cc48f0a3196b0986fc7a7a9dd320ac75e89e33302a7ff4ea6b9dc4f7b00
main      |           public key 0xBd7055AB500cD1b0b0B14c82BdBe83ADCc2e8D06
main      | Account 8 private key 0x418cc0b07bfef998f577384b185b97ad544204b5be43ac9b3abf16db2012ab5c
main      |           public key 0xe8898A4E589457D979Da4d1BDc35eC2aaf5a3f8E
main      | Account 9 private key 0x698eece6f9915b08b4d1a63958dc4f3996ee5a8d685b29d17c28beab912a77cd
main      |           public key 0xED6A91b1CFaae9882875614170CbC989fc5EfBF0
main      |
```

Figura 3.5: Cuentas generadas al desplegar el contrato.

3.2.4. Etherscan

Etherscan es un explorador de bloques y una plataforma de analíticas para Ethereum [50]. En el explorador se pueden consultar, entre otras cosas, las transacciones, direcciones y detalles de cada bloque que interactúa con Ethereum. Nos permite consultar el estado de las transacciones que estemos ejecutando y su correcta finalización (ver figura 3.6).

Txn Hash	Method	Block	Age	From		To	Value	Txn Fee
0x14ff6ad07e7322bcb67…	New Token And In…	7782794	133 days 14 hrs ago	0x22e16273dd8e65d5cc…	OUT	0xa3809a525b92a8a290…	0 Ether	0.06723594
0x93890c36500e9b2266…	0x60806040	7775363	134 days 21 hrs ago	0x22e16273dd8e65d5cc…	OUT	Contract Creation	0 Ether	0.000304109
0xb349f108c7e32549bb…	0x60806040	7765558	136 days 14 hrs ago	0x22e16273dd8e65d5cc…	OUT	Contract Creation	0 Ether	0.000304109
0x5161cc5ecfe4f00dc1f2…	New Instance	7639252	158 days 16 hrs ago	0x22e16273dd8e65d5cc…	OUT	0xefe6da5fdfefabe96e41…	0 Ether	0.05145203
0xebdf04e49c033b87f97…	New Token	7639251	158 days 16 hrs ago	0x22e16273dd8e65d5cc…	OUT	0xefe6da5fdfefabe96e41…	0 Ether	0.01728493
0x6bce265e5bc7b81836…	0x60806040	7501683	182 days 15 hrs ago	0x22e16273dd8e65d5cc…	OUT	Contract Creation	0 Ether	0.000448831
0x021b7289ae0786332a…	0x60806040	7490394	184 days 15 hrs ago	0x22e16273dd8e65d5cc…	OUT	Contract Creation	0 Ether	0.000096585
0x78a131b585a06ae0b9…	Transfer	7490390	184 days 15 hrs ago	0x31b98d14007bdee637…	IN	0x22e16273dd8e65d5cc…	18.75 Ether	0.000021

Figura 3.6: Transacciones ejecutadas por una de las cuentas de Rinkeby utilizadas para el proyecto.

3.2.5. Node.js

Node.js es un entorno asíncrono en tiempo de ejecución dirigido por eventos para JavaScript y está diseñado para construir aplicaciones web escalables. En contraste con los modelos concurrentes más populares actualmente, en los que se utilizan hilos del sistema operativo, las aplicaciones web basadas en hilos son menos eficientes y más difíciles de usar (ver figura 3.7).

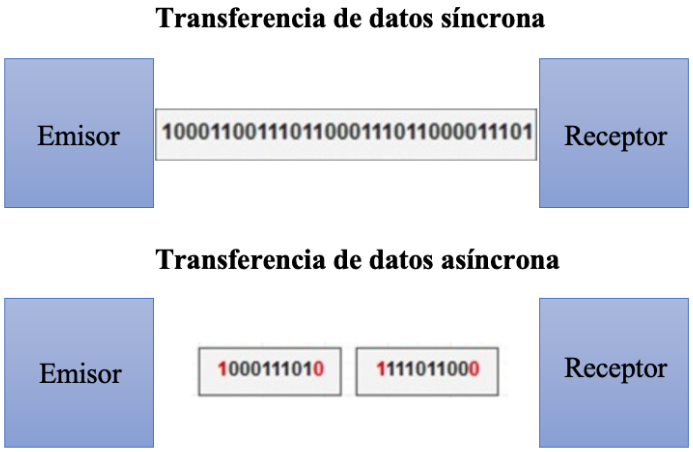

Figura 3.7: Diferencias entre la transmisión síncrona y asíncrona de datos [7].

Este es el sistema que integra Aragon en su plantilla para una aplicación descentralizada (ver sección 3.2.6) y, por lo tanto, será el que usemos durante el desarrollo de nuestras aplicaciones.

3.2.6. Front-end con React y plantilla de Aragon

React [51] es una de las bibliotecas de JavaScript para construir interfaces de usuario más populares en la actualidad. Se caracteriza por:

- **Vistas declarativas**: Te permite diseñar vistas para cada estado de la aplicación y será React quien se encargue de actualizar y renderizar los componentes de forma eficiente. Esto hace que el código sea más predecible y fácil de depurar.
- **Basado en componentes**: Creación de componentes encapsulados que manejan su propio estado. Al escribir la lógica de los componente en JavaScript y no en plantillas, se facilita el paso de la información de forma cómoda a través de la aplicación y se mantiene el estado fuera del DOM.
- **Compatibilidad**: React puede renderizar desde el servidor usando Node, lo que será especialmente útil en nuestro proyecto.
- **Hooks**: Son una característica añadida a React a partir de la versión 16.8. Permiten cambiar el enfoque anterior centrado en clases, por otro funcional que facilite la re-utilización de la lógica entre diferentes componentes haciendo más sencillo entender su comportamiento.

Además de su popularidad y potencial, la principal razón para que hayamos escogido esta biblioteca para crear nuestra interfaz, es debido a que Aragon proporciona una plantilla [52] (*boilerplate* en inglés) de React integrada con Node.js, que permite tener una aplicación de Aragon funcionando en tan solo unos minutos (ver figura 3.8). Para hacer modificaciones sobre esta, será suficiente con añadir tu contrato de Solidity, crear tu interfaz con React y gestionar la comunicación entre ambas utilizando la API que nos proporciona Aragon.

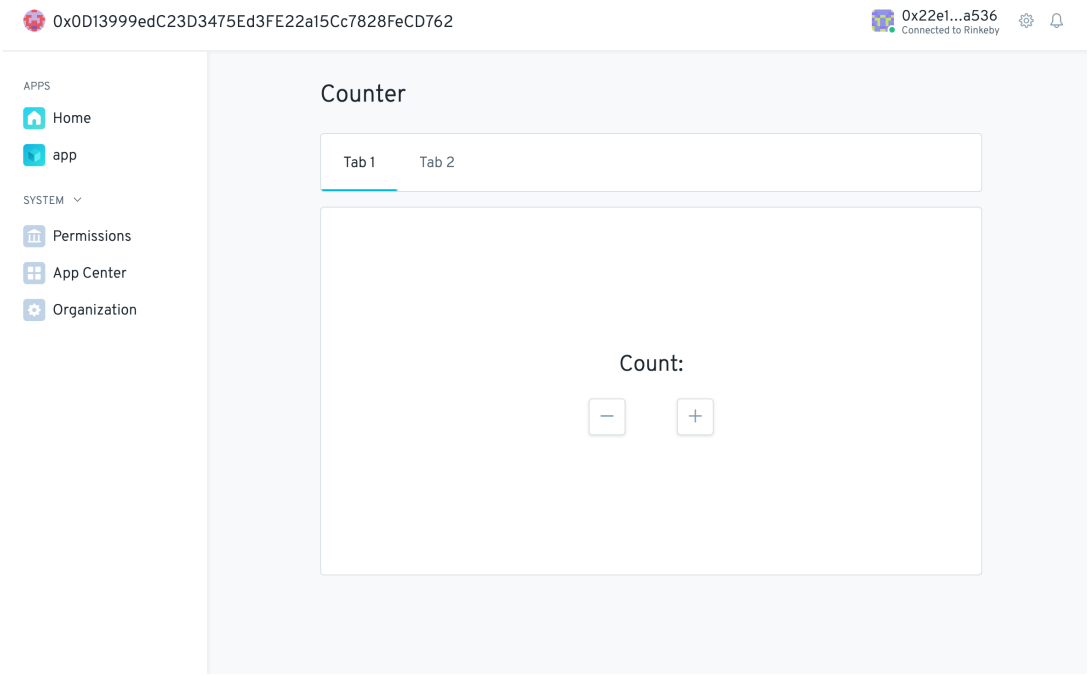

Figura 3.8: Aspecto de la plantilla de React proporcionada por Aragon.

Capítulo 4

Primeros desarrollos

4.1. Lotería

El primer paso para comenzar a entender lo que es una aplicación descentralizada y cómo funcionan los *smart contracts*, fue realizar una pequeña implementación que pruebe su funcionamiento. La primera tarea de este TFG, por tanto, consistió en realizar un contrato que gestiona el premio de una lotería. Este proyecto está abierto al público en un repositorio de Github [53] bajo una licencia *GNU General Public License v3.0*.

El comportamiento del sistema viene dado por el diagrama de casos de uso de la figura 4.1.

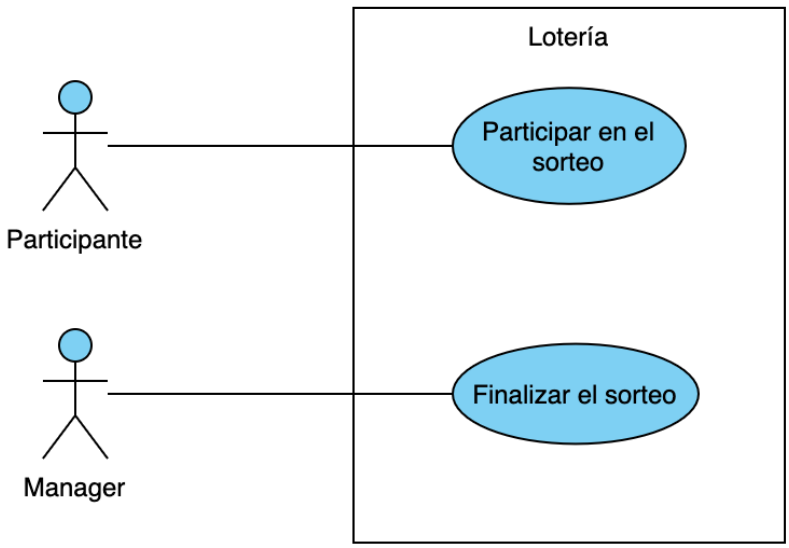

Figura 4.1: Diagrama de casos de uso de la Lotería.

4.1.1. Smart Contract

Usando el IDE de Remix [54], generamos una versión del *smart contract* que permitía inscribirse en un sorteo apostando una cantidad determinada de ETH y finalmente, el administrador indicaba al contrato que determinase al ganador.

Para ello, se declararon las siguientes variables:

- ***players***. *Array* que contiene las direcciones de los participantes del sorteo.
- ***manager***. Administrador que se encarga de avisar al contrato para que genere un ganador.
- ***totalAmount***. Cantidad total apostada en el sorteo.

Por otro lado, implementamos dos funciones principales para la lógica del sorteo:

- ***enter()***. Comprueba que la cantidad apostada sea mayor de un mínimo determinado para participar. Añade el participante al sorteo y aumenta la cantidad de ETH al valor total. Los participantes envían dinero al contrato llamando a esta función.
- ***chooseWinner()***. Elige un número aleatorio para seleccionar al ganador y se transfiere el premio a la cuenta del ganador.

Ambas funciones se declaran como *payable*, lo que les permite recibir fondos. En este caso, necesitábamos recibir y enviar los ETH apostados y del premio final, respectivamente.

Por otro lado, usamos modificadores para comprobar si el usuario que deseaba generar el ganador, era el administrador. A continuación, podemos ver el código fuente del smart contract realizado para la gestión de la lotería.

```
1
2  // SPDX-License-Identifier: GPL-3.0
3  pragma solidity ^0.7.4;
4
5  contract Lottery {
6      address public manager;
7      address payable[] public  players;
8      uint public totalamount;
9      uint public numrandom;
10
11     constructor () {
12         manager = msg.sender;
13         totalamount = 0;
14     }
15
16     modifier onlyOwner() {
17         require(manager == msg.sender);
18         _;
19     }
20
```

```
21      modifier isWinner() {
22          require(players[numrandom] == msg.sender);
23          _;
24      }
25
26      function enter() public payable {
27          require (msg.value > .01 ether);
28          players.push(msg.sender);
29          totalamount += msg.value;
30      }
31
32      function random() private view returns (uint) {
33          uint randomHash = uint(keccak256(abi.encodePacked(block.
               difficulty, block.timestamp)));
34          return randomHash;
35      }
36
37      function choseWinner() public onlyOwner payable{
38           numrandom = random()%players.length;
39           players[numrandom].transfer(totalamount);
40           totalamount = 0;
41           delete players;
42      }
43
44  }
```

4.1.2. Aplicación con React

Una vez tenemos nuestro primer contrato implementado, llega el momento de generar un proyecto que le de vida a la dapp. Para ello usamos una plantilla (*boilerplate*) [55] que nos proporciona React y creamos nuestra aplicación modificando los ficheros correspondientes y añadiendo los que sean necesarios. De esta manera, obtenemos una interfaz que nos permite interactuar con nuestro contrato.

Se realizó una interfaz muy simple (ver figura 4.2) que posibilita a los usuarios introducir la cantidad que desean apostar para participar. Por otro lado, cuenta con un botón que el *manager* podrá pulsar cuando desee para finalizar el sorteo. Adicionalmente, se mostrarán mensajes informativos que comunicarán al usuario si las acciones se realizaron correctamente.

Figura 4.2: Interfaz de la aplicación de lotería con React.

4.1.3. Desplegar el smart contract en Rinkeby

Después de implementar nuestra aplicación descentralizada, era el momento de desplegar el contrato en la red de pruebas de Rinkeby. Esta red nos permite simular el entorno de Ethereum como si fuera real, pero sin consumir gas de verdad.

Primero creamos una red de pruebas en Rinkeby mediante el uso de la plataforma Infura (ver figura 4.3). Luego, procedimos creando diferentes cuentas en esa red con objetivo probar la participación de distintos usuarios en el sorteo (ver figura 4.4). Una vez creadas las cuentas, las añadíamos a nuestro monedero virtual de Metamask (ver sección 3.2.2).

Desde el lado del back-end, utilizamos el módulo Truffle para poder conectarnos a Metamask. Este nos proporciona acceso a las redes de Ethereum, y conectará nuestra app con la red de prueba de Rinkeby que se usará para desplegar el *smart contract*.

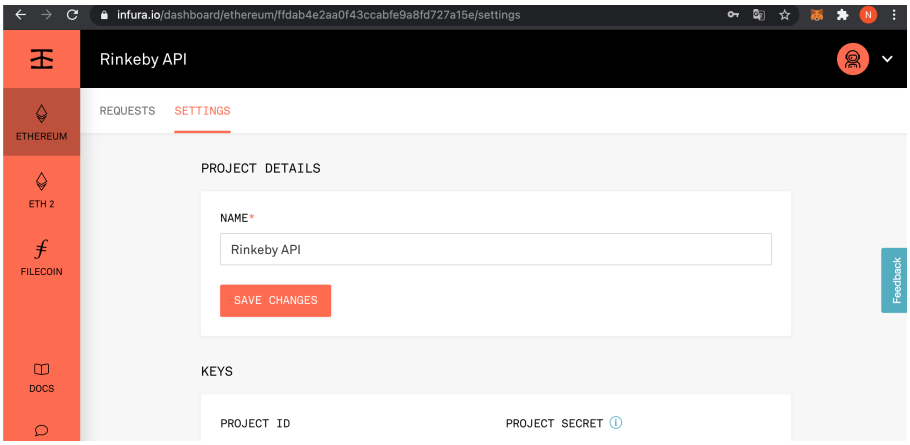

Figura 4.3: Configuración de un proyecto de Infura para Rinkeby.

Después de terminar las configuraciones para desplegar el contrato, estará listo para

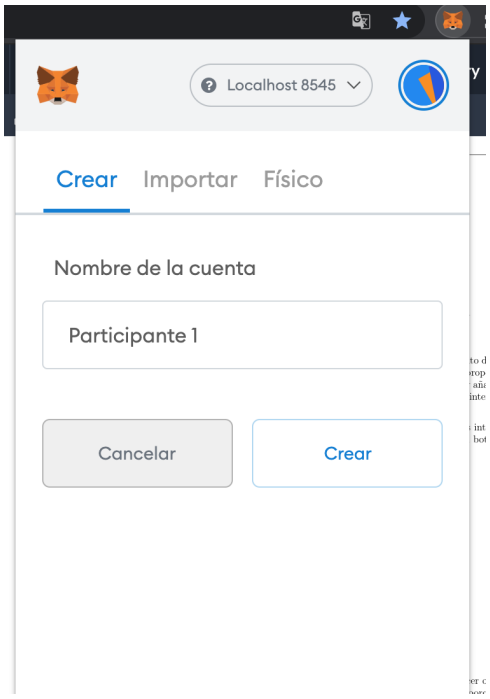

Figura 4.4: Crear una nueva cuenta en Metamask.

interactuar con él en la red y revisar las transacciones realizadas en Etherscan.

4.2. Gestor de tareas

Para continuar familiarizándonos con la tecnología *blockchain*, decidimos hacer otro pequeño proyecto, pero esta vez haciendo uso de las herramientas que nos proporciona la plataforma de Aragon. Esto eleva el nivel de dificultad ya que no solo estaremos trabajando con Solidity y React, tecnologías nuevas para el equipo hasta la fecha, si no que también tendremos que aprender como funcionan las APIs de Aragon (ver sección 2.4.3). Este proyecto está abierto al público en un repositorio de Github [56] bajo una licencia *GNU Affero General Public License v3.0*.

4.2.1. Objetivos

Con este proyecto se pretendía conocer todas las posibilidades que nos ofrecen las librerías de desarrollo de Aragon, por lo que la complejidad técnica no debía ser demasiado alta. Para alcanzar este objetivo, optamos por crear un gestor de tareas descentralizado.

Buscábamos que nuestro gestor de tareas guardase una lista cada una de ellas con la siguiente información:

- **Nombre de la tarea**
- **Prioridad**: A elegir entre alta, media o baja.
- **Estado**: El estado inicial de una tarea será asignado como 'en progreso' y podremos cambiarlo posteriormente a 'finalizada' cuando se haya completado.

Todas las tareas se mostrarán en una tabla dinámica en el *front-end*. Para interactuar con la lista de tareas, ofreceremos las siguientes acciones:

- **Crear tarea**: Dado un nombre de tarea y una prioridad, se crea y añade una nueva tarea a la lista de tareas.
- **Modificar prioridad**: Dada una tarea existente y una prioridad, se asigna la prioridad a la tarea escogida.
- **Finalizar tarea**: Dada una tarea existente, cambia el estado de esta a 'finalizada'.
- **Eliminar tarea**: Dada una tarea existente, la elimina de la lista, dejando así de mostrarse en la tabla de las tareas.

El comportamiento del sistema viene dado por el diagrama de casos de uso de la figura 4.5.

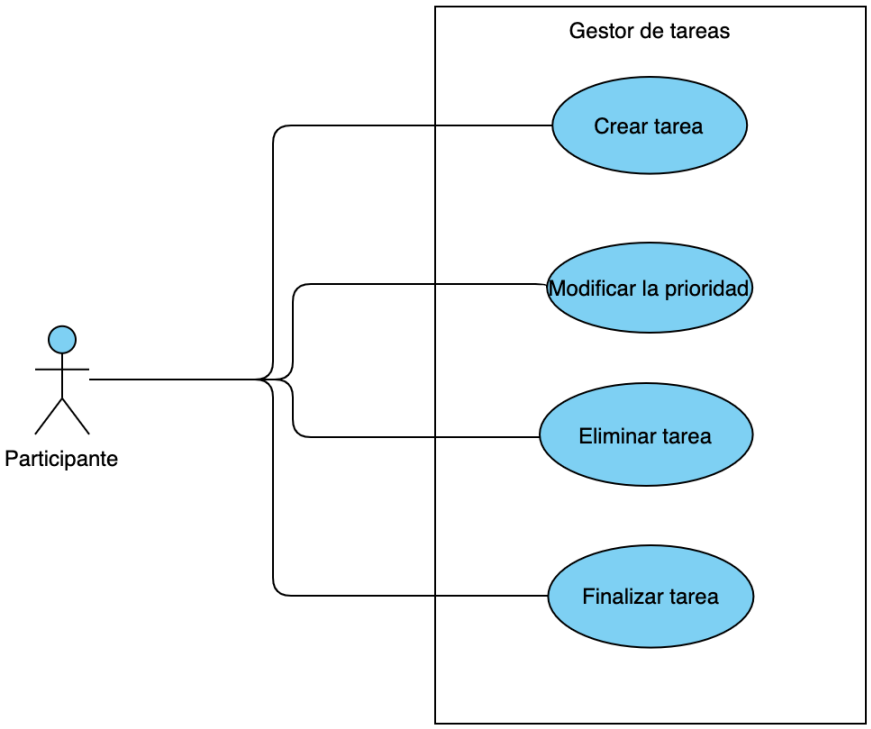

Figura 4.5: Diagrama de casos de uso del gestor de tareas.

4.2.2. Lógica de la aplicación

Al tratarse de una funcionalidad bastante sencilla, no fue excesivamente complejo crear un smart contract con Solidity que pudiera hacerse cargo de ella. Los conocimientos obtenidos realizando el primer proyecto fueron de gran utilidad para el desarrollo de este.

Lo que más nos sorprendió durante el desarrollo del *back-end*, fue lo limitadas que están las estructuras de datos en Solidity [57]. Incluyen únicamente *arrays*, *mappings* y *structs*.

Además, los *arrays* solo permiten su inserción y eliminación en la última posición, lo que conlleva a que los mapas sean la mejor opción en la mayoría de los casos.

Para este proyecto, se implementó una lista que almacena las diferentes tareas. Esto nos obliga a que, para borrar una tarea, tengamos que desplazar una posición hacia a atrás todas las que se encuentren en posiciones superiores a esta (ver figura 4.6). Esto es viable para un número reducido de tareas pero, si hubiéramos necesitado almacenar un número mayor, habría sido obligatorio usar un mapa para tratarlas evitando de esta manera, que los gastos en *ether* asciendan demasiado.

```
function deleteTask(string  name) external  auth(DELETETASK_ROLE){
    uint i = 0;
    while(i < tasks.length && !compareStrings(tasks[i].name, name)) {
        i++;
    }
    if(i != tasks.length) {
        for (uint j = i; j < tasks.length - 1; j++){
            tasks[j] = tasks[j+1];
        }
        delete tasks[tasks.length - 1];
        tasks.length--;
        numTasks = numTasks.sub(1);
        emit DeleteTask(msg.sender, name);
    }
}
```

Figura 4.6: Implementación de la eliminación de un elemento de una lista en Solidity.

La comunicación entre el *back-end* y el *front-end* la realizamos haciendo uso de la librería AragonAPI proporcionada por Aragon (ver subsección 2.4.3). Esta nos permite acceder al estado y a las funciones del *back-end* desde el *front-end*. Esta parte del desarrollo suponía algo novedoso para nosotros y para enfrentarnos a ello, se consultó de manera extensa la documentación de Aragon[40].

4.2.3. Interfaz de usuario

Para construir la interfaz de usuario, se hizo un amplio uso de los componentes que nos proporciona la librería AragonUI (ver subsección 2.4.3). Mezclando esta librería con conocimientos de CSS y React, se pudo alcanzar la experiencia de usuario que estábamos buscando sin demasiadas complicaciones.

Optamos por una interfaz sencilla en la que todas las funcionalidades de la aplicación estuvieran a la vista (ver figura 4.7). Gracias a la modularidad de React, se pudo crear la tabla de forma dinámica para que ajustase su tamaño según el número de tareas. Además, usando elementos de AragonUI, se crearon desplegables que muestran las opciones disponibles para cada acción (ver figura 4.8). Si aceptamos, aparecerá otra ventana, esta vez de Metamask, mostrándonos el coste en *ether* de la transacción y solicitando la confirmación de esta (ver figura 4.9).

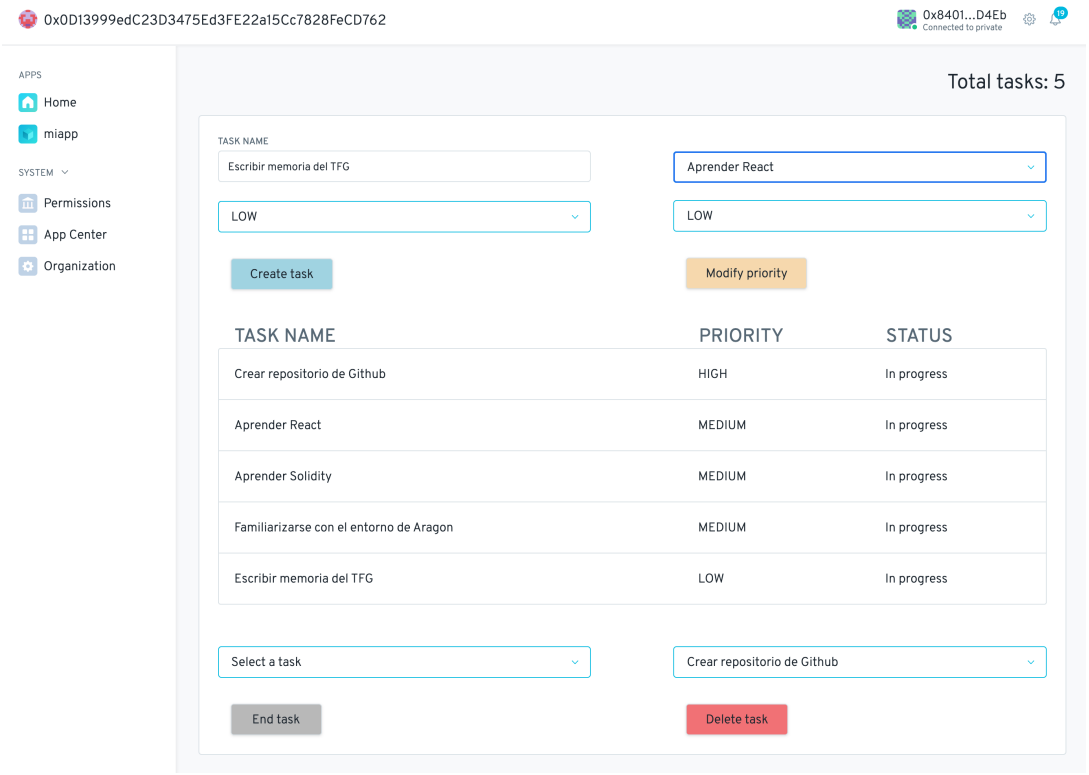

Figura 4.7: Vista general de la interfaz creada para el gestor de tareas descentralizado.

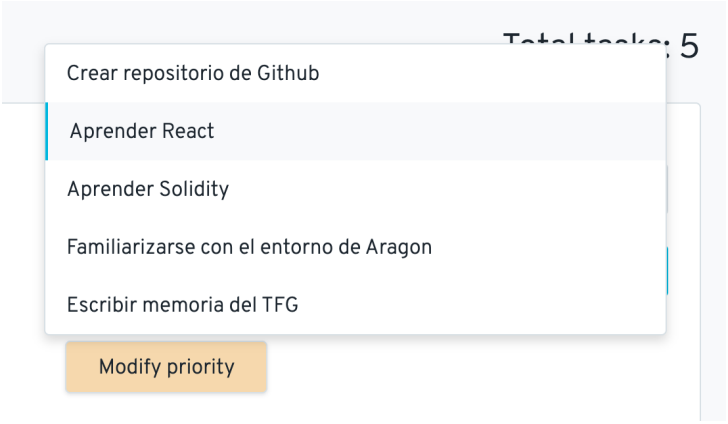

Figura 4.8: Desplegable que muestra las tareas cuya prioridad puede ser modificada.

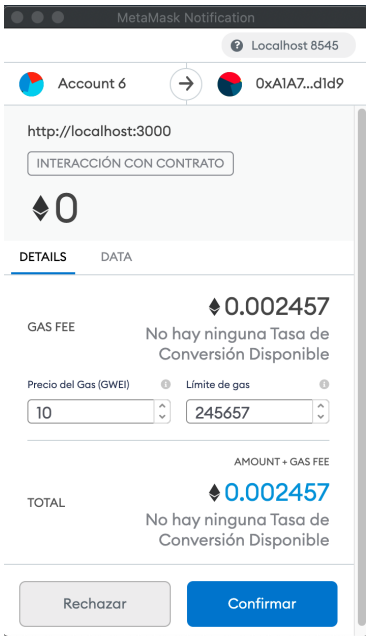

Figura 4.9: Notificación de Metamask con información de la transacción a realizar y petición de confirmación.

La integración de Metamask en Aragon facilita el proceso de ejecución de las funcionalidades haciendo uso de diálogos que guían al usuario en la situación en la que este encuentre. Si se intenta realizar una transacción sin tener una cuenta conectada, solicitará que se conecte. Una vez conectada, se mostrará un tabla de diálogo detallando la transacción y la dirección del contrato (ver figura 4.10).

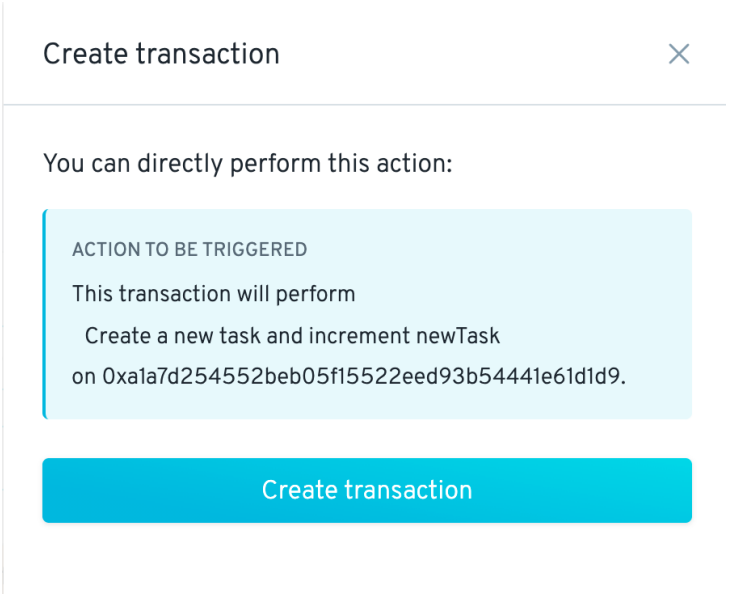

Figura 4.10: Diálogo que muestra Aragon indicando información sobre la transacción a realizar.

4.3. Experimentando con los diferentes frameworks

Después de comprobar el funcionamiento de los smart contract y de cómo implementar una pequeña DAO en el framework de Aragon, el siguiente paso fue examinar el desarrollo en otras plataformas.

4.3.1. DAOstack

Comenzamos realizando pruebas con DAOstack que desde su página web principal, ofrece la opción de lanzar una DAO. Ver figura 4.11.

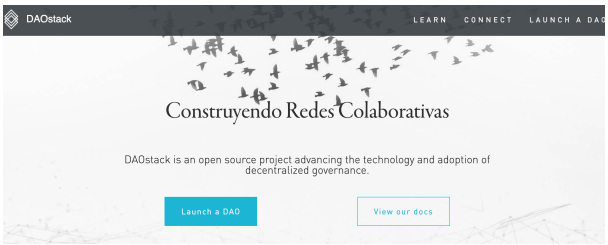

Figura 4.11: Página principal de DAOStack.

Partiendo desde este punto, accedemos a la dapp Alchemy [58] para crear DAOs. Alchemy nos muestra un formulario en el que podemos rellenar los datos de la organización (Nombre y símbolo del token, figura 4.12), la configuración de votación (figura 4.13) y los miembros que participan en ella (figura 4.14).

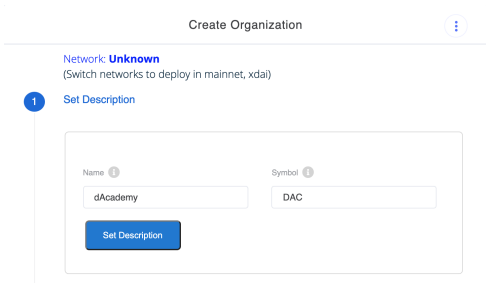

Figura 4.12: Crear una nueva organización con Alchemy.

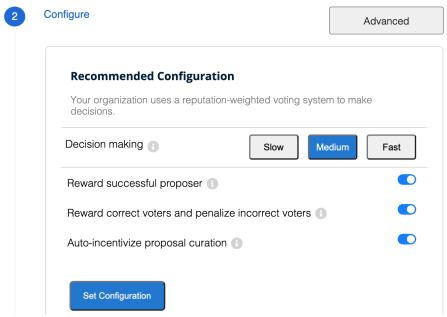

Figura 4.13: Configuración de votación para la DAO.

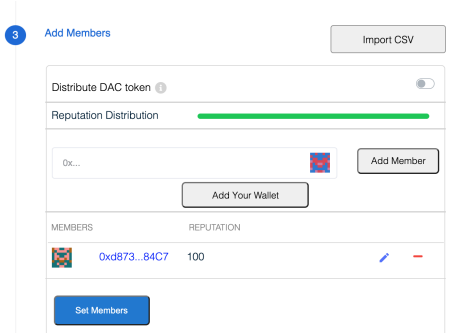

Figura 4.14: Añadir miembros a la organización.

En principio, parece que los pasos que debemos seguir para completar la creación de una DAO son sencillos. Cuando completamos el formulario y pulsamos sobre "Install organization" se realizan las operaciones necesarias para crear y desplegar la DAO en la red determinada (en nuetro caso la red de pruebas de Rinkeby) y seguidamente, nos muestra una URL dónde encontraremos nuestra DAO (figura 4.15).

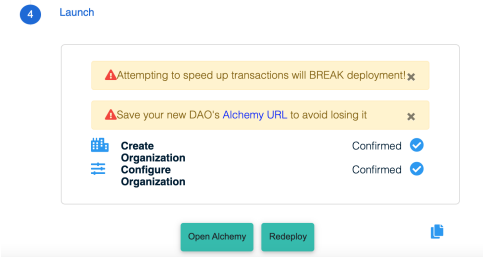

Figura 4.15: Lanzar la aplicación a la red.

Al intentar abrir nuestra organización nos encontramos con un mensaje de error que nos indica que algo no ha ido bien (figura 4.16). Se realizan varias pruebas entre los miembros del grupo que no aportan datos concluyentes a la hora de intentar subsanar el error mencionado.

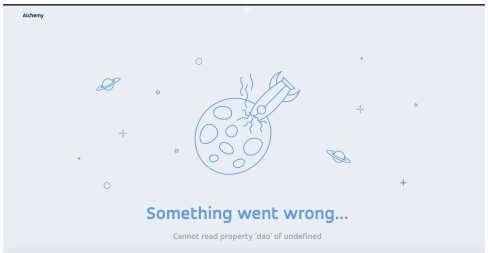

Figura 4.16: Error al intentar abrir la organización.

Puesto que la documentación de DAOstack solo indica como redes disponibles mainnet y xDAI, se vuelve a realizar otro intento de la prueba con la red xDAI. Para ello, se configura la red como RPC personalizado (ver figura 4.17), pero los resultados obtenidos en esta prueba tampoco son satisfactorios puesto que obtenemos el mismo error.

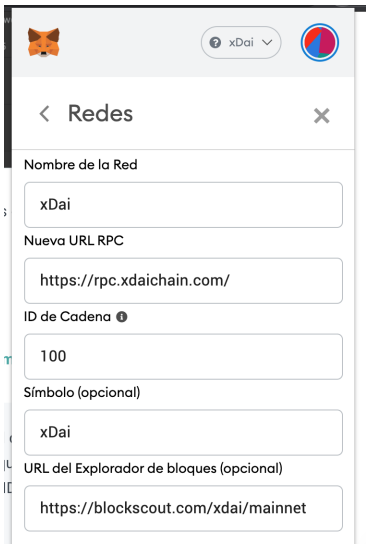

Figura 4.17: Creación del RPC personalizado para la red xDAI.

La documentación y la información obtenida del propio framework y de otros usuarios acerca de cómo crear la DAO, no ofrece una solución posible para continuar con la investigación y se continúan las pruebas en el resto de frameworks.

4.3.2. Colony

Tampoco fue posible crear una DAO con este framework puesto que actualmente tienen deshabilitada la opción de crear colonias. Colony está trabajando para disminuir el costo del gas en Ethereum, por lo que no quieren que los nuevos usuarios creen organizaciones haciendo uso del precio actual (ver figura 4.18). A pesar de que para este trabajo no haya resultado positivo puesto que no hemos podido probar la creación de DAOs en Colony, que se trabaje para solucionar este tipo de problemas de Ethereum es algo beneficioso para un futuro cercano.

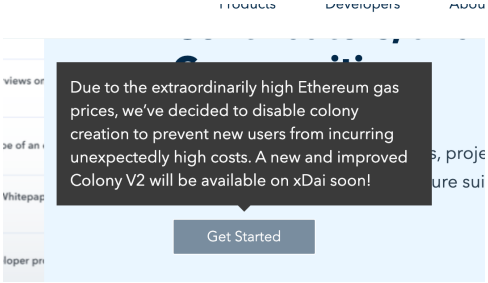

Figura 4.18: Mensaje informativo sobre la creación de colonias actualmente.

4.3.3. Aragon

Realizamos distintas pruebas para crear una DAO usando esta plataforma, tanto con la interfaz de cliente de Aragon, como desde la terminal.

Aragon nos ofrece la posibilidad de crear una DAO siguiendo una serie de pasos para realizar la configuración y lanzar la aplicación. En nuestro caso, a través de la red de pruebas de Rinkeby.

Primero accedemos al cliente de Aragon para la red Rinkeby (Figura 4.19), al comenzar a construir la DAO, lo primero que nos permitirá será elegir la plantilla que se adapte mejor a nuestro propósito. Cada una contiene una serie de aplicaciones de Aragon preinstaladas (figura 4.20). El siguiente paso es darle nombre a la organización (figura 4.21) y después, ajustaremos la configuración que queramos para cada aplicación (figura 4.22). Una vez tenemos los ajustes definidos, determinaremos el nombre del token y los propietarios del mismo (figura 4.23). Antes de finalizar, podremos revisar toda la configuración para asegurarnos de que es correcta (figura 4.24) y después podremos crear la DAO.

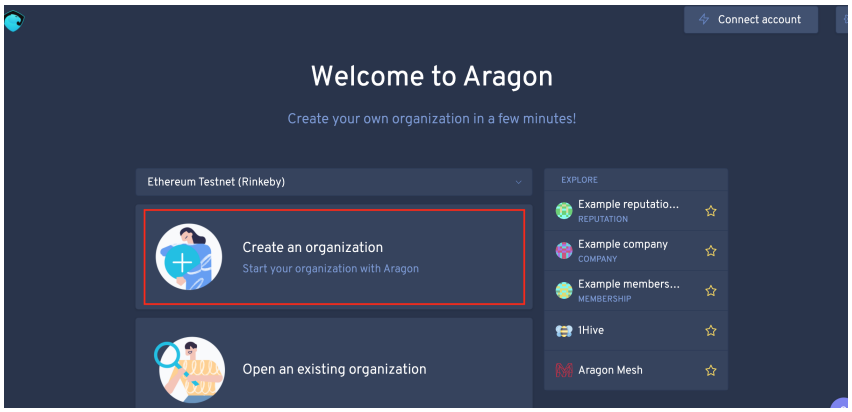

Figura 4.19: Vista del cliente de Aragon para Rinkeby.

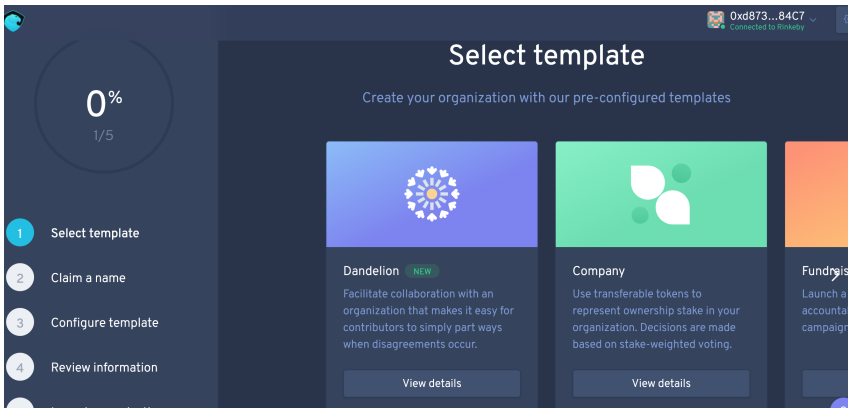

Figura 4.20: Selección de platillas para crear una DAO con Aragon.

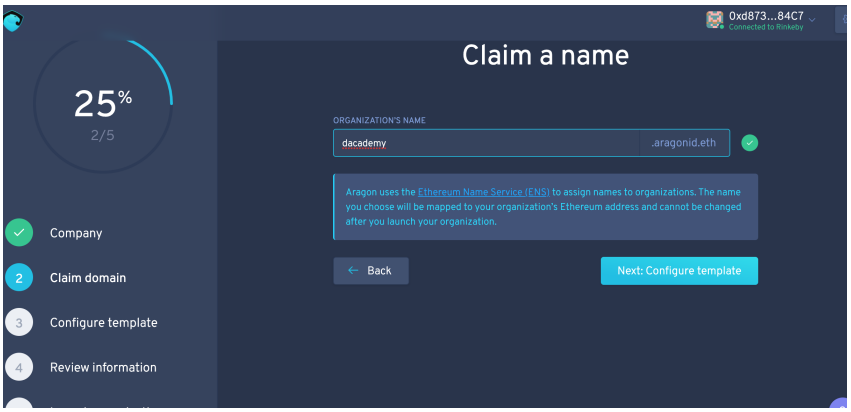

Figura 4.21: Selección del nombre de la organización.

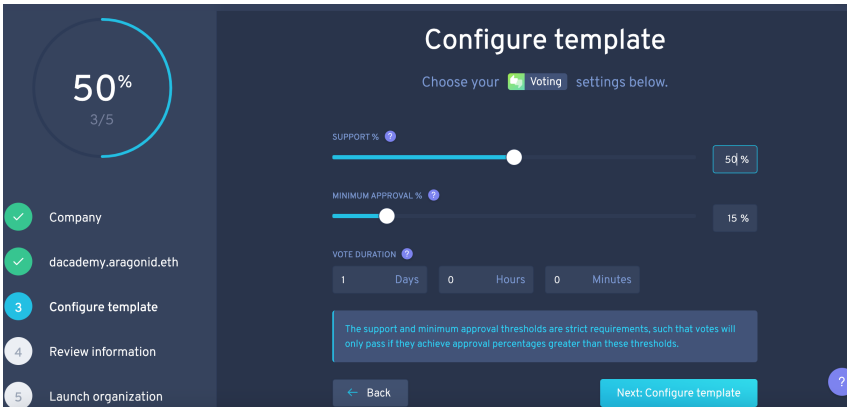

Figura 4.22: Ajustes de configuración para la aplicación Voting.

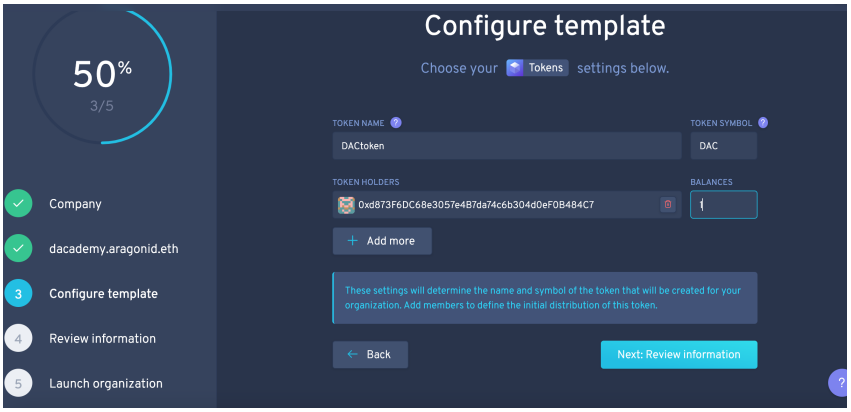

Figura 4.23: Ajustes del token de la organización.

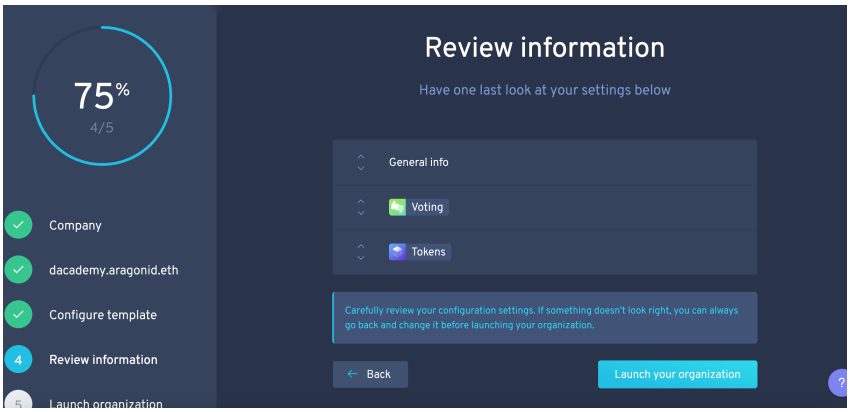

Figura 4.24: Comprobar los ajustes definidos en la organización.

Nuestra DAO nos permitirá interactuar con las aplicaciones que tenga instaladas (ver figura 4.25).

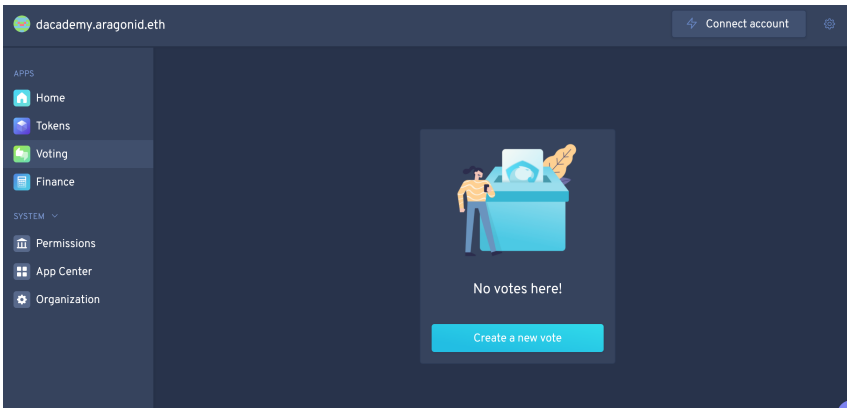

Figura 4.25: Vista de la aplicación Voting en una DAO de Aragon.

Por otro lado, podemos generar nuestro propio código desde la terminal. Para crear esta plantilla y desarrollar una aplicación de Aragon, escribimos el comando *npx create-aragon-app [nombre_de_la_dapp]*. Este comando genera un código ejecutable que podemos modificar para adaptarlo a nuestro caso de uso y generar nuestra propia DAO (ver figura 4.26).

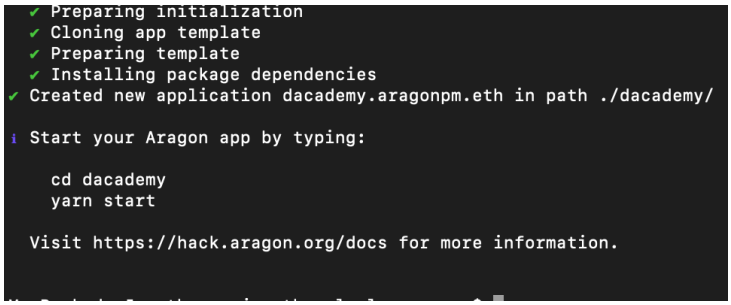

Figura 4.26: Crear una aplicación mediante el comando *npx create-aragon-app* desde el terminal.

4.3.4. Comparación entre frameworks

Aragon se enfoca en promover la tecnología libre y de código abierto a través de la creación de DAOs, potencia la libertad mediante la creación de herramientas liberadoras que aprovechan las tecnologías descentralizadas. Sus objetivos se centran en defender las elecciones de los usuarios, asegurando la participación descentralizada y colaborativa entre ellos.

DAOstack se centra más en la coordinación de todos los individuos en la organización. Impulsa el sistema de votación mediante propuestas y cuenta con el Consenso Holográfico que mencionamos en puntos anteriores (ver sección 2.4.1).

Colony en cambio, defiende una división meritocrática de los recursos y del trabajo. Fomenta la participación de los miembros de la colonia para ser compensados en función del trabajo que realicen. Las colonias existen para permitir la colaboración entre sus miembros y dirigir los esfuerzos colectivos hacia objetivos comunes.

Como conclusión acerca de nuestra experiencia probando los distintos frameworks, podemos destacar:

- Los tres fameworks ofrecen una interfaz para que el usuario cree una DAO siguiendo unos pasos determinados, aunque en nuestro caso solo ha se ha podido finalizar correctamente en Aragon (ver sección 4.3).
- Aunque los tres cuentan con un sistema de votación, DAOstack y Aragon son mayormente impulsados por el voto, mientras que Colony se centra más en la meritocracia y apoyo a la comunidad mediante tareas [59].
- Cada framework permite unas redes distintas para su lanzamiento.
  - **Aragon**: localhost, Rinkeby y mainnet.
  - **DAOstack**: mainnet y xDAI.
  - **Colony**: Goerli, localhost y mainnet.
- Los tres poseen funcionalidades para tokens, reputación y fondos.
- Aragon y Colony permiten asignación de permisos mediante un sistema de control de acceso que permite conectar las aplicaciones de forma segura, mientras que DAOstack no.
- Aragon proporciona plantillas para crear la organización y la posibilidad de personalizar una plantilla propia a partir de añadir aplicaciones mediante smart contracts. DAOstack por el contrario, no permite una gran personalización.

Tras realizar las pruebas oportunas con los diferentes frameworks, se toma la decisión de hacer uso de Aragon para el desarrollo de la DAO. Puesto que con los otros frameworks no obtenemos resultados satisfactorios y que Aragon posibilita el desarrollo de la aplicación aportando las herramientas necesarias, nos parece la opción más acertada continuar la implementación sobre esta plataforma.

Capítulo 5

dAcademy: Una red descentralizada de cursos certificados

5.1. Nuestra idea

Para conseguir una idea innovadora y adecuada al problema en cuestión, decidimos realizar una sesión de brainstorming (ver subsección 3.1.4). La sesión fue realizada el 21 de febrero de 2021. Tras realizar la sesión, decidimos filtrar y organizar la ideas resultantes, obteniendo así la siguiente lista de potenciales aplicaciones:

- **Ofertas de trabajo**: Plataforma descentralizada donde los usuarios pueden ofrecer y/o reclamar servicios o puestos de trabajo a cambio de una compensación en tokens que quedará registrada en la blockchain.
- **Obras de arte**: Plataforma en la que usuarios con talento podrán adquirir reputación a la vez que venden o financian sus obras de arte en la comunidad con el uso de tokens.
- **Cursos de formación variada**: Plataforma donde se podrán ofrecer cursos con certificación la cual tendrá mayor o menor validez en función de la reputación en la comunidad. Los profesores obtendrán tokens como compensación al valor educativo aportado.
- **Noticias**: Plataforma de noticias donde se podrá certificar la validez de cada una de ellas atendiendo a la reputación dada por la comunidad a esa noticia en concreto o a su autor.
- **Preguntas y respuestas**: Plataforma donde los usuarios podrán plantear preguntas ofreciendo una compensación a la mejor respuesta. También se plantea la posibilidad, de realizar preguntas a usuarios concretos que tengan reputación en la comunidad, a cambio de un pequeño pago dependiendo de la reputación del usuario.

Tras hablarlo con los tutores del proyecto, decidimos seguir adelante con el proyecto de cursos de formación variada y llamarlo dAcademy.

dAcademy es una plataforma que permite a los usuarios adquirir conocimientos y habilidades a través de cursos de formación en una red descentralizada que asegura la validez de la certificación mediante la reputación que la comunidad le ha otorgado. A su vez, los usuarios podrán también ejercer como profesores obteniendo su correspondiente remuneración en ether. Es decir, la DAO tiene distintos integrantes que tienen la posibilidad de ofrecer cursos a la comunidad o de aprovechar los que crean otros usuarios recibiendo o pagando una cantidad de ether determinada (ver figura 5.1). De esta manera, son los usuarios los que aportan el valor y la reputación a los títulos otorgándoles la validez conjunta de la comunidad.

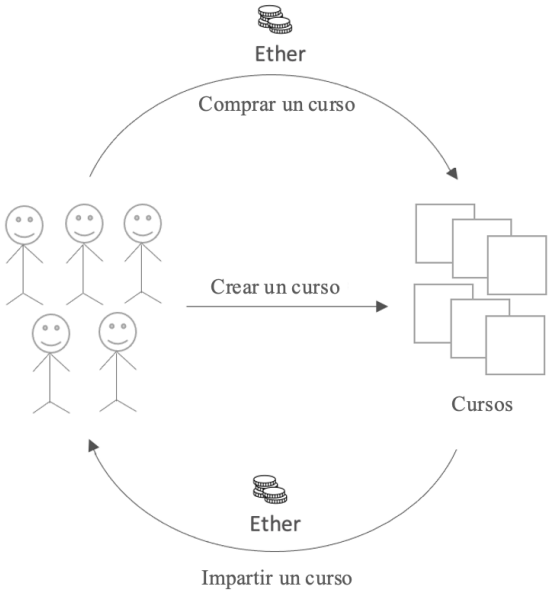

Figura 5.1: Red de cursos dAcademy.

Con esta aplicación descentralizada, queremos facilitar la transmisión de conocimientos en la comunidad para que cualquiera pueda actuar como profesor u alumno. Al realizarse las transacciones en ether, personas de lugares del mundo completamente diferentes podrán adquirir u ofertar cursos sin necesidad de intermediarios financieros, lo que simplifica y abarata las relaciones económicas entre los integrantes de la comunidad.

Además, la descentralización alinea los incentivos de profesores y alumnos. Por un lado, los alumnos no se sentirán presionados a elegir su formación teniendo en cuenta la reputación de la entidad que la respalda, pues solo se tendrá en cuenta la reputación de cada curso y profesor de forma individual. Por otro, los profesores no podrán acomodarse en la institución que los respalda, si no que tendrán que ofrecer un buen servicio a los alumnos para obtener valoraciones positivas de estos.

Finalmente, cada curso realizado y opinión emitida, quedan registrados de manera pública en la blockchain, por lo que cualquiera puede consultar la formación de los usuarios y el valor de esta en función de lo que la comunidad opine en conjunto [41]. De esta forma, las opiniones excéntricas o interesadas quedarán diluidas entre el resto de opiniones, mostrando con el paso del tiempo una representación fiel de cómo la comunidad valora el curso. Es así como este sistema asegura la veracidad de la información y hace imposible la falsificación de los títulos.

5.2. Especificación de requisitos

En esta sección se detallará una descripción completa del comportamiento del sistema que se va a desarrollar. Incluiremos un conjunto de casos de uso (ver figura 5.2), usando notación UML con referencia a la última versión, que buscarán abarcar todas las posibles interacciones que los usuarios tengan con la aplicación.

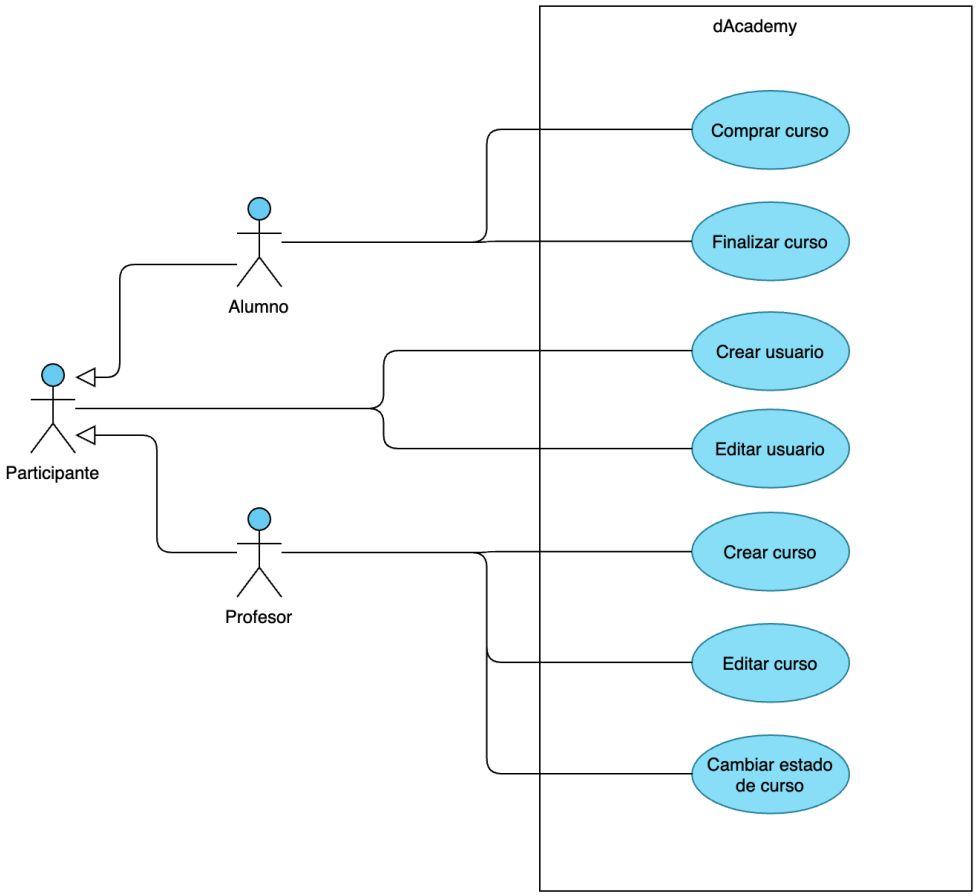

Figura 5.2: Diagrama de casos de uso.

5.2.1. Casos de uso

Para detallar debidamente toda la información referente a cada caso de uso incluiremos, para cada uno, la siguiente información:

- **Tabla**: Contendrá una descripción de la funcionalidad del caso de uso, su valores de entrada y salida, las condiciones iniciales necesarias para que se lleve a cabo y los efectos posteriores que tendrán tras su ejecución.
- **Diagrama de actividad**: Indicará de forma gráfica el flujo de acciones que compondrán cada caso de uso desde su inicio hasta su finalización.

Crear usuario

Nombre	Crear usuario
Actores	Participante
Objetivo	Crear un nuevo usuario.
Precondiciones	
Postcondiciones	- Se crea un nuevo usuario.
Escenario básico	Un usuario se registra en la aplicación. Si la cuenta de Ethereum asociada al usuario ya tiene un perfil en la aplicación se muestra un error.

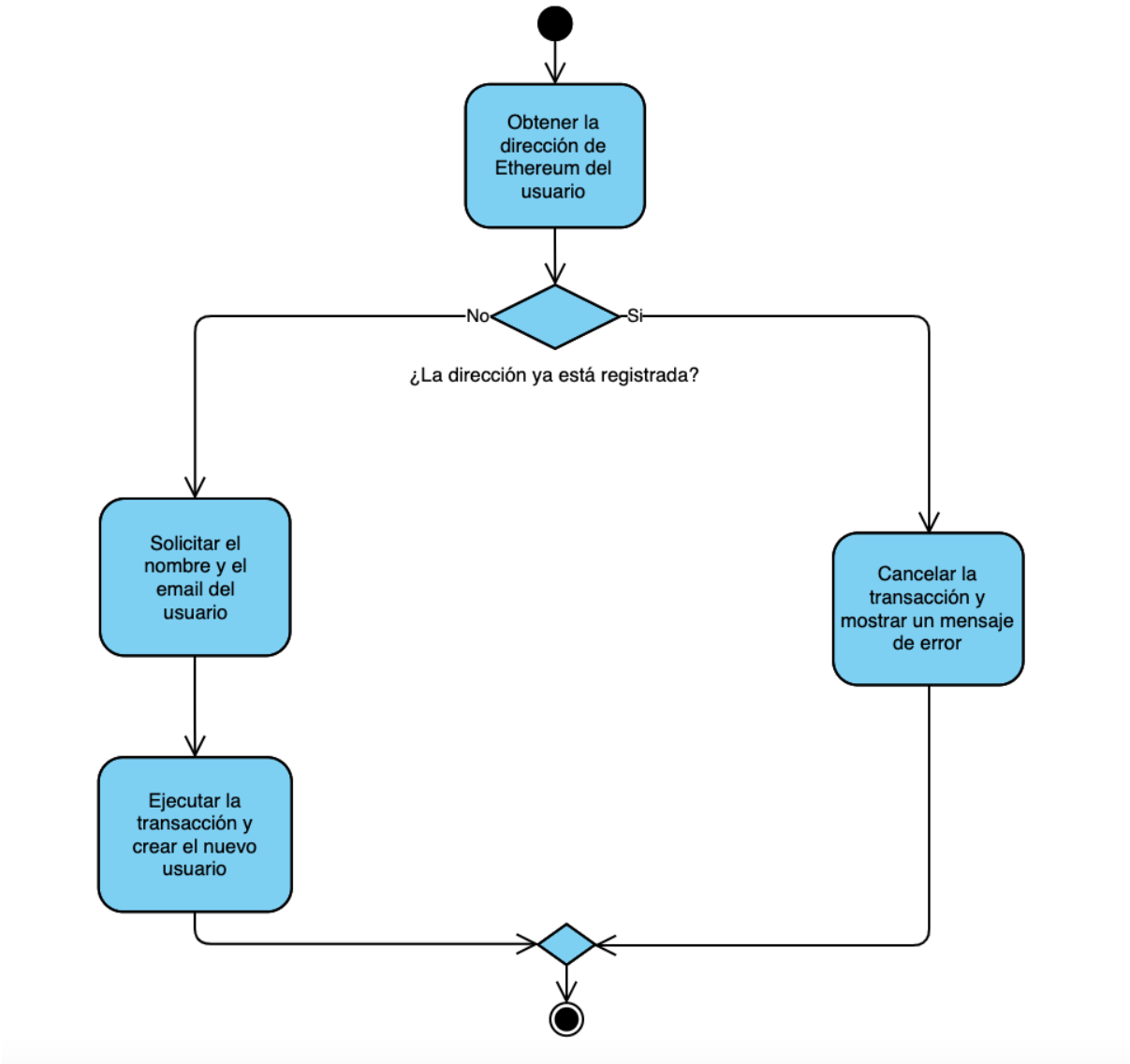

Figura 5.3: Diagrama de actividad del requisito funcional: crear usuario.

Editar usuario

Nombre	Editar usuario
Actores	Participante
Objetivo	Un usuario edita su nombre y/o email.
Precondiciones	- El usuario está registrado en la aplicación.
Postcondiciones	- El usuario se actualiza con los nuevos campos.
Escenario básico	Un usuario edita su nombre y/o su email. Si los campos que desea editar no le pertenecen se muestra un error.

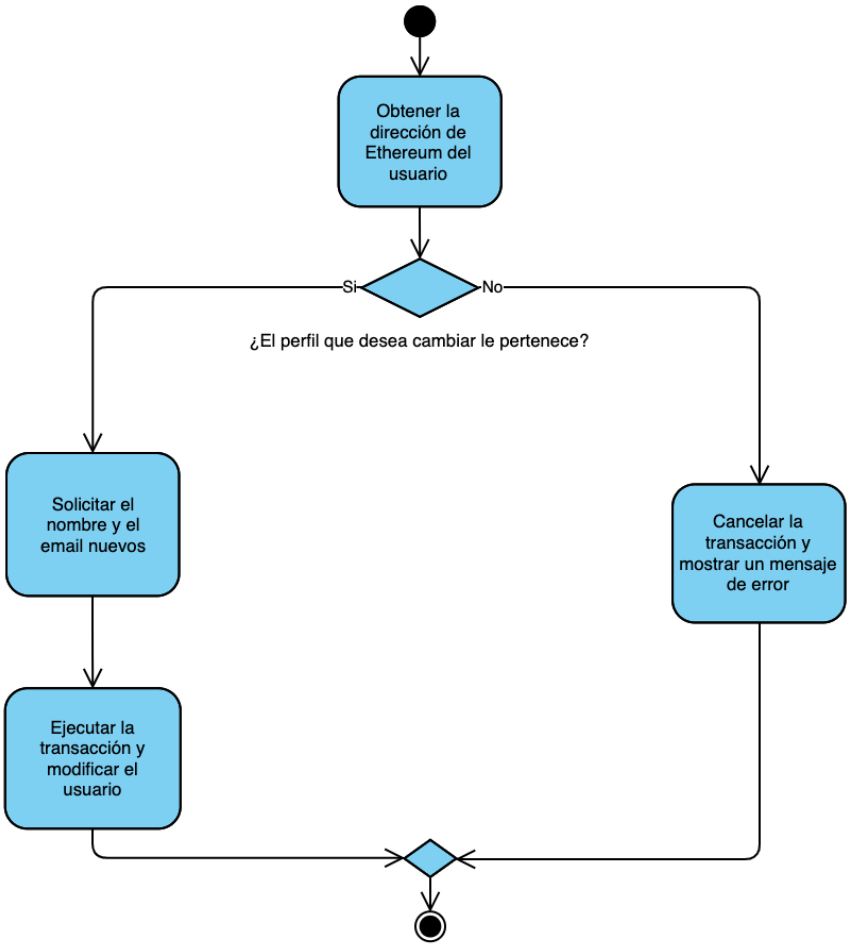

Figura 5.4: Diagrama de actividad del requisito funcional: editar usuario.

Crear curso

Nombre	Crear curso
Actores	Profesor
Objetivo	Crear un curso.
Precondiciones	- El usuario está registrado en la aplicación.
Postcondiciones	- La cantidad de diez veces el precio en ether ha sido transferida del usuario al contrato. - El curso se crea y pasa a estar disponible para el resto de usuarios. - El curso es asignado al usuario que lo crea.
Escenario básico	Un usuario crea un curso pagando una fianza de diez veces el valor de este para asegurar diligencia a la hora de impartirlo. Si el usuario no tiene suficiente ether, se muestra error.

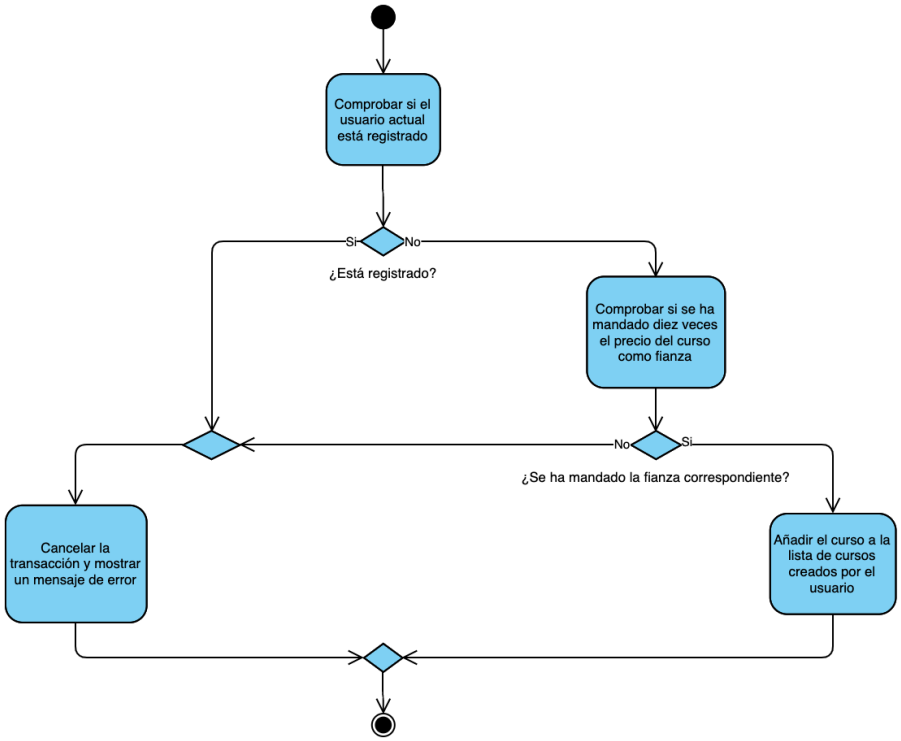

Figura 5.5: Diagrama de actividad del requisito funcional: crear curso.

Editar curso

Nombre	Editar curso
Actores	Profesor
Objetivo	Editar un curso.
Precondiciones	- El curso debe pertenecer al usuario que lo desea editar.
Postcondiciones	- El curso se actualiza con los nuevos campos.
Escenario básico	Un usuario edita la descripción de un curso que imparte. Si el curso no le pertenece se muestra un error.

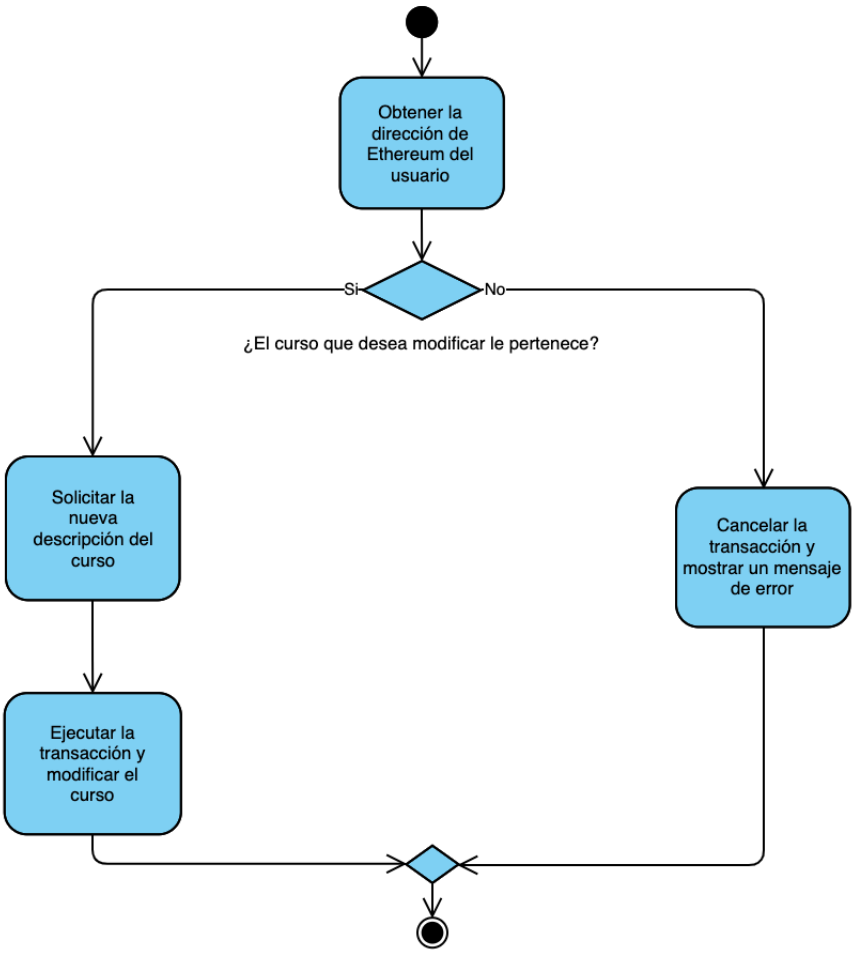

Figura 5.6: Diagrama de actividad del requisito funcional: editar curso.

Cambiar estado de curso

Nombre	Cambiar disponibilidad del curso
Actores	Profesor
Objetivo	Se cambia el estado del curso.
Precondiciones	- El usuario puede modificar el estado únicamente de un curso que le pertenezca.
Postcondiciones	- El curso se actualiza con el nuevo estado.
Escenario básico	Un usuario cambia el estado de un curso que imparte. Si se encuentra habilitado lo cambiará a deshabilitado y viceversa.

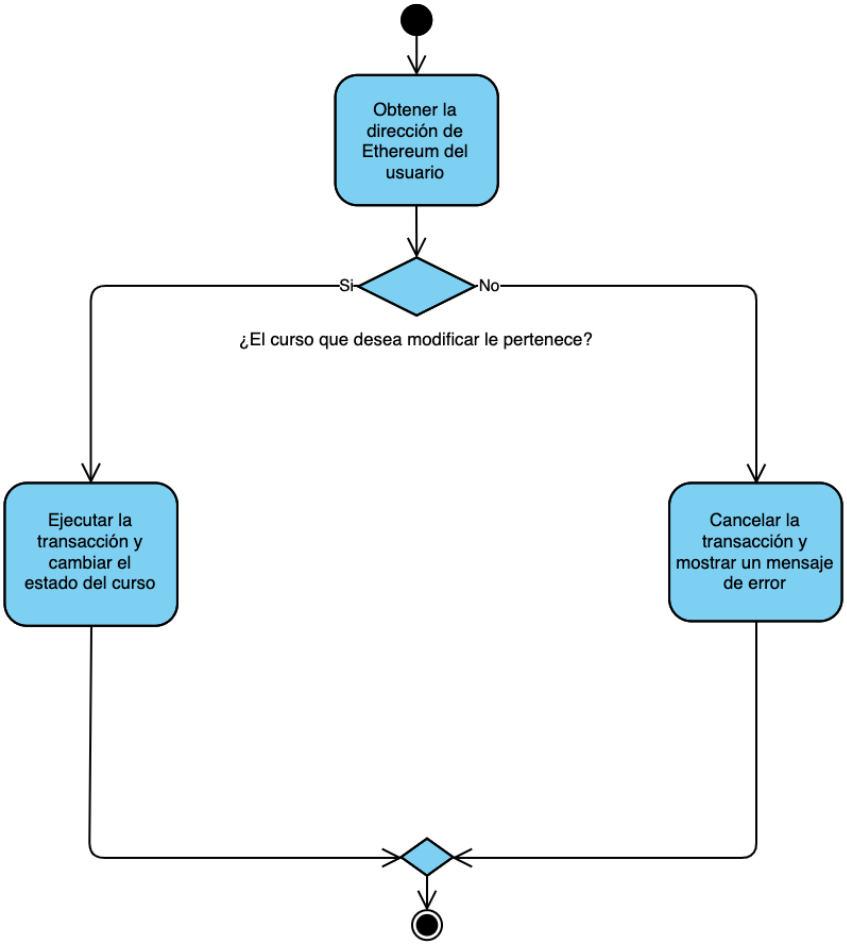

Figura 5.7: Diagrama de actividad del requisito funcional: Cambiar estado de un curso.

Comprar curso

Nombre	Comprar curso
Actores	Alumno
Objetivo	Un usuario adquiere un curso.
Precondiciones	
Postcondiciones	- El precio en ether ha sido transferido desde la cuenta del usuario a la del creador del curso. - El curso pasa a ser accesible para el usuario.
Escenario básico	Un usuario compra un curso pagando una cantidad determinada. El precio del curso es transferido en ether desde la cuenta del usuario que lo compra a la del creador del curso.

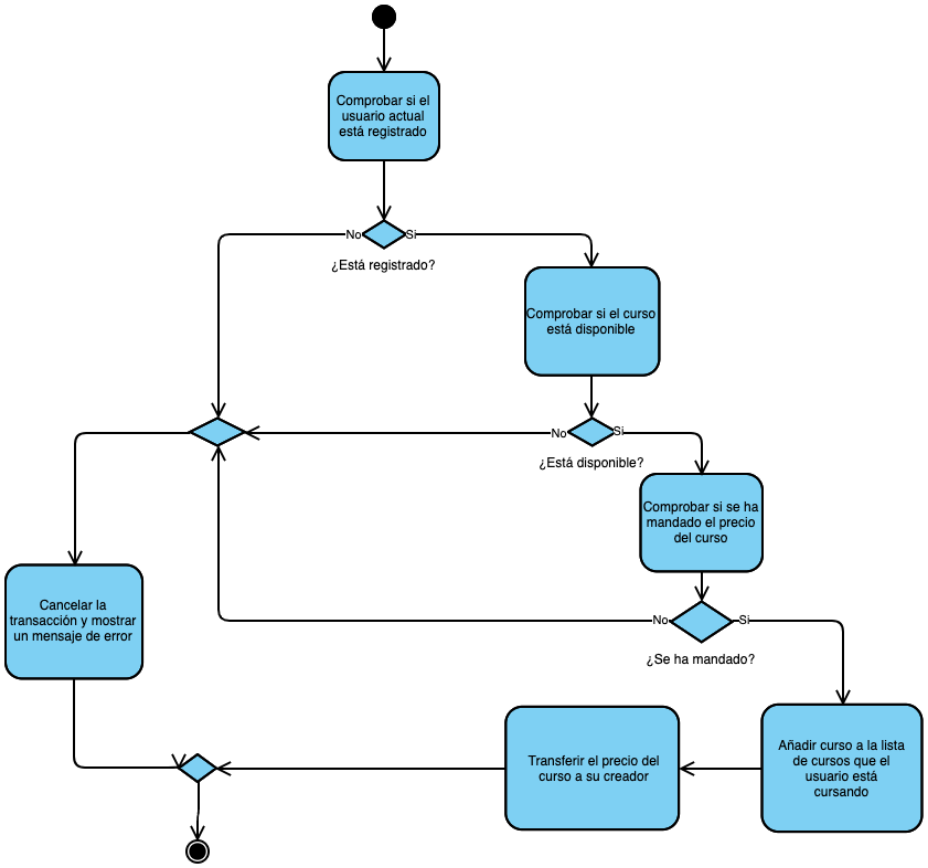

Figura 5.8: Diagrama de actividad del requisito funcional: comprar curso.

Finalizar curso

Nombre	Finalizar curso
Actores	Alumno
Objetivo	El usuario finaliza uno de los cursos que estaba realizando.
Precondiciones	- El usuario debe de haber adquirido el curso previamente.
Postcondiciones	- La valoración ha sido publicada. - La reputación del curso ha sido actualizada. - Si aún no había recibido toda la fianza, el creador del curso habrá recibido una décima parte de esta.
Escenario básico	Un usuario, tras tomar un curso, indica que lo ha finalizado y deja una valoración de este. Si aún no se ha devuelto toda la fianza al creador del curso, se le devuelve una décima parte de esta.

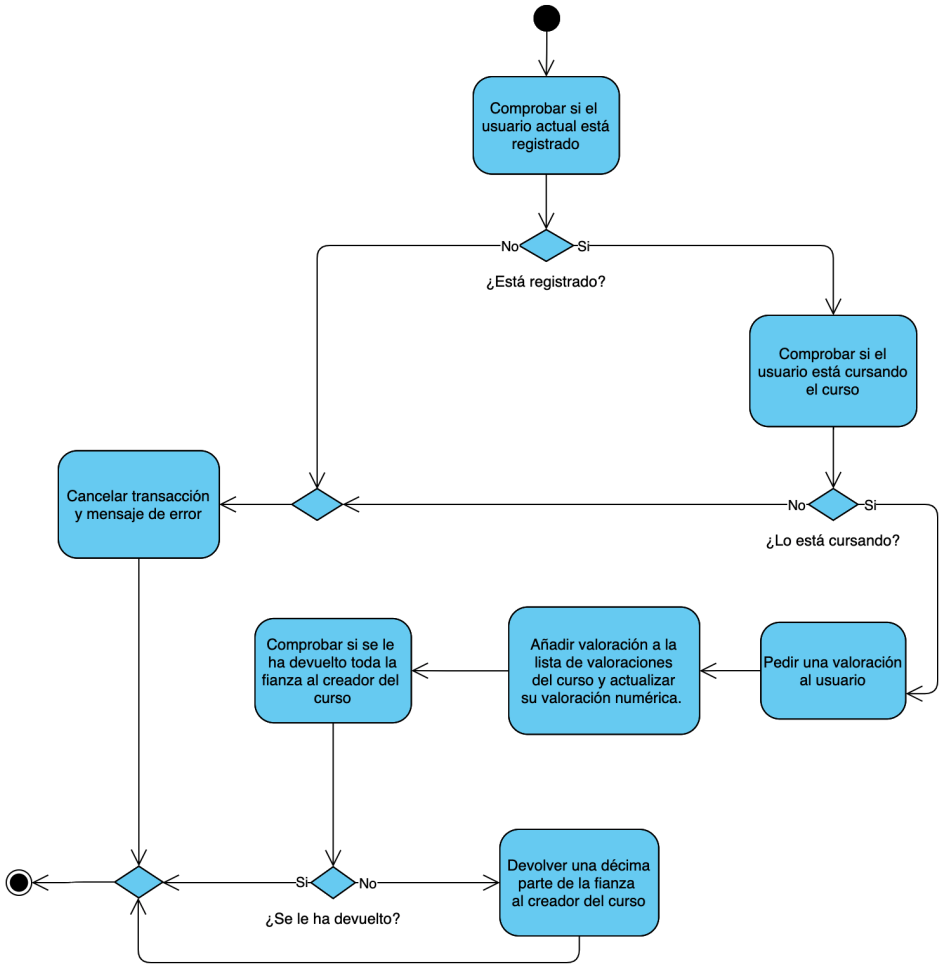

Figura 5.9: Diagrama de actividad del requisito funcional: finalizar y valorar curso.

5.2.2. Requisitos no funcionales

Las tecnologías descentralizadas presentan limitaciones que debemos tener en cuenta a la hora de desarrollar sobre ellas. A continuación, vamos a enumerar los requisitos no funcionales que presenta el desarrollo de esta DAO.

- **Coste**. Las tecnologías utilizadas requieren un coste en ETH, que debe pagar el usuario en cada transacción que suponga un cambio de estado en la blockchain, para ejecutar la lógica implementada en los *smart contracts*. Por ello, es un requisito importante minimizar al máximo este tipo de costes, implementando un código simple para el tratamiento de las acciones, en las que el usuario realice los mínimos cambios posibles en la blockchain.
- **Almacenamiento**. Existen distintos tipos de almacenamiento en la blockchain: Pila (stack), Memoria (Memory) y variables de estado (Storage). Cada uno de estos tipos cuenta con una serie de operaciones que tienen un alto coste de gas. Este coste debe ser pagado en ethers por el usuario en la transacción correspondiente. Debido a que guardar datos en Ethereum es bastante caro, se debe tratar de evitar el almacenamiento en la cadena de bloques y hacer uso de otros sistemas como IPFS. En la implementación del smart contract se han seleccionado unos datos muy simples y de poco tamaño que permitan minimizar este coste.
- **Usabilidad**. Otro requisito sumamente importante es conseguir una interfaz que no sea demasiado compleja para el usuario. Debe presentar todas las acciones disponibles que se pueden ejecutar, en una vista clara y limpia. De esta forma, el usuario se sentirá cómodo y seguro de las acciones que realiza interactuando con el sistema, sabiendo en todo momento lo que puede hacer y lo que no. Una buena interfaz le permite aprender rápidamente todo su funcionamiento. Por otro lado, se debe tener en cuenta la necesidad de utilizar monederos digitales que requieren un registro previo para disponer de una cuenta de Ethereum. Los usuarios pueden no estar familiarizados con el uso de monederos digitales, lo que supone una complicación más a la hora de interactuar con DAOs.
- **Rendimiento**. El rendimiento de la aplicación está limitado debido a que se encuentra desarrollada sobre la plataforma Aragon. La tecnología aún es bastante mejorable y los tiempos de ejecución son bastante altos en determinadas acciones. Además, las transacciones que deben ejecutarse en cada acción, requieren un tiempo para ser aprobadas. Esto hace que el uso del sistema tome un tiempo extra de ejecución aunque para la mayoría de las acciones, no es necesario que se reciba una respuesta inmediata.

5.2.3. Diseño de la interfaz de usuario: mockups

Para el desarrollo de la interfaz de la aplicación, se elaboraron distintos mockups con un primer diseño de las vistas. Siguiendo la estructura de la interfaz de Aragon, se presentaron componentes con un diseño simple y limpio, consiguiendo de esta manera, una interfaz intuitiva de cara al usuario.

A continuación, se muestran los mockups realizados.

En primer lugar, se realiza la vista principal donde se listan todos los usuarios registrados (ver figura 5.10). Contiene además, una barra de búsqueda con la que filtrar a los usuarios

por nombre y un botón que permite añadir nuevos usuarios.

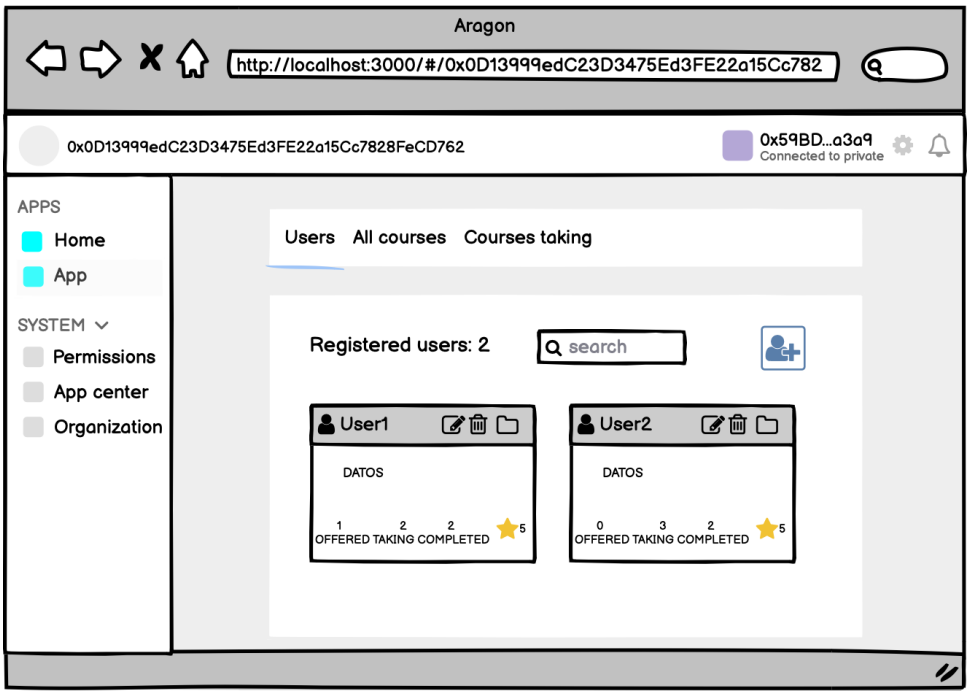

Figura 5.10: Vista principal de los usuarios registrados.

En la siguiente vista (ver figura 5.11), se muestran todos los cursos registrados. De la misma forma que en la vista anterior, se puede buscar por nombre del curso o añadir otros nuevos.

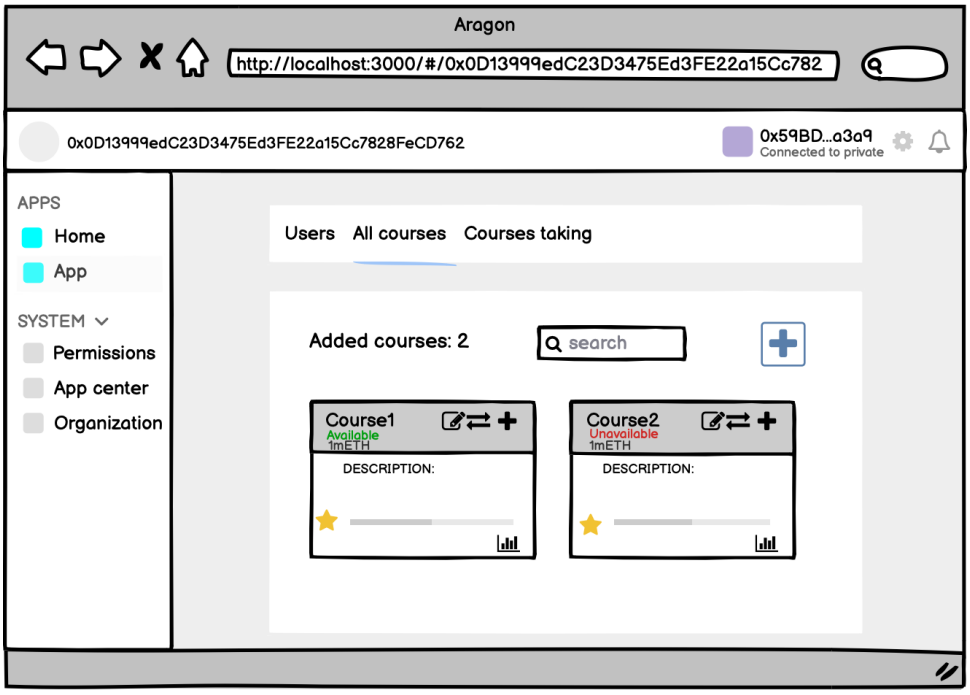

Figura 5.11: Vista principal de los cursos añadidos.

A esta otra vista, al igual que a las dos anteriores, accedemos mediante la barra de pes-

tañas. En este caso, veremos los cursos que tiene matriculado el usuario que se encuentre conectado en ese momento. (ver figura 5.12).

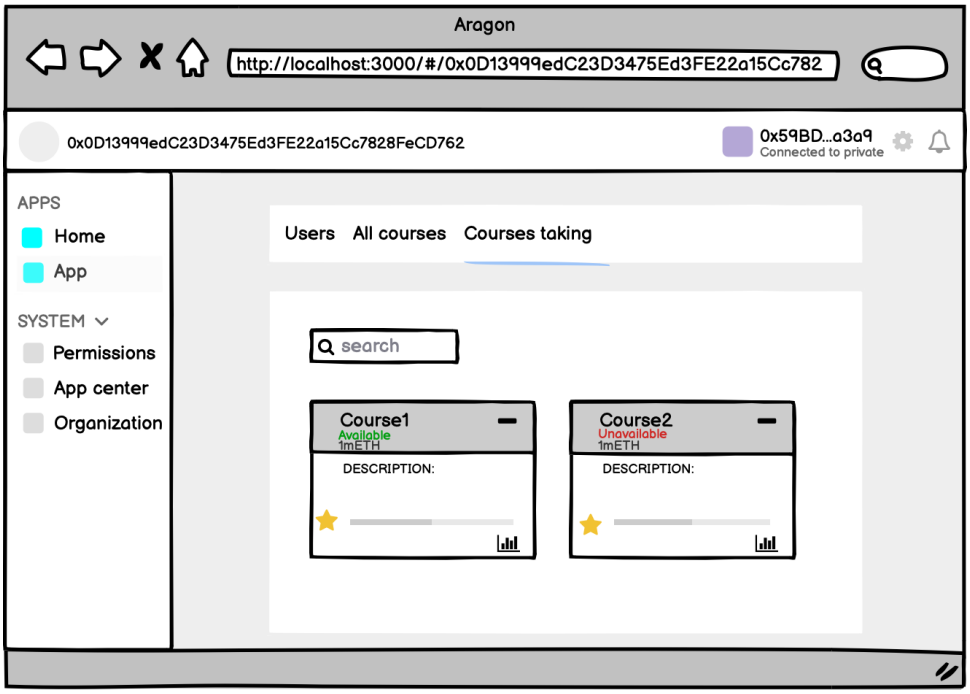

Figura 5.12: Vista de los cursos en los que está matriculado el usuario actual.

Para tener una primera visión de los formularios, se opta por usar una ventana modal que muestre los campos necesarios para la creación de un nuevo usuario, un nuevo curso o para aportar una valoración a un curso (ver figura 5.13).

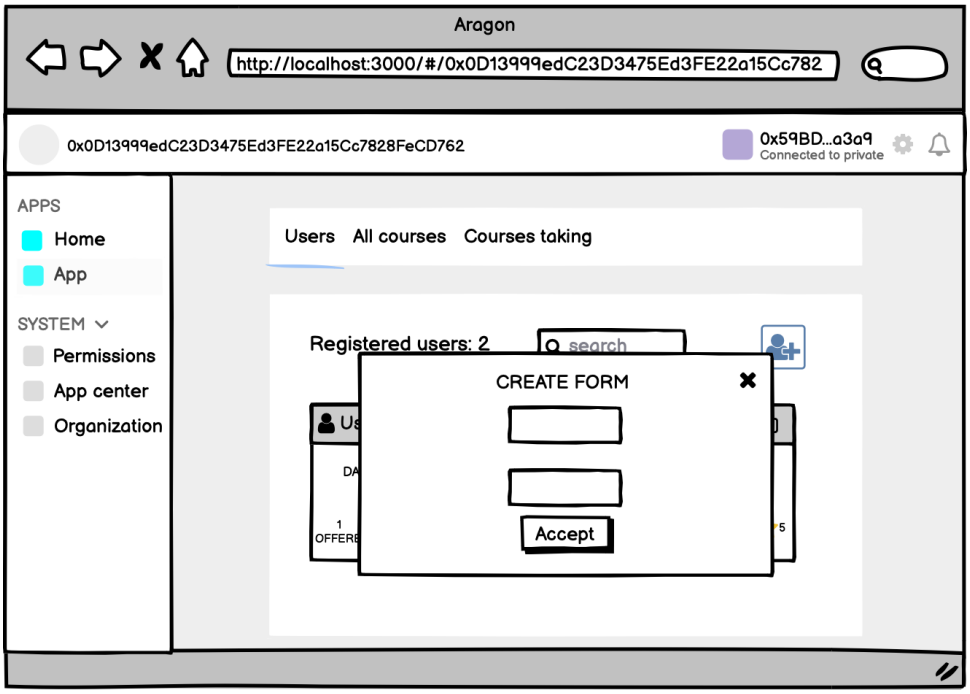

Figura 5.13: Vista del formulario para crear un nuevo usuario en la red. Mismo diseño para crear un curso o una valoración.

Del mismo modo que el formulario anterior, en la figura 5.14 se muestra como se verá un formulario para editar los campos de un usuario o de un curso.

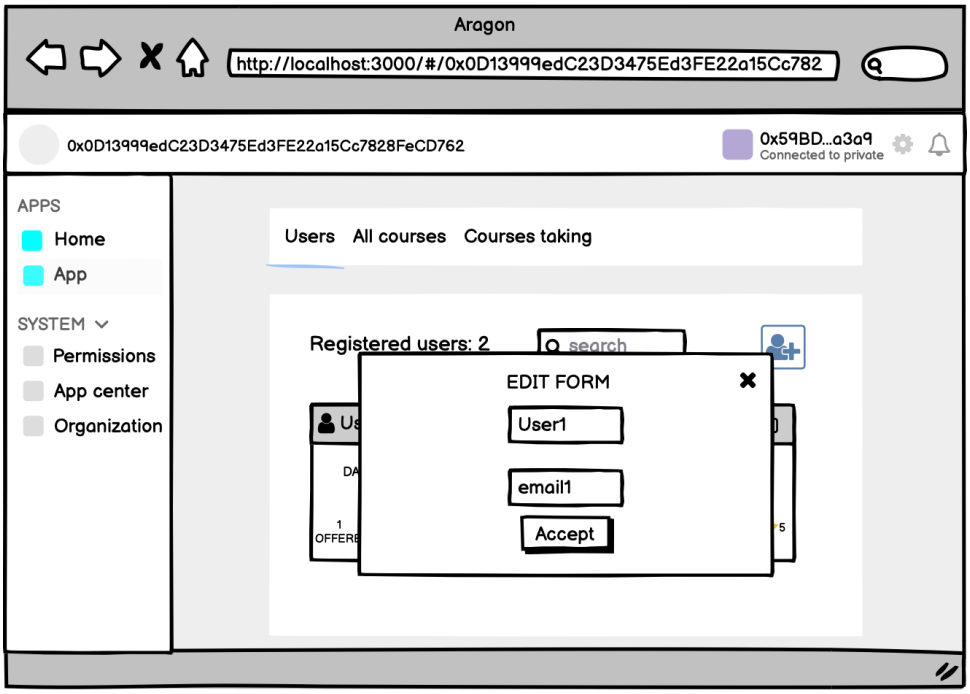

Figura 5.14: Vista del formulario para editar un usuario registrado. Mismo diseño para editar un curso.

En el mockup de la figura 5.15 se presenta cómo se mostrarán las valoraciones de un curso mediante una ventana modal.

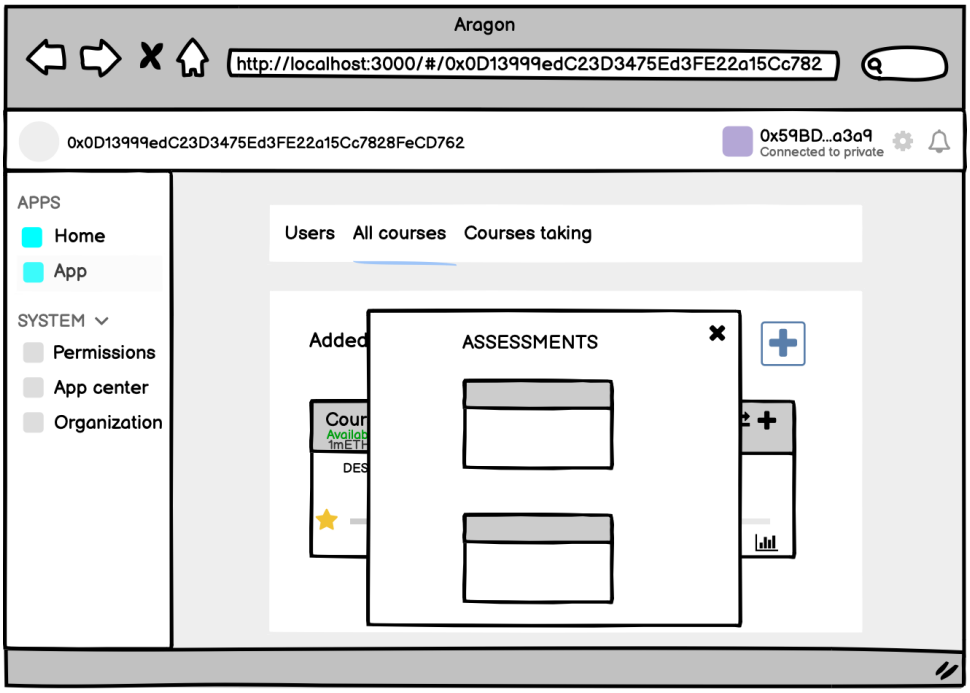

Figura 5.15: Vista de las valoraciones de un curso.

Por último, podemos ver en la figura 5.16 cómo representamos, de nuevo usando una ventana modal, la vista que mostrará los cursos ofrecidos, matriculados o que ha completado un usuario concreto.

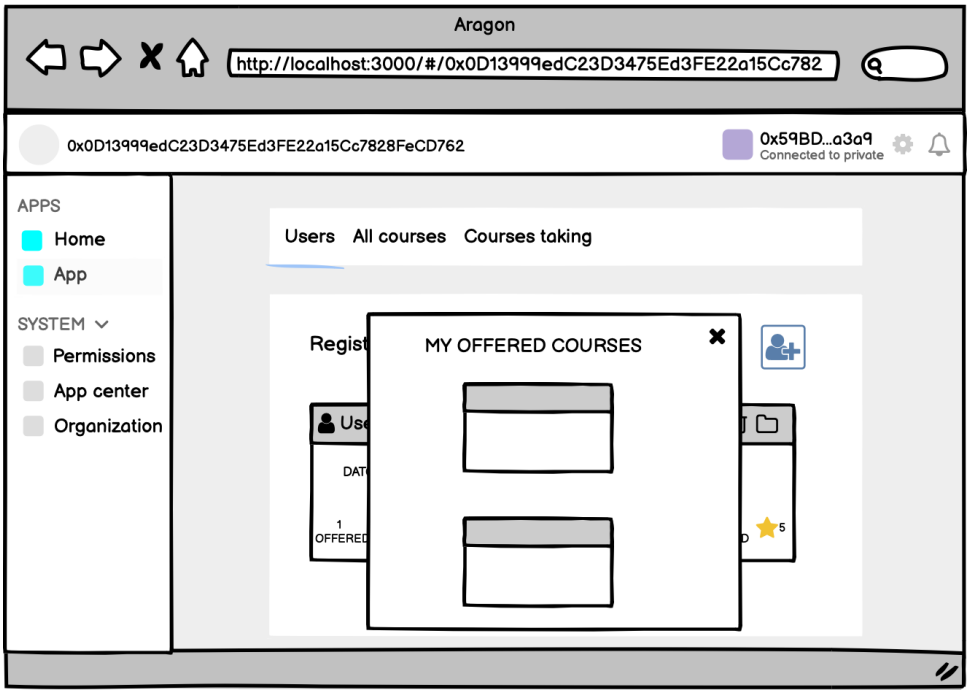

Figura 5.16: Vista de los cursos ofrecidos por un usuario concreto.

5.3. Implementación

Tras tener claros los requisitos y el aspecto que debería tener nuestra aplicación, comenzamos con la implementación. Esta se divide en varias partes: el back-end, que consta de smart contracts codificados con Solidity, el front-end, que está formado por vistas implementadas con React, y la integración de ambos que se realiza a través de un script en JavaScript proporcionado por Aragon al que llamamos *background script* (ver figura 5.17). El flujo de información de la aplicación funciona de la siguiente manera:

1. La vista se comunica con el contrato a través del módulo Aragon API (concretamente usando React API) invocando métodos de este, lo que llamamos *intentos*.
2. El smartcontract, al ejectuar el método, avisa al background script mediante un *evento*.
3. El background script reenvía la información recogida en el evento a la vista mediante un *estado*.

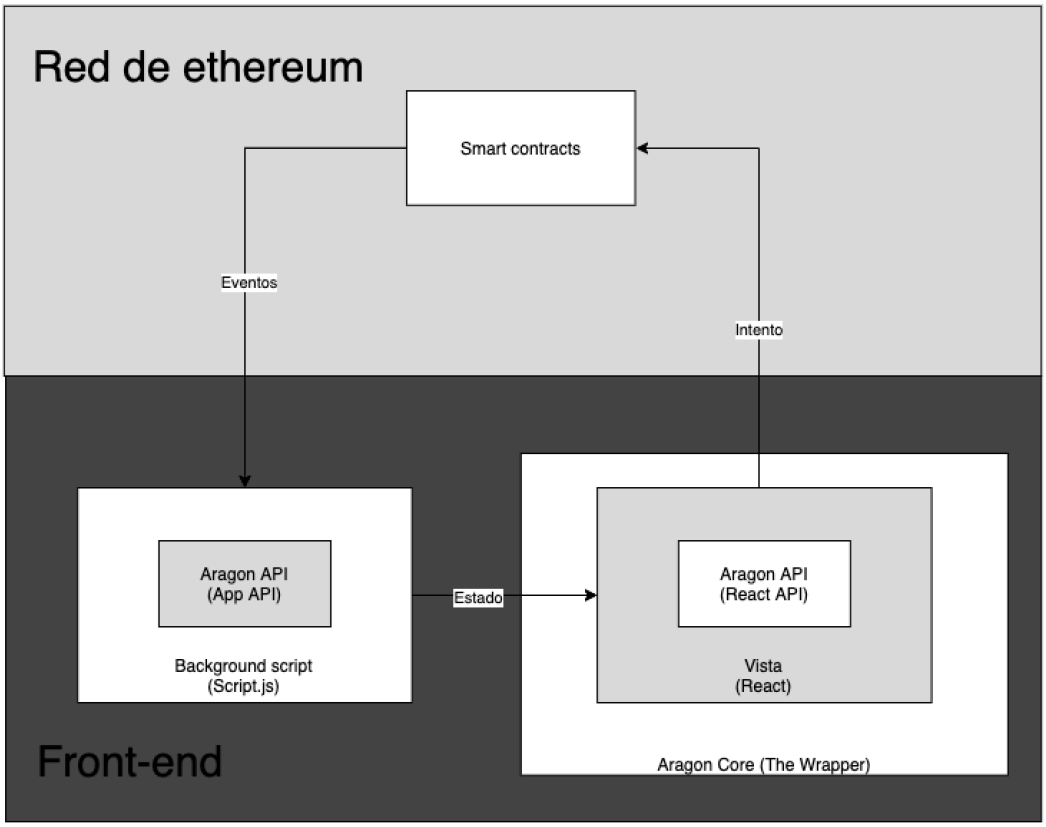

Figura 5.17: Esquema general de la implementación de nuestra aplicación con Aragon [8].

5.3.1. Back-end

El back-end de nuestra aplicación está completamente desarrollado en Solidity. Para obtener la funcionalidad buscada, hemos creado un único contrato llamado *Courses*, que hereda del contrato *AragonApp* para poder integrarse correctamente con el framework. Este contrato alberga todas las estructuras de datos necesarias para llevar a cabo nuestro proyecto, así como las funciones que realizaran las acciones deseadas. En la figura 5.18 se muestra una perspectiva general del contenido de este contrato, la cual será detallada en las siguientes subsecciones.

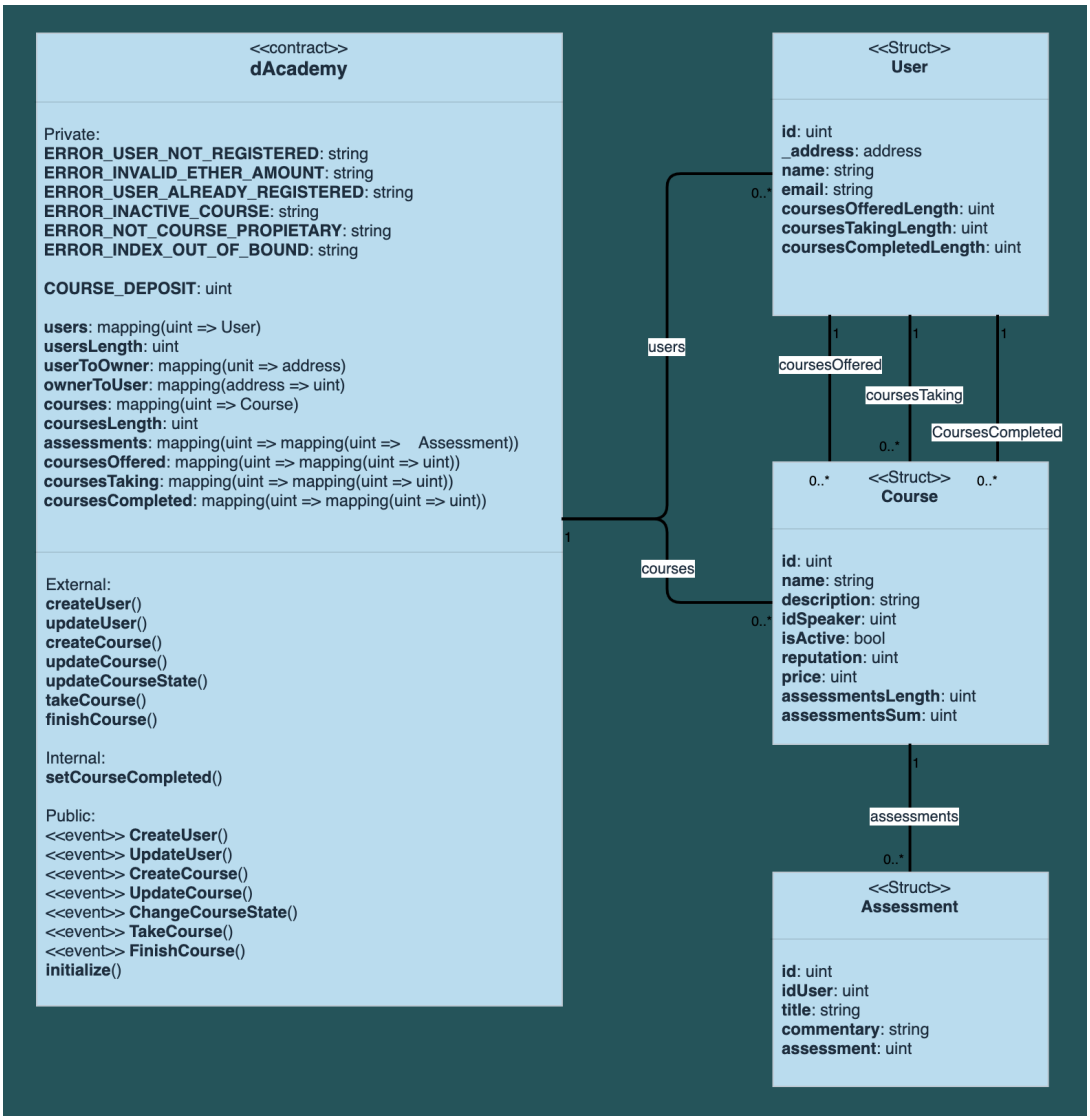

Figura 5.18: Diagrama UML del back-end con Solidity.

Estructuras de datos

Para poder gestionar correctamente nuestro sistema de cursos, hemos declarado tres estructuras:

- User (ver tabla 5.1).
- Course (ver tabla 5.2).
- Assessment (ver tabla 5.3).

Hemos generado las estructuras de datos necesarias para almacenar nuestra información usando diferentes mapas (ver tabla 5.4). Debemos mencionar, que los mapas pertenecientes a las estructuras, se definen fuera de las mismas. Esto se debe a las limitaciones que presenta el lenguaje Solidity a la hora de acceder a los datos desde el front-end. Por tanto estas variables que en un diseño convencional estarían dentro de las estructuras, quedarán fuera debido a las restricciones del lenguaje.

User		
Tipo	*Nombre*	*Descripción*
uint	id	Identificador del usuario.
address	_address	Dirección del usuario .
string	name	Nombre del usuario.
string	email	Email del usuario.
uint	coursesOfferedLength	Cantidad de cursos que el usuario ofrece.
uint	coursesTakingLength	Cantidad de cursos que el usuario está cursando.
uint	coursesCompletedLength	Cantidad de cursos que el usuario ha completado.

Tabla 5.1: Campos del struct User.

Course		
Tipo	*Nombre*	*Descripción*
uint	id	Identificador del curso.
string	name	Nombre del curso.
string	description	Descripción del curso.
uint	idSpeaker	Identificador del creador del curso.
bool	isActive	Indicador de si el curso está disponible.
uint	reputation	Reputación del curso. Se calculará como la media de las valoraciones.
uint	price	Precio del curso en wei.
uint	assessmentsLength	Número de valoraciones del curso.
uint	assessmentsSum	Suma del valor de todas las valoraciones. Esta variable se guarda para facilitar la actualización de la reputación del curso.

Tabla 5.2: Campos del struct Course.

Assessment		
Tipo	*Nombre*	*Descripción*
uint	id	Posición de la valoración en el array de valoraciones del curso.
uint	idUser	Identificador del usuario que puso la valoración.
string	title	Título de la valoración.
string	commentary	Texto de la valoración.
uint	assessment	Valor numérico de la valoración.

Tabla 5.3: Campos del struct Assessment.

Mapas		
Tipo	*Nombre*	*Descripción*
mapping (uint =>User)	users	Relaciona identificador de usuario con usuario.
mapping (uint =>address)	userToOwner	Relaciona identificador de usuario con su dirección.
mapping (address =>uint)	ownerToUser	Relaciona dirección con identificador de usuario.
mapping (uint =>Course)	courses	Relaciona identificador de curso con curso.
mapping(uint =>mapping (uint =>Asessment))	assessments	Relaciona identificador de curso con mapa de identificador de valoración a valoración.
mapping(uint =>mapping (uint =>uint))	coursesOffered	Relaciona identificador de curso con mapa de identificador de curso ofertado a curso.
mapping(uint =>mapping (uint =>uint))	coursesTaking	Relaciona identificador de curso con mapa de identificador de curso cursando a curso.
mapping(uint =>mapping (uint =>uint))	coursesCompleted	Relaciona identificador de curso con mapa de identificador de curso completado a curso.

Tabla 5.4: Estructuras de datos en el smartcontract.

Funciones externas

Toda la lógica de nuestro proyecto quedará implementada en las funciones del smart contract. Estas se definen como externas, para poder llamarlas desde otra aplicación, en este caso del front-end. Los atributos y el propósito de cada función se encuentran detallados en la tabla 5.5.

Funciones externas		
Nombre	*Atributos*	*Descripción*
createUser	name: string email: string	Crea un usuario asignado a la dirección de la cuenta que ejecuta la función con el nombre y el email dados. Solo es posible crear un usuario por cuenta.
updateUser	id: uint name: string email: string	Actualiza el nombre y el email del usuario correspondiente al identificador dado. A cada dirección solo se le permite modificar su propio usuario.
createCourse	name: string description: string price: uint	Crea un nuevo curso asociado a la cuenta que ejecuta la función con los parámetros dados. Se deberá depositar diez veces el precio del curso en ether como fianza para crearlo.
updateCourse	id: uint description: string	Actualiza la descripción del curso asociado al identificador dado. Cada usuario solo podrá modificar los cursos que él mismo ha creado.
updateCourseState	id: uint	Cambia el estado del curso correspondiente al identificador dado. Si está disponible cambia a no disponible y viceversa. Solo podrá ejecutarse por el usuario que haya creado el curso.
takeCourse	id: uint	Permite a un usuario adquirir un curso. Tendrá que abonar su precio en ether, y este será transferido al creador del curso.
finishCourse	takingCourseId: uint title: string commentary: string assessment: uint	Permite a un usuario finalizar un curso y dejar un comentario con un título, cuerpo y una valoración numérica. Además, si aún no se le ha devuelto toda la fianza al creador del curso, este recibirá una décima parte de ésta.

Tabla 5.5: Funciones externas del smart contract.

Eventos

Los eventos juegan un papel fundamental en Solidity y, especialmente, en Aragon. Esto es debido a que son los encargados de la comunicación entre el back-end y el front-end (ver figura 5.17). En nuestro caso, hemos decidido crear un evento por cada una de las funciones externas. Esto es apropiado ya que todas las funciones externas pueden potencialmente realizar cambios que recibe directamente el front-end a través del estado, y para ello necesita la notificación de un evento. Los eventos creados son los siguientes:

- **CreateUser**: Evento para la creación de un usuario.
- **UpdateUser**: Evento para la actualización de un usuario.
- **CreateCourse**: Evento para la creación un curso.

- **UpdateCourse**: Evento para la actualización de la información de un curso.
- **UpdateCourseState**: Evento para la actualización del estado de un curso.
- **TakeCourse**: Evento para la compra de un curso.
- **FinishCourse**: Evento para la finalización de un curso de los que estás cursando y dejar una valoración.

Roles

Los roles son una funcionalidad que Aragon añade al back-end de Solidity. Con ellos, podemos decidir qué tipo de acciones puede realizar cada usuario. Estos tienen un potencial enorme a la hora de diseñar organizaciones escalables que puedan ir evolucionando de forma autónoma. La idea original de nuestro proyecto es que todos los usuarios puedan crear u ofrecer cursos sin restricciones por lo que, a priori, no parece que vayamos a necesitar un uso muy intensivo de esta funcionalidad. Sin embargo, hemos decidido crear un rol para cada función externa y, de esta manera, poder atender a posibles necesidades de implementación futuras en la línea de la restricción de permisos a ciertos usuarios. Los roles definidos han sido los siguientes:

- **CREATEUSER_ROLE**: Permiso para crear un usuario.
- **UPDATEUSER_ROLE**: Permiso para actualizar un usuario.
- **CREATECOURSE_ROLE**: Permiso para crear un curso.
- **UPDATECOURSE_ROLE**: Permiso para actualizar la información de un curso.
- **UPDATECOURSESTATE_ROLE**: Permiso para actualizar el estado de un curso.
- **TAKECOURSE_ROLE**: Permiso para comprar un curso.
- **FINISHCOURSE_ROLE**: Permiso para finalizar un curso de los que estás cursando y dejar una valoración.

5.3.2. Front-end

Para que el usuario pueda interactuar con los datos que la aplicación ofrece, contamos con una estructura de front-end similar a cualquier otra aplicación de React. Se parte desde los archivos *index.html* e *index.js*. En este último, se hace uso del componente *AragonAPI* a través del módulo *@aragon/api-react* (ver figura 5.19) para conectar la interfaz de nuestra aplicación con el *smart contract*. También usamos el componente *App* que será explicado más adelante. Además, contamos con la función *reducer* para procesar el estado inicial.

```
ReactDOM.render(
  <AragonApi reducer={reducer}>
    <App />
  </AragonApi>,
  document.getElementById('root')
)
```

Figura 5.19: Uso de @aragon/api-react.

A continuación, se muestra en la figura 5.20 el diagrama de componentes de la aplicación, donde podemos ver las dependencias entre los componentes.

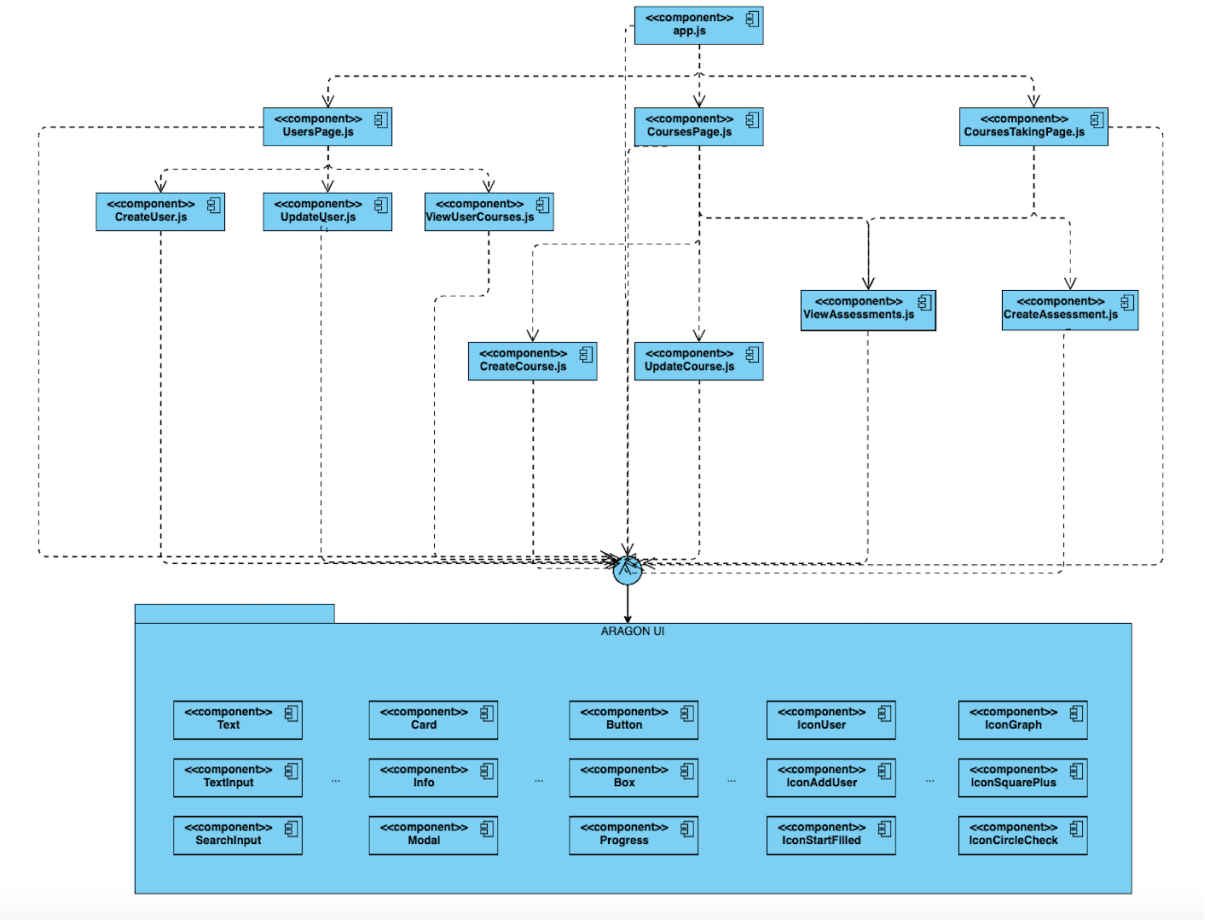

Figura 5.20: Diagrama de componentes para el front-end.

El componente *App* contiene la instancia de *useAragonApi()*, que nos devuelve los datos necesarios para interactuar con el *smart contract*. Con ello, podemos obtener *api* para llamar a los métodos implementados en el *smart contract* y *appState* para obtener el estado actual de la aplicación. También ofrece otras opciones como *connectedAccount*, que nos sirve para saber qué cuenta de Ethereum está conectada en cada momento.

App es la base de la interfaz y cuenta con un componente de pestañas para separar las tres vistas principales: usuarios, cursos y cursos matriculados (ver figura 5.21).

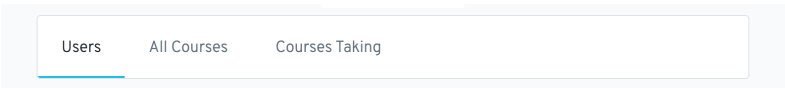

Figura 5.21: Componente *Tabs* para cambiar en las diferentes vistas.

UsersPage.js

En esta vista se muestran todos los usuarios registrados en la aplicación. Para ello, se hace uso del componente *Card* en cual se muestran los datos del usuario (e-mail y *account*), su reputación actual, el número de cursos de ofrece, que ha completado y que está cursando en este momento (ver figura 5.22). Para cada usuario se muestran además, tres botones que permiten realizar las acciones de editar perfil, eliminar perfil y ver los cursos que ofrece ese usuario. Un usuario solo podrá editar o borrar un perfil si le pertenece.

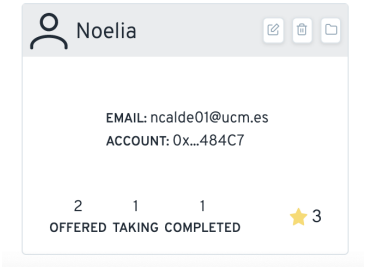

Figura 5.22: Vista de cómo se ve el componente *Card* para un usuario.

Por otro lado, contamos con un componente de búsqueda (*SearchInput*) que nos permite filtrar por nombre de usuario y un botón para acceder al formulario de registro e ingresar un nuevo usuario a la aplicación. Vista completa en la figura 5.23.

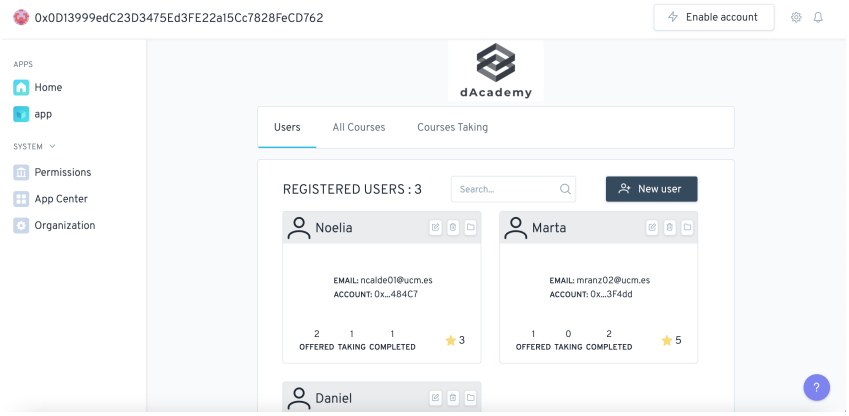

Figura 5.23: Vista principal dónde se muestran los usuarios registrados.

Para crear un nuevo usuario, contamos con el componente *CreateUser*. Este se compone de un *Modal* que contiene entradas de texto en las que el usuario puede escribir los datos solicitados (ver figura 5.24). De la misma manera, se utiliza el componente *UpdateUser* para mostrar el formulario de edición. En este último caso, los campos del formulario se encontrarán rellenos con los datos actuales y el usuario podrá decidir si modificarlos o no. (Ver figura 5.25).

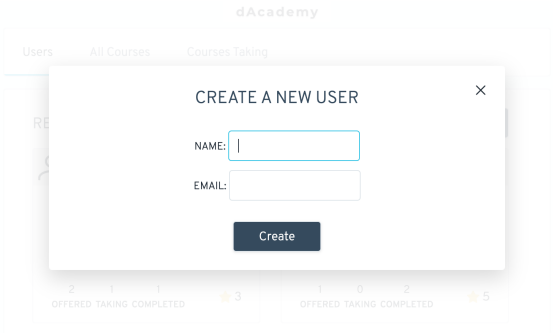

Figura 5.24: Formulario de registro para un nuevo usuario.

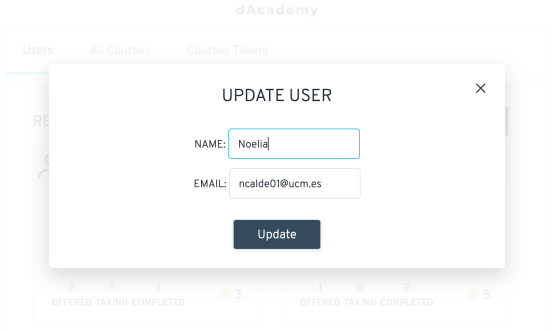

Figura 5.25: Formulario para editar los datos de un usuario.

Si deseamos consultar rápidamente los cursos de un usuario en concreto, entonces podemos acceder a esa vista desde el tercer botón de su tarjeta (figura 5.22). De esta manera, se abrirá un nuevo modal que contendrá exclusivamente los cursos que imparte ese usuario. (Ver figura 5.26).

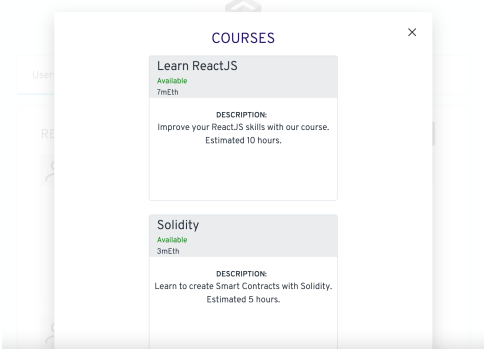

Figura 5.26: Vista de los cursos que imparte un usuario determinado.

CoursesPage.js

Esta otra vista expone todos los cursos que se encuentran agregados en la aplicación. Pueden estar disponibles para que cualquier usuario pueda matricularse, o pueden no estarlo si el usuario que imparte ese curso lo ha deshabilitado temporalmente. De la misma forma que para mostrar usuarios, hacemos uso del componente *Card* para mostrar cada curso. Se presenta la información del curso junto con su disponibilidad, su precio y su valoración (ver figura 5.27). En cada tarjeta, aparecen los botones para editar, comprar, cambiar el estado del curso (habilitado/deshabilitado) o ver valoraciones y comentarios de otros usuarios. Un usuario solo podrá editar o cambiar el estado de un curso si este le pertenece.

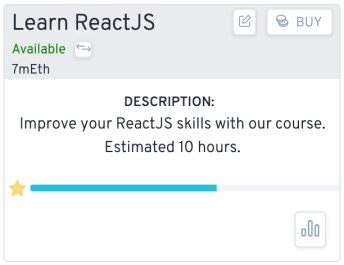

Figura 5.27: Vista de cómo se ve el componente *Card* para un curso.

En la vista contamos, como en la anterior, con una barra de búsqueda con la que podemos filtrar por nombre del curso. Además, tenemos un botón que nos permite acceder al formulario para crear un nuevo curso. Ver vista completa en la figura 5.28.

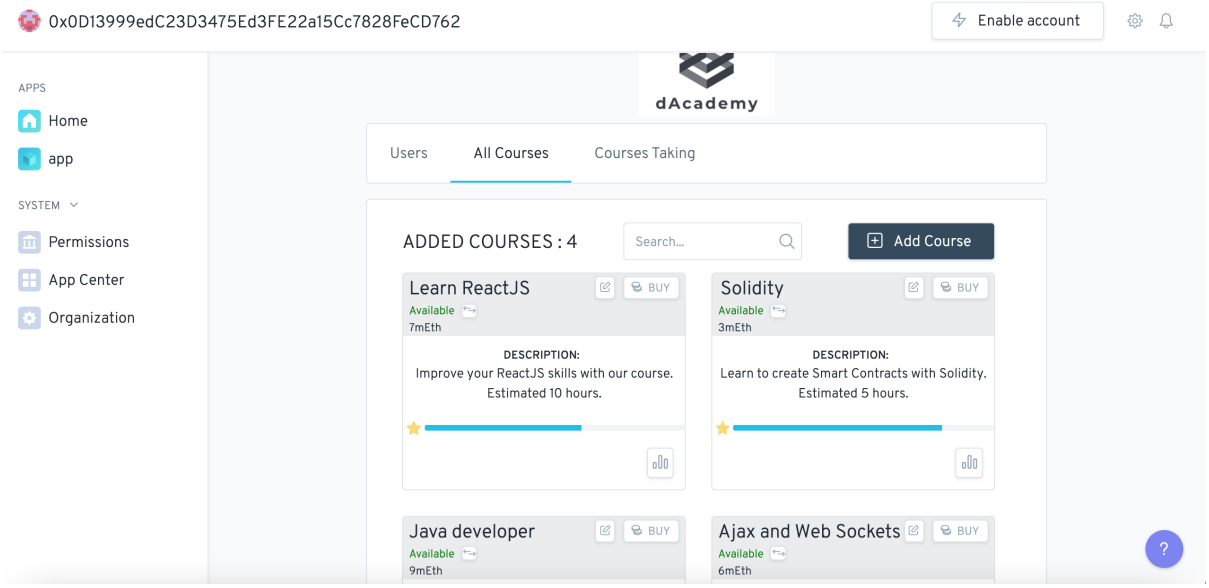

Figura 5.28: Vista principal dónde se muestran los cursos añadidos.

El formulario para crear un nuevo curso, se implementa en el componente *CreateCourse* y como el resto de formularios, se compone de un Modal y varias entradas de texto en las que introducir la información correspondiente. (Ver figura 5.29). Del mismo modo, el componente *UpdateCourse* ofrece el formulario para editar la descripción de un curso. (Ver figura 5.30).

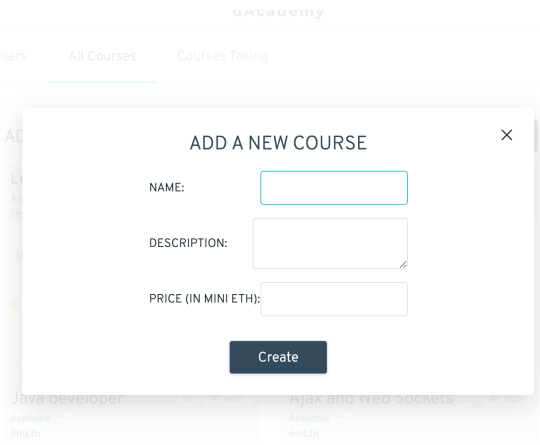

Figura 5.29: Formulario para añadir un nuevo curso.

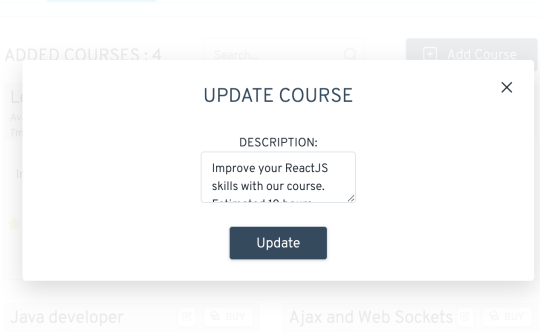

Figura 5.30: Formulario para editar un curso existente.

Por otro lado, disponemos del componente *ViewAssessments* para visualizar los comentarios aportados por otros usuarios a un curso específico. Pulsando en el botón de la esquina derecha inferior, podremos acceder a ellos. (Ver firgura 5.31).

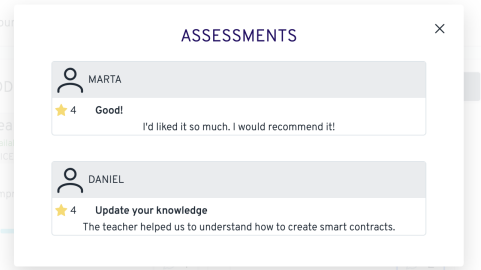

Figura 5.31: Valoraciones de los usuarios para un curso determinado.

CoursesTakingPage.js

Esta vista solo estará disponible si estamos conectados y registrados en la aplicación, puesto que se trata de los cursos en los que estamos matriculados en el momento actual. En caso contrario, nos mostrará un mensaje de error (Ver figuras 5.32 y 5.33).

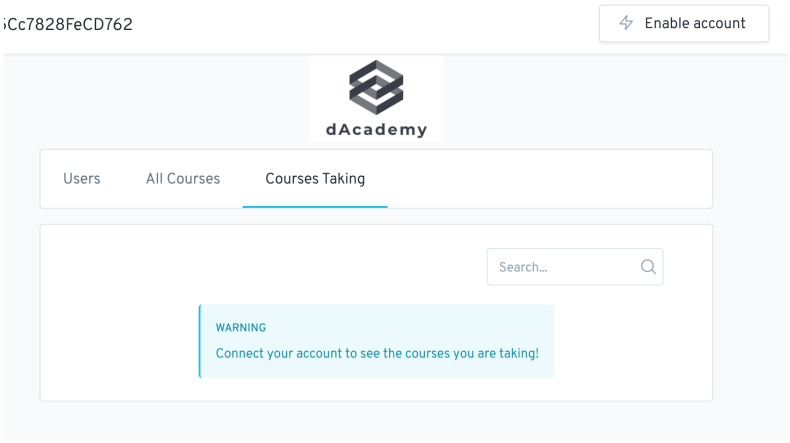

Figura 5.32: Mensaje informativo: No estás conectado en la cuenta de Metamask.

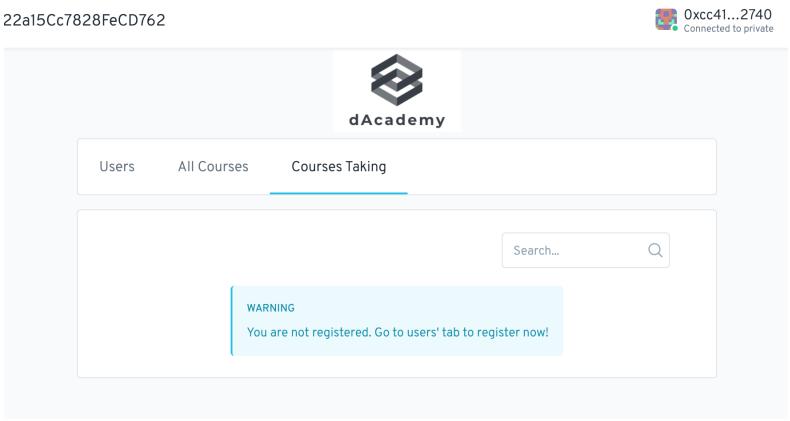

Figura 5.33: Mensaje informativo: No estás registrado en la aplicación.

Si estás correctamente conectado con tu cuenta y tienes un perfil dentro de la aplicación, entonces podrás ver los cursos en los que estás matriculado actualmente. Si no dispones de ningún curso matriculado, se mostrará un mensaje informativo (ver figura 5.34). Los cursos se muestran de la misma forma que la vista anterior, con el componente Card. En este caso, contamos con un botón para finalizar el curso que nos llevará a un formulario para realizar una valoración antes de terminarlo.

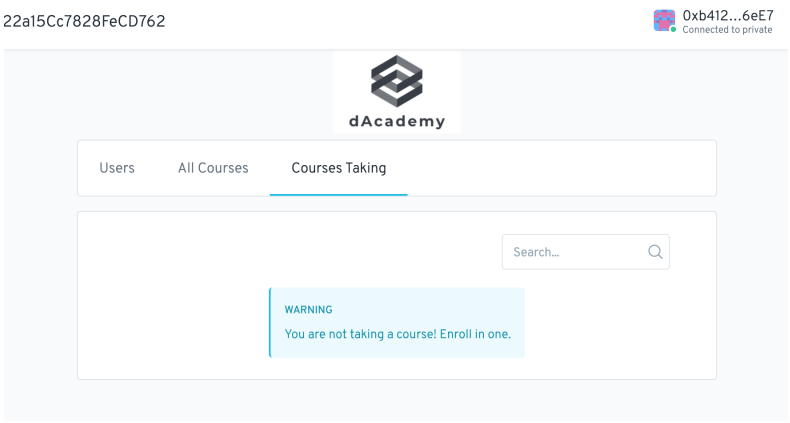

Figura 5.34: Mensaje informativo si no estás matriculado en ningún curso.

El formulario de valoración de un curso, se define mediante el componente CreateAssessment de la misma manera que los anteriores y nos permite valorar un curso para validar su reputación y la del usuario que lo imparte. (Ver figura 5.35). Ver vista completa en la figura 5.36.

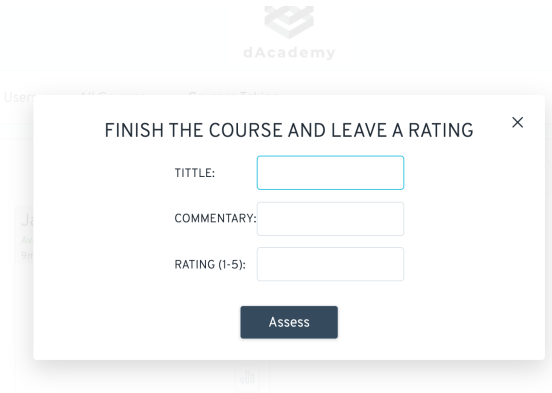

Figura 5.35: Vista del formulario para dar una valoración a un curso.

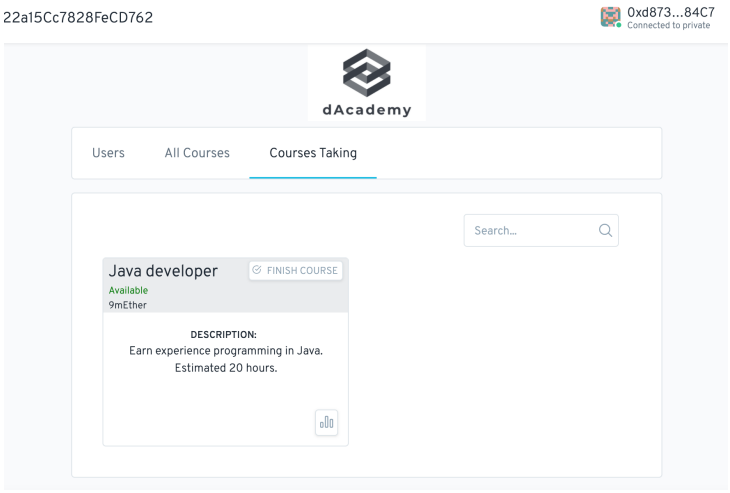

Figura 5.36: Vista general de los cursos matriculados por el usuario conectado.

5.3.3. Problemas encontrados

Debido a que esta tecnología es muy emergente y sus componentes aún son inmaduros, el desarrollo de la aplicación ha ido atravesando diversos problemas que se describen a continuación.

Compatibilidad con los Sistemas Operativos

Uno de los retos a los que nos hemos enfrentado han sido los equipos de trabajo con los que contábamos los miembros del equipo. Al intentar ejecutar la aplicación en Windows (*npm start*) obtuvimos varios errores, entre ellos el que se muestra en la figura 5.37.

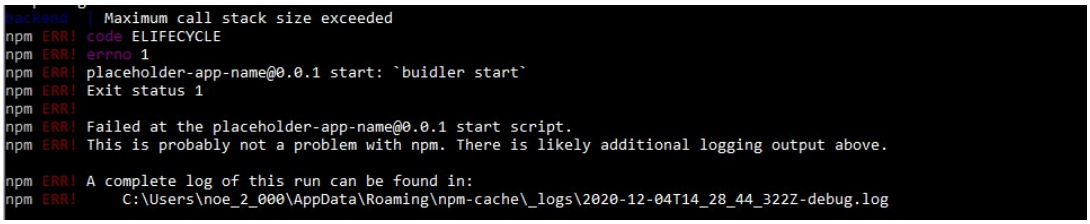

Figura 5.37: Error producido al ejecutar el comando de inicio de la aplicación en un sistema operativo Windows.

El tipo de información que se detallaba en el log no ofrecía pistas que indicaran cómo solucionar el error, únicamente se anunciaba que el fallo se produjo al intentar ejecutar el script de inicio. Tras semanas buscando una solución a este problema, puesto que las medidas encontradas en artículos y foros [60] no servían de ayuda, descubrimos que algunos módulos que eran necesarios para el inicio de la aplicación, no se encontraban disponibles en este sistema operativo. Para solventar el problema, se realizaron pruebas en otros sistemas operativos: MacOS y Linux.

Nonce incorrecto al realizar una transacción en Metamask

Al ejecutar distintas pruebas en una cadena de bloques de prueba teniendo Metamask conectado, es posible que acabe realizando erróneamente el cálculo del nonce de cada transacción, puesto que Metamask almacena una caché con cierta información de la red a la que está conectado. Esta información contiene las transacciones completadas y las utiliza para derivar en siguiente nonce.

Este problema nos lleva a un error en el momento de realizar cualquier transacción, ya que se calcula un nonce erróneo y no te permite ejecutarla (ver figura 5.38).

Para solventar este error, se proponen distintas soluciones:

- Restablecer la cuenta de Metamask. Esto se permite en las opciones de configuración (ver figura 5.39) y hará que el nonce se establezca de nuevo en 0.
- Desinstalar y volver a instalar Metamask.
- Configurar el nonce personalizado llevando la cuenta manualmente (ver figura 5.40).

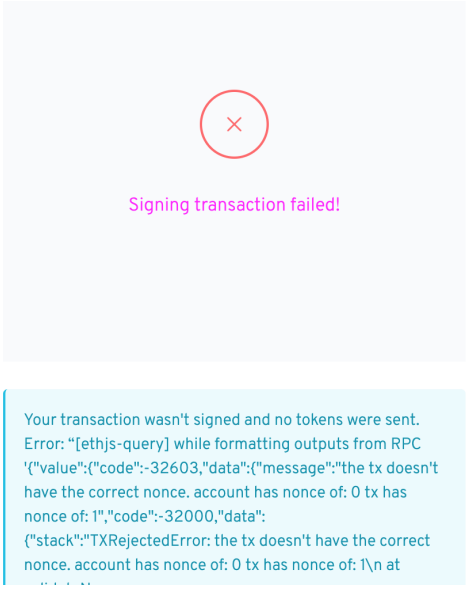

Figura 5.38: Error al realizar la transacción debido a un nonce incorrecto.

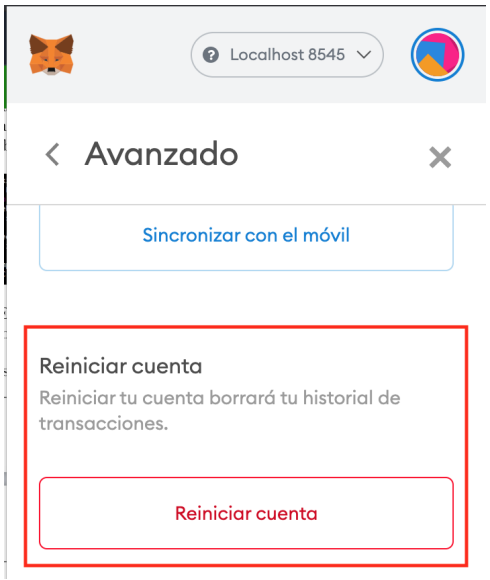

Figura 5.39: Cómo resetear la cuenta de Metamask.

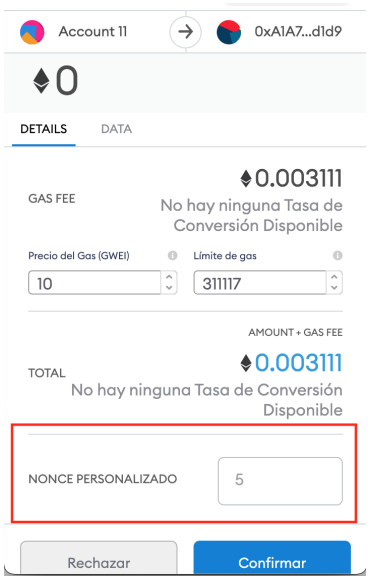

Figura 5.40: Insertar un nonce manualmente al realizar una transacción.

Error de red al iniciar la aplicación

En ocasiones, Aragon mostraba un error de red al iniciar la aplicación como se muestra en la figura 5.41. Esto sucedía justo después de parar la aplicación cuando esta se estaba ejecutando, lo que resultaba un poco desconcertante. Este error se solucionaba reiniciando el ordenador, y seguramente tiene que ver con algún parámetro de la configuración de red en el ordenador que no fuimos capaces de detectar.

```
backend  | Deploying Aragon bases (ENS, DAOFactory, and APM)...
Error BDLR110: Network connection timed-out.
Please check your internet connection and networks config

For more info go to https://buidler.dev/BDLR110 or run Buidler with --show-stack-traces
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! app@0.0.1 start: `buidler start`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the app@0.0.1 start script.
npm ERR! This is probably not a problem with npm. There is likely additional logging output above.

npm ERR! A complete log of this run can be found in:
npm ERR!     /Users/davalve/.npm/_logs/2021-04-22T17_43_56_608Z-debug.log
```

Figura 5.41: Error de red al iniciar la aplicación.

Problemas con git y desarrollo en diferentes ordenadores

Al inicio del desarrollo, tuvimos abundantes problemas a la hora de subir la aplicación a un repositorio de Github y ejecutarla posteriormente en otros ordenadores (ver figura 5.42). Tras las correspondientes investigaciones, llegamos a la conclusión de que debíamos excluir las dependencias y los archivos autogenerados de la aplicación para poder portar esta a otros ordenadores. El problema fue finalmente solucionado borrando de git los archivos mencionados y creando un .gitignore adecuado.

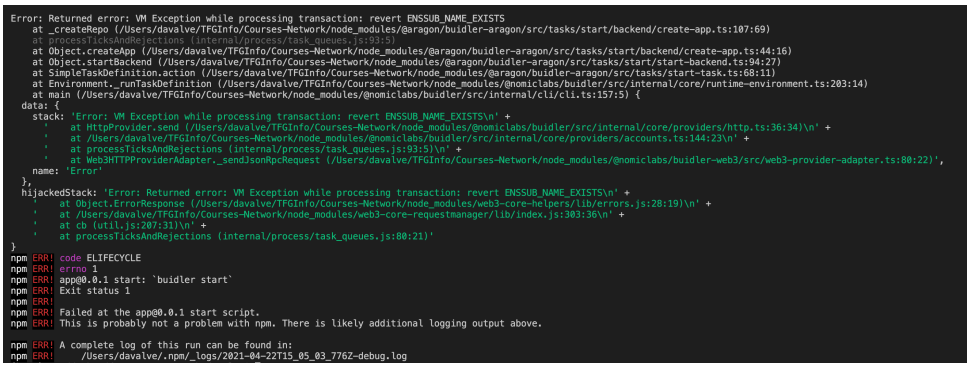

Figura 5.42: Error al ejecutar la aplicación en un ordenador diferente tras descargar de Github.

Documentación y casos prácticos insuficientes

El desarrollo en blockchain es todavía inmaduro y necesita crecer y progresar lo suficiente para que el uso de esta tecnología no sea tan complejo. Cuenta con muy pocas referencias, la documentación a veces es escasa y los ejemplos prácticos, además de ser pocos, son demasiado simples y no ofrecen la ayuda necesaria, o son proyectos que requieren una serie de conocimientos bastante avanzados para su seguimiento, lo que impedía su comprensión dado el nivel de conocimiento del equipo en ese momento.

A pesar de ello, esto no es más que otra razón de peso por la que es tan necesario el desarrollo con esta tecnología y la investigación en este tipo de casos.

5.4. Como descargar y ejecutar el código

Para ejecutar el código de nuestra aplicación tendremos que cumplir lo siguientes pre-requisitos:

- La aplicación debe ser ejecutada en un ordenador con Linux o macOS como sistema operativo. Nuestra aplicación **no es compatible con windows** debido a la incompatibilidad de ciertas librerías de Aragon.
- Tener instaladado Node.js [61].
- Tener instalada la extensión de Metamask para el navegador elegido [23]. Recomendamos el uso de Google Chrome o Firefox. Desaconsejamos el uso de Safari, ya que no soporta la extensión de Metamask. Una vez instalada, es importante seleccionar la red Localhost:8545 como se muestra en la figura 5.43.
- Tener instalado git [62].

Una vez cumplimentados estos pre-requisitos, solo tendremos que abrir una terminal en la carpeta donde queramos descargar la aplicación y ejecutar los siguientes comandos:

```
git clone https://github.com/P2PModels/Courses-Network.git
cd Courses-Network
npm install
npm start
```

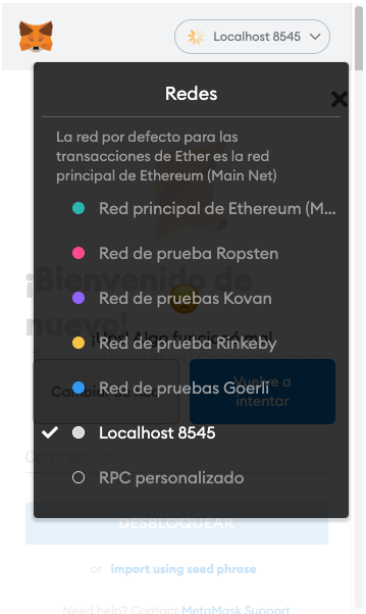

Figura 5.43: Selección de red en MetaMask.

Estos comandos se encargarán de descargar el código de la aplicación de nuestro repositorio de Github [63], instalar las librerías necesarias para su correcto funcionamiento y ejecutar la aplicación. Tras esto, se debería abrir una ventana en el navegador con la aplicación. Si no es el caso, podemos abrirla manualmente utilizando la dirección que se indica en la última linea de la información mostrada en la consola.

Una vez abierta la aplicación, tendremos que añadir cuentas de prueba a MetaMask para poder interactuar con ella. Para ello, buscamos las direcciones de prueba generadas por Aragon en los mensajes que aparecieron en consola tras ejecutar *npm start* (ver figura 5.44).

```
main    | Starting Aragon app development...
main    | App name: app
main    | App id: 0x740e14bd8efcf3fa51e40db0c746650c88690730bd094e78c8981ba6d61981b1
main    | Accounts mnemonic "explain tackle mirror kit van hammer degree position ginger unfair soup bonus"
main    | Account 0 private key 0xa8a54b2d8197bc0b19bb8a084031be71835580a01e70a45a13babd16c9bc1563
main    |           public key 0xb4124cEB3451635DAcedd11767f004d8a28c6eE7
main    | Account 1 private key 0xce8e3bda3b44269c147747a373646393b1504bfcbb73fc9564f5d753d8116608
main    |           public key 0x8401Eb5ff34cc943f096A32EF3d5113FEbE8D4Eb
main    | Account 2 private key 0x8716d2701596f51aa39d061a685d5ae5ec946eb2c7adb059d29024b5bb3b02c8
main    |           public key 0x306469457266CBBe7c0505e8Aad358622235e768
main    | Account 3 private key 0x62d7bb725787d84b059eb4950f6eea060d898183250ca3ea673a36b8e113018f
main    |           public key 0xd873F6DC68e3057e4B7da74c6b304d0eF0B484C7
main    | Account 4 private key 0x705df2ae707e25fa37ca84461ac6eb83eb4921b653e98fdc594b60bea1bb4e52
main    |           public key 0xDcC5dD922fb1D0fd0c450a0636a8cE827521f0eD
main    | Account 5 private key 0x6b12b45143fc6c7721d0ffbb9811905e773868376501fd1f46c64bf34ae29991
main    |           public key 0x27E9727FD9b8CdDdd0854F56712AD9DF647FaB74
main    | Account 6 private key 0x33f3f34569f997abb165d6967895d963a2b15ec609efcec844e65b60ee8340c7
main    |           public key 0x9766D2e7FFde358AD0A40BB87c4B88D9FAC3F4dd
main    | Account 7 private key 0x5a013cc48f0a3196b0986fc7a7a9dd320ac75e89e33302a7ff4ea6b9dc4f7b00
main    |           public key 0xBd7055AB500cD1b0b0B14c82BdBe83ADCc2e8D06
main    | Account 8 private key 0x418cc0b07bfef998f577384b185b97ad544204b5be43ac9b3abf16db2012ab5c
main    |           public key 0xe8898A4E589457D979Da4d1BDc35eC2aaf5a3f8E
main    | Account 9 private key 0x698eece6f9915b08b4d1a63958dc4f3996ee5a8d685b29d17c28beab912a77cd
main    |           public key 0xED6A91b1CFaae9882875614170CbC989fc5EfBF0
main    |
```

Figura 5.44: Cuentas de prueba de Aragon.

Una vez tengamos las claves privadas (*private key* en inglés) de las cuentas que queremos utilizar, tenemos que agregarlas a MetaMask. Para hacerlo, hemos de seguir los siguientes pasos:

1 - Pulsamos en el icono del zorro que nos aparece arriba a la derecha de la ventana

del explorador tras instalar la extensión (ver figura 5.45).

2 - Iniciamos sesión con una cuenta MetaMask si aún no lo hemos hecho.

3 - Pulsamos en el botón circular de arriba a la derecha y seleccionamos la opción *Importar cuenta*.

4 - Pegamos la clave privada de la cuenta que deseamos añadir (ver figura 5.46).

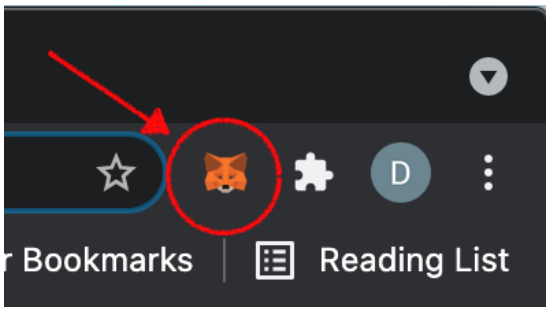

Figura 5.45: Icono de Metamask en el explorador (imagen tomada con Google Chrome).

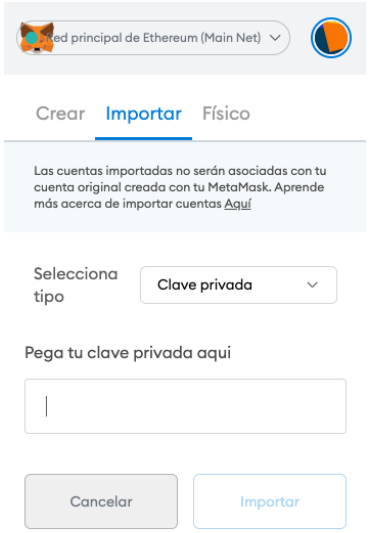

Figura 5.46: Cómo añadir una nueva cuenta a MetaMask usando una clave privada.

Tras seguir estos pasos, ya tendremos la aplicación disponible en nuestros equipos y una cuenta de prueba en MetaMask lista para interactuar con ella y realizar transacciones.

Capítulo 6

Evaluación con usuarios

6.1. Objetivo de la evaluación

Se desea evaluar la facilidad de uso de la DAO dAcademy y de las tecnologías necesarias para su utilización, como sería por ejemplo, el uso del monedero virtual Metamask. Con esta evaluación buscamos asegurarnos de que la interfaz y la funcionalidad desarrollada es adecuada para los usuarios, independientemente del nivel de sus conocimientos en blockchain.

Para obtener una primera apreciación de la usabilidad comprobaremos los siguientes puntos:

- La efectividad de la aplicación para la adquisición de aptitudes y habilidades mediante cursos con los que contribuyen los miembros de la plataforma.
- Comprobar que la interfaz resulta intuitiva para los usuarios que hacen uso de ella.
- Asegurarnos de que los usuarios pueden realizar todas las acciones correctamente y que las funcionalidades están distribuidas de forma coherente.
- Comprobar si el usuario se siente cómodo utilizando este tipo de aplicaciones.
- Obtener una realimentación por parte de los usuarios para realizar futuras mejoras.

6.2. Descripción

La evaluación se ha llevado a cabo en un periodo aproximado de 5 días, en los cuales diversos usuarios han interactuado con nuestra DAO descubriendo su funcionamiento. Debido a la situación actual de Covid, el número de estos usuarios se ha visto reducido a 13, siendo en su mayoría amigos y familiares.

Para obtener resultados precisos y cercanos a la realidad, los usuarios evaluados se encuentran en distintos rangos de edad (de 22 a 28 años) y competencia en esta tecnología, aunque cuentan con el requisito mínimo de tener capacidad de uso del equipo informático sobre el que se ejecutará la aplicación.

Las evaluaciones cuentan con un moderador que, en la medida de lo posible, se encargará de explicar el propósito de la DAO dAcademy y el motivo de la evaluación, también dictará las tareas a realizar que se enumeran más abajo. Además, tomará notas acerca de cómo se encuentra el usuario mientras realiza las acciones y el tiempo que toma para ejecutar cada una de ellas.

Tras completar la lista de tareas a realizar, se pedirá a los usuarios evaluados que completen un formulario para valorar la usabilidad (ver sección 6.3). Una vez finalizado el formulario, se permitirá que el usuario realice aportaciones para mejorar la funcionalidad e interfaz de la aplicación.

En el desarrollo de está evaluación, se pide a diversos usuarios que ejecuten una serie de tareas, para probar el funcionamiento de la aplicación y comprobar que pueden efectuarlas sin dificultades.

1. Conecta tu cuenta de Ethereum con Metamask.
2. Regístrate en la aplicación añadiendo un nuevo usuario con tus datos.
3. Edita algún campo del usuario que has creado.
4. Localiza los cursos que ofrece algún usuario en concreto.
5. Busca algún curso sobre Solidity.
6. Matricúlate en el curso que quieras mirando las valoraciones y comentarios de otros usuarios.
7. Crea un curso de lo que tu quieras para ofrecer a otros usuarios.
8. Deshabilita el curso que acabas de crear para que el resto de usuarios sepan que no se encuentra disponible actualmente.
9. Accede a la pestaña de los cursos que estás estudiando para finalizar alguno y darle una valoración.

6.3. Cuestionario

Tras completar cada una de las tareas, se pide a los usuarios que rellenen un cuestionario de usabilidad [64] para valorar la experiencia de usuario durante el proceso de prueba de la aplicación.

Las respuestas indican en su mayoría, el grado de acuerdo o desacuerdo con la pregunta.

A continuación, se enumeran las cuestiones definidas.

Preguntas generales

1. Encuentro que la aplicación es muy difícil de usar y necesito ayuda de una persona para usarla.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

2. Encuentro esta aplicación innecesariamente compleja.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

3. Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

4. Imagino que la mayoría de la gente aprendería a usar esta aplicación de forma muy rápida.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

5. Me siento cómodo al usar esta aplicación.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

6. Las funciones de esta aplicación están bien integradas.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

7. Creo que la aplicación es muy inconsistente.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

8. En el caso de que realice algún curso, me gustaría utilizar esta aplicación.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

Metamask

1. He podido conectar fácilmente mi cuenta de Metamask a la aplicación.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

2. Me ha resultado fácil acceder a mi cuenta de Metamask.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

3. La interfaz de Metamask hace que seguir los pasos para realizar una acción me resulte sencillo.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

Blockchain

1. Conozco otras aplicaciones de blockchain.

Si

No

2. Conozco y comprendo las ventajas que tienen las tecnologías blockchain descentralizadas frente a los clásicos sistemas centralizados.

Si

No

3. Me siento cómodo usando blockchain teniendo en cuenta mi nivel de conocimiento sobre la tecnología.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

4. Creo que la tecnología blockchain es confiable para realizar transacciones de dinero.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

5. La volatilidad del valor de las criptomonedas me hace desconfiar de ellas.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

6. Creo que me sentiría cómodo usando blockchain si su uso estuviera más extendido.

Totalmente en desacuerdo 1 2 3 4 5 Totalmente de acuerdo

6.3.1. Resultados

A continuación, se detallarán los resultados obtenidos tras completar cada tarea y se destacarán los puntos o elementos en los que se han encontrado dificultades o no se han considerado adecuados.

- **Conecta tu cuenta de Ethereum con Metamask.** En general, los usuarios no han tenido dificultades para realizar esta acción. Además, puesto que el uso de Metamask no era el objetivo principal de esta evaluación, se facilitó a los usuarios los pasos que debían seguir para añadir una nueva cuenta.
- **Regístrate en la aplicación añadiendo un nuevo usuario con tus datos.** Tarea completada sin inconvenientes.
- **Edita algún campo del usuario que has creado.** El icono para editar ha resultado bastante intuitivo, por lo que los usuarios han realizado esta acción fácilmente.
- **Localiza los cursos que ofrece algún usuario en concreto.** El icono utilizado para esta acción no era lo suficiente claro. Los usuarios intentaron realizar esta acción desde de la parte inferior donde se muestra el número de los cursos que ofrece, está tomando o ha terminado un usuario.
- **Busca algún curso sobre Solidity.** Tarea completada sin inconvenientes.
- **Matricúlate en el curso que quieras, mirando las valoraciones y comentarios de los otros usuarios.** En general, los usuarios no asocian el icono asignado a las valoraciones para acceder a las mismas. A pesar de ello, la acción se realiza correctamente.
- **Crea un curso de lo que tu quieras para ofrecer a otros usuarios.** Tarea completada sin inconvenientes.
- **Deshabilita el curso que acabas de crear para que el resto de usuarios sepan que no se encuentra disponible actualmente.** La tarea se realiza correctamente sin complicaciones aunque los usuarios se sentirían más cómodos si pudieran realizar esta acción desde la pestaña de usuarios.
- **Accede a la pestaña de los cursos que estás estudiando para finalizar alguno y darle una valoración.** Tarea completada sin inconvenientes.

A continuación, se muestran los resultados obtenidos tras la cumplimentación del cuestionario por parte de los usuarios.

Preguntas generales

1. Encuentro que la aplicación es muy difícil de usar y necesito ayuda de una persona para usarla.

Un 46.2 % de los usuarios están totalmente en desacuerdo con esta afirmación, asegurando que el uso de la aplicación es bastante simple y sencillo.

2. Encuentro esta aplicación innecesariamente compleja.

El 77 % ha considerado la aplicación sencilla.

3. Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación.

El 69.2 % de los usuarios no necesitó aprender demasiadas cosas para conseguir un buen manejo de la aplicación.

4. Imagino que la mayoría de la gente aprendería a usar esta aplicación de forma muy rápida.

El 46.2 % está totalmente de acuerdo en que el aprendizaje de uso es bastante rápido.

5. Me siento cómodo al usar esta aplicación.

El 76.9 % de los usuarios afirma sentirse cómodo usando la aplicación. El porcentaje restante no lo tiene claro.

6. Las funciones de esta aplicación están bien integradas.

El 92.3 % cree que las funcionalidades presentes en la aplicación se encuentran bien integradas.

7. Creo que la aplicación es muy inconsistente.

Un 77 % de los usuarios asegura estar en desacuerdo con esta afirmación.

8. En el caso de que realice algún curso, me gustaría utilizar esta aplicación.

El 69.3 % de los usuarios usaría dAcademy para completar algún curso. El 30.8 % restante, no lo tiene claro pero no descarta la opción.

Metamask

1. He podido conectar fácilmente mi cuenta de Metamask a la aplicación.

El 69.2 % no ha presentado dificultades para conectarse a su cuenta.

2. Me ha resultado fácil acceder a mi cuenta de Metamask.

El 76.9 % asegura que ha accedido fácilmente.

3. La interfaz de Metamask hace que seguir los pasos para realizar una acción me resulte sencillo.

El 84.6 % de los usuarios valora la interfaz de Metamask como fácil e intuitiva.

Blockchain

1. Conozco otras aplicaciones de blockchain.

Un 46.2 % de usuarios asegura conoce otras aplicaciones implementadas en blockchain.

2. Conozco y comprendo las ventajas que tienen las tecnologías blockchain descentralizadas frente a los clásicos sistemas centralizados.

A pesar de que menos de la mitad de los usuarios evaluados conoce aplicaciones en blockchain, un 76.9 % afirma conocer las ventajas que tienen estas tecnologías.

3. Me siento cómodo usando blockchain teniendo en cuenta mi nivel de conocimiento sobre la tecnología.

El 69.3 % de los usuarios evaluados dice sentirse cómodo haciendo uso de la blockchain.

4. Creo que la tecnología blockchain es confiable para realizar transacciones de dinero.

El 77 % confía en la blockchain para realizar transacciones.

5. La volatilidad del valor de las criptomonedas me hace desconfiar de ellas.

El 61.5 % de los usuarios no tiene clara su posición y han ofrecido una valoración media a esta pregunta. Esto puede deberse a que no cuentan con la información suficiente para determinar su postura.

6. Creo que me sentiría cómodo usando blockchain si su uso estuviera más extendido.

Un 92.3 % de los usuarios asegura que se sentiría más cómodo y que haría uso de la blockchain si estuviera más normalizada.

En términos generales, los usuarios no han presentado dificultades para realizar un uso completo de la aplicación. La mayoría está de acuerdo en que se presenta una interfaz sencilla que permite aprender rápidamente su funcionamiento.

Por otro lado, los resultados indican el poco conocimiento que se tiene actualmente acerca de esta tecnología puesto que sigue siendo muy innovadora y para muchos es algo totalmente nuevo y desconocido. Aun así, los usuarios evaluados se muestran dispuestos a entender y dar uso de la blockchain si, tal vez en un futuro, se diese a conocer a un público más generalista.

A continuación, se muestra una tabla en la que se enumeran las recomendaciones aportadas por los usuarios tras la evaluación.

Nº	**Recomendación**	**Prioridad**
1	Añadir un mensaje informativo para señalar que se está pidiendo una valoración antes de finalizar el curso.	MEDIA
2	El icono para acceder a los comentarios de los usuarios, no resulta intuitivo. Se recomienda cambiarlo a otro más reconocible.	ALTA
3	Permitir el acceso a los cursos ofrecidos, matriculados y completados desde cada usuario. Sustituir el botón actual para acceder a los cursos ofrecidos, por otro en la parte inferior y añadir, además, acceso a los cursos matriculados y completados.	ALTA
4	Para obtener una mayor comodidad, permitir editar un curso desde una pestaña del usuario que lo imparte. Modificar la estructura de la interfaz para permitir estas acciones.	BAJA
5	Deshabilitar los botones cuando una acción no esté permitida. (Por ejemplo, editar un perfil que no te pertenezca.).	MEDIA
6	Añadir el número de usuarios que ha valorado un curso.	BAJA

Tabla 6.1: Informe de hallazgos y recomendaciones tras la evaluación.

6.3.2. Mejoras realizadas tras la evaluación

Con motivo de la evaluación y las recomendaciones aportadas por los usuarios evaluados, se han implementado las siguientes mejoras:

1. Añadir un mensaje informativo en el formulario de valoración. Figura 6.1 Para ofrecer una mejor ayuda visual al usuario, se añade un mensaje informativo que indica el propósito del formulario.

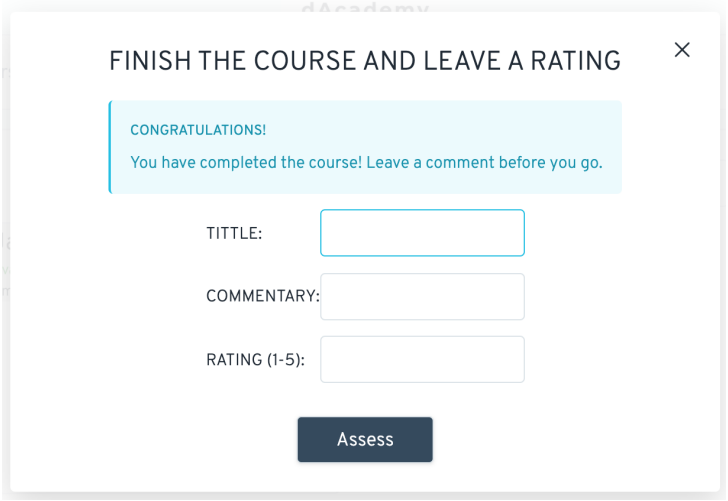

Figura 6.1: Nuevo mensaje informativo al finalizar un curso.

2. Cambio del icono para acceder a las valoraciones de un curso. Figura 6.2 Debido a la confusión producida por el icono anterior, se ha optado por el siguiente icono que resulta más reconocible de cara al público.

Además, se incluye la recomendación **6. Añadir el número de usuarios que han valorado un curso**.

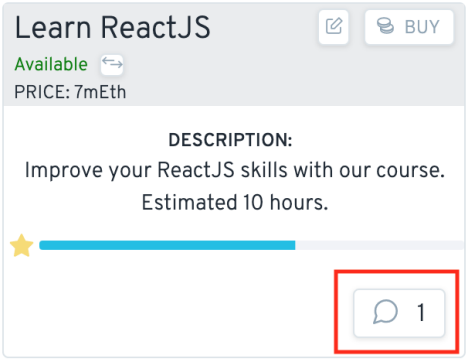

Figura 6.2: Nuevo icono de acceso a las valoraciones más reconocible para los usuarios.

3. Permitir el acceso a los cursos ofrecidos, matriculados y completados desde cada usuario.

Se ha eliminado el icono anterior (ver figura 5.22 el icono de carpeta) por nuevos botones que permiten el acceso a cada tipo de cursos desde la parte inferior, como vemos en la figura 6.3.

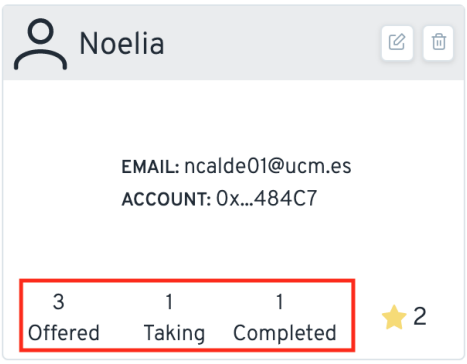

Figura 6.3: Botones para acceder a los cursos ofrecidos, matriculados y completados de un usuario.

Pulsando sobre *Offered* accedemos a los cursos ofrecidos de ese usuario (figura 6.4), sobre *Taking* accedemos a los cursos en los que el usuario está matriculado (figura 6.5) y sobre *Completed* a los cursos que ha completado (figura 6.6).

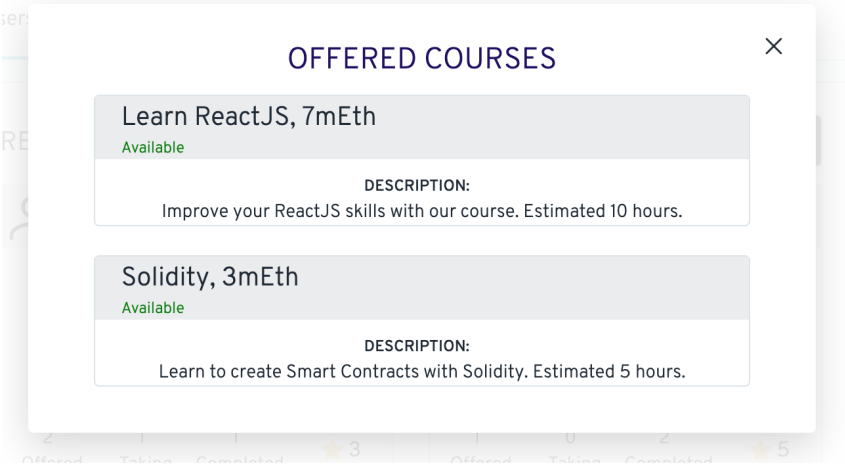

Figura 6.4: Nuevo formato de los cursos ofrecidos de un usuario.

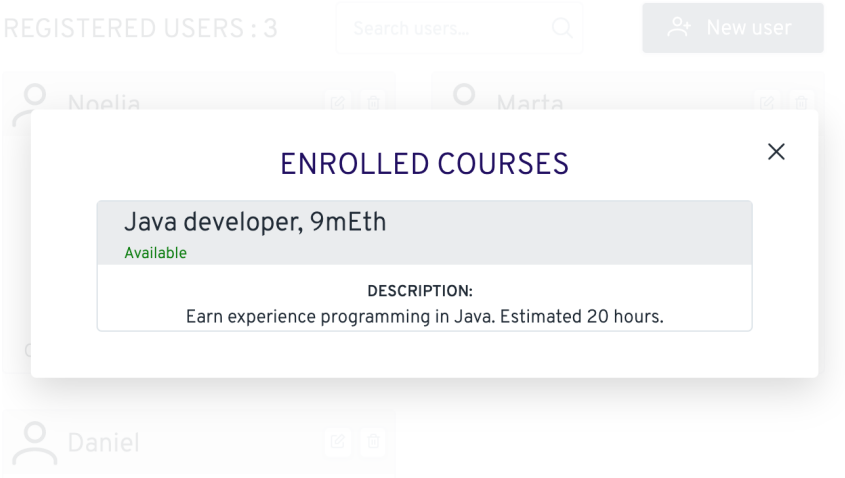

Figura 6.5: Nuevo formato de los cursos matriculados de un usuario..

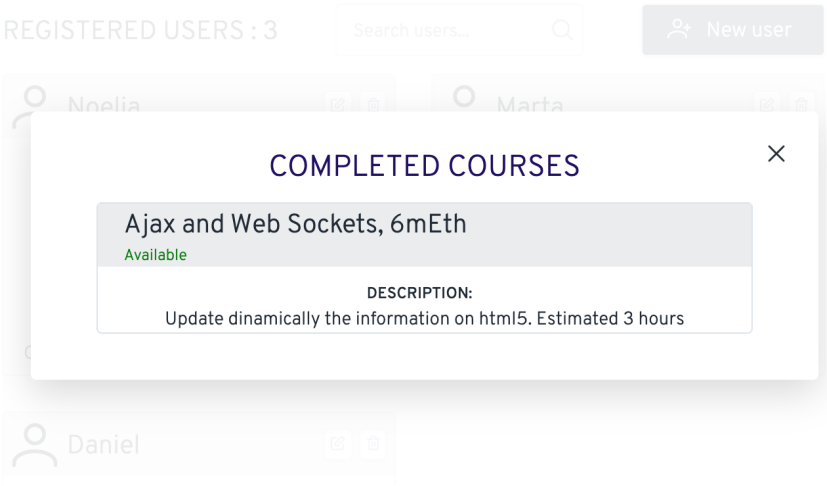

Figura 6.6: Nuevo formato de los cursos completados de un usuario.

5. Deshabilitar/habilitar los botones.

Para mejorar la experiencia de usuario y que éste sepa en cada momento lo que puede hacer o no, se han deshabilitado los botones cuyas funcionalidades no se encuentren disponibles debido a que no se posean los permisos necesarios para ejecutarlas.

Los botones de editar o eliminar un usuario se encontrarán deshabilitados si el usuario no se corresponde con la cuenta conectada. De la misma manera, se deshabilitan los botones de editar y cambiar el estado de un curso. Por el contrario, se deshabilita el botón de compra de aquellos cursos que sí pertenezcan al usuario conectado, puesto que un usuario no puede comprar su propio curso. (Ver ejemplo en la figura 6.7)

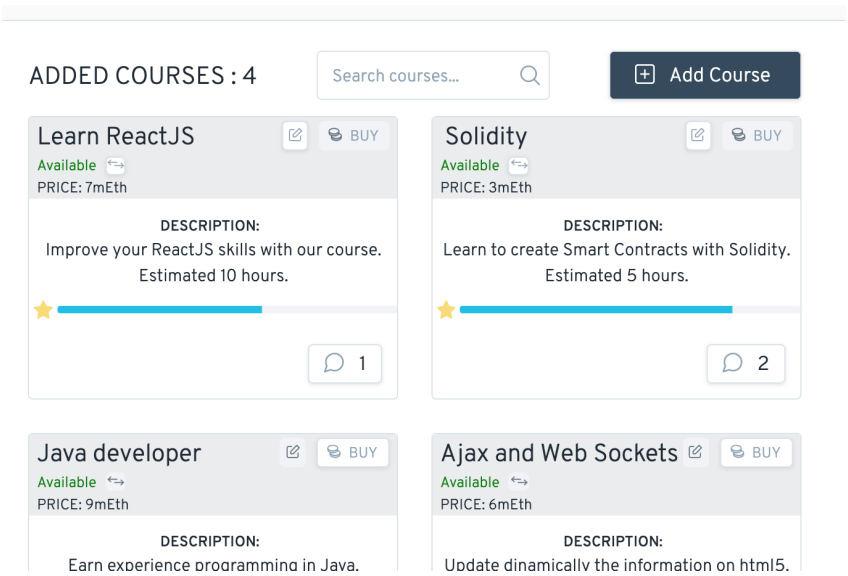

Figura 6.7: Cómo se ven los botones deshabilitados o habilitados.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

Blockchain es una tecnología que no deja indiferente a nadie. Durante los últimos años, la magnitud de las comunidades de desarrolladores y de las inversiones en el campo, han aumentado exponencialmente. Estamos viviendo una revolución que, aunque por ahora parezca solo tecnológica, muy probablemente tenga efectos económicos, políticos y sociales muy profundos en nuestra sociedad, de la misma forma que sucedió en los inicios de Internet.

Sin embargo, como hemos podido experimentar, aún existen muchos retos que atravesar. Las tecnologías son todavía poco maduras e inestables y son utilizadas por gente con altos conocimientos tanto técnicos, como del ecosistema blockchain en general. Probablemente, en los siguientes años presenciaremos avances tecnológicos en las plataformas y una mayor permeabilidad de estas tecnologías en la sociedad. Este progreso dependerá principalmente de cómo de eficaz se vuelva la tecnología, a la hora de solucionar los problemas mayores que se generen por la existencia de sistemas centralizados en grupos sociales concretos.

Respecto al desarrollo de DAOs, es todavía un mundo totalmente reservado para los desarrolladores. Durante el desarrollo de este proyecto, hemos comprobado como estudiantes de informática de último año, han tenido que enfrentarse a abundantes problemas, incluso tras pasar por una etapa de aprendizaje bastante extensa. Por lo tanto, no es de extrañar que los desarrolladores de blockchain sean escasos, puesto que se trata de una tecnología que requiere mucha especialización. Además, la variedad de plataformas para el desarrollo de DAOs sigue siendo muy limitada como hemos visto a lo largo de este trabajo, que finalmente solo se ha podido implementar nuestro proyecto en uno de los tres frameworks inicialmente propuestos.

A pesar de no haber conseguido el objetivo principal, implementar una DAO en los distintos frameworks para comprobar las diferencias que presentaban entre ellos, hemos construido sobre Aragon una aplicación descentralizada que esperamos mejorar en el futuro sirviendo de apoyo a otros desarrolladores, fomentando la creación de DAOs y del uso de este y otros frameworks. Para ello, esperamos también que sirva para mejorar las plataformas de desarrollo, y de esta manera, impulsen aún más la creación de nuevas

DAO de manera sencilla y con esto el crecimiento de la tecnología blockchain.

Gracias a las pruebas realizadas con la aplicación por parte de distintos usuarios, hemos podido simular cómo sería la interacción de una persona con un bajo nivel de conocimiento en esta tecnología, con una aplicación blockchain. La conclusión obtenida determina que las aplicaciones blockchain podrían ser usadas sin ningún problema por el público mayoritario, siempre y cuando se facilite una interfaz lo suficientemente intuitiva. Sin embargo, existen obstáculos que pueden hacer que el usuario medio se decante por seguir usando las aplicaciones tradicionales, por ejemplo, la necesidad de interactuar con un monedero digital o el tener que pagar comisiones por cada transacción. Por lo tanto, para que el uso de las aplicaciones blockchain se generalice, ha de existir un incentivo lo suficientemente fuerte como para motivar al usuario a afrontar las dificultades iniciales de blockchain.

En general, en cuanto a nuestro trabajo, esperamos que algún día pueda llegar a ser una aplicación más dentro del ecosistema de Aragon, y que genere una comunidad de usuarios que se beneficien mutuamente de sus servicios. Si no se da el caso, al menos servirá como guía y documentación a personas que quieran emprender nuevos proyectos en este novedoso y apasionante mundo.

7.2. Trabajo futuro

El hecho de trabajar sobre una plataforma que está en constante desarrollo provoca que, las aplicaciones que se implementen sobre ella, deban estar preparadas para actualizarse y adaptarse constantemente a las nuevas versiones. Esto, a la larga, es la única forma de obtener aplicaciones funcionales y robustas que se adapten a las demandas de los usuarios, pero en el corto plazo puede dar lugar a cambios repentinos en el funcionamiento de la aplicación que desagraden al usuario. Así, abrazando la filosofía del continuo cambio, presentaremos en esta sección los flujos de desarrollo que pensamos debería seguir nuestra aplicación en el futuro.

En función de los resultados obtenidos al finalizar el proyecto y teniendo en cuenta los problemas encontrados durante el desarrollo del mismo se proponen un conjunto de mejoras que podrían implementarse a partir de este trabajo.

Nuevas funcionalidades

- Implementar un campus donde alumnos puedan acceder al material de sus cursos.
- Implementar una vista donde los profesores puedan añadir y modificar el contenido de sus cursos y que este se encuentre accesible a sus alumnos. Como comentamos en el punto anterior, por ejemplo, a través de un campus virtual.
- Implementar una vista donde cada usuario tenga un mejor acceso a los cursos que él ha creado.
- Permitir subir imágenes para el perfil de los usuarios.
- Implementación de un *chat* que permita la comunicación entre estudiantes y profesores.
- Añadir un apartado que muestre el *ranking* con los usuarios mejor valorados.

- Añadir una sección de *tendencias* que, de la misma forma que el punto anterior, muestre los cursos que más éxito están teniendo en la actualidad.
- Uso de un token propio para el uso dentro de la aplicación. Actualmente el token utilizado para las transacciones es Ether.
- Mejorar la lógica para reducir los tiempos de espera.Simplificar las funciones y mejorar la eficiencia de las ejecuciones para conseguir una respuesta más rápida de la aplicación.

Conclusions and future work

Conclusions

Over the last few years, the size of the developer communities and investments in the field have increased exponentially. We are living a revolution that, although for now it seems only technological, will most likely have very profound economic, political and social effects on our society, in the same way as it happened in the early days of the Internet.

However, as we have experienced, there are still many challenges to overcome. The technologies are still immature and unstable and are used by people with high technical knowledge as well as knowledge of the blockchain ecosystem in general. Probably, in the following years we will witness technological advances in the platforms and a greater permeability of these technologies in society. This progress will depend mainly on how effective the technology becomes in solving the major problems generated by the existence of centralized systems in specific social groups.

As far as the development of DAOs is concerned, it is still a world entirely reserved for developers. During the development of this project, we have seen how final-year computer science students have had to deal with a lot of problems, even after going through a fairly extensive learning phase. It is therefore not surprising that blockchain developers are in short supply, as it is a technology that requires a lot of specialisation. Moreover, the variety of platforms for the development of DAOs is still very limited, as we have seen throughout this work, which has finally only been possible to implement our project in one of the three frameworks initially proposed.

Despite not having achieved the main objective, to implement a DAO in the different frameworks to test the differences between them, we have built on Aragon a decentralised application that we hope to improve in the future by supporting other developers, encouraging the creation of DAOs and the use of this and other frameworks. In doing so, we also hope that it will serve to improve the development platforms, and in this way, further boost the creation of new DAOs in a simple way and with this, the growth of blockchain technology.

Thanks to the tests carried out with the application by different users, we have been able to simulate how a person with a low level of knowledge of this technology would interact with a blockchain application. The conclusion reached is that blockchain applications could be used without any problem by the majority of the public, as long as a sufficiently intuitive interface is provided. However, there are obstacles that may make the average user choose to continue using traditional applications, for example, the need to interact

with a digital wallet or having to pay fees for each transaction. Therefore, for the use of blockchain applications to become widespread, there must be a strong enough incentive to motivate the user to deal with the initial difficulties of blockchain.

In general, as far as our work is concerned, we hope that one day it can become one more application within the Aragon ecosystem, and that it will generate a community of users who will mutually benefit from its services. If this is not the case, at least it will serve as a guide and documentation for people who want to undertake new projects in this new and exciting world.

7.3. Future work

The fact of working on a platform that is constantly being developed means that the applications that are implemented on it must be prepared to be constantly updated and adapted to new versions. This, in the long run, is the only way to obtain functional and robust applications that adapt to the demands of users, but in the short term it can lead to sudden changes in the functioning of the application that displease the user. So, embracing the philosophy of continuous change, we will present in this section the development flows that we think our application should follow in the future.

Based on the results obtained at the end of the project and taking into account the problems encountered during the development of the project, we propose a set of improvements that could be implemented from this work.

New functionalities

- Implement a campus where students can access their course material.
- Implement a view where teachers can add and modify the content of their courses and make it accessible to their students. As mentioned in the previous point, for example, through a virtual campus.
- Implement a view where each user has better access to the courses he has created.
- Allow uploading images for user profiles.
- Implementation of a chat to allow communication between students and teachers.
- Add a section that shows the ranking with the best rated users.
- Add a trends section which, in the same way as the previous point, shows the courses that are currently most successful.
- Use of a proprietary token for use within the application. Currently the token used for transactions is Ether.
- Improve logic to reduce waiting times. Simplify functions and improve execution efficiency to achieve faster application response.

Capítulo 8

Contribuciones al proyecto

8.1. Contribuciones de Daniel Valverde Menasalvas

Durante la primera etapa del proyecto (ver tabla 3.1), yo me dediqué a familiarizarme con las tecnologías necesarias en el futuro desarrollo. Empecé con Solidity, que era un lenguaje en el que no tenía nada de experiencia. Los ejemplos y tutoriales de la página oficial[65] me fueron de gran ayuda. A continuación, como nunca había hecho desarrollo web, tuve que aprender HTML, CSS y JavaScript a un nivel básico que me permitiera utilizar el framework de React. Finalmente, realicé el tutorial de la página oficial de React[51] y leí la documentación haciendo hincapié en las funcionalidades de los *hooks*, ya que permiten un desarrollo con mayor fluidez y más posibilidades. Es esta primera etapa del proyecto, fue fundamental el apoyo de nuestra directora de TFG, Maria Cruz Valiente, que redactó unos pequeños tutoriales de desarrollo con Solidity y nos resolvió las dudas que nos iban surgiendo.

Tras la etapa de inicio, nos aventuramos con nuestros primeros desarrollos. En el proyecto de la lotería, yo creé un smartcontract y mi compañera otro. Esto significó duplicar el trabajo, pero nos sirvió para adquirir soltura con Solidy. En el front-end de la aplicación de lotería no hice grandes aportaciones ya que era un desarrollo sencillo y mi compañera tenía más experiencia que yo con la tecnología.

Para nuestro segundo desarrollo, ya utilizamos el framework de Aragon. El smartcontract fue desarrollado conjuntamente tras algunas discusiones a cerca de los casos de uso que queríamos cubrir. Respecto al front-end, yo me encargué de refinar ciertos aspectos de la interfaz como conseguir mostrar una tabla de forma dinámica en React o presentar opciones posibles a elegir en un desplegable.

Previo a la sesión de brainstorming, realizamos un estudio del estado del arte de las DAOs en aquel momento. Yo me centré en el estudio de las aplicaciones orientadas a las finanzas descentralizadas (DeFi). Aunque nuestro proyecto finalmente no estuvo directamente relacionado con el mundo DeFi, esto me sirvió para entender el potencial de las aplicaciones descentralizadas y para poder aportar más ideas en el brainstorming.

Tras el brainstorming, comenzamos con el desarrollo de la aplicación. A continuación, listo mis contribuciones a la aplicación dAcademy:

- Integración del proyecto con git. Búsqueda de solución de problemas en diferentes

ordenadores mediante la exclusión de archivos generados del proyecto y la creación de un fichero *.gitignore* adecuado.

- Back-end
  - Creación de códigos de errores.
  - Diseño del sistema de creación de cursos, fianzas y pagos.
  - Diseño del sistema de valoraciones (desarrollo conjunto).
  - Modificación de la función crear curso para añadir fianza.
  - Creación de la función comprar curso.
  - Creación de la función para finalizar y valorar curso.
- Front-end
  - Integración del sistema de pagos con el front-end mediante AragonAPI.
  - Conversión de las unidades monetarias para que fueran manejables por el usuario.
  - Creación de barra para la representación de la reputación de cada curso.
  - Creación de página donde se muestran los cursos en proceso del usuario actual.
  - Estructuración de los modales de crear usuario, editar usuario, crear curso, editar curso y crear validación en diferentes componentes de React y diferentes archivos.
  - Estructuración de las páginas de usuario, cursos y cursos cursando en diferentes componentes de React y diferentes archivos.

Finalmente, dedicamos alrededor de un mes a la escritura de la memoria. Los capítulos que realicé o en los que participé de forma activa son los siguientes:

- **1. Introducción**: Escritura conjunta.
- **2. Estado del arte**: Secciones 2.1. Blockchain, Bitcoin, 2.2. Ethereum y 2.5. Blockchain en educación.
- **3. Metodología y tecnología utilizada**: Subsecciones 3.1.1. Etapas de trabajo, 3.1.4. Método para la obtención de ideas: Brainstorming, 3.2.4. Etherscan, 3.2.5. Node.js, 3.2.6. Front-end con React y plantilla de Aragon.
- **4. Primeros desarrollos**: Sección 4.2. Gestor de tareas.
- **5. dAcademy: Una red descentralizada de cursos certificados**: Sección 5.1. Nuestra idea (escritura conjunta), 5.3. Implementación (introducción), subsecciones 5.2.1. Casos de uso (escritura conjunta), 5.3.1. Back-end, 5.3.3. Problemas encontrados (escritura conjunta), 5.4. Como descargar y ejecutar el código.
- **7. Conclusiones y trabajo futuro**: Escritura conjunta.

8.2. Contribuciones de Noelia Calderón Mateo

En los primeros meses del curso, comenzamos a investigar sobre la tecnología blockchain. Yo en particular, realicé varias lecturas de diversos artículos acerca de blockchain, Bitcoin, Ethereum y DAOs, proporcionados por el tutor. La idea era tener claros estos conceptos, para más tarde poder profundizar en ellos durante el desarrollo del TFG.

Para familiarizarme con el lenguaje Solidity, necesario para la implementación de los smarts contracts, realicé varios ejercicios apoyándome del canal de YouTube de Nicolás Palacios, "Tutoriales de Solidity"[66] y siguiendo estos ejemplos, redacté un documento con algunos apuntes sobre el lenguaje, que compartí con mis compañeros. Por otro lado, realicé un pequeño tutorial descrito por la codirectora del TFG, María Cruz Valiente, en el cuál se implementaba un smart contract de un contador muy simple.

Para el primer acercamiento a los smart contracts realizamos una pequeña lotería (ver sección 4.1). Aplicando los conocimientos adquiridos con los tutoriales y por supuesto, la documentación oficial de Solidity, realicé una versión del contrato para la lotería con la funcionalidad completa, que más tarde completamos con las aportaciones de mis compañeros. Además, para esta fase realicé una interfaz sencilla que sirviera de comunicación para poder explotar las funcionalidades presentes en el contrato. Para ello, primero completé otro tutorial, propuesto por María Cruz, que describía cómo generar una aplicación con React. Esta aplicación se desarrolló haciendo uso de React, HTML y CSS. No había usado React previamente, pero al ser bastante similar a JavaScript, el cual sí había utilizado en diversas asignaturas, no resultó complejo para el desarrollo de esta pequeña aplicación.

El siguiente proyecto se trataba de una aplicación simple para la gestión de tareas haciendo uso del framework de Aragon (ver sección 4.2). Para esta fase comencé realizando un nuevo tutorial descrito por María Cruz, que explicaba los pasos para usar una plantilla de Aragon a través de la cuál poder realizar la DAO. Comencé por escribir parte de las funcionalidades del smart contract y tras debatir con mis compañeros cómo queríamos implementarlo fuimos completándolo conjuntamente. Para el front-end, desarrollé una interfaz básica con botones y entradas para llamar a los métodos del contrato, dejando la funcionalidad disponible para su interacción.

Para el proyecto dAcademy (ver sección 5), realizamos conjuntamente un brainstorming a partir del cual obtuvimos la propuesta para el desarrollo de la DAO. A continuación, listaré mi aportación a cada parte del proyecto.

Desarrollo del smart contract:

- Aunque la estructura general se definió conjuntamente, me encargué de la escritura del código de la misma.
- Funcionalidades:
    - Crear usuario.
    - Editar usuario.
    - Borrar usuario.
    - Crear curso.

    - Editar curso.
    - Modificar el estado de un curso.
    - Crear valoraciones (Funcionalidad modificada posteriormente por mi compañero).
    - Insertar valoración a un usuario.

Desarrollo del front-end con React:

- Generé la base de la aplicación, gracias a la plantilla de Aragon.
- Codifiqué el archivo script.js que se encarga de obtener los datos del back-end para poder utilizarlos en el front-end.
- Funcionalidades:
    - Crear usuario.
    - Editar usuario.
    - Borrar usuario.
    - Mostrar usuarios.
    - Crear curso.
    - Editar cursos.
    - Modificar el estado del curso.
    - Mostrar cursos.
    - Crear valoración.
    - Mostrar valoraciones.
    - Añadir reputación a un usuario.
- Componentes de la interfaz:
    - Navegación mediante pestañas (tabs).
    - Diseño del componente en el cual se muestra cada uno de los usuarios y de los cursos.
    - Diseño de los botones e iconos de la interfaz.
    - Mensajes de información.
    - Formulario para crear usuario.
    - Formulario para editar usuario.
    - Formulario para crear curso.
    - Formulario para editar curso.
    - Formulario para crear valoración.
    - Ventana modal para mostrar las valoraciones.

- Logo de la aplicación.
- Habilitar o deshabilitar los botones dependiendo de su disponibilidad para el usuario.

Además, tras la evaluación con usuarios he llevado a cabo las tareas de mejora de las funcionalidades presentes en la aplicación (ver sección 6.3.2), así como la subsanación de errores que surgieron en el proceso de prueba.

Por último, paso a enumerar los puntos en los cuales he participado en la redacción de la memoria:

- **1. Introducción**: Escritura conjunta.
- **2. Estado del arte**: Secciones 2.3. DAOs y 2.4. Frameworks.
- **3. Metodología y tecnología utilizada**: Subsecciones 3.1.2. Software libre, 3.1.3. Método de desarrollo, 3.2.1. Backend con Solidity y Remix, 3.2.2. Cartera virtual Metamask, 3.2.3. Red de pruebas: Rinkeby y Localhost 8545.
- **4. Primeros desarrollos**: Secciones 4.1. Lotería y 4.3. Experimentando con los diferentes frameworks.
- **5. dAcademy: Una red descentralizada de cursos certificados**: Sección 5.1. Nuestra idea (escritura conjunta), 5.2.1. Casos de uso (escritura conjunta), 5.2.2. Requisitos no funcionales, 5.2.3. Diseño de la interfaz de usuario: mockups, 5.3.2. Front-end, 5.3.3. Problemas encontrados (escritura conjunta).
- **6. Evaluación con usuarios**.
- **7. Conclusiones y trabajo futuro**: Escritura conjunta.

Bibliografía

[1] "Investing, criptomonedas - cotización en el mercado." https://es.investing.com/crypto/. Fecha de acceso: 2021-05-06.

[2] "Trazabilidad blockchain." https://www.neocheck.es/neoblock/. Fecha de acceso: 2021-05-23.

[3] "Principales exchanges spot de criptomonedas." https://coinmarketcap.com/es/rankings/exchanges/. Fecha de acceso: 2021-05-23.

[4] "Mapping the most important ethereum forks." https://www.visualcapitalist.com/mapping-major-ethereum-forks/. Fecha de acceso: 2021-05-23.

[5] "Introduction to aragon." https://hack.aragon.org/docs/getting-started. Fecha de acceso: 2021-05-23.

[6] "Metodología scrum (metodología ágil)." https://www.diegocalvo.es/metodologia-scrum-metodologia-agil/. Fecha de acceso: 2021-05-23.

[7] "What is the difference between synchronous and asynchronous data transfer." https://pediaa.com/what-is-the-difference-between-synchronous-and-asynchronous-data-transfer/. Fecha de acceso: 2021-05-23.

[8] P. J. Colombo, "Committee manager app: una aplicación para la gestión de comités de organizaciones autónomas descentralizadas," 2019. (Trabajo de Fin de Grado). Facultad de Informática, Universidad Complutense de Madrid, Grado en ingeniería del software.

[9] "Qué es la economía colaborativa." https://es.eserp.com/articulos/que-es-la-economia-colaborativa/.

[10] "Airbnb." https://www.airbnb.es/.

[11] "Uber." https://www.uber.com/es/es-es/.

[12] "Ulule." https://es.ulule.com/.

[13] "Vibbo." https://www.vibbo.com/.

[14] V. Buterin, "Ethereum whitepaper." https://ethereum.org/en/whitepaper/.

[15] M. Swan, *Blockchain. Blueprint for a new economy.* O'Reilly, 2015.

[16] "Bitcoin: A peer-to-peer electronic cash system." https://bitcoin.org/bitcoin.pdf. Datos de publicación no disponibles. Fecha de acceso: 2021-05-06.

[17] "Blockchain explorer." https://www.blockchain.com/explorer. Fecha de acceso: 2021-05-15.

[18] D. Rozas, A. Tenorio-Fornés, S. Díaz-Molina, and S. Hassan, "When ostrom meets blockchain: Exploring the potentials of blockchain for commons governance," *SAGE Open*, vol. 11, no. 1, p. 21582440211002526, 2021.

[19] A. Tenorio-Fornés, V. Jacynycz, D. Llop-Vila, A. Sánchez-Ruiz, and S. Hassan, "Towards a decentralized process for scientific publication and peer review using blockchain and ipfs," *Hawaii International Conference on System Sciences*, 2019.

[20] P. Tasatanattakool and C. Techapanupreeda, "Blockchain: Challenges and applications," in *2018 International Conference on Information Networking (ICOIN)*, pp. 473–475, 2018.

[21] D. Rozas, A. Tenorio-Fornés, and S. Hassan, "Analysis of the potentials of blockchain for the governance of global digital commons," *Frontiers in Blockchain*, vol. 4, p. 15, 2021.

[22] P. D. Filippi and S. Hassan, "Blockchain technology as a regulatory technology: From code is law to law is code," 2018.

[23] "Metamask." https://metamask.io/. Fecha de acceso: 2021-05-19.

[24] "Elige tu monedero - bitcoin." https://bitcoin.org/es/elige-tu-monedero. Fecha de acceso: 2021-05-15.

[25] A. M. Antonopoulos and G. Wood, *Mastering Ethereum*. O'Reilly Media, Inc., 2018.

[26] V. Buterin, "A next-generation smart contract and decentralized application platform." https://ethereum.org/en/whitepaper/. Publicado originalmente en un repositorio de Github. Fecha de acceso: 2021-05-07.

[27] V. Buterin, "A prehistory of the ethereum protocol." https://vitalik.ca/general/2017/09/14/prehistory.html. Fecha de acceso: 2021-05-07.

[28] "Dao (decentralized autonomous organization)." https://masterethereum.com/dao-decentralized-autonomous-organization-master-blockchain-online/.

[29] S. Hassan and P. D. Filippi, "Decentralized autonomous organization. internet policy review, 10(2)." https://doi.org/10.14763/2021.2.1556, 2021.

[30] S. Wang, W. Ding, J. Li, Y. Yuan, L. Ouyang, and F.-Y. Wang, "Decentralized autonomous organizations: Concept, model, and applications," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 870–878, 2019.

[31] Q. DuPont, *Experiments in Algorithmic Governance: A history and ethnography of "The DAO, .ª failed Decentralized Autonomous Organization.* 2017.

[32] "Moloch dao." https://www.molochdao.com/. Fecha de acceso: 2021-05-08.

[33] "Proyectos financiados en molochdao." https://www.molochdao.com/projects.

[34] "Makerdao." https://makerdao.com/es/. Fecha de acceso: 2021-05-08.

[35] A. Fischer and M.-C. Valiente, "Blockchain governance. internet policy review, 10(2)." https://doi.org/10.14763/2021.2.1554, 2021.

[36] "Decentraland." https://decentraland.org/. Fecha de acceso: 2021-05-08.

[37] "Aragon." https://aragon.org/. Fecha de acceso: 2021-05-08.

[38] "Daostack." https://daostack.io/. Fecha de acceso: 2021-05-08.

[39] "Colony." https://colony.io/. Fecha de acceso: 2021-05-08.

[40] "Aragon developer portal." https://hack.aragon.org/docs/getting-started. Fecha de acceso: 2021-05-08.

[41] G. Chen, B. Xu, M. Lu, and N.-S. Chen, "Exploring blockchain technology and its potential applications for education," *Smart Learning Environments*, vol. 5, no. 1, p. 1, 2018.

[42] "Blockcerts." https://www.blockcerts.org/. Fecha de acceso: 2021-06-12.

[43] "Appii." https://appii.io/. Fecha de acceso: 2021-06-12.

[44] "Sony global education." https://www.sonyged.com/. Fecha de acceso: 2021-06-12.

[45] "Odem." https://odem.cloud/. Fecha de acceso: 2021-06-12.

[46] "Blockchain education network." https://www.blockchainedu.org/. Fecha de acceso: 2021-06-12.

[47] "Bitdegree." https://www.bitdegree.org/. Fecha de acceso: 2021-06-12.

[48] "Lluvia de ideas." https://es.wikipedia.org/wiki/Lluvia_de_ideas. Fecha de acceso: 2021-05-15.

[49] "Documentación de solidity." https://solidity-es.readthedocs.io/es/latest/index.html.

[50] "Ethereum (eth) blockchain explorer." https://etherscan.io/. Fecha de acceso: 2021-05-08.

[51] "React - una biblioteca de javascript para construir interfaces de usuario." https://es.reactjs.org/. Fecha de acceso: 2021-05-08.

[52] "Aragon buidler boilerplate." https://github.com/aragon/aragon-react-boilerplate. Fecha de acceso: 2021-05-08.

[53] N. C. Mateo and D. V. Menasalvas, "Aplicación descentralizada de lotería." https://github.com/Noeliac/lottery. Fecha de acceso: 2021-05-15.

[54] "Ethereum ide - remix." http://remix.ethereum.org/. Fecha de acceso: 2021-05-08.

[55] "React boilerplate." https://github.com/react-boilerplate/react-boilerplate. Fecha de acceso: 2021-05-08.

[56] N. C. Mateo and D. V. Menasalvas, "Gestor de tareas descentralizado." https://github.com/P2PModels/task-manager-app. Fecha de acceso: 2021-05-10.

[57] "Types - solidity 0.5.3 documentation." https://docs.soliditylang.org/en/v0.5.3/types.html#. Fecha de acceso: 2021-05-10.

[58] "Alchemy, dapp para crear daos en daostack." https://v1.alchemy.do/daos/create. Fecha de acceso: 2021-05-08.

[59] Y. El Faqir, J. Arroyo, and S. Hassan, "An overview of decentralized autonomous organizations on the blockchain," in *Proceedings of the 16th International Symposium on Open Collaboration*, OpenSym 2020, (New York, NY, USA), Association for Computing Machinery, 2020.

[60] "'errno 1' code answer." https://www.codegrepper.com/code-examples/javascript/errno+1. Fecha de acceso: 2021-05-08.

[61] "Descarga | node.js." https://nodejs.org/es/download/. Fecha de acceso: 2021-05-19.

[62] "Git." https://git-scm.com/. Fecha de acceso: 2021-05-23.

[63] "P2pmodels/courses-network." https://github.com/P2PModels/Courses-Network. Fecha de acceso: 2021-05-08.

[64] "Cuestionario de usabilidad para dacademy." https://docs.google.com/forms/d/e/1FAIpQLSeS5OqE-6SyHQXJXoQiQ_tRdzEZHq5JkzZF87zkGk9Zi4ZtbA/viewform. Fecha de acceso: 2021-05-23.

[65] "Solidity by example." https://docs.soliditylang.org/en/v0.8.4/solidity-by-example.html. Fecha de acceso: 2021-05-19.

[66] N. Palacios, "Tutoriales solidity - español - contratos inteligentes, icos y apps descentralizadas para principiantes." https://www.youtube.com/watch?v=zd5WOHF1kFM&list=PLVR6_kyVYQd7z0CeV9xcy-gf6jKrO6cTP. Fecha de acceso: 2021-05-23.

[67] "Blockchain solutions: The way to transform your business processes." https://dashbouquet.com/blog/blockchain/blockchain-solutions-the-way-to-transform-your-business-processes. Fecha de acceso: 2021-05-23.

Noelia Calderón Mateo y Daniel Valverde Menasalvas
2021
Ult. actualización 14 de junio de 2021
LaTeX lic. LPPL & powered by TEFLON CC-ZERO