
Finanzas Descentralizadas: Plataforma DeFi integrada para la inversión en fondos de criptodivisas e intercambios de tokens

Por:

**Ismail Azizi González
Gonzalo Martínez Berzal
Muad Rohaibani Alkadri**



**UNIVERSIDAD
COMPLUTENSE
MADRID**

**Grado en Ingeniería de Computadores
Grado en Ingeniería Informática
Grado en Ingeniería de Software**

Director:

Jesús Correas Fernández

Colaborador:

Valentín García

Madrid, 2021-2022

RESUMEN

En los últimos años la tecnología ha avanzado rápidamente, y con la llegada del Blockchain, eliminando los intermediarios financieros, y las criptodivisas, como principal apoyo del Blockchain, el concepto de las inversiones ha cambiado por completo. Ha sido tal la revolución que han traído consigo estas nuevas tecnologías, que ha dejado atrás las inversiones financieras basadas en acciones dando paso a las inversiones en criptomonedas, con un valor basado en la ley de la oferta y la demanda, pero con una volatilidad superior.

CryptoFund es un fondo de inversiones indexado de criptodivisas desarrollado con Solidity y tecnología de Diseño Web, desplegado en la Test-net de GOERLI, y que surge entre otras razones de la falta de productos financieros que estén basados en las tecnologías mencionadas anteriormente. Este trabajo, se vale de contratos programados en Solidity y un interfaz de usuario para proporcionar las funcionalidades de un fondo de inversión, compra y venta tokens.

PALABRAS CLAVE

Blockchain, Criptodivisas, Productos Financieros, Smart Contracts, Ethereum, Solidity, Trading, Exchange

ABSTRACT

In recent years technology has advanced rapidly, and with the arrival of the Blockchain, eliminating financial intermediaries, and cryptocurrencies, as the main support of the Blockchain, the concept of investments has changed completely. Such has been the revolution brought about by these new technologies, that it has left behind financial investments based on shares giving way to investments in cryptocurrencies, with a value based on the law of supply and demand, but with a higher volatility.

CryptoFund is a cryptocurrency indexed investment fund developed with Solidity and Web Design technology, deployed on GOERLI's Test-net, and which arises among other reasons from the lack of financial products that are based on the technologies mentioned above. This work uses contracts programmed in Solidity and a user interface to provide the functionalities of an investment fund, buying and selling tokens.

KEYWORDS

Blockchain, Cryptocurrencies, Financial Products, Smart Contracts, Ethereum, Solidity, Trading, Exchange.

ÍNDICE

RESUMEN.....	3
PALABRAS CLAVE	3
ABSTRACT.....	5
KEYWORDS.....	5
1. INTRODUCCIÓN.....	9
1.1. MOTIVACIONES Y SITUACIÓN ACTUAL.....	9
1.2. OBJETIVOS	10
1.3. PLAN DE TRABAJO	11
1.3.1. ETAPA DE PLANTEAMIENTO DEL PROYECTO	11
1.3.2. ETAPA DE DESARROLLO	12
1.4. ESTRUCTURA DEL DOCUMENTO	13
2. INTRODUCTION	15
2.1. MOTIVATIONS AND CURRENT SITUATION	15
2.2. OBJECTIVES	16
2.3. WORK PLAN.....	17
2.3.1. PLANNING STAGE.....	17
2.3.2. DEVELOPMENT STAGE	18
2.4. STRUCTURE OF THE DOCUMENT	19
3. SISTEMAS DE BLOCKCHAIN Y PRODUCTOS FINANCIEROS BASADOS EN CRIPTOMONEDAS.....	21
3.1. SISTEMAS DE BLOCKCHAIN.....	21
3.2. TOKENS.....	26
3.3. PRODUCTOS FINANCIEROS.....	29
3.3.1 EXCHANGES	31
3.3.2 PRÉSTAMOS	32
3.3.3 FONDOS DE INVERSIÓN	33
4. APLICACIONES SIMILARES.....	35
4.1. CRYPTO20.....	35
4.2. UNISWAP	36
4.3. PAXOS	38
5. DISEÑO.....	39
5.1. ARQUITECTURA DEL SISTEMA	40

5.2.	DISEÑO Y OPERACIONES DEL SERVIDOR OFF CHAIN.....	41
5.3.	DISEÑO DE CONTRATOS ON CHAIN	42
5.3.1.	PROXY	44
5.3.2.	SCEXCHANGE	44
5.3.3.	SCSTORAGE	45
5.3.4.	SCCOMMISSION.....	46
5.3.5.	SCOPS.....	48
5.3.6.	ANÁLISIS DE VULNERABILIDADES	48
5.3.7.	OPTIMIZACIÓN DE CONSUMO DE GAS	50
5.4.	DISEÑO DE INTERFAZ DE USUARIO.....	53
5.4.1.	EVOLUCIÓN DE LA INTERFAZ	53
5.4.2.	DESCRIPCIÓN DETALLADA DE LA INTERFAZ DE USUARIO....	55
5.4.3.	OTRAS INTERFACES	58
5.5.	HERRAMIENTAS UTILIZADAS	60
5.6.	BIBLIOTECAS UTILIZADAS	62
6.	DISPONIBILIDAD	65
6.1.	UBICACIÓN Y DISTRIBUCIÓN	65
6.2.	INSTRUCCIONES DE USO	66
7.	CONCLUSIONES.....	71
7.1.	CONCLUSIONES.....	71
7.2.	TRABAJO FUTURO	72
8.	CONCLUSIONS	75
8.1.	CONCLUSIONS	75
8.2.	FUTURE WORK.....	76
9.	CONTRIBUCIONES AL PROYECTO	79
9.1.	ISMAIL AZIZI.....	79
9.2.	GONZALO MARTÍNEZ	80
9.3.	MUAD ROHAIBANI	82
10.	REFERENCIAS	85

1.INTRODUCCIÓN

1.1. MOTIVACIONES Y SITUACIÓN ACTUAL

La tecnología Blockchain ha supuesto una revolución para el mundo financiero, al eliminar la necesidad de intermediarios financieros en muchas operaciones, reduciendo el coste de las operaciones y ofreciendo oportunidades de negocio impensables hace unos años. Aunque todos los actores del mercado están estudiando las consecuencias tanto actuales como futuras de esta revolución, solo una pequeña parte del mundo financiero tradicional se ha volcado en esta tecnología.

En esta situación, toman especial importancia los sistemas de blockchain programables, entre los que destaca el sistema Ethereum, que ha sentado las bases del futuro de todo el mercado de productos financieros descentralizados (DeFi) y se plantea como la infraestructura sobre la que se construirán los sistemas de productos financieros en un futuro próximo. Como resultado de la introducción de estos sistemas de blockchain, se ha producido una verdadera explosión de las criptomonedas que, o utilizan la misma arquitectura que la blockchain de Ethereum o una tecnología equivalente desplegada en una infraestructura de red propia, o están directamente desarrolladas en Ethereum y existen como *tokens* dentro del sistema en lo que se conocen como “sub-monedas”.

Un aspecto fundamental de los sistemas DeFi es que tienen la capacidad de operar con distintas criptomonedas simultáneamente, aprovechando las oportunidades de negocio que surgen del intercambio de criptomonedas en cada momento

Este trabajo surge de la necesidad real de una empresa para proporcionar productos financieros en criptomonedas, con características específicas, diferentes a los productos proporcionados por las soluciones disponibles en el mercado. Como resultado se ha desarrollado CryptoFund, un fondo de inversiones indexado de criptodivisas. Es un sistema que resuelve las necesidades específicas de esta empresa.

El sistema está desarrollado en la plataforma Ethereum y ofrece a los usuarios del sistema la posibilidad de realizar las operaciones básicas de un fondo de inversión, mediante el uso de un servidor off-chain y una aplicación on-chain. Además, CryptoFund ha sido desplegado en la red de pruebas GOERLI con resultados muy prometedores.

1.2. OBJETIVOS

El objetivo principal de este proyecto es crear un fondo de inversiones indexado de criptodivisas funcional. Al ser un proyecto nuevo se tienen que sentar unas bases que permitan su correcto funcionamiento y a su vez debe permitir que se pueda escalar en un futuro. Para ello se necesita cumplir una serie de objetivos específicos que se enumeran a continuación:

- Diseño de una interfaz de usuario, simple y completa, que permita al usuario poder realizar las acciones de comprar y vender tokens sin necesidad de que este usuario tenga conocimientos sobre programación ni tenga que hacer sus propios scripts.
- Conexión con una Blockchain real, para que, además de poder realizar pruebas en redes de prueba (Test nets), se pueda dar un uso real.
- Conexión con un servidor fuera de la Blockchain, denominado servidor off-chain, con estrategias de compraventa (trading) de criptomonedas, para poder realizar cálculos en base a una estrategia de trading sin tener que aumentar el coste de ejecución de los contratos.
- Posibilidad de operar con distintas monedas, para que no solo se pueda operar con las más conocidas como Bitcoin o Ethereum.
- Conexión con una cartera virtual, para poder realizar operaciones reales a largo plazo y no dejar el proyecto en usos teóricos de la aplicación.
- Creación de token propio, en el que se basará el fondo de inversión y que estará compuesto por diversas criptodivisas. El precio del token y su composición variaran en base a la estrategia de trading y el precio de las diversas criptodivisas de las que está compuesto.

1.3. PLAN DE TRABAJO

Este proyecto se ha organizado en un plan de trabajo formado por dos etapas principales: una primera etapa de planteamiento del proyecto, dónde se investigó sobre las tecnologías que íbamos a utilizar y una etapa de desarrollo dónde se implementó la aplicación web.

Para organizar el trabajo realizamos reuniones periódicas cada dos semanas que comenzaron el 15 de octubre de 2021, y que se pueden dividir en dos etapas:

1.3.1. ETAPA DE PLANTEAMIENTO DEL PROYECTO

Desde el 15 de octubre de 2021 al 25 de enero de 2022. Estas reuniones se realizaron de forma presencial, durante dichas reuniones fijamos el tema del trabajo y acotamos las funcionalidades que consideramos que podríamos abarcar. Se descartaron ideas como el uso de estrategias de backtesting y el uso de una estrategia de trading compleja, ya que consideramos que se necesitaría más tiempo del que disponíamos. Por otro lado, durante este periodo nos formamos en las tecnologías que íbamos a utilizar para realizar el proyecto, tanto revisando la documentación técnica de Ethereum [1] como de otros sistemas similares como Crypto20 y Uniswap, que son punteros en este ámbito.

Cabe destacar la incorporación de Valentín García [2], responsable de Fair Co. [3], a estas reuniones a partir del día 25 de noviembre de 2021, que puso a su disposición todos sus conocimientos sobre la tecnología Blockchain, así como su experiencia laboral con dicha tecnología. La llegada de Valentín nos ayudó para poder acotar los temas del trabajo y decidir a cuáles se les daría prioridad en un escenario real, además de darnos una perspectiva más realista sobre el mundo del Blockchain.

Por otro lado, la empresa de Valentín contrató a un alumno de económicas para extraer conclusiones financieras que trasladaríamos al mundo cripto, además de ayudarnos a la creación de una estrategia de trading.

1.3.2. ETAPA DE DESARROLLO

Esta etapa comenzó con el comienzo del primer cuatrimestre, que coincide con la mitad del proyecto, y las reuniones se realizaron de manera online a través de Google Meet. En la primera reunión dividimos el trabajo de desarrollo en dos partes a realizar por dos subequipos del proyecto: la parte de los contratos y el servidor off-chain, de la que se encargaron Ismail y Muad, y la parte de la interfaz de usuario, de la que se encargó Gonzalo. Una vez realizado reparto de tareas Ismail y Muad se organizaron para desarrollar la parte de los contratos y Gonzalo trabajó de manera independiente.

Ismail y Muad empezaron mediante varias reuniones en Google Meet para definir primero la arquitectura de los Smart Contracts y su conexión con el servidor, fueron depurando la estructura de los distintos componentes durante el primer mes mediante reuniones semanales y cada dos les daban retroalimentación Valentín y Jesús sobre cualquier aspecto que podrían mejorar. Una vez ya definida la arquitectura se inició la parte de programación de los contratos necesarios y pruebas del funcionamiento del sistema. Durante esta fase se realizaron reuniones de seguimiento cada dos semanas.

Antes de comenzar el desarrollo de la interfaz de usuario Gonzalo realizó una reunión con Valentín para revisar las diversas interfaces de usuario que tienen tanto los fondos de inversión indexados como las páginas que se dedican al intercambio de criptodivisas. A partir de esa reunión Gonzalo comenzó con la programación de la interfaz de usuario de manera independiente, mostrando los avances que iba realizando en las reuniones que realizábamos cada dos semanas.

1.4. ESTRUCTURA DEL DOCUMENTO

La estructura de la memoria está dividida en diez capítulos, cuyo contenido aparece resumido a continuación:

- Capítulo 1. Introducción. Exposición de los motivos por los que se realiza el proyecto, la situación actual, los objetivos que se plantean para el proyecto y un resumen del plan de trabajo que se ha realizado.
- Capítulo 2. Introduction. Traducción al inglés de Capítulo 1.
- Capítulo 3. Sistemas de Blockchain y criptodivisas. Exposición del funcionamiento de la tecnología Blockchain y las Blockchains más relevantes y funcionamiento de las criptodivisas que se han utilizado para construir el sistema de desarrollado.
- Capítulo 4. Aplicaciones Similares. Análisis e investigación de diversas aplicaciones disponibles en el mercado que proporcionen funcionalidades similares a las ofrecidas por este proyecto.
- Capítulo 5. Implementación. Descripción detallada del diseño de la arquitectura del sistema, el servidor off-chain, la aplicación on-chain y la interfaz de usuario.
- Capítulo 6. Disponibilidad. Incluye la ubicación del repositorio con el código fuente del sistema incluyendo los contratos y la interfaz de usuario, así como, el proceso de instalación de los diversos componentes del sistema
- Capítulo 7. Conclusiones. Describe las conclusiones que hemos extraído del desarrollo trabajo y propone nuevas funcionalidades y mejoras para implementar en el futuro.
- Capítulo 8. Conclusions. Traducción al inglés del Capítulo 7.
- Capítulo 9. Contribuciones al proyecto. Describe detalladamente las tareas realizadas por cada uno de los miembros del grupo.
- Capítulo 10. Bibliografía. Fuentes consultadas, APIs utilizadas para el desarrollo de la aplicación.

2.INTRODUCTION

2.1. MOTIVATIONS AND CURRENT SITUATION

Blockchain technology has revolutionized the financial world by eliminating the need for financial intermediaries in many transactions, reducing the cost of operations and offering business opportunities that were unthinkable a few years ago. Although all market players are studying both the current and future consequences of this revolution, only a small part of the traditional financial world has turned to this technology.

In this situation, programmable blockchain systems take on particular importance, among which the Ethereum system stands out, which has laid the foundations for the future of the entire decentralized financial products (DeFi) market and is posed as the infrastructure on which financial product systems will be built in the near future. As a result of the introduction of these blockchain systems, there has been a veritable explosion of cryptocurrencies that either use the same architecture as the Ethereum blockchain or equivalent technology deployed on a proprietary network infrastructure, or are directly developed in Ethereum, existing as tokens within the system in what are known as sub-currencies.

A fundamental aspect of DeFi systems is that they have the ability to trade different cryptocurrencies simultaneously, taking advantage of the business opportunities that arise from the exchange of cryptocurrencies at any given time.

This work is motivated by from the real need of a company to provide financial products in cryptocurrencies, with specific characteristics, different from the products provided by the solutions available in the market. This has resulted in the development of CryptoFund, a cryptocurrency indexed investment fund. It is a system that solves these specific needs of this company.

The system is developed on the Ethereum platform and offers system users the possibility to perform the basic operations of an investment fund, through the use of an off-chain server and an on-chain application. In addition, CryptoFund has been deployed on the GOERLI test network with very promising results.

2.2. OBJECTIVES

The main objective of this project is to create a functional cryptocurrency indexed investment fund. Since it is a new project, it is necessary to lay the foundations that allow its correct operation and at the same time it must be designed to scale in the future. To this end, a series of specific objectives that are listed below must be met:

- Design of a user interface, simple and complete, that allows the user to perform the actions of buying and selling tokens without the need for this user to have programming skills or have to write their own scripts.
- Connection with a real Blockchain, so that in addition to being able to perform tests in test networks (Test nets) it can be put to real use.
- Connection with a server outside the Blockchain, called off-chain server, with cryptocurrency trading strategies, to be able to perform calculations based on a trading strategy without having to increase the cost of contract execution.
- Possibility of trading with different currencies, so that the users can trade with currencies besides such as Bitcoin or Ethereum.
- Connection with a virtual wallet, to be able to eventually perform real operations and not leave the project in theoretical uses of the application.
- Creation of its own token, on which the investment fund will be based and which will be composed of various cryptocurrencies. The price of the token and its composition will vary based on the trading strategy and the price of the various cryptocurrencies of which it is composed.

2.3. WORK PLAN

This project has been organized in a work plan consisting of two main stages: a first stage of the project approach, where we researched about the technologies we were going to use and a development stage where the web application was implemented.

In order to organize the work, we hold regular meetings every two weeks, starting on October 15, 2021, those meetings can be divided into two stages, the project planning stage and the development stage:

2.3.1. PLANNING STAGE

From October 15, 2021 to January 25, 2022. These meetings were held in person, during these meetings we set the theme of the work and narrowed down the functionalities that we considered that we could cover. Ideas such as the use of backtesting strategies and the use of a complex trading strategy were discarded, as we considered that it would require more time than we had available. On the other hand, during this period we trained ourselves in the technologies that we were going to use to carry out the project, both reviewing the technical documentation of Ethereum [1] and other similar systems such as Crypto20 and Uniswap, which are leading in this field.

It is worth mentioning the incorporation of Valentín García [2], head of Fair Co [3], to these meetings from November 25, 2021, who made available all his knowledge about Blockchain technology, as well as his work experience with this technology. Valentin's arrival helped us to narrow down the topics of the work and decide which ones would be prioritized in a real scenario, as well as giving us a more realistic perspective on the Blockchain world.

On the other hand, Fair Co. hired a student of economics to draw financial conclusions that we would transfer to the crypto world, in addition to helping us create a trading strategy.

2.3.2. DEVELOPMENT STAGE

This stage started with the beginning of the first quarter, which coincides with the middle of the project, and the meetings were held online through Google Meet. In the first meeting we divided the development work into two parts to be carried out by two sub-teams of the project, the contracts part, which Ismail and Muad were in charge of, and the user interface part, which Gonzalo was in charge of. Once the tasks were distributed, Ismail and Muad organized themselves to develop the contracts part of the project and Gonzalo worked independently.

Ismail and Muad started with several meetings in Google Meet to first define the Smart Contracts architecture and its connection with the server, they were debugging the structure of the different components during the first month through weekly meetings and every two meetings Valentín and Jesús gave them feedback on any aspect that could be improved. Once the architecture was defined, the programming of the necessary contracts and testing of the system began. Follow-up meetings were held for two weeks during this phase.

Before starting the development of the user interface Gonzalo held a meeting with Valentin for the different user interfaces that both indexed investment funds and cryptocurrency exchange sites have. After that meeting Gonzalo started with the programming of the user interface independently, showing the progress he was making in the meetings we held every two weeks.

2.4. STRUCTURE OF THE DOCUMENT

The structure of the report is divided into ten chapters, the contents of which are summarized below:

- Chapter 1. Introduction. An explanation of the reasons for the project, the current situation, the objectives of the project and a summary of the work plan that has been carried out.
- Chapter 2. Introduction. English translation of Chapter 1.
- Chapter 3. The Blockchain systems and cryptocurrencies. Presentation of the functioning of Blockchain technology and the most relevant Blockchains and functioning of the cryptocurrencies that have been used to build the developed system.
- Chapter 4. Similar Applications. Analysis and research of several applications available in the market that provide similar functionalities to those offered by this project.
- Chapter 5. Implementation. Detailed description of the design of the system architecture, the off-chain server, the on-chain application and the user interface.
- Chapter 6. Availability. Includes the location of the repository with the source code of the system including the contracts and the user interface, as well as the installation process of the various components of the system.
- Chapter 7. Conclusions. Describes the conclusions we have drawn from the development work and proposes new functionalities and improvements to be implemented in the future.
- Chapter 8. Conclusions. English translation of Chapter 7.
- Chapter 9. Contributions to the project. Describes in detail the tasks performed by each of the members of the group.
- Chapter 10. Bibliography. Sources consulted; APIs used for the development of the application.

3.SISTEMAS DE BLOCKCHAIN Y PRODUCTOS FINANCIEROS BASADOS EN CRIPTOMONEDAS

En esta sección se explicarán cómo funcionan los sistemas de blockchain, explicando también conceptos como Proof of Work, Proof of Stake y Proof of Authority, que son de vital importancia; así como las principales Blockchains que hay. Por otro lado, también se explicará cómo funcionan los tokens ERC-20 y los métodos más importantes; y también cuáles son los principales productos financieros que hay en la actualidad.

3.1. SISTEMAS DE BLOCKCHAIN

La tecnología blockchain fue introducida en 2008 con la creación de Bitcoin [2]. Un sistema Blockchain de forma muy simplificada consiste en un registro electrónico de transacciones. Estas transacciones son transferencias de dinero digital entre distintas cuentas con características únicas; Que son inmutables, verificables, seguras y completamente descentralizadas. Es parecido a un banco tradicional solo que se realiza todo sin la necesidad de ningún tercero. Es una conexión únicamente entre los usuarios. Todo esto resuelve los problemas fundamentales de las transacciones electrónicas.

Las transacciones se agrupan en bloques que se van procesando por unos nodos de la red denominados "mineros". Por tanto, un bloque es un conjunto de transacciones. Cada bloque tiene una cabecera en la que se almacenan diversos datos (el minero que lo ha generado, el instante en el que se ha generado el bloque, etc.). Un dato fundamental de la cabecera del bloque es una referencia al bloque anterior, en forma de hash criptográfico. Un hash criptográfico es una función que, a partir de un dato de cualquier longitud, devuelve un valor de tamaño fijo (habitualmente de 256 bits en los sistemas de blockchain actuales) que tiene las siguientes propiedades: (1) es determinista: para el mismo dato de entrada, devuelve el mismo resultado; (2) no es reversible: dado un hash, no es factible calcular el dato con el que fue generado; (3) es resistente a las colisiones: no es factible calcular dos datos de entrada diferentes que produzcan el mismo resultado; y (4) cualquier cambio en los datos de entrada produce un hash completamente diferente.

Los sistemas de blockchain toman su nombre "cadena de bloques" de este hash criptográfico: cada bloque contiene el hash criptográfico del bloque anterior. De esta forma, cualquier cambio en un bloque producirá un hash

completamente distinto, por lo que el siguiente bloque deberá contener el nuevo hash para formar una cadena de bloques válida. Si alguien intenta modificar un bloque intermedio en una cadena de bloques, deberá recalcular el hash de todos los bloques posteriores.

El sistema de blockchain se mantiene en una red de nodos. Como se ha comentado antes, los mineros son los nodos que se encargan de generar el siguiente bloque, habitualmente a cambio de una comisión o recompensa. Otros nodos solo hacen consultas o envían transacciones para ser procesadas, pero todos los nodos validan que los bloques generados son correctos (el hash del bloque anterior es válido) y las transacciones están firmadas por los emisores y son correctas.

La red de nodos es descentralizada: no existe un nodo central que organice el funcionamiento del sistema. Los nodos utilizan un mecanismo de consenso para decidir cuál es el siguiente bloque de la cadena. Dependiendo del mecanismo de consenso, los nodos pueden competir para generar el siguiente bloque. Distinguimos cuatro protocolos de consenso principales:

- **Proof of Work (PoW):** Prueba de trabajo, primer protocolo de todos, se adoptó con la creación de Bitcoin [2] introduciendo también el concepto de Blockchain al mundo.
Consiste en que el primer usuario en resolver un problema computacionalmente complejo de cálculo de hashes recibe una recompensa y valida el bloque. Cuanta más potencia de cálculo tenga el mismo más fácil es validar el bloque y darle al minero una recompensa de minado en criptomonedas. Actualmente se ha vuelto una competición por ver quién realiza el minado más rápido, pero eso está siendo negativo en cuanto al gasto energético que se ha disparado. Tampoco es muy justo para todos los usuarios ya que el nodo de mayor potencia de cómputo es el más probable a descifrarlo por eso han surgido otros protocolos como Proof of Stake y Proof of Authority.
- **Proof of Stake [3] (PoS):** Prueba de participación, el objetivo de este protocolo es sustituir al Proof of Work y aportar una mayor seguridad y escalabilidad a las redes que lo implementen.
Los usuarios que minan en PoS se les denomina validadores, la decisión de quien valida un bloque se hace de forma aleatoria, aunque hay usuarios que tienen mayor probabilidad de ser elegidos dependiendo del tiempo que lleven participando en la red, la cantidad de monedas que posee el usuario. Esto lleva a incentivar los usuarios en usar más su sistema blockchain y que participen en

su proyecto con alguna cantidad de dinero y así tener mayor probabilidad de ser elegidos. También los desarrolladores pueden definir otras formas más de elección. No necesita una gran potencia de cálculo como en PoW, por lo tanto termina siendo una tecnología más respetuosa con el medio ambiente.

- **Proof of Authority [4] (PoA):** La prueba de Autoridad, su objetivo es ser práctica y eficiente a la vez. Fue propuesta por el cofundador de Ethereum, aunque aún no es tan conocida como PoW y PoS. Este protocolo se fija en que los usuarios (también se les llama validadores) proporcionen su identidad real y reputación como garantía para así saber quién está en el sistema realizando validaciones. De esta forma validadores intentarán cuidar su identidad y reputación dentro de la Blockchain. Se eligen arbitrariamente estos validadores, aunque su minero es limitado. Por lo que el sistema puede lograr tener una alta escalabilidad y velocidad. También el control de acceso al sistema es muy alto ya que solo los usuarios a los que se les ha dado permiso pueden participar.
- **Proof of Stake Authority [5] (PoSA):** consiste en una mezcla de Proof of Stake y Proof of Authority en la que los validadores confirman de forma alternativa las transacciones, pero la generación de nuevos bloques se hace mediante el método de Proof of Authority. Estos validadores están limitados a 21 y son elegidos entre el ranking de los que son activos con mayor número de tokens del sistema, a diferencia de la red PoS de Ethereum, que tiene cerca de 70.000 validadores. Todo esto resulta en comisiones y minado mucho más barato que los anteriores protocolos mencionados.

Hablando de las Blockchains programables, esta tecnología nos permite también registrar información más general que un registro de transacciones mediante smart contracts. Los Smart contracts son parecidos a los contratos legales solo que en forma de pequeños programas, escritos en su lenguaje de programación y asegurados por la Blockchain: su ejecución es verificable, podemos tener garantías de que se ha ejecutado correctamente y que no han realizado ningún ataque que pueda manipular y cambiar los datos de almacenamiento. Estos Smart Contracts proporcionan muchas más funcionalidades que un contrato tradicional. Por ejemplo, el sistema desarrollado en este trabajo implementa un fondo de inversión de criptodivisas usando Smart Contracts bajo la red de Ethereum que explicaremos posteriormente.

En este trabajo se utilizará la expresión “on-chain” para hacer referencia a los componentes que se ejecutan dentro de un sistema de blockchain, que en el caso de este trabajo es un conjunto de contratos desplegados en una red Ethereum. Por otra parte, la expresión “off-chain” hace referencia a aquellos procesos o sistemas que se ejecutan en servidores externos al sistema de blockchain.

Existen muchos sistemas de blockchain disponibles, en este trabajo se utilizan los 3 que se describen a continuación:

- Ethereum [1]: Fue fundada en 2015 por Vitalik Buterin un programador con el objetivo de darle un uso más amplio a las aplicaciones descentralizadas. Fue el primer sistema que propone el concepto de blockchain programable que incluye una máquina virtual para la ejecución de Smart Contracts on-chain de forma segura y verificable.

Tiene una blockchain parecida a Bitcoin ya que utiliza Proof of Work también. Junto a este blockchain programable, se diseña su propio lenguaje de programación denominado “Solidity”, permitiendo así a los usuarios la posibilidad de programar código en una Blockchain. Todo este código es ejecutado en una máquina virtual denominada Ethereum Virtual Machine (EVM), después de pasar las instrucciones a códigos de operación y posteriormente a bytecode. Actualmente es la máquina virtual más utilizada para ejecutar contratos en sistemas blockchain.

Ethereum además introduce un concepto innovador para evitar ataques de denegación de servicio (DoS), que es la noción de consumo de gas.

Todas las operaciones realizadas en Ethereum que modifiquen su estado (por ejemplo, una transacción de Ether) tienen un coste en gas. De esta forma, cada instrucción de la EVM consume una cantidad de gas cuando se ejecuta.

Cuando se va a procesar una transacción, se ha de indicar el valor en Ether de cada unidad de gas, así como la cantidad máxima de gas que se puede consumir. Este mecanismo evita ataques DoS basados en transacciones que no terminan y que podrían bloquear el sistema: en cuanto se llega al límite máximo de gas, se revierte la transacción, lo que permite continuar procesando otras transacciones.

Además, la asignación de un precio en Ether a cada unidad de gas permite retribuir a los mineros que procesan las transacciones.

Otro concepto es el de “address”, la dirección de los usuarios desde donde realizan las distintas operaciones, se usa para identificar a

los usuarios y los contratos desplegados en el sistema, al ser un valor único que no se puede modificar.

Una gran desventaja de la red principal de Ether es el alto coste de gas por transacción. Por este motivo han surgido otras blockchains para mejorar estos problemas como Binance Smart Chain y Polygon.

- Binance Smart Chain [6] (BSC): Lanzada por primera vez en septiembre de 2020 por Changpeng Zhao también creador de “Binance” el mayor portal de intercambio de criptodivisas del mundo. Después de los problemas anteriormente mencionados en Ethereum, relacionados con el coste de las transacciones, surge esta Blockchain y gana una popularidad bastante rápido debido a su compatibilidad con la máquina virtual de Ethereum (EVM) y sus transacciones que son más rápidas y significativamente más baratas.

Esta Blockchain funciona con Proof of Staked Authority (PoSA).

Una gran desventaja de BSC es la centralización, al ser muy dependiente de una sola empresa en varios aspectos.

- Polygon [7]: Es un sistema de Blockchain centrado en ofrecer soluciones de escalabilidad a proyectos implementados bajo la red de Ethereum. Ya después de la gran adopción que está teniendo esta tecnología en los últimos años, han aparecido nuevos proyectos y muy prometedores que funcionan bajo la misma Blockchain, por tanto, ahora mismo Ethereum está al límite de sus capacidades y Polygon pretende dar una alternativa mediante la creación de una red de nodos descentralizados que sirven para formar una cadena lateral (denominada sidechain) conectada a la de Ethereum actuando, así como apoyo. Esta sidechain funciona mediante Proof of Stake, lo que permite una alta velocidad de transacción dentro de la Blockchain y unos costes de comisión menores a un céntimo.

Hemos elegido estos tres sistemas porque funcionan todos con la máquina virtual de Ethereum Virtual Machine y son programables para los usuarios. Los contratos que forman el componente on-chain se instalan en la red de Ethereum, y las otras dos nos ayudan para conseguir un menor coste de operación y ejecuciones más rápidas.

3.2. TOKENS

Los tokens hacen posible que existan la mayoría de los productos y proyectos en Ethereum y otras blockchains programables. Son unidades de valor emitidas por una entidad privada, que no tienen ninguna validez por sí mismos, pero que sirven en ciertos contextos. Los tokens se asemejan a unas fichas de casino, las cuales tienen uso dentro del establecimiento, pero fuera de él son inútiles. Nos centraremos en los tokens basados en las redes EVM. Estos tokens se definen en un contrato que lleva un registro de los propietarios de dichos tokens y las cantidades que estos poseen e implementan funciones básicas para la gestión de los tokens, como la realización de transferencias. En Ethereum por lo general vienen definidos inicialmente por estándares EIP (Ethereum Improvement Proposals) [8], que indican la estructura básica que debe tener el contrato que defina a un tipo de token. Estos EIP pueden estar en cuatro fases: fase de consulta inicial de la propuesta, en la que están abiertos a discusión; fase de revisión donde si la propuesta inicial ha salido adelante serán pronto convertidos en un estándar y utilizados después de realizar los últimos cambios; fase de utilización o estandarización donde el token es ya un estándar aceptado y se convierte en un ERC (Ethereum Request for Comments); y la última sería la fase no aceptada, en la que la propuesta ha sido rechazada y no se estandarizará en la red.

Mediante el uso de estándares ERC, los tokens del mismo tipo son iguales en cuanto a sus funciones básicas (transferir, crear, destruir, comprobar balance, etc.) y se puede utilizar una interfaz universal para todos. Esto no quiere decir que no haya tokens que implementen funcionalidades únicas a cada uno, pero es muy útil la utilización de un estándar para poder manejar los tokens independientemente de su implementación específica.

El estándar que hemos utilizado en este trabajo es el ERC20, que es el más común y el primer estándar de token que fue definido. Se define en el EIP-20. Este estándar define ciertas funciones:

- **totalSupply()**: indica el número de tokens que hay en circulación.
- **balanceOf(address owner)**: indica el saldo de la cuenta indicada por la dirección 'owner'.
- **transfer(address to, uint value)**: con esta función, un usuario puede transferir la cantidad de tokens indicada por 'value' desde su cuenta a la de otro usuario, indicada por la dirección 'to'.
- **approve(address spender, uint value)**: con esta función, el usuario da a la cuenta de otro usuario indicada por la dirección 'spender' permiso para utilizar la cantidad de tokens indicada por 'value' como si fuesen suyos.

- **transferFrom(address from, address to, uint value)**: el usuario envía la cantidad de tokens indicada por 'value' desde la cuenta correspondiente a la dirección 'from', a la indicada por la dirección 'to' (siempre y cuando la cuenta de origen tenga el saldo suficiente). Para realizar esta transferencia el usuario debe tener permiso de la cuenta 'from', dado en la función 'approve', para utilizar sus tokens.
- **allowance(address owner, address spender)**: devuelve la cantidad que tiene permitida gastar la cuenta con dirección 'spender' sobre los fondos de la cuenta con dirección 'owner'.

Además de las funciones, el estándar define ciertos eventos. Los eventos son notificaciones que se activan cuando sucede una acción determinada y que almacenan información en la blockchain. Esta información puede ser leída por una página web, una aplicación de cartera, etc... Se definen los siguientes:

- **Transfer(address from, address to, uint256 value)**: se activa cada vez que se transfieren tokens (incluidas las transferencias de valor cero).
- **Approval(address owner, address spender, uint256 value)**: se activa en cada llamada con éxito a la función approve.

Opcionalmente pueden estar presentes las siguientes funciones de consulta. Estas funciones no son obligatorias y solo se utilizan para mejorar la usabilidad:

- **function name()**: devuelve el nombre del token.
- **function symbol()**: devuelve el símbolo del token.
- **function decimals()**: devuelve los decimales que el token usa. Como las cantidades se representan con enteros, el token puede tener un campo decimals que indica que hay que dividir las cantidades entre 10 elevado al número de decimales para obtener las cantidades reales de tokens.

El token ERC20 es con diferencia el estándar más utilizado, aunque se han considerado otras opciones en este trabajo. Por ejemplo, el propuesto en el EIP-223 y el ERC777. Ambos estándares tienen retrocompatibilidad con los tokens ERC20.

El EIP-223 pretende resolver uno de los problemas de los tokens ERC20, el cual no permite saber que un contrato está recibiendo tokens de dicho tipo, y de esta manera, si se envían tokens a un contrato que no esté preparado para manejarlos (realizar transferencias con dichos tokens o que lleguen de manera no esperada a dicho contrato), los tokens podrían llegar a perderse para siempre sin manera de recuperarlos. De esta

manera y como se indica en esta propuesta, las equivocaciones de los usuarios pueden dar lugar a millones de dólares perdidos.

Los tokens ERC20 también pueden ser susceptibles a ataques de falso depósito. Estas vulnerabilidades pueden darse por la combinación de dos motivos: una implementación errónea del contrato del token ERC20 y la mala verificación por parte de un exchange (centralizado o descentralizado) o cualquier contrato cuya funcionalidad incluya el depósito de los tokens ERC20. Este tipo de vulnerabilidad fue objeto de estudio en abril de 2020, y se descubrió una gran cantidad de tokens con estos problemas, unos 7.000 de 176.000 analizados [9].

Los contratos con una mala implementación de las funciones `transfer()` o `transferFrom()`, pueden ser utilizados por el atacante mediante el uso de un exchange. Cuando el atacante llame a la función `deposit()` para depositar tokens en el exchange, el exchange llamará a la función `transfer()` o `transferFrom()`, y si estas están mal implementadas pueden ser susceptibles a estos falsos depósitos. Estas vulnerabilidades suceden si el exchange no comprueba que estos fondos hayan sido depositados, y asume que el atacante tenía saldo suficiente. Si el contrato del token implementa las funciones de manera que en lugar de revertir si no hay saldo suficiente, solo devuelven un valor booleano, el exchange creerá que el atacante ha depositado los fondos, ya que la ejecución no ha revertido, cuando realmente no se ha depositado nada. También puede suceder que alguna de estas funciones no esté implementada, llegando así la llamada a la función de fallback, donde podría llegar a ejecutarse código arbitrario.

Estos ataques son posibles debido a que el estándar ERC20 define la interfaz que debe seguir el contrato, pero no la implementación de las funciones.

La propuesta EIP-223 pretende arreglar estos problemas del ERC20 e implementar en los contratos la función `tokenReceived()`. Esta función debe ser llamada cada vez que un contrato reciba los tokens, y en caso de no estar presente, la transferencia fallaría. De esta manera, aseguramos que los contratos siempre son conscientes de las transferencias de dichos tokens y que no acabarán bloqueados en un contrato y se podrá gestionar cada transferencia e incluso bloquear la llegada de tokens no deseados. En caso de que el destinatario no sea un contrato, y sea una cuenta de propiedad externa, la transferencia se podrá realizar siempre.

El estándar ERC777, definido en el EIP-777, es también una extensión del ERC20, y al igual que el EIP-223, se obliga a implementar una función de fallback en los contratos para poder recibir tokens de este tipo, y también se sustituye la función `transfer()` por una función `send()` con un parámetro

más de tipo bytes. Esto hace que las transferencias se asemejen más a una transferencia normal de Ether. Este estándar propone muchas más funcionalidades que el EIP-223 y es el que se ha convertido en un estándar aceptado al contrario que el EIP-223. Este tipo de tokens especifica no solo la interfaz que deben seguir los ERC777, además actúa como registro público, descentralizado y con una dirección universal en cualquier blockchain EVM, y donde cada contrato o cuenta debe registrar las funciones que define e implementa. Este registro comprueba que los contratos implementen dichas funciones y lo almacena en un mismo contrato todo, de esta manera, para cualquier token de este tipo, sus funciones estarán verificadas y registradas en un mismo sitio. Los tokens ERC777, además de definir las interfaces, también definen la implementación obligatoria de dichas funciones de la interfaz, evitando así algunas vulnerabilidades que puedan surgir debido a una implementación incorrecta.

El uso de estos estándares alternativos al ERC20, aunque podría ser útil, se ha descartado a favor de la simplicidad y el uso masivo de este tipo de tokens más clásicos. Otra razón para utilizar ERC20, es que los tokens que se utilizarán para comprar o vender con las estrategias de trading son tokens ERC20, así que, de esta manera, mantenemos el mismo estándar en todo nuestro proyecto.

3.3. PRODUCTOS FINANCIEROS

Hay una gran variedad de productos financieros, tanto en las redes blockchain, como Ethereum y otras redes EVM, como en plataformas más tradicionales. Hay gran variedad de productos financieros, que podemos dividir en dos clases: centralizados y descentralizados.

Los productos centralizados se asemejan a los productos financieros tradicionales, donde hay una institución o una empresa que gestiona los servicios que se proveen y donde la empresa es dueña del producto. Estos productos deben estar regulados por el estado o país que corresponda, y debe aplicar ciertas medidas, tales como KYC (Know Your Customer) [10] o AML (Anti-Money Laundering) [11], que impiden el uso anónimo de dichos servicios, ya que necesitamos registrar una cuenta con nuestros datos personales y enviar una foto o escaneo de un documento de identidad para poder utilizar dicho servicio. Todas las transacciones que realicen los usuarios las registrará y almacenará la empresa como información privada que no es visible por otros usuarios. Los productos suelen tener una liquidez muy alta gracias a que la empresa gestiona y

mantiene su producto, respaldándolo con grandes cantidades de dinero si es necesario y las operaciones son más baratas y rápidas, debido a que no están ubicados en redes blockchain, sino en servidores privados de la empresa, y, por lo tanto, no necesitamos confirmación ni minado de las transacciones. Dentro de estos productos centralizados tenemos exchanges (plataformas de intercambio), fondos de inversión tradicionales, brokers (intermediarios que nos permiten comprar y vender acciones), etc...

El otro tipo de productos financieros son los descentralizados, que vamos a gestionar con el sistema CryptoFund, ya que son los presentes en las redes blockchain. Estos productos se basan en contratos inteligentes, que implementan la funcionalidad del producto financiero. Los contratos pueden definir desde algo sencillo, como un token, a aplicaciones complejas como DeX (Decentralised Exchange), sitios de préstamo, o aplicaciones de staking, donde el usuario deja su dinero para recibir un beneficio fijo o variable en función del tiempo.

Estas aplicaciones, al funcionar dentro del sistema de blockchain, tienen su código de byte disponible para que cualquiera pueda verlo. Además de esto, muchas de las aplicaciones son de código abierto, así que su implementación de alto nivel también está disponible para el resto de los usuarios. Esto da lugar a que muchas veces haya aplicaciones que hagan sus propias versiones adaptadas (forks) de otras ya existentes e incluso que añadan su propia funcionalidad adicional a la aplicación original. Esto beneficia bastante a los desarrolladores de estas aplicaciones descentralizadas, ya que cualquiera puede sugerir sus cambios y aportar ideas que crea que mejoran la aplicación, y se puede tener una realimentación a nivel no solo de experiencia de usuario, también a nivel de código, optimizaciones y corrección de bugs. Otra ventaja de las aplicaciones basadas en contratos inteligentes es que no necesitan un servidor para estar alojados, y mientras la red de blockchain esté operativa, los contratos seguirán existiendo y siendo funcionales, todos los días del año, a cualquier hora. Esto permite a los usuarios crear sus propias aplicaciones sin necesidad de una cantidad de recursos muy grande, ya que solo hace falta conocer el lenguaje de programación y un poco de Ether (o la moneda de la red de blockchain correspondiente) y, debido a la existencia de aplicaciones open-source, es muy fácil empezar con una base preestablecida y evitando las complicaciones de hacer todo desde cero. Una desventaja de las aplicaciones descentralizadas frente a las centralizadas son los tiempos de operación y los costes de las mismas. Los tiempos de operación son más largos frente a las plataformas centralizadas, ya que nuestra transacción debe ser confirmada en un bloque por algún minero o validador. En Ethereum los tiempos de bloque

pueden tardar entre unos pocos segundos y casi un minuto en algún caso, aunque en otras redes Proof of Stake, como por ejemplo BSC, los bloques tardan 3 segundos, y en otras redes pueden tardar aún menos. Los costes pueden verse incrementados, ya que, dependiendo del nivel de congestión de la red, será necesario pagar un coste más alto por cada unidad de gas para que nuestra transacción se realice.

A continuación, se describen brevemente las clases de productos financieros que hay presentes, tanto en las redes blockchain como en aplicaciones financieras tradicionales.

3.3.1 EXCHANGES

Los *exchanges* o sitios de intercambio son sitios donde se pueden intercambiar criptomonedas a cambio de una pequeña comisión porcentual. Se dividen en exchanges centralizados (CeX) y exchanges descentralizados (DeX). Los CeX permiten operar con cantidades mayores, ya que poseen mucha más liquidez, y de manera más rápida. También permiten cambiar moneda de curso legal (dinero fiat) por criptomonedas, pero con la desventaja de que cualquier operación tendrá que realizarse dentro de su plataforma. Los DeX permiten operar directamente desde una cuenta en la red blockchain, y sin necesidad de depositar los fondos en ningún sitio. Además de estos, se abre la posibilidad de, en una misma transacción, realizar múltiples operaciones entre los contratos o plataformas que queramos. De esta manera se abre la opción al arbitraje (estrategia financiera que consiste en aprovechar la diferencia de precio entre diferentes mercados sobre un mismo activo financiero para obtener un beneficio económico) entre varios exchanges.

Como ejemplo de estos CeX, tenemos Binance [12], FTX [13] y Coinbase [14], siendo Binance el más grande de los tres, llegando de media a un volumen en 24h de entre de 50 mil y 100 mil millones de dólares.

Los DeX más relevantes son Uniswap(V3) [15] y PancakeSwap [16], siendo estas plataformas las más grandes de la red Ethereum y Binance Smart Chain respectivamente, y llegando a un volumen en 24 horas de 1.300 y 700 mil millones de dólares.

3.3.2 PRÉSTAMOS

Son bastante comunes los sitios de préstamo en el ecosistema de las criptomonedas. Son muy utilizados, ya que, con la gran cantidad de tokens y monedas existentes, es muy difícil tener activos de todo tipo en cualquier momento, ya que necesitaría una inversión muy grande. Es por eso que existen los préstamos con colateral, en los cuales se deposita una cantidad mayor a la prestada en una moneda que tenga el usuario, y a cambio recibe una fracción de lo depositado en otra moneda, a cambio de una pequeña comisión. Si, por ejemplo, el usuario solo posee Ether, pero para realizar una operación de arbitraje necesita utilizar el token USDC, es posible depositar Ether y sacar un porcentaje de esa cantidad para realizar la operación, y más tarde devolverlo para recibir el colateral de vuelta.

Normalmente, los CeX como Binance dan esta opción, mientras que es una característica menos común en los DeX. Normalmente las plataformas descentralizadas que realizan préstamos suelen estar dedicadas íntegramente a los préstamos y a proveer liquidez para los mismos. Un ejemplo de esta plataforma es Aave [17], la cual tiene alrededor de 14 mil millones de dólares de liquidez entre 7 blockchains distintas.

Aunque, como hemos comentado, los DeX no suelen hacer estos préstamos con colateral, suelen implementar una funcionalidad llamada "flash swaps". Estos intercambios consisten en que se presta una cantidad que, de un activo en concreto, sin necesidad de depositar colateral, con la condición de que esta cantidad prestada sea devuelta al final de la misma transacción en la que nos ha sido prestada, para que realicemos los intercambios que queramos sin necesidad de tener los activos a nuestra disposición. Para el prestamista es seguro realizar estos cambios, gracias a la instrucción REVERT de la máquina virtual de Ethereum, la cual permite revertir todos los cambios que haya realizado una transacción. De esa manera, si el prestamista no recibe de vuelta los fondos, simplemente la transacción se revierte y no se pierde nada.

3.3.3 FONDOS DE INVERSIÓN

Al estar diseñando un fondo de inversión, en este trabajo hemos tenido que estudiar en el funcionamiento general de estos y aplicarlo a la tecnología blockchain. Un fondo de inversión general está organizado por una institución en la que los partícipes depositan su dinero para que un equipo lo invierta en varios activos financieros. El dinero que se ha recaudado entre todos los partícipes del fondo lo llamamos patrimonio, que se divide en participaciones. Cada partícipe tendrá un número de participaciones en función del dinero invertido. Este patrimonio va variando constantemente en función de los partícipes nuevos que entren y salgan, también si los activos en los que se ha invertido se revalorizan o bajan de precio. Otro concepto importante es la comisión que se le cobra al usuario por dejar ahí depositado sus fondos y estar realizando la empresa todas las inversiones, consiste generalmente en un porcentaje anual a lo que se ha invertido, suele ser entre el 5% y el 8%.

En el ámbito de los fondos de inversión hay cuatro conceptos fundamentales que se utilizarán en los siguientes capítulos:

- NAV (Net Asset Value): Representa el valor neto de una entidad, en este caso del fondo. Se calcula sumando el valor total de activos menos el total de pasivos.
- Participación: el patrimonio se divide en partes iguales llamadas participaciones.
- Partícipe del fondo: cada una de las personas que ha aportado parte de su capital al fondo
- Sociedad Gestora: Organización encargada de adoptar las decisiones y políticas de inversión. También se encarga de la gestión y administración del fondo.

4.APLICACIONES SIMILARES

Para poder crear una aplicación resuelve los problemas de una empresa real y pueda ser una alternativa a algunas otras aplicaciones ya existentes, el primer paso lógico es explorar la competencia que hay en el mercado actual e intentar averiguar qué características destacan de ellos y qué es lo que pueden aportar a nuestro proyecto.

Tras una búsqueda exhaustiva, se comprendió que había una gran cantidad de fondos de inversión, pero muy pocos de ellos se dedican al mundo cripto, por otro lado, también hay bastantes aplicaciones dedicadas al mundo cripto, pero no son tan numerosas como los ya mencionados fondos de inversión.

Por todo lo mencionado anteriormente, hemos seleccionado tres aplicaciones web que consideramos que tienen un fin claramente relacionado con nuestra aplicación o que nos han servido de inspiración tanto a la hora de decidir que funcionalidades implementar tanto como a la hora de diseñarla.

4.1. CRYPTO20



Crypto20 [18] [19] es un fondo indexado autónomo basado en Blockchain y criptodivisas lanzado en febrero de 2018 por un grupo de empresarios e inversores. Es el primer fondo indexado que se ha creado en este campo y el único exitoso de los pocos que hay actualmente.

Es un sistema que realiza tracking de las 20 criptomonedas con mayor capitalización del mercado. Tiene su propia criptodivisa, C20. El usuario, al invertir en este token, le está dando su capital para que lo inviertan de la mejor forma mediante compras y ventas automatizadas (eso hace que las operaciones sean de menor comisión) en 20 de los mayores sistemas Blockchain como Bitcoin, Ethereum, Cardano, etc. y suba así el valor del token mediante estas distintas acciones. Todos los beneficios obtenidos

se reinvierten en el fondo para tener un margen de crecimiento cada vez mayor. El usuario puede depositar y retirar sus tokens cuando él quiera no se le obliga a retener su dinero. El fondo recibe a cambio de un porcentaje de comisión. El fondo se va reequilibrando semanalmente según unos parámetros fijados, siendo así un sistema autosuficiente.

También tienen otro proyecto al que llaman Crypto10 [20] [20], que se basa en la misma idea de fondo de inversión, aunque es con las 10 mayores criptomonedas. La forma que tienen de operar es distinta: en este caso, en cuanto se detecta que el mercado está empezando a ser muy bajista (porque en este mercado cuando el valor de Bitcoin baja, el mercado entero baja también), se pasa todo lo invertido en criptodivisas a dólares y posteriormente se compran de nuevo criptomonedas cuando sea el momento oportuno. Como en el caso anterior es un sistema que funciona de forma automática.

4.2. UNISWAP



UNISWAP

Uniswap [21] es un DeX (Decentralised Exchange) cuya primera versión fue lanzada en noviembre de 2018. Debido a la naturaleza descentralizada, nadie controla dicho sitio de intercambio, y su versión v1 sigue existiendo a día de hoy, aunque la versión más reciente es la v3.

Este exchange se basa en pools de pares de tokens ERC20, cuyo precio viene determinado por la fórmula de producto constante. De esta forma, si tenemos una pool con 10 tokens de tipo A y 20 tokens de tipo B, cada token A valdrá 2 tokens de tipo B. Esta proporción debe mantenerse siempre constante, con lo cual, el precio queda determinado de manera automática por esta fórmula. Esto también abre la posibilidad a que la gente realice operaciones de arbitraje, ya que, según se vayan intercambiando tokens de un tipo, el precio de estos irá bajando, y el otro token del par subirá de precio, pudiéndose producir un desajuste de precios, y una oportunidad de beneficio para un comprador rápido que la aproveche.

Cada intercambio tiene una comisión porcentual (0,3% para todas las pools en el caso de UniswapV2 y 1%, 0,3%, 0,05% o 0,01%, dependiendo de la pool, en el caso de UniswapV3). Cada vez que se realiza un intercambio, se cobra esta comisión y se distribuye de manera proporcional entre todos los proveedores de liquidez. La plataforma tiene una comisión propia que podría llegar a activarse, pero hasta el momento ha estado siempre deshabilitada. Los propietarios del sitio no obtienen ingresos directamente, sino que proveen liquidez y obtienen ingresos como cualquier usuario normal.

Su versión v1 fue popular, pero no fue hasta la versión v2 que Uniswap se estableció como uno de los exchanges más importantes en Ethereum e incluso en otras redes EVM en las que ni siquiera existía. Esto es debido a que, al ser de código abierto, hay infinidad de exchanges que son forks de Uniswap v2, por ejemplo, PancakeSwap, que es uno de los más grandes en todas las redes EVM y el más grande de la red BSC. Uniswap v2 introducía una funcionalidad muy interesante y que, debido al funcionamiento de Ethereum y el opcode REVERT, era posible y sencilla de implementar: los intercambios flash. Como se ha mencionado en el capítulo anterior, los flash swaps, no son más que préstamos sin colateral ni aval, en los que el préstamo y la devolución de dicho préstamo sucede dentro de la misma transacción, con la peculiaridad de que, al ser un sitio de intercambio, habrá que pedir prestado tokens de un tipo, y devolver tokens del otro. Esta operación es muy interesante y útil para gente que intente realizar operaciones de arbitraje dentro del mismo exchange o entre distintos sitios, ya que permite aprovechar un precio favorable sin riesgo y sin necesidad de perder activos. A día de hoy, Uniswap v2, aunque ha perdido importancia, sigue siendo bastante relevante y tiene un uso considerable.

Finalmente, su versión v3, la más reciente, es uno de los DeX con más volumen y liquidez de todas las redes EVM, y además de seguir implementando más o menos la misma estructura, cambia la manera en la que se provee la liquidez y en la que se calcula el precio. En esta versión los usuarios que quieran depositar sus tokens en un par, en lugar de depositarlos en la proporción marcada por el precio actual, que sigue la fórmula del producto constante, pueden elegir los rangos de precio en los que quieren proveer su liquidez. De esta manera, se pueden concentrar mucho más los activos en los rangos de precio elegidos y maximizar los beneficios aún más, al no estar repartiendo los activos por todo el rango de precios posible.

4.3. PAXOS



Paxos es una institución financiera y una empresa tecnológica neoyorkina fundada en 2012 y especializada en la tecnología Blockchain. La oferta de productos de esta empresa incluye criptocorretaje (broking), servicios de tokenización de activos y servicios de liquidación. Pero sus productos más importantes son itBit, PAX y PAX Gold.

- ItBit: es una bolsa de activos digitales aprobada por el Departamento de Finanzas del Estado de Nueva York para comercializar cinco activos digitales: bitcoin (BTC), Ethereum (ETH), Bitcoin Cash (BCH), Litecoin (LTC) y PAX Gold (PAXG).
- PAX: es un token ERC20 que está 100% respaldado por dólares estadounidenses. Por cada PAX que se intercambie, debe haber al menos un dólar estadounidense en las cuentas bancarias de Paxos. Al igual que otras monedas estables como la moneda del dólar estadounidense, PAX busca combinar la estabilidad y confiabilidad del dólar estadounidense con los beneficios de una moneda digital. Es decir, que se puede transferir a través de Internet a cualquier persona del planeta, de inmediato, en cualquier momento y con un coste casi nulo.
- PAX Gold: es un proyecto Blockchain open source construido en la Blockchain de Ethereum. La peculiaridad de este token ERC-20 es que se trata de un token estable vinculado al oro. Cuando alguien compra un token PAXG, compra un lingote de oro. Cada token PAXG está respaldado por un lingote de oro que se encuentra almacenado por Paxos Trust Company. Al estar vinculado a oro físico su precio también está vinculado directamente al precio real del oro en el mercado.

5.DISEÑO

En esta sección se describe de manera detallada la arquitectura del sistema CryptoFund: el diseño y operaciones realizadas por el servidor off chain; el diseño de los contratos, haciendo hincapié en el análisis de posibles vulnerabilidades y el uso de técnicas para reducir el consumo de gas; y el diseño de la interfaz de usuario, explicando su evolución, interfaces de usuario de otras aplicaciones con usos parecidos y las herramientas y bibliotecas que se han utilizado para desarrollarla.

Para el desarrollo de este sistema Blockchain hemos tenido que adaptar los conceptos de un fondo de inversión original a uno que use esta tecnología.

El NAV de CryptoFund es el resultado de sumar todo el dinero que se ha depositado en el fondo de inversión con la compra del token propio, y las participaciones son las unidades de este token ERC20 creado para gestionar el fondo y darle un valor a cada participación.

Los usuarios invierten en el fondo comprando unidades de este token en la página web.

El sistema dispone de una pequeña estrategia de trading automatizada que es la encargada de invertir el dinero de los participantes en criptodivisas.

El valor del token del fondo va fluctuando dependiendo de las operaciones de trading que haga el sistema: si son ganadoras aumenta el valor, pero si son perdedoras disminuye. Este valor depende también de la cantidad de participantes que tenga el sistema: cuantos más participantes haya, mayor es el valor del token.

Cuando un participante quiere retirar sus tokens se calcula la comisión a pagar dependiendo del tiempo que haya dejado depositado sus participaciones en el fondo.

Es un sistema autosuficiente, ya que realiza automáticamente las inversiones de los participantes. Se puede ir puliendo la estrategia de trading para conseguir que los inversores saquen un mayor beneficio y atraer a más usuarios que inviertan.

5.1. ARQUITECTURA DEL SISTEMA

El sistema se compone de tres elementos principales como podemos ver en la Figura 1: el subsistema on-chain formado por un conjunto de contratos desplegados en la blockchain, una interfaz de usuario basada en una página web que sirve para que los usuarios más inexpertos puedan utilizar los contratos de manera sencilla y visual, y un servidor que utilizaremos para implementar las estrategias de trading con las que intentaremos generar beneficios.

La comunicación entre la interfaz de usuario y los contratos se realiza mediante el contrato **SCExchange**, el cual sirve de punto de compra y venta para los usuarios que quieran hacer uso de nuestro servicio. Este contrato es el único con funciones que pueden ejecutar los usuarios, y se comunica con los contratos **SCStorage** y **Token** para generar las participaciones en el fondo de inversión y depositar el dinero pagado por ellas. También hacemos uso del contrato **Proxy**, para tener las direcciones almacenadas de manera dinámica, y leemos del contrato **SCStorage**, para saber el NAV de nuestro fondo.

El servidor conecta con el contrato de operaciones SCOps, que utiliza los fondos de **SCStorage** para realizar intercambios y poder generar dinero. El servidor off-chain contendrá la estrategia de trading y cada vez que necesite operar, se lanzará un script en JavaScript que comunique con el contrato de operaciones y realice las operaciones de intercambio, compra o venta según corresponda. El servidor también conecta con el contrato **SCCommissions** para recoger las comisiones dependientes del tiempo, que se generan a raíz de que la gente deposite su dinero en nuestro fondo.

Los contratos están disponibles de manera pública en la blockchain, y cualquiera puede ver su implementación y leer su estado. Los contratos se comunican entre sí para realizar las operaciones necesarias para comprar y vender, almacenar fondos o cobrar comisiones. Solo el propietario de los contratos tiene acceso a las funciones de borrar contratos, cambiar algún parámetro, recoger las comisiones, etc...

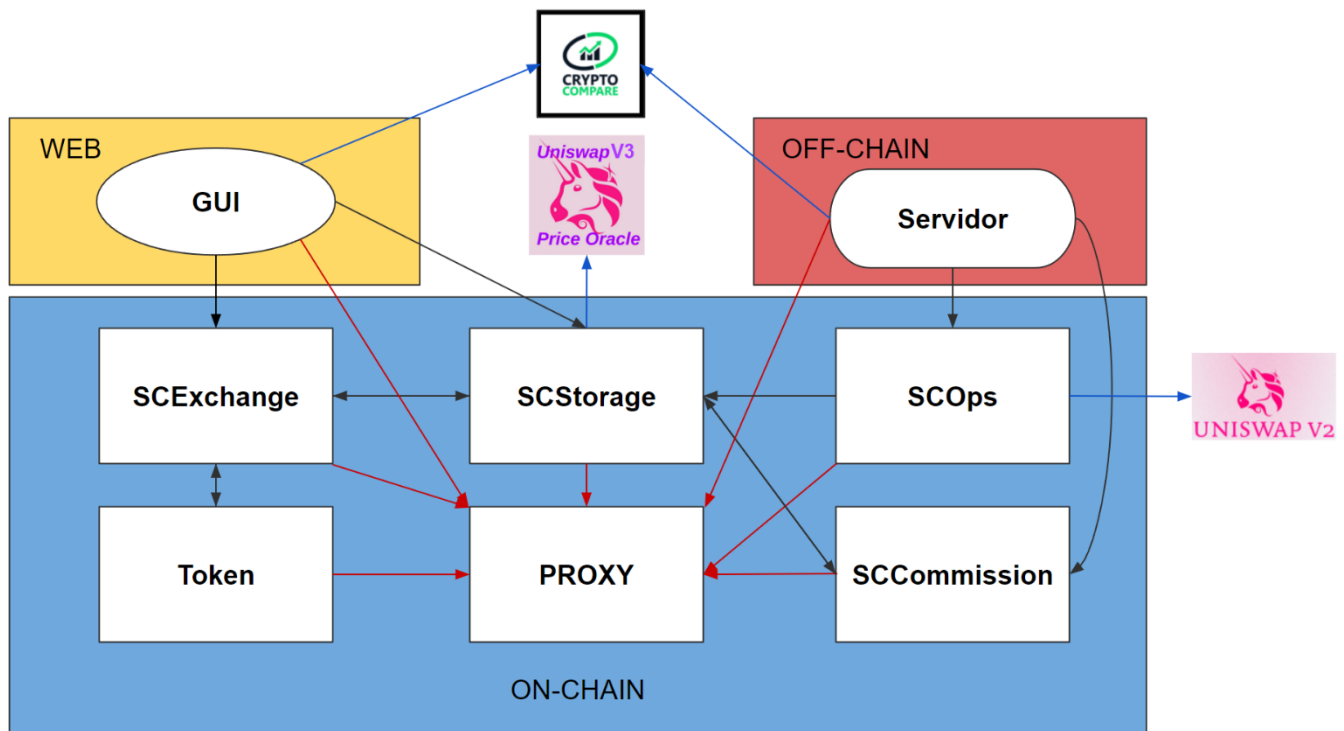


Fig. 1 Arquitectura del fondo de inversiones CryptoFund

5.2. DISEÑO Y OPERACIONES DEL SERVIDOR OFF CHAIN

En el servidor off-chain se va a utilizar un componente que realice la consulta de los precios de apertura y cierre de las criptomonedas que forman parte del fondo. Esta consulta de precios se realizará periódicamente en intervalos de 6 horas, mediante el uso de un planificador de tareas (en este caso, Cron). Cuando tengamos estos datos, los volcaremos a un archivo para utilizarlos más adelante.

El servidor off-chain implementa una estrategia básica:

- Si en el intervalo de 6h, el cierre ha sido menor a la apertura, predeciremos que el valor de la moneda va a aumentar, con lo cual operaremos en largo (compramos).
- Si en el intervalo de 6h, la apertura ha sido menos que el cierre, predeciremos que el valor de la moneda va a disminuir, con lo cual tendremos que operar en corto (vendemos).

Esta estrategia simula una estrategia más sofisticada que es la que se implementaría en un producto real. De esta manera, con cortos y largos dependiendo de la tendencia, buscaríamos aumentar nuestro NAV y dar más valor a nuestro fondo.

Para comprar y vender, utilizaremos UniswapV2, uno de los DeX más grandes de Ethereum. Cuando la estrategia determine que hay que realizar una compra, se cambiarán las stablecoins almacenadas por la moneda que corresponda, y cuando la estrategia determine que es necesario vender, si hubiese activos, se venderían y almacenarían en stablecoin para conservar su valor (si no hay fondos de una moneda en concreto en ese momento, como en estos exchanges no se puede realizar una operación en corto, no haríamos nada).

5.3. DISEÑO DE CONTRATOS ON CHAIN

Los contratos que forman el subsistema on-chain han sido desarrollados con la intención de que puedan ser fácilmente actualizables e independientes. Debido a la inmutabilidad de las redes blockchain, una vez desplegado un contrato, no puede ser cambiado. De esta manera, si en algún momento algún contrato necesita ser actualizado, todo el sistema debe volver a desplegarse. Para evitar esto, y que se puedan actualizar los contratos cuando sea necesario, CryptoFund hace uso de un contrato **Proxy** que almacena las direcciones de los contratos que componen el sistema on-chain. De esta manera, cada llamada que necesite hacerse pasa a través de este contrato que actúa de intermediario y redirige a la dirección que tiene almacenada.

El uso de un proxy tiene limitaciones, ya que en caso de que se añadan funciones que no estén presentes en las interfaces de los contratos llamados, los contratos que implementan estas interfaces no podrán llamar a las funciones añadidas. Esto podría arreglarse mediante el uso de una función que reciba el selector de la función como parámetro, y los parámetros codificados en un array de bytes y que realice una llamada de bajo nivel `call`, `delegatecall` o `staticcall` según corresponda. Esta solución sería posible, ya que estas instrucciones solo necesitan la dirección, pero para este trabajo no se ha implementado, ya que, esta solución tiene ciertos problemas y su implementación no entra dentro de los objetivos de este trabajo.

Una parte importante a considerar en el diseño son los modificadores de funciones de contrato que hemos utilizado. Los modificadores son comprobadores que se evalúan al principio de la función y solo si se

cumplen se ejecuta la función. Si no se cumple se revierte la ejecución, y falla la transacción. Hemos usado los siguientes modificadores:

- **OnlyDeployer:** Este modificador se encarga de comprobar si el usuario que quiere ejecutar la función es el propietario del contrato.
- **Blacklisted:** Se encarga de comprobar que el usuario está en la lista negra, en caso de serlo no puede entrar.
- **Coin (address stableAddr):** Dependiendo de la stablecoin con la que nos quieran pagar que recibe por parámetro, se comprueba que se acepta ese tipo de moneda.
- **Locked:** Comprobamos que el contrato no esté cerrado con una cerradura (lock), que se puede acceder a él.
- **NonInit:** Comprueba si el contrato está inicializado o no.

Todos los contratos tienen varias funciones en común que son de vital importancia y desempeñan la misma funcionalidad;

- **updateProxy (address newAddr) OnlyDeployer:** función que actualiza la dirección del proxy con la nueva dirección que se indica por parámetro, por si se ha tenido que eliminar el anterior y desplegar uno nuevo. Solo la puede ejecutar el creador del contrato.
- **deleteContract (address newContract) OnlyDeployer:** Función que se encarga de eliminar el contrato y crear uno nuevo con la dirección que se proporciona. Se utiliza para desplegar una versión nueva de un contrato. Esto solo lo puede realizar el creador del contrato.
- **receive():** Función que se utiliza para rechazar transferencias de ether al contrato. Este tipo de transferencias es incorrecto porque nuestros tokens solo se pueden adquirir comprándolos en la página web, pues con esta función lo que hace es revertir la ejecución ya que nuestro propósito es solo tener el dinero de los tokens en el sistema y evitar tener cualquier otro tipo de fondos que no sea controlado por nosotros.
- **fallback():** Función que se ejecuta automáticamente cuando algún usuario llama a cualquier función no válida en el contrato. Revierte la ejecución.

Distinguimos los siguientes Smart Contracts en nuestro sistema:

5.3.1. PROXY

Como se ha indicado anteriormente, se utiliza un contrato Proxy que almacena las direcciones de los contratos utilizados. De esta manera, los usuarios no tienen que preocuparse por donde está su dinero, ya que la web toma las direcciones de este Proxy, y los programadores pueden acceder a estas direcciones leyendo dicho contrato. Cada contrato, cuando se destruye, envía los fondos e inicializa los campos del contrato nuevo. Las direcciones de cada contrato pueden cambiar si son llamadas por dicho contrato, operación que ocurre solo cuando este se destruye.

Además de esta funcionalidad, el proxy implementa una función ***multicall***. Esta función permite combinar varias llamadas en una, ahorrando los 21000 de gas base la cual recibe dos parámetros, un array de bytes con las llamadas

5.3.2 SCEXCHANGE

Smart contract encargado de realizar la compra y venta de tokens que quiera el usuario. Este contrato está conectado con nuestra interfaz de página web, es con el que interactúan los usuarios para realizar sus movimientos. En él distinguimos las siguientes funciones principales:

- **initContract (address proxyAddr) OnlyDeployer NonInit:** Función que inicia el contrato, se encarga de pasarle la dirección del proxy al contrato, solo se puede ejecutar una única vez por el creador del contrato.
- **lockOn() OnlyDeployer:** Función que solo puede ejecutar el creador, se encarga de bloquear el contrato, para que nadie pueda acceder a las funciones del contrato. Se utiliza para realizar tareas de mantenimiento del sistema.
- **lockOff() OnlyDeployer:** Función contraria a la anterior, se encarga de desbloquear el contrato para que puedan acceder los usuarios de nuevo. Solo la puede ejecutar el creador
- **buyTokensOutput (address stableAddr, uint tokensOut) Blacklisted Locked Coin(stableAddr):** Función que se encarga de

realizar una compra con la stablecoin elegida y una cantidad de tokens también elegida. El usuario que ejecute esta función tiene que estar fuera de la Blacklist, y el contrato no puede estar cerrado con lock. Si los datos proporcionados son correctos se realiza correctamente la compra y se transfiere la cantidad de tokens desde nuestro contrato de Token a la cuenta del usuario, se guarda en el contrato de Storage el dinero que ha usado para pagar los tokens el usuario, y se realiza el apunte contable para calcular la comisión cuando quieran vender.

- **buyTokensInput (address stableAddr, uint qtyIn) Blacklisted Locked Coin(stableAddr):** Función que también es igual a la anterior solo que en vez de proporcionar el número de tokens se pasa directamente el dinero en dólares que quieren aportar y la función calcula los tokens que le corresponden para después enviarlos a la cuenta del usuario.
- **sellTokens (uint qty) Blacklisted Locked:** Función encargada de vender los tokens indicados por el usuario como parámetro, no la pueden ejecutar usuarios bloqueados del sistema y tampoco si está cerrado el contrato con el lock. Esta función llama al contrato de comisiones para que calcule el NAV y reste la comisión que debe pagar al usuario dependiendo del tiempo que haya depositado su dinero en el sistema, y transfiere en stablecoin la cantidad que corresponde desde el contrato de Storage que almacena todos los fondos.
- **addCoin (address coinAddr, uint stable) OnlyDeployer:** Función para agregar una nueva moneda stablecoin con la que puedan pagar los usuarios. Solo el creador del contrato puede hacer esto.
- **delCoin (address coinAddr) OnlyDeployer:** Función contraria a la anterior, solo el creador del contrato puede eliminar una moneda del fondo.

5.3.3. SCSTORAGE

Este contrato funciona a modo de “banco” ya que se encarga de almacenar todos los fondos que han ido pagando los usuarios para invertir en nuestro sistema. Por lo tanto, si cualquier contrato quiere depositar o sacar fondos tiene que realizarlo a través de este contrato. En él distinguimos las siguientes funciones:

- **initContract (address proxyAddr) OnlyDeployer NonInit:** Función que inicia el contrato, se encarga de pasarle la dirección del proxy al contrato. Solo se puede ejecutar una única vez por el creador del contrato.
- **updateNAV (uint comission):** Se encarga de actualizar el NAV restando la comisión cobrada que se pasa por parámetro y de devolver el nuevo valor de este. Esta función solo la ejecuta el contrato de Exchange cuando van a realizar una venta.
- **transferFunds (address to, uint totalToTransfer):** Función que se encarga de transferir los stable al vender sus tokens. Recibe como parámetros la dirección a la que hay que enviar el dinero y el total. Esta función está restringida al contrato de Exchange y es llamada cuando se va a realizar una venta.
- **oracleNAV():** Función a la que llama el servidor para saber cuál es el valor del NAV y de la última vez que se hizo un apunte contable para calcular las comisiones.
- **calculateNAV():** Función interna (solo se puede llamar desde dentro del contrato) que se encarga de calcular el valor del NAV y devolverlo actualizado.
- **addCoin (address coinAddr, uint stable) OnlyDeployer:** Función para agregar una nueva moneda, ya sea stablecoin o criptodivisa. Solo el creador del contrato puede hacer esto.
- **delCoin (address coinAddr) OnlyDeployer:** Función contraria a la anterior. Solo el creador del contrato puede eliminar una moneda del fondo.

5.3.4. SCCOMMISSION

Este contrato se encarga de calcular las comisiones que deben cobrarse en función del tiempo y almacenarlas, transfiriéndolas desde el contrato **Storage** a este contrato. En esta versión de CryptoFund, la comisión ha sido fijada en un 5% anual, aunque esta es parametrizable y puede cambiar a lo largo del tiempo. Cada vez que un usuario realiza una operación de compra y venta, se calculan las comisiones desde la última vez que se realizó una operación y se cobran. Para ello se tienen las siguientes variables y funciones:

- **uint feeYear:** comisión anual (5%) multiplicada por 1e18 (para ajustar los decimales ya que en Solidity no existen). Esta variable tiene un *setter*, por si decidimos cambiar la comisión.
- **uint r:** comisión anual ajustada a un día, multiplicada por 1e18. Esta variable tiene un *setter*, por si decidimos cambiar la comisión.
- **uint lastUpdated:** almacena la marca de tiempo de la última vez que se cobró la comisión.
- **uint accumCommission:** almacena las comisiones acumuladas, las cuales solo se cobran con una operación de venta. Siempre que la operación realizada por el usuario sea de compra, solo añadiremos la cantidad a cobrar a esta variable, y se cobrará luego (de esta manera se ahorran transferencias, ahorrando gas).
- **function initContract(address proxyAddr, uint accumCommission, uint feeYear, uint r) OnlyDeployer NonInit:** Función que inicia el contrato, se encarga de pasarle la dirección del proxy al contrato y de inicializar las variables de comisión acumulada (será 0 si es el primer contrato que se crea, y el valor del contrato antiguo si se ha destruido uno antiguo y se despliega uno nuevo), la comisión anual y la comisión anual en término de un día. Solo se puede ejecutar una única vez por el creador del contrato.
- **function computeCommission() internal view returns (uint):** función donde se calcula la comisión a cobrar hasta el momento desde la última vez que se cobró. Como para calcular el interés compuesto necesario para cobrar la comisión se necesita el uso de decimales, debido a que Solidity no soporta los decimales, se ha calculado la comisión mediante el desarrollo de Taylor. Devuelve el valor de la comisión calculada.
- **function payCommissions(uint buyOrSell) external returns (uint):** función que ejecuta la transferencia de las comisiones si la operación que la llama es una venta y que suma al acumulado de la comisión si la operación es una compra. Esta función llama a **computeCommission()** para calcular cuánto hay que cobrar. Devuelve la cantidad total de comisiones a cobrar.
- **function collectCommission(address[] tokensToTransfer, uint[] qty) external OnlyDeployer:** función que permite al dueño de los contratos transferir a su cuenta los fondos acumulados en el contrato de comisiones. Tiene dos arrays como parámetros: el primero, **tokensToTransfer** que contiene las direcciones de los tokens que queremos transferir, y el segundo, **qty** que contiene las cantidades de dichos tokens.

5.3.5. SCOPS

Este contrato es el que realiza las interacciones con los sitios de exchange que se utilizarán para comprar y vender los distintos tokens que compondrán el fondo. Para ello se conectará el contrato con Uniswap, y se permitirá al contrato de operaciones tener acceso a los fondos del contrato **Storage**. Para operar se hará uso de las siguientes funciones:

- **function initContract(address proxyAddr) OnlyDeployer NonInit:** Función que inicia el contrato, se encarga de pasarle la dirección del proxy al contrato, solo se puede ejecutar una única vez por el creador del contrato.
- **function uniSwap(address[] path, uint amount) OnlyDeployer:** esta función conecta con los contratos de UniswapV2. Utilizando los contratos de las pools y las librerías necesarias, realiza los intercambios siguiendo la ruta del parámetro **path**, y tomando como input la cantidad **amount**. Las monedas intercambiadas son depositadas en **Storage**.

5.3.6. ANÁLISIS DE VULNERABILIDADES

En cualquier sistema financiero es fundamental analizar los posibles riesgos y en particular los sistemas on-chain, pues el código de byte de los contratos es accesible públicamente que. Para ello hay que tener en cuenta los distintos tipos de ataque por si hay algún intento de robo de fondos. Por lo tanto, hemos realizado un análisis, y posteriormente nos hemos encargado de asegurarlo contra los ataques más relevantes en sistemas Blockchain. Los tipos de ataque analizados son [22]:

- **Ataque de reentrada:** Este ataque ocurre cuando una función hace una transferencia de su contrato a uno externo del que no se tiene confianza. Ese contrato, al recibir los fondos, puede ejecutar una función que vuelva a llamar al contrato inicial y hacer una retirada antes de que se haya actualizado el valor de su balance a tiempo, haciendo así una retirada de dinero cuando en realidad no podría. Este ataque ha provocado grandes pérdidas de fondos como el conocido ataque DAO de 2016 [23].
En CryptoFund para evitar este tipo de ataque no se opera con ether sino con tokens únicamente y no se hace ninguna llamada a un contrato externo por lo tanto no se pueden realizar ataques de reentrada en el sistema.

- **Overflows:** Ataque que consiste en enviar un valor bastante grande a un contrato, cerca del máximo soportado por Solidity, y si al realizar cualquier operación se sobrepasa este máximo el programa lo que hace es automáticamente empezar a contar desde el principio.

Esta vulnerabilidad se puede resolver de dos formas, o usando la librería de SafeMath que se encarga de comprobar que no haya ningún overflow o usando una versión del compilador mayor o igual que la 0.8.0 en la que el compilador añade automáticamente comprobaciones de overflow en todas las operaciones aritméticas. En este proyecto se ha optado por usar una versión mayor o igual que 0.8.0 para evitarlo.
- **Underflow:** En Solidity hay tipos de variables que no tienen signo, por tanto, si se realiza una operación que tiene un resultado negativo al no soportarlo producen un resultado positivo incorrecto. Para evitar este problema se plantean también las mismas dos soluciones que en el anterior: utilizar la librería SafeMath o utilizar una versión de las últimas, mayor o igual que la 0.8.0.
- **Delegatecall:** El ataque consiste en un tipo de llamada de bajo nivel, que se llama “delegatecall”, en la que es posible suplantar el emisor. Esto ocurre cuando un contrato A llama a una función de otro contrato externo B mediante delegatecall y B llama desde esa función a otro contrato aparece que el emisor es A y eso lo puede usar el atacante para ejecutar funciones que solo puede ejecutar A. Para evitar estos problemas hemos implementado un array que comprueba si el contrato que ejecuta las funciones es de confianza y todas las llamadas de delegatecall que realizamos en el sistema son a funciones de contratos de CryptoFund, nunca son a contratos que desconocidos.
- **tx.origin:** Solidity tiene una variable global llamada “tx.origin” que al igual que “msg.sender” se puede usar para identificar la dirección de los usuarios que ejecutan funciones con la diferencia que tx.origin se queda con la dirección que inició la transacción: Si A empieza y llama a una función del contrato B, luego B llama a una función del contrato C a C le llega que msg.sender = B mientras que tx.origin = A, se queda solo con el del origen. No se recomienda el uso de esta variable para identificar quien ha llamado una función del contrato. La mejor opción es usar msg.sender, ya que con tx.origin puede un atacante usar la dirección del creador de un contrato para llamar a funciones que realicen la autorización de entrada mediante tx.origin

y entrar a funciones exclusivas del creador como puede ser eliminar un contrato.

- **Denegación de servicio:** Con este tipo de ataque se intenta que un Smart contract se convierta en inoperable durante un periodo de tiempo o permanentemente y puede bloquear los fondos que están almacenados en este contrato. Por ejemplo, si un atacante intenta crear arrays excesivamente grandes para forzar que se consuma el máximo valor de gas que se puede consumir por función, entonces se bloquea el contrato y pasa a ser inoperable. Generalmente para solucionar este problema se puede limitar el uso de gas de la función para evitar que se quede bloqueado el contrato. En CryptoFund no se permite que los usuarios introduzcan un array como parámetro de entrada, solo pueden pasar enteros que corresponden con lo que quieran comprar o vender. Tampoco se llama a contratos externos que pueden gastar todo el gas que queda de la función para bloquearlo.

5.3.7. OPTIMIZACIÓN DE CONSUMO DE GAS

Es importante en las blockchains EVM el desarrollo de contratos cuyo código sea lo más eficiente posible. El gas asociado a las computaciones necesarias se traduce en costes monetarios, con lo cual es muy importante conseguir el menor coste de gas posible para cada contrato que desarrollemos.

Para ello hemos usado una serie de técnicas, basadas en la forma en la que el compilador genera el código ejecutable EVM:

- Almacenar variables de la región de memoria persistente del contrato (storage) en variables locales de pila (stack) si se va a leer repetidas veces: Los accesos a storage son mucho más caros que los accesos a las variables locales de pila. Si se necesita acceder múltiples veces a una variable de storage, es mucho más barato guardarlo en una variable local. También para realizar una escritura, si es necesario realizar varias operaciones distintas con el valor de esa variable, es más barato almacenarlo en una variable de stack, y asignar el valor final, ya que las asignaciones a storage también son caras.
- Utilizar parámetros calldata en lugar de memory: calldata es una región donde se almacenan los datos de las llamadas externas

como un array de bytes a los que se puede acceder con el uso de un offset. De esta manera, se pueden leer los parámetros de entrada sin necesidad de copiarlos a memoria, pues se consumiría más gas en el proceso. Esta técnica solo se puede utilizar cuando no se vayan a modificar los parámetros dados, ya que calldata es una región de solo lectura.

- Utilizar funciones external en lugar de public siempre que tengan parámetros: las funciones external, aunque solo se pueden llamar desde fuera del propio contrato, leen sus parámetros desde calldata que como hemos visto antes es más barato, mientras que las funciones public, aunque pueden ser llamadas desde dentro o fuera del contrato, necesitan copiar los parámetros a memoria, ya que no saben si son llamadas desde funciones externas o internas.
- Utilizar funciones internal siempre que sea posible: las funciones internal solo pueden ser llamadas desde el propio contrato o contratos derivados, pero, son mucho más eficientes que las funciones public o external, ya que en el código EVM ejecutable no se utiliza una llamada (los opcode CALL, DELEGATECALL, STATICCALL) que consume una cantidad considerable de gas. Para llamar a estas funciones, en cambio, se realiza un salto en el código (JUMP) que apenas consume gas.
- Fijar las funciones como view (funciones constantes, que no pueden modificar el estado del contrato) o pure (funciones puras, las cuales no pueden modificar ni leer el estado del contrato): de esta manera, el compilador traduce las llamadas con el opcode STATICCALL en lugar de CALL. STATICCALL es más barato y se asegura de que no cambia el estado, y en caso de que sí, revierte la ejecución. También es útil para mitigar vulnerabilidades.
- Hacer uso de la llamada de bajo nivel de Solidity delegatecall siempre que se pueda: teniendo en cuenta las consideraciones de seguridad necesarias (msg.sender y msg.value van a ser conservados entre delegatecall al contrario que entre call normales), podemos aprovechar este tipo de llamadas, ya que ahorran gas con respecto a las llamadas normales, y además facilitan implementar ciertas funcionalidades en contratos modulares (por ejemplo, llamar a una función que solo pueda ejecutar owner desde otra función que solo pueda ejecutar

owner, ahorrando el tener que hacer dos llamadas separadas). Es útil para "empaquetar" varias transacciones que normalmente se realizarían en varios pasos, gastando el gas base de cada llamada más de una vez.

- Empaquetado de transacciones: permite agrupar varias transacciones en una misma, y ahorrar el coste de gas base por cada transacción (21000) y de esta manera pagarlo una sola vez y no varias. Lo hemos implementado mediante la función ***multicall***.
- Empaquetado de variables de storage: declaración adyacente de variables de storage de tipos que Ethereum pueda empaquetar juntos en una misma palabra. Por ejemplo, si declaramos 32 variables uint8 juntas, es decir, 32 enteros sin signo de 8 bits, Ethereum puede empaquetarlos en la misma palabra de 256 bits, ahorrando así espacio, y, por lo tanto, gas. Esta técnica no puede ser aplicada en variables locales ni de memoria, y, de hecho, es más barato utilizar un entero que ocupe una palabra completa a un uint8 en este caso. Esto es debido a que en memoria o en la pila, la máquina virtual necesita realizar operaciones adicionales para saber cuánto ocupa dicha variable en la palabra, y con una variable que ocupa una palabra entera no es necesario.
- Vaciado de variables de storage siempre que sea posible: cuando se añaden ceros a una variable de storage, Ethereum recompensa por liberar memoria, devolviendo una cantidad de gas basada en el número de ceros que se hayan añadido, y por lo tanto cuanto espacio de storage se haya liberado.
- Realizar operaciones antes de hacer selfdestruct: la operación de eliminar un contrato devuelve gas, ya que se está liberando el storage asociado a dicho contrato, con lo cual, si antes de borrarlo es necesario hacer alguna operación, saldrá más barata que haciéndola independientemente. Por ejemplo, en CryptoFund, cada vez que se destruye un contrato, se inicializa el siguiente y de esta manera, se ahorra gas.
- Utilizar estructuras de datos baratas siempre que se pueda: se priorizará el uso de mappings al uso de arrays de storage siempre que se pueda, ya que los mappings, aunque ocupan una cantidad de memoria considerable, están optimizados para poder acceder al valor de cada elemento con coste constante, mientras que,

para acceder a un valor de un array, se necesitaría realizar una búsqueda si no se sabe su índice, siendo el coste lineal.

- Programar los requires lo antes posible: en caso de fallo, las operaciones que se revierten solo consumen el gas utilizado hasta el momento, con lo cual, conviene hacer las comprobaciones sobre si hay que revertir o no al principio, de esa manera se habrá consumido la menor cantidad de gas posible.
- Hacer uso de librerías si necesitamos reutilizar código en varios de nuestros contratos: en caso de que necesitemos repetir ciertas funcionalidades, usaremos librerías, ya que las funciones a estas se llaman mediante DELEGATECALL, que son más baratas que llamadas normales, y se reducirá el número de bytes que ocupa la aplicación en la blockchain, reduciendo así el coste de gas de desplegar los contratos.

5.4. DISEÑO DE INTERFAZ DE USUARIO

5.4.1. EVOLUCIÓN DE LA INTERFAZ

Tras visitar una serie de fondos de inversión y aplicaciones web relacionadas con el mundo de las criptomonedas, se decidió que la interfaz de usuario debería tener un diseño muy limpio y sin sobrecarga de datos. Se realizó un primer prototipo de la interfaz de usuario mediante el uso de la herramienta Balsamiq [24], con el objetivo de tener un primer diseño base antes de comenzar con el desarrollo de la interfaz, como se muestra en la figura 3.

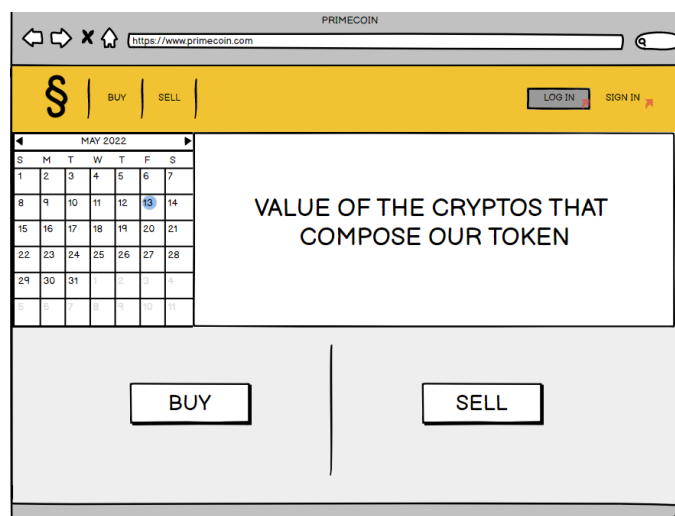


Fig. 3 Prototipo de la aplicación en Balsamiq

Una vez acabado el prototipado en Balsamiq, el diseño fue enseñado a todos los miembros del grupo, y se comenzó con su desarrollo. El diseño inicial constaba de 4 páginas independientes: la pantalla de inicio, de compra, de venta y de “Log In”. Además, en estas pantallas se muestra el precio de las criptomonedas de las que está compuesto el token en tiempo real, así como el precio del propio token. Una gran parte de este primer diseño se descartó, dando paso a un segundo diseño que se convirtió en el definitivo. Los cambios principales respecto al primer diseño son la reducción de pantallas de 4 a 1, en la que se pueden realizar todas las acciones mencionadas en el diseño del primer modelo; el cambio del “Log In”, que iba a depender de una base de datos por un “Log In” externo mediante una cartera (Wallet) y la implementación de dos botones que redirigen al repositorio de GitHub donde está el código del proyecto y a la documentación. Esta interfaz se muestra en la Figura 4.

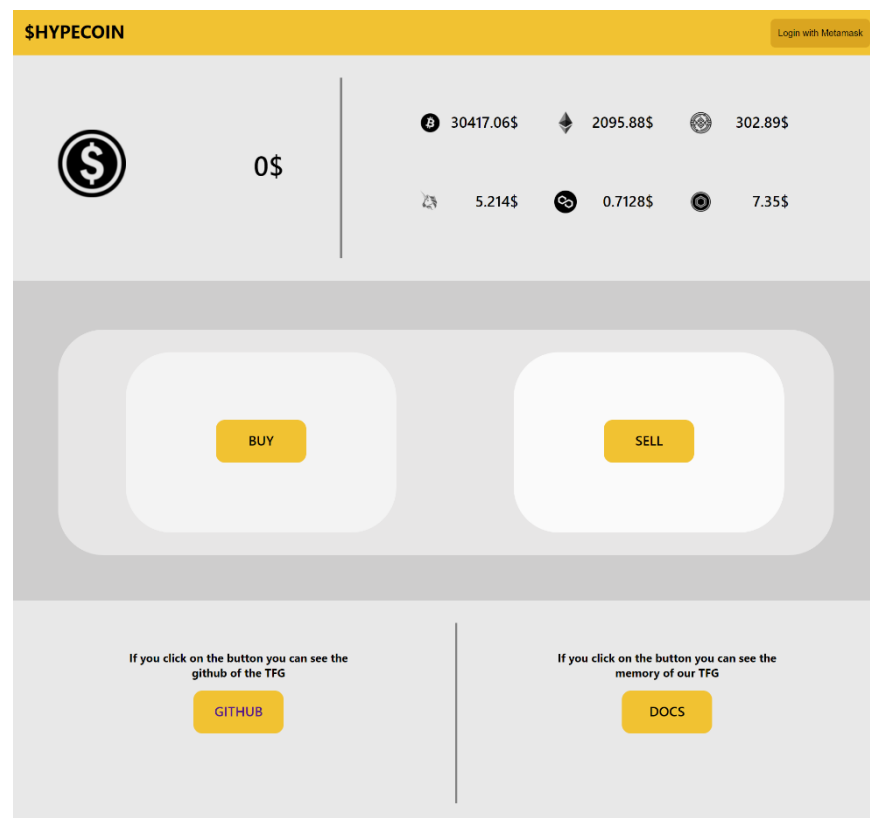


Fig. 4 Pantalla completa de la interfaz de usuario

5.4.2. DESCRIPCIÓN DETALLADA DE LA INTERFAZ DE USUARIO

A continuación, se detallan las acciones que se pueden realizar en toda la interfaz de usuario, así como cada uno de los componentes relevantes de la interfaz.

- Log in con Metamask: este botón, situado en la parte superior derecha de la pantalla, comprueba si la extensión de Metamask está instalada en el navegador, y actúa de dos formas dependiendo de si está instalada o no. Si la extensión no está instalada el botón se pone en gris, cambia el texto del botón a un texto que informa de que la extensión no está instalada y se deshabilita el botón (Figura 6). Por otro lado, en caso de que la extensión esté instalada, el botón estará habilitado (Figura 5), y al pulsar sobre él se abrirá una ventana de Metamask que permitirá ingresar las credenciales de Metamask. Si la identificación en Metamask se realiza correctamente, el texto del botón cambiará a “Sign out of Metamask” y mostrará en un nuevo contenedor el identificador de la cartera (Figura 7) del usuario; en caso de que el Log in no se realice correctamente el botón no cambia y sigue permitiendo realizar el Log in.

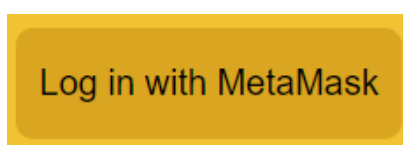


Fig. 5 Botón Metamask instalado

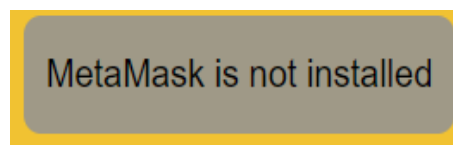


Fig. 6 Botón Metamask sin instalar

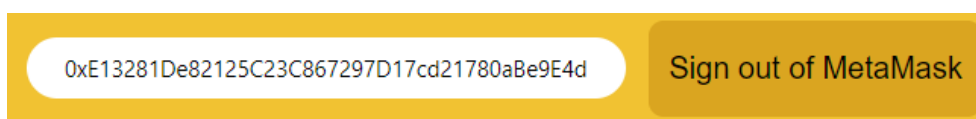


Fig.7 Botón con la dirección de la cuenta de Metamask

- Precio de las divisas: en este caso se diferencian dos casos: el precio nuestro token, que se obtiene llamando a los contratos (Figura 8) y el precio del resto de las criptodivisas (Figura 9). Esto se consigue consultando la API de cryptocompare [referencia a la web] y cada precio aparece situado a la derecha del icono propio de dicha criptomoneda.

Todos los precios de las criptodivisas se actualizan cada vez que se refresca la página.



Fig. 8 Precio Token propio

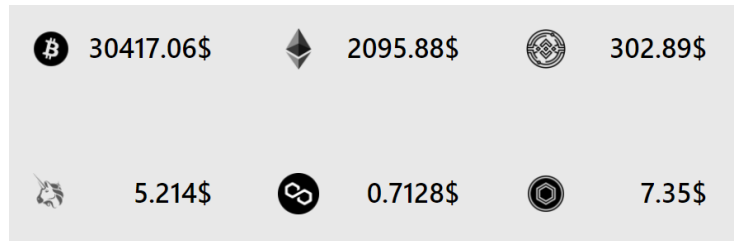


Fig. 9 Precio Tokens externos

- Botón de compra: al pulsar en el botón de compra (Figura 10) se abre una nueva interacción (Figura 11) en la que se muestra un texto en el que se explica cómo utilizar la interfaz de compra, un checkbox que permite cambiar entre introducir una cantidad de tokens o de dinero, un campo de entrada que permite introducir el número de tokens que quieres comprar o la cantidad de dinero que se desea usar para comprarlos (dependiendo de si el checkbox está marcado o no), un desplegable que permite seleccionar entre una serie de criptodivisas para realizar la compra y un botón de continuar y uno de cancelar la compra.

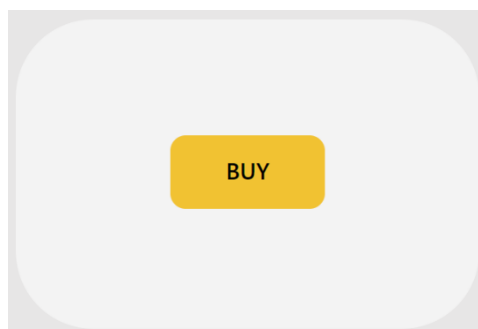


Fig. 10 Botón de compra

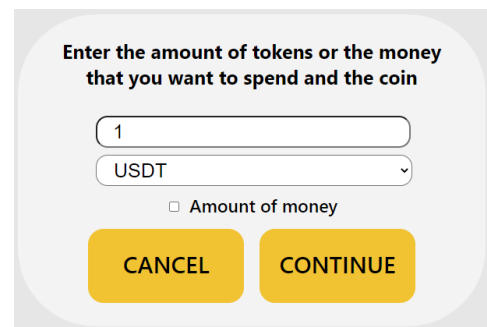


Fig. 11 Interfaz de compra

- Botón de venta: al pulsar en el botón de venta (Figura 12) se abre una nueva interacción (Figura 13) en la que se muestra un texto en el que se explica cómo utilizar la interfaz de venta, y un campo de entrada que permite introducir el número de tokens que se desea vender. En este caso no se muestra un

desplegable para seleccionar la criptomoneda en la que recibir el dinero porque las retiradas siempre se realizan en USDT. Por último, se muestran dos botones, uno para continuar y otro para cancelar la acción de venta de tokens.

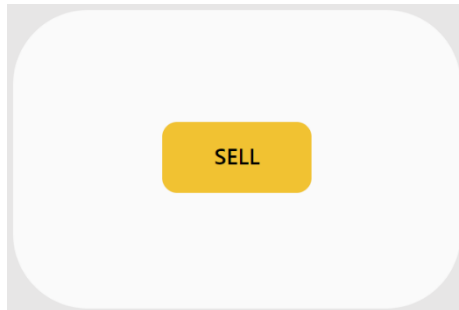


Fig. 12 Botón de venta

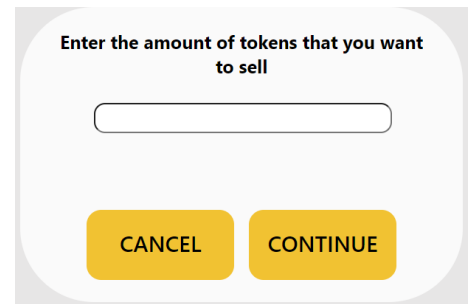


Fig. 13 Interfaz de venta

- Botones de Documentación y Github: estos dos botones realizan la misma acción, pero redirigiendo a sitios distintos, el botón de "Docs." (Figura 14) redirige a la memoria del Trabajo de Fin de Grado y el botón de "Github" (Figura 15) redirige a un repositorio en Github dónde está todo el código referente a la interfaz de usuario y a los contratos del proyecto.

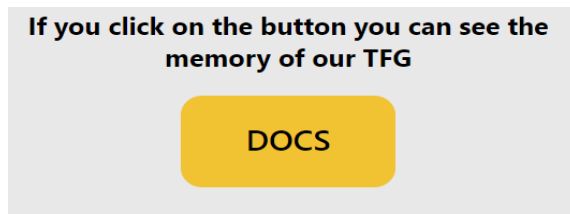


Fig. 14 Enlace a documentación

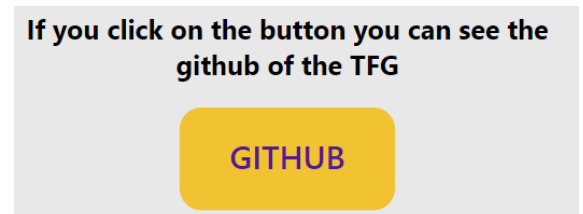


Fig. 15 Enlace Github proyecto

5.4.3. OTRAS INTERFACES

Como se ha mencionado anteriormente, antes de realizar el diseño de la interfaz de usuario se realizó una reunión en la que se analizaron las interfaces de usuario de otras páginas web con una finalidad similar a la de este proyecto. Las interfaces seleccionadas fueron la de Crypto20 [18], Uniswap [21], Paxos [25], Stoic [26], Metamask [27], Curve [28], Pancake [16], Blackrock [29], e Interactive Brokers [30].

- CRYPTO20: al ser uno de los pocos fondos de inversión indexados de criptodivisas, Crypto20 tenía que ser una referencia obligatoria para el desarrollo de nuestra interfaz de usuario. En este caso, su interfaz de usuario pide realizar un “Log In” previo antes de poder acceder a la aplicación. La interfaz está compuesta de 4 pantallas distintas: “Dashboard”, “Wallets”, “Invest” y “Performance”. Todas ellas contienen la información necesaria para poder realizar operaciones dentro de la aplicación y con un estilo simple y uniforme.

La página de inicio (Dashboard) incluye noticias diarias, así como recordatorios, un contenedor con las carteras a las que tiene acceso y los movimientos de los precios del fondo. La pantalla de “Wallets” permite al usuario acceder a las diferentes carteras y permite realizar las acciones de invertir, retirar y recibir. La pantalla de “Invest” muestra las distintas opciones de inversión que ofrecen: Bitcoin Alpha, Margin Lending, Crypto10 Hedged, Crypto20, Hyperion VC y Global Tech Plus. La pantalla de “Performance” permite realizar un seguimiento del rendimiento de la cartera del usuario, viendo sus rendimientos en tiempo real.

- UNISWAP: la interfaz de usuario de Uniswap se caracteriza por ser muy simple y mostrar muy poca información, para de este modo no saturar al usuario. Permite la conexión con carteras a través de Metamask, WalletConnect, Coinbase Wallet y Fortmatic, lo que es muy importante, ya que esta aplicación tiene como objetivo principal poder realizar intercambios de una criptodivisa a otra.

La interfaz dispone de 4 opciones: “Intercambiar”, “Fondo común”, “Votaciones” y “Gráficos”. La página de inicio (Intercambiar), muestra un único contenedor que permite

seleccionar el token desde el que se quiere realizar el intercambio, el token que se quiere recibir y la dos posibles entradas de texto, para introducir el número de tokens que se quieren intercambiar en una de las dos criptodivisas seleccionadas. La pantalla de “fondo común” permite observar los ingresos obtenidos de los préstamos de liquidez. La pantalla de “Votaciones” permite realizar votaciones mediante el uso de las dichas UNI. Y la pantalla de “Gráficos” redirige a otra página externa en la que se muestran gráficos con la evolución del precio de diversas criptodivisas.

- **PAXOS:** la interfaz de Paxos está mucho más saturada de información que las dos anteriores, tiene una página principal con un menú que permite acceder a los distintos productos que ofrecen divididos en 3 categorías “Crypto for Enterprise”, “Crypto for investors” y “Settlement for Institutions”. Tanto en los productos de la categoría “Crypto for Enterprise” como los de “Settlement for Institutions” la pantalla muestra información sobre el producto y un botón que permite contactar con ellos. Por otro lado, los productos de la categoría “Crypto for investors” muestra información sobre los productos y da la opción de crear o iniciar sesión en su aplicación de inversiones.
- **OTRAS INTERFACES:** las interfaces de Stoic, Metamask y Pancake eran muy parecidas a la de Uniswap, interfaces muy simples e intuitivas. Lo más destacable de la interfaz de Curve es que apuesta por un estilo retro que la hace destacar, ya que las otras usan diseños más actuales. Por otro lado, Blackrock e Interactive Brokers son páginas web relacionadas con inversiones financieras al uso, por lo que se incidirá más en ellas.

5.5. HERRAMIENTAS UTILIZADAS

Los contratos los hemos desarrollado en el lenguaje Solidity, el lenguaje más utilizado con diferencia en las redes EVM. Hemos utilizado la última versión disponible a día de hoy (0.8.14) y hemos activado la optimización para 5000 ejecuciones, ya que como es una primera versión, esperamos un uso moderado y los contratos se irán actualizando.

El servidor off-chain será una máquina con Linux. Nos vendrá bien ya que haremos uso de la funcionalidad cron, que es un planificador que viene con el sistema operativo y que usaremos para programar la estrategia. En este servidor necesitaremos tener instalado Node.js con la librería ethers.js, y tener nuestra clave privada (preferiblemente en un archivo encriptado) para poder realizar transacciones desde nuestra cuenta, que será la que ha desplegado los contratos.

Programaremos con cron la consulta de los precios de las monedas utilizadas mediante la API de Cryptocompare. Tras la consulta, se realiza la ejecución de un script de JavaScript que enviará una transacción la para operar de la manera establecida.

Para el desarrollo de la interfaz de usuario se han utilizado las herramientas enumeradas a continuación:

- **XAMPP:** es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl. En este caso concreto solo se ha hecho uso del servidor web Apache, ya que nuestra aplicación web carece de base de datos y no emplea los lenguajes de script PHP y Perl.
- **VISUAL STUDIO CODE:** es un editor de texto de código abierto desarrollado por Microsoft. En cuanto a su uso en este proyecto, ha sido el editor de código elegido para el desarrollo de la interfaz de usuario.
- **LENGUAJES PROGRAMACIÓN WEB:** Los tres lenguajes de programación que se enumeran a continuación se han utilizado para el desarrollo de la interfaz de usuario, y se complementan entre ellos:

- HTML (HyperText Markup Language): es el componente más básico de las webs y se usa para definir el significado y la estructura del contenido web.
 - CSS (Cascading Style Sheets): es un lenguaje de estilos utilizado para describir la presentación de documentos HTML. Este lenguaje se ha utilizado para dar un aspecto uniforme y atractivo a la interfaz de usuario.
 - JAVASCRIPT: es un lenguaje de programación que se utiliza como lenguaje de scripting para páginas web, este lenguaje es el que nos ha permitido realizar las conexiones con Metamask y los contratos, así como darle un mayor dinamismo a la aplicación web. Por otro lado, de no ser por el uso del JavaScript, no se podría haber hecho una interfaz completa y sin un exceso de sobrecarga de datos.
-
- BALSAMIQ: esta herramienta de prototipado está incluida en Google Drive y permite crear prototipos interactivos de manera que además de poder mostrar un diseño muy aproximado de la interfaz que se quiere implementar, también se pueden recrear las distintas interacciones que se desean realizar con dicha interfaz de usuario.
 - INSPECTOR DEL NAVEGADOR: esta herramienta viene incluida en todos los navegadores web, y permite acceder al código de la página. En este proyecto se ha utilizado como apoyo para la generación del código CSS y para depurar la interfaz de usuario, ya que además de acceder al código permite colocar puntos de parada en el código JavaScript, lo que permite comprobar si dicho código se ejecuta como es esperado.
 - REMIX: Es un entorno de desarrollo integrado de desarrollo (IDE) basado en un navegador web que contiene un compilador y un entorno en tiempo de ejecución para Solidity. Se ha usado este entorno en el trabajo para programar y ejecutar los contratos en la testnet de Ethereum.

5.6. BIBLIOTECAS UTILIZADAS

- **ETHERS.JS:** La biblioteca ethers.js es una biblioteca completa y compacta para interactuar con la Blockchain de Ethereum y su ecosistema. Fue diseñada originalmente para su uso con ethers.io y desde entonces se ha ampliado a una biblioteca de propósito más general. En nuestra aplicación hemos utilizado esta biblioteca para conectar la interfaz de usuario con los contratos.
- **WEB3:** es un nuevo tipo de servicio de internet construido utilizando cadenas de bloques descentralizadas, es decir, los sistemas de registro compartido que utilizan criptomonedas como Bitcoin y Ethereum. En nuestro caso, hemos utilizado web3 junto con la API de Metamask para realizar la conexión con la cuenta de Metamask y poder obtener información de las cuentas de la cartera de Metamask.
- **API CRYPTOCOMPARE:** es una plataforma que se especializa en proporcionar a los usuarios información en tiempo real en las tendencias del mercado. En este proyecto hemos utilizado las API de Cryptocompare para mostrar los precios de ciertas criptodivisas en tiempo real cada vez que se refresca nuestra página.
- **API METAMASK:** La extensión de navegador Metamask hace las funciones de cartera para Ethereum y tokens ERC-20. Además, permite conectar con diferentes Apps y también permite acceder a otras blockchain seleccionadas y compatibles. En nuestro caso, hemos utilizado la API de Metamask junto con web3 para realizar la conexión con la cuenta de Metamask y poder obtener información de las cuentas de la cartera de Metamask.
- **ERC20 OPENZEPELIN:** La implementación de estos contratos es la más comunmente utilizada en cuanto a los tokens ERC20. Al no definir este estándar la implementación, no hay un modelo oficial de estos tokens, pero los desarrollados por OpenZeppelin son los que más han logrado fijar una implementación específica.
- **ORÁCULO UNISWAPV3:** Este oráculo permite consultar los precios de una moneda concreta on-chain. Una peculiaridad de los sistemas blockchain, es que no se pueden realizar llamadas

desde dentro del sistema a sistemas externos, con lo cual, no se podría hacer uso de una API externa para consultar on-chain de precios. Por ello en este trabajo se hace uso de esta librería, la cual nos da el precio de las monedas acorde al precio que tengan en la plataforma UniswapV3. Se utiliza para calcular internamente el NAV.

- **CONTRATOS Y LIBRERÍAS UNISWAPV2:** Se utiliza el sitio de intercambio UniswapV2 para transformar las stablecoins en tokens no estables y viceversa. Para ello, se requiere el uso de ciertos contratos de este DeX y sus dependencias, en forma de librerías adicionales, para realizar los intercambios.

6. DISPONIBILIDAD

En este capítulo se explica dónde se puede encontrar el código fuente de la interfaz de usuario, el servidor y los contratos de CryptoFund. Además, se han incluido instrucciones de instalación y de uso con una cartera Metamask."

6.1. UBICACIÓN Y DISTRIBUCIÓN

El prototipo de CryptoFund ha sido publicado en un repositorio público de GitHub:

<https://github.com/GonzaloMartinezBerzal/CryptoFund>

En él pueden encontrarse los siguientes directorios:

- **CONTRACTS:** en este directorio se encuentran todos los contratos de Solidity que forman la parte on-chain del proyecto. Además de estos contratos, están presentes las siguientes carpetas:
 - Interfaces: esta carpeta contiene las interfaces de los contratos.
 - Libraries: esta carpeta contiene las librerías utilizadas por los contratos.
- **UI:** en este directorio se encuentran los archivos correspondientes a la interfaz de usuario del proyecto, así como una carpeta llamada "img" donde están las imágenes utilizadas.
- **OFF-CHAIN SERVER:** en este directorio se encuentran los archivos correspondientes a la estrategia del servidor off-chain.

También se hace uso de algunas dependencias externas:

- **UniswapV2:** Los contratos UniswapV2Pair y UniswapV2Callee, y sus dependencias asociadas, disponibles en el repositorio público de UniswapV2-Core <https://github.com/Uniswap/v2-core>
- **UniswapV3:** El contrato OracleLibrary, y sus dependencias asociadas, disponible en el repositorio público de UniswapV3-Periphery <https://github.com/Uniswap/v3-periphery>

Además de los directorios mencionados anteriormente también se incluyen: un fichero README.md que incluye una pequeña descripción del repositorio y una pequeña guía de uso del prototipo; y un fichero LICENSE, correspondiente a la licencia del prototipo.

6.2. INSTRUCCIONES DE INSTALACIÓN Y DE CONFIGURACIÓN DE METAMASK

Debido a que el prototipo utiliza un servidor local, es necesario instalar una serie de herramientas que nos permitan probarlo.

- Instalar XAMPP.
- Una vez instalado, abrir XAMPP, en el panel de control de XAMPP, pulsar el botón de *Start* situado a la derecha de Apache.
- Pulsar el botón Explorer, situado a la derecha del panel de control de XAMPP
- Entrar en la carpeta de htdocs.
- Introducir los archivos y carpetas del directorio UI de Github en la carpeta htdocs.
- Buscar en el navegador <http://localhost/>.

Una vez ya esté configurado XAMPP hay que instalar MetaMask para poder comprar y vender tokens en la testnet de nuestro fondo:

- Descargar metamask desde el link <https://metamask.io/download/> como se ve en Fig.16
- Agregar la extensión al navegador como se ve en Fig.17
- Una vez añadida la extensión se nos abrirá una página para importar o crear una nueva cartera como se ve en Fig.18 Se seleccionará la opción de importar cartera.
- Se abrirá una pestaña que pregunta por la frase secreta ubicada en el archivo metamask.txt y se pedirá definir una contraseña como se ve en Fig.19. Esta contraseña solo sirve para restringir el acceso a la aplicación de cartera, pero no tiene ninguna relación con la dirección de Ethereum, la cual tiene su clave privada independiente.
- Después de introducir la frase secreta ya se tendrá acceso a la cartera como se ve en Fig.20.
- Hay que activar la opción de poder visualizar las redes de prueba para ello hay que seleccionar “Red principal de Ethereum” como se ve en Fig.21 y después se selecciona “Show/hide test networks”.
- Se activa la opción “Show test networks” como se ve en Fig.22

- Finalmente, se selecciona la pestaña de red y se fija la opción “Red de prueba Goerli” como se ve en Fig.23.

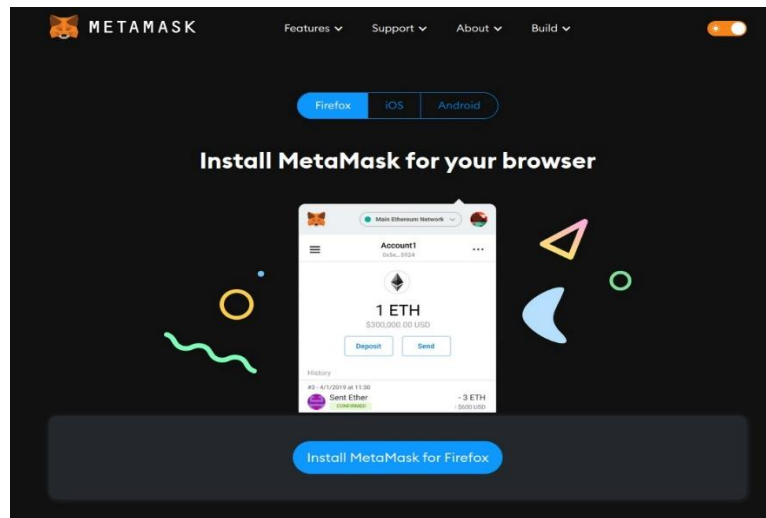


Fig.16. Página principal de MetaMask

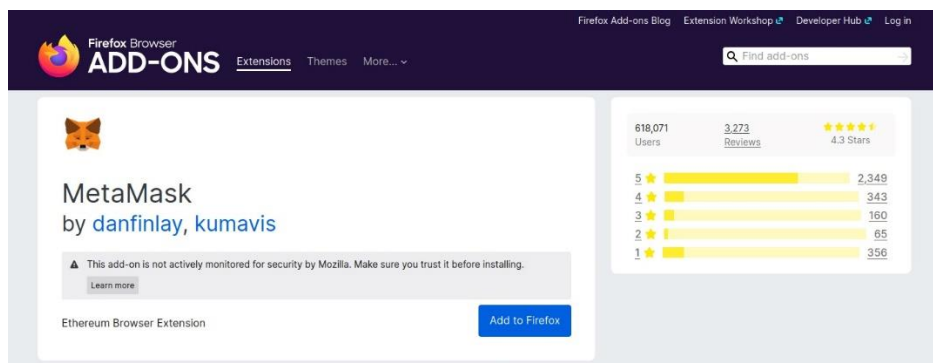


Fig.17. Página del Add-On de MetaMask para Firefox



¿Es nuevo en MetaMask?

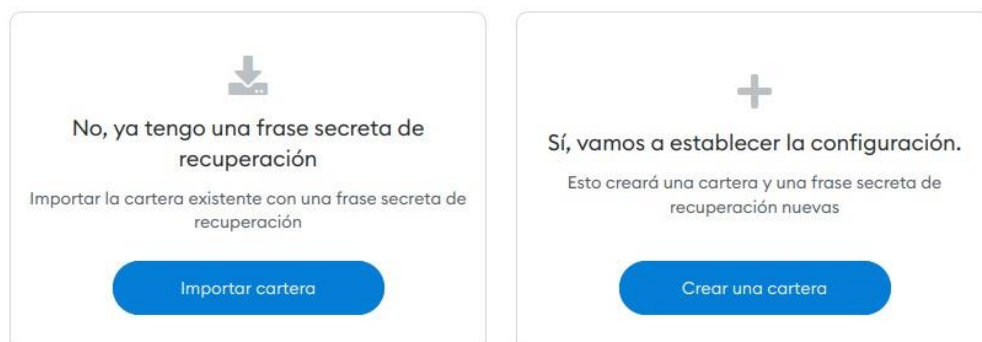



Fig.18 Página de bienvenida a MetaMask, donde se observan dos botones, el izquierdo para importar una cartera ya existente y el derecho para crear una cartera nueva.



The screenshot shows the MetaMask interface for importing an account. At the top, the MetaMask logo and a '< Volver' button are visible. The main heading is 'Importar una cuenta con la frase secreta de recuperación'. Below this, a subtitle reads 'Ingrese su frase secreta aquí para restaurar su bóveda.' The 'Secret Recovery Phrase' section includes a dropdown menu currently set to 'I have a 12-word phrase'. A blue information box states: 'You can paste your entire secret recovery phrase into any field'. Below this, there are 12 numbered input fields for the recovery phrase words, each with a copy icon to its right. The 'Contraseña nueva (mín. de 8 caracteres)' section has a single input field. The 'Confirmar contraseña' section also has a single input field. At the bottom, there is a checkbox labeled 'Leí y estoy de acuerdo con' followed by a link to 'Términos de uso'.

Fig.19 Página para importar una cartera existente mediante la frase secreta y para definir una nueva contraseña de acceso a la aplicación de cartera.

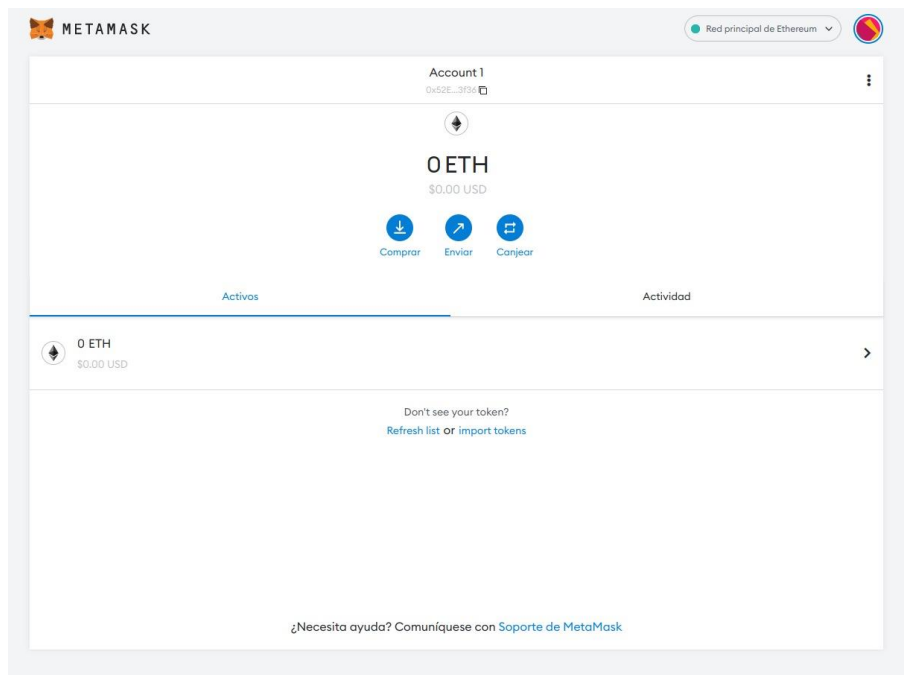


Fig. 20 Vista general de una cuenta de Ethereum desde la aplicación de cartera, donde se muestra el saldo de Ether y los tokens en posesión.

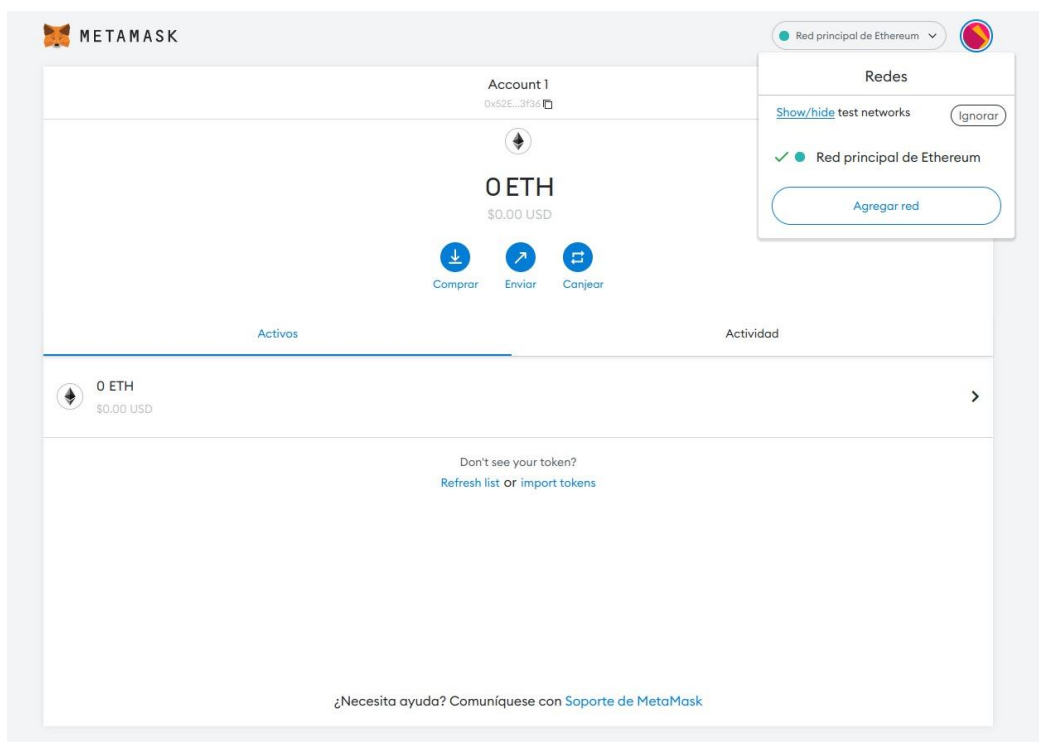


Fig.21 Vista de la cartera tras seleccionar la opción de “Red principal de Ethereum”.

Show test networks
Select this to show test networks in network list

☒ ACTIVADO

Personalizar nonce de transacción
Active esta opción para cambiar el nonce (número de transacción) en las pantallas de confirmación. Esta es una función avanzada, úsela con precaución.

☐ DESACTIVADO

Temporizador con bloqueo automático (minutos)
Establezca el tiempo de inactividad en minutos antes de que se bloquee MetaMask.

5

Guardar

Sincronizar datos con 3Box (experimental)
Active esta opción para crear una copia de seguridad de la configuración con 3Box. Actualmente, esta función es experimental. Úsela bajo su propio riesgo.

☐ DESACTIVADO

Puerta de enlace de IPFS
Escriba la dirección URL de la puerta de enlace de IPFS CID para usar la resolución de contenido de ENS.

dweb.link

Guardar

Ignorar el recordatorio de respaldo de la frase de recuperación
Active esta opción para ignorar el recordatorio de respaldo de la frase de recuperación. Le recomendamos que respalde la frase secreta de recuperación para evitar la pérdida de fondos.

Fig.22 Opciones del Metamask para poder activar las redes de prueba.

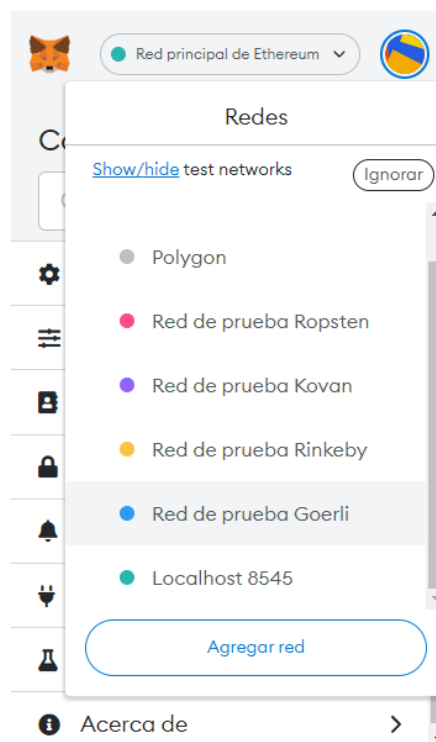


Fig.23 Redes disponibles tras activar la opción de visualizar las redes de prueba.

7. CONCLUSIONES

7.1. CONCLUSIONES

Una vez finalizado el proyecto, se cuestionan los progresos y los objetivos planteados al comienzo del trabajo, indicados en el Capítulo 1.

El objetivo principal de este trabajo es la creación de un fondo de inversión indexado de criptodivisas que emplea estrategias de trading.

El primer objetivo específico planteado es la creación de una interfaz de usuario simple y completa, que incluya las acciones de comprar y vender tokens. Este primer objetivo se ha cumplido, ya que, tras investigar sobre las interfaces de otros productos parecidos, hemos logrado crear una interfaz intuitiva y limpia que permite a los usuarios interactuar con ella sin grandes complicaciones.

El segundo objetivo específico era conseguir que se pudiera conectar a una Blockchain real, de manera que se pudiera utilizar en el mundo real. Este objetivo se ha cumplido ya que nuestros contratos se pueden desplegar en cualquier red y, además, hacen uso de Uniswap y Curve.fi que son aplicaciones con contratos desplegados en la red de Ethereum.

El tercer objetivo específico era poder mantener una conexión con un servidor off-chain para poder incluir una estrategia de trading funcional. Este objetivo también se ha cumplido, utilizamos una estrategia de trading trivial, que se puede actualizar en un futuro.

El cuarto objetivo específico es incluir la posibilidad de operar con diversas monedas que no fueran únicamente las más conocidas como Bitcoin y Ethereum. Este objetivo se ha cumplido, ya que nuestro trabajo permite operar con tokens que teóricamente no tienen valor pero que tienen correspondencia 1 a 1 con los tokens de la Blockchain correspondiente.

El quinto objetivo específico era permitir la conexión de una cartera desde la interfaz de usuario, de manera que la gente pueda utilizarla sin necesidad de utilizar scripting. Este objetivo ha sido cumplido, al permitir que la cartera de “Metamask”, se pueda conectar a la interfaz de usuario de manera que se puede operar desde las direcciones guardadas en dicha cartera.

El sexto objetivo específico es la creación de un token propio compuesto del precio de ciertas criptodivisas en el que basar el fondo de inversión. Hemos cumplido este objetivo mediante la creación de un Token ERC20

que se vende al valor de la media ponderada de los precios de Bitcoin, Ethereum, Polygon, etc.

Como hemos conseguido cumplir todos los objetivos que propusimos al comienzo del proyecto, consideramos que este trabajo ha sido un éxito. Además, la posibilidad de colaborar con una empresa nos ha permitido aplicar los conocimientos teóricos que hemos adquirido en un plano más realista, así como aprender el funcionamiento del mundo Blockchain a nivel empresarial.

7.2. TRABAJO FUTURO

En este trabajo hemos cumplido todos los objetivos que nos habíamos planteado y hemos conseguido desarrollar una aplicación funcional y que servirá como primera versión del producto para una empresa. Esto no quiere decir que no se pueda ampliar el proyecto y añadir funcionalidades adicionales, que pueden surgir por necesidad a lo largo del tiempo, o que pueden ser sugeridas por los propios usuarios que pueden hacernos llegar feedback para mejorar el producto.

El proceso de desarrollo iba muy enfocado a realizar una aplicación modular, en la que cada contrato se pueda reemplazar llegado el momento, si se quisiese actualizar o se descubriese alguna vulnerabilidad nueva. La arquitectura es perfecta para realizar este tipo de cambios, e incluso podrían añadirse más contratos reemplazando solo el de Proxy.

Algunos cambios que podemos realizar en nuestro sistema podrían ser los siguientes:

- Implementar una estrategia de trading más consistente con un análisis de backtesting profundo en lugar de la estrategia trivial utilizada hasta el momento. Esta estrategia la desarrollará gente experta en las finanzas, ya que nosotros no somos especialistas en este campo, y, por lo tanto, solo hemos implementado esta estrategia como prueba de concepto.
- Desplegar los contratos que componen la aplicación en más de una blockchain EVM, para así poder llegar a más usuarios, independientemente de si utilizan Ethereum, Binance Smart Chain, Polygon, etc. Esto también nos permitiría utilizar estrategias con DeX que solo existan en una blockchain y no en otras (por ejemplo,

UniswapV2 que es exclusivo de Ethereum o PancakeSwap que es exclusivo de Binance Smart Chain).

- Unificar los fondos de inversión desplegados en las diferentes blockchains. De esta manera, además de poder utilizar todos los productos disponibles para nuestras estrategias, permitimos que de manera sencilla los usuarios envíen los tokens que quieran utilizar entre las redes. Para esto podríamos hacer uso de un bridge ya existente, como, por ejemplo, Multichain, o crear nuestro propio sistema. Para nuestro sistema de cross-chain, podríamos utilizar un HTLC (Hash Time-Locked Contract) o alguna otra técnica más sofisticada, según nuestras necesidades.
- Mejorar la interfaz de usuario, adaptándola constantemente a este mundo ya que siempre hay nuevas tecnologías y librerías para las conexiones con la interfaz, para que encaje con los contratos que vayamos también actualizando. De esta manera el usuario podrá tener una experiencia de uso sencilla y agradable sin que note los cambios de contrato.
- Agregar al sistema otra estrategia de inversión automatizada distinta al trading que consta en realizar staking (depositar fondos, normalmente para proveer liquidez) en alguna plataforma que nos lo permita. De esta manera se obtendría un beneficio económico por tener los fondos guardados e iría aumentando el valor del token de CryptoFund acorde a las comisiones que se vayan generando. Para ello se podría hacer un análisis de todas las plataformas para encontrar aquella que nos diese un mayor APY (Annual Percentage Yield).
- Uso de otras plataformas alternativas al DeX Uniswap, que permitan realizar compra y venta de futuros. De esta manera se podrían implementar las operaciones en corto, y no solo realizar una venta cuando haya fondos disponibles.
- Implementación de estrategias de arbitraje que permitan aumentar nuestro saldo de stablecoins en lugar de estrategias de trading. De esta forma, se conseguiría aumentar el NAV con menos riesgo de pérdida si se desplomase el valor de alguna moneda, pero con más costes de operativa, ya que realizar operaciones de arbitraje requiere operar más veces y no siempre obtener beneficio.

8. CONCLUSIONS

8.1. CONCLUSIONS

Once the project is completed, the progress and objectives set at the beginning of the work are assessed, being those mentioned in Chapter 1, Section 2.

The main objective is the creation of a cryptocurrency indexed investment fund using trading strategies. To meet this main objective, we have used the specific objectives as it is detailed in the following paragraphs.

The first specific objective was to create a simple user interface, which includes the actions of buying and selling tokens. This first objective has been met, since, after researching the interfaces of other similar products, we have managed to create an intuitive and clean interface that allows users to interact with it without major complications.

The second specific objective was to make it connectable to a real Blockchain, so that it could be used in the real world. This goal has been met as our contracts can be deployed on any network and, in addition, make use of Uniswap and Curve.fi which are applications with contracts deployed on the Ethereum network.

The third specific objective was to be able to maintain a connection to an off-chain server in order to include a functional trading strategy. This objective has also been met, we use a trivial trading strategy, which can be updated in the future.

The fourth specific objective is to include the possibility of trading with various currencies that were not only the most known ones such as Bitcoin and Ethereum. This objective has been met, as our work allows trading with "Wrapped" tokens, which are tokens that are theoretically worthless but have 1-to-1 correspondence with tokens on the corresponding Blockchain.

The fifth specific objective was to enable the connection of a wallet from the user interface, so that people can use it without scripting. This objective has been met, by allowing the "Metamask" wallet to be connected to the user interface so that trading can be done from the addresses stored in that wallet.

The sixth specific objective is the creation of a proprietary token composed of the weighted average of the price of certain cryptocurrencies on which to base the investment fund. We have met this objective by creating an

ERC20 token which we sell at the value of the weighted average of the prices of Bitcoin, Ethereum, Polygon, etc.

Since we managed to meet all the objectives we proposed at the beginning of the project, we consider this work a success. In addition, the possibility of collaborating with a company has allowed us to apply the theoretical knowledge we have acquired on a more realistic level, as well as to learn how the Blockchain world works at a business level.

8.2. FUTURE WORK

In this work we have met all the objectives we had set ourselves and we have managed to develop a functional application that will serve as the first version of the product for a company. This does not mean that the project cannot be extended and additional features added to it, which may arise by necessity over time, or that may be suggested by the users themselves who can give us feedback to improve the product.

The development process was very focused on making a modular application, in which each contract can be replaced whenever it needs to be updated or if a new vulnerability is discovered. The architecture is perfect to make this kind of changes, and even more contracts could be added by replacing only the Proxy contract.

Some improvements that can be made to this system follow:

- Implement a more consistent trading strategy with deep backtesting analysis instead of the trivial strategy used so far. This strategy will be developed by people who are experts in finance, as we are not specialists in this field, and therefore we have only implemented this strategy as a proof of concept.
- Deploy the contracts that make up the application on more than one EVM blockchain, so that we can reach more people, regardless of whether they use Ethereum, Binance Smart Chain, Polygon, etc. This also benefits us when using strategies with DeX that only exist on one blockchain and not on others (e.g., UniswapV2 which is exclusive to Ethereum or PancakeSwap which is exclusive to Binance Smart Chain).

- Unify the investment funds deployed on the different blockchains. In this way, in addition to being able to use all the products available for our strategies, we allow users to easily send the tokens they want to use between networks. For this we could make use of an existing bridge, such as Multichain, or create our own system. For our cross-chain system, we could use an HTLC (Hash Time-Locked Contract) or some other more sophisticated technique, depending on our needs.
- Improve the user interface, constantly adapting it to this world as there are always new technologies and libraries for the connections with the interface, so that it fits with the contracts that we are also updating. This way the user can have a simple and pleasant user experience without noticing the contract changes.
- Add to the system another automated investment strategy other than trading that consists of staking in a platform that allows us to obtain an interest for having the funds saved and we would increase the value of our token according to the commissions that we generate. To do this we can make an analysis of all platforms to find the one that would give us a higher APY (Annual Percentage Yield).

9. CONTRIBUCIONES AL PROYECTO

En esta sección se describirán detalladamente las distintas contribuciones individuales de cada uno de los miembros del equipo al proyecto.

9.1. ISMAIL AZIZI

Este proyecto surgió como una idea de Valentín, el colaborador de este proyecto y responsable de la empresa FairCo en la que me encuentro trabajando hasta el momento. Me propuso hacer un fondo de inversión activo para este trabajo y para posteriormente seguir desarrollándolo y poder llegar a implementarlo como un producto real. Actualmente no existe ningún fondo de inversión activo, descentralizado y basado en la compra-venta de criptomonedas. Lo más parecido a esta idea es Crypto20, pero este es un fondo pasivo, es decir, compra ciertas monedas a principio un período de tiempo, y se espera que el valor de estas suba para generar beneficio. La diferencia con esta nueva idea es que el fondo CryptoFund implementaría unas estrategias activas de compra-venta a lo largo del tiempo, y no se basa solo en la compra en un momento dado para generar dinero.

La propuesta fue trasladada a los miembros del grupo y al tutor del trabajo, y después de recibir el visto bueno por todos los involucrados, comenzó el desarrollo.

Las primeras semanas se basaban principalmente en reuniones en las que poníamos en común los conocimientos que teníamos y que íbamos adquiriendo en las áreas en las que nos íbamos a enfocar cada uno. El foco de mi trabajo iba enfocado sobre todo a la optimización del consumo de gas. Elegí esta parte del proyecto debido a que al estar mi trabajo centrado en el desarrollo de aplicaciones en sistemas blockchain, estoy familiarizado con los problemas que pueden causar unos costes de gas excesivos, sobre todo en la red Ethereum. Los costes de algunas transacciones pueden superar los cientos, incluso los miles de dólares dependiendo del momento y de la rapidez con la que necesitemos que se confirme nuestra transacción. Esto muchas veces hace ciertas operaciones que deberían ser realizadas, inviables.

Mis aportaciones iniciales se basaban en el estudio del Yellowpaper [31] de Ethereum, el cual da una especificación formal del coste de todas las operaciones de la máquina virtual de Ethereum. Aunque el lenguaje en el que hemos programado es Solidity, es necesario conocer en profundidad

cómo traduce el compilador las instrucciones de alto nivel, a las instrucciones de bajo nivel, para poder estimar los costes con precisión. Además de este artículo, también ha sido necesario consultar la página oficial de Ethereum que contiene la descripción detallada de estas instrucciones de bajo nivel [32]. Esta página contiene el coste de algunas instrucciones y referencia una página de Github con el desglose de los costes de las instrucciones que dependen de ciertos factores (como, por ejemplo, ocupar memoria persistente de un contrato, cuyo coste depende del espacio ocupado) [33].

Además de conocer la especificación de las instrucciones de bajo nivel, otro objetivo de este trabajo es conseguir que los contratos sean independientes y actualizables. Mi aportación en este apartado fue tener en cuenta ciertos factores del diseño. Es importante saber estructurar los contratos de manera que sean eficientes, pero sin comprometer la modularidad. Debido a la naturaleza inmutable de los contratos, una vez están desplegados en la red blockchain, no se pueden cambiar. Por ello, crear un contrato Proxy que almacene y redireccione a los contratos es muy útil, ya que, si actualizamos algún contrato, en lugar de cambiar toda la estructura de contratos, solo tendríamos que reemplazar la dirección antigua por la nueva.

El sistema necesita un servidor off-chain que gestione la compra-venta de las criptomonedas con la estrategia que esté implementada. Junto con mis compañeros, desarrollamos un script programado periódicamente que realice las operaciones pertinentes en cada momento. Mi rol en esta tarea era guiar a mis compañeros para programar el script y ejecutarlo mediante un planificador, ya que, gracias a mi trabajo, ya he realizado tareas similares a esta.

9.2. GONZALO MARTÍNEZ

Tras la primera reunión que realizamos con el fin de aproximarnos a un tema, revisé todos los apuntes que tenía de asignaturas relacionadas con el Trabajo de Fin de Grado. Durante las siguientes reuniones, en las que poco a poco fuimos concretando más las tareas que podíamos realizar, me llamó la atención la parte de backtesting, por lo que me estuve informando hablando con algún profesor que sabía que tenía conocimientos sobre Backtesting y buscando información en Internet sobre esta metodología. Fue ahí cuando encontré la herramienta Ninja Trader, un simulador de trading gratuito, que permitía probar las estrategias de trading con datos del pasado, estuve investigando dicha

herramienta, pero la idea de realizar backtesting sobre nuestro proyecto fue descartada. Además, durante este periodo también estuve leyendo los documentos que pasaban tanto Ismail como Jesús, que al tener más conocimientos que yo sobre el mundo Blockchain me parecieron muy interesantes.

Más adelante se repartieron los distintos componentes de la aplicación y yo realicé la interfaz de usuario (apartado 5.1) para evitar que los usuarios del fondo tuvieran que saber scripting para poder interactuar con nuestros contratos. Esta idea me pareció también muy atractiva, debido a que en las prácticas de empresa estaba realizando tareas de Frontend y me gustaba mucho.

Tras comentarlo con mis compañeros y con Jesús me dispuse a realizar unos primeros prototipos en Balsamiq para empezar a tener una idea básica sobre la que comenzar el desarrollo. Pasadas dos semanas, enseñé los prototipos al grupo y aportaron una serie de cambios como eliminar el “Log In” tradicional e implementar un “Log In” con una cartera de criptodivisas. Al final de esa reunión, Valentín, colaborador independiente del proyecto, me propuso realizar una reunión para enseñarme páginas punteras en este sector para poder ver cómo eran sus interfaces.

Durante la reunión mencionada anteriormente estuvimos revisando las páginas (Apartado 5.4.3.) de Crypto20, Stoic, Metamask, Uniswap, Curve, Pancake, Blackrock e Interactive Brokers y observamos que las 6 primeras, que eran las relacionadas con el mundo de las criptomonedas y el Blockchain, tenía una interfaz muy simple, intuitiva y con no mucha información para saturar a los usuarios, mientras que las otras dos, correspondientes a los fondos financieros al uso, tenían interfaces mucho más complejas y mostraban más información.

Tras realizar la reunión con Valentín comencé un primer maquetado rápido de la aplicación web con HTML y CSS, pero casi sin ningún tipo de funcionalidad, ya que aún no había utilizado nada de JavaScript. Este periodo de primer desarrollo duró dos semanas y después de esas dos semanas mostré los avances al grupo, esta primera interfaz estaba formada por 4 pantallas, cada una asignada a una de las funcionalidades que queríamos implementar (Inicio, Compra, Venta y Log In). Durante la reunión se sugirió intentar reducir el número de pantallas a una que englobara todas las funcionalidades.

Las siguientes dos semanas de trabajo las dediqué a implementar las mejoras que se sugirieron en la reunión mencionada anteriormente, así como a investigar sobre las APIs de Metamask y de Cryptocompare para

poder realizar el Log In con la cartera de Metamask y poder mostrar el precio en tiempo real de las criptomonedas de las que está compuesto el fondo.

Durante las dos últimas semanas de desarrollo trabajé con Ismail para realizar la conexión entre los contratos y la interfaz y pulí algunos detalles de la interfaz como los flujos de compra y venta de los botones de la interfaz de usuario.

9.3. MUAD ROHAIBANI

Después de realizar la propuesta de CryptoFund la empresa en la que trabaja Ismail y actualmente yo también, me interesé principalmente por la lógica y el funcionamiento de los fondos de inversión y el poder adaptarlo a la tecnología Blockchain.

Empecé informándome sobre cómo funcionan los fondos de inversión para tener claros distintos conceptos en el campo de las finanzas, al ser algo que no dominaba. Después, para hacerme una idea de cómo adaptar estos conceptos a la tecnología Blockchain, estudié el whitepaper de Crypto20, primer y único fondo de inversión Blockchain exitoso. En concreto, analicé detalladamente como estaban programados los Smart Contracts, qué estrategia usaban, cómo se realizaban las operaciones de compra y venta dentro del mercado de forma automatizada.

Intenté pensar en una forma de desarrollar un fondo de inversión que use Blockchain y sea distinto al único actual, así que propuse a mi tutor que podía ser buena idea desarrollar nuestro fondo con técnicas de inteligencia artificial. Iba a encargarme de desarrollar una estrategia de compra-venta de criptodivisas, estuve investigando varias semanas la forma de aplicarlo a los mercados financieros, revisé todos los apuntes de inteligencia artificial que disponía de la asignatura cursada aquí, me informé en distintas páginas web. Pero finalmente se decidió orientar el proyecto más hacia la arquitectura del sistema evitar abordar demasiados aspectos al mismo tiempo ya que requería bastante trabajo esa parte, de forma que finalmente usamos una estrategia sencilla.

Nos encargamos Ismail y yo de la parte de implementación de los Smart Contracts y definición de la arquitectura del sistema. Comenzamos mediante varias reuniones, primero para definir la arquitectura de los Smart Contracts, cómo se iban a conectar entre ellos, cuál iba a estar conectado con nuestra interfaz de página web, cómo iba a funcionar el servidor dentro del sistema, las funciones principales que íbamos a

necesitar y las variables. También contamos con la asistencia de Valentín García en la ayuda para definir la arquitectura con sus conocimientos en el campo de las finanzas.

Al estar trabajando en diseñar un producto en el que va a haber dinero depositado de usuarios es muy importante que el sistema sea seguro y que haya una gestión de los posibles intentos de hackeo que puede haber. Así que yo me centré en el análisis de vulnerabilidades dentro del sistema y me puse a investigar los distintos tipos de ataques que se pueden realizar y adaptamos el código para evitar cualquier tipo de ataque conocido.

El cálculo de comisiones fue una parte compleja que tuvimos que pulir con la ayuda de Valentín García mediante varias reuniones investigando las fórmulas matemáticas que podemos usar y cómo aplicarlas al cálculo de comisiones dentro del fondo dependiendo del tiempo que los usuarios tengan depositado su dinero.

Una vez ya definida la arquitectura, empezamos a programar los Smart Contracts, tuvimos que adaptar algunos cambios ya definidos anteriormente en la arquitectura al darnos cuenta en la fase de programación de alguna situación no prevista en la fase de diseño.

Ya con todo programado pasamos a la fase de ejecución de los contratos en la red de pruebas de Ethereum llamada Goerli y corregimos todos los fallos de implementación para poder realizar bien todas las pruebas y garantizar que el sistema funciona correctamente.

10. REFERENCIAS

- [1] V. Buterin, «Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,» 2014. [En línea]. Available: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.
- [2] S. Nakamoto, «Bitcoin Whitepaper,» 31 10 2008. [En línea]. Available: <https://bitcoin.org/bitcoin.pdf>.
- [3] Ethereum, «Ethereum, What is Proof of Stake?,» [En línea]. Available: <https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/>.
- [4] Binance, «Proof of Authority Explicado,» [En línea]. Available: <https://academy.binance.com/es/articles/proof-of-authority-explained>.
- [5] RugDocWiki, «Proof of Stake Authority,» [En línea]. Available: <https://wiki.rugdoc.io/es/docs/proof-of-stake-authority-posa/>.
- [6] C. Zhao, «Binance Smart Chain,» [En línea]. Available: <https://docs.binance.org/smart-chain/guides/bsc-intro.html>.
- [7] R. Wyatt, «Polygon whitepaper,» [En línea]. Available: <https://polygon.technology/lightpaper-polygon.pdf>.
- [8] Ethereum Protocol Development Governance and Network Upgrade Coordination, «Propuestas de mejora de Ethereum (EIP),» [En línea]. Available: <https://ethereum.org/es/eips/>.
- [9] R. Ji, N. He, L. Wu, H. Wang, G. Bai y Y. Guo, «DEPOSafe: Demystifying the Fake Deposit Vulnerability in Ethereum Smart Contracts,» 11 06 2020. [En línea]. Available: <https://arxiv.org/pdf/2006.06419.pdf>.
- [10] Binance, «¿Qué es la verificación KYC y por qué es cada vez más importante para las criptomonedas?,» 18 06 2021. [En línea]. Available: <https://www.binance.com/es/blog/ecosystem/qu%C3%A9-es-la-verificaci%C3%B3n-kyc-y-por-qu%C3%A9-es-cada-vez-m%C3%A1s-importante-para-las-criptomonedas-421499824684902130>.
- [11] Binance Academy, «Anti Money Laundering (AML) | Binance Academy,» [En línea]. Available: <https://academy.binance.com/en/glossary/anti-money-laundering>.
- [12] «Binance,» [En línea]. Available: <https://www.binance.com/es>.

- [13] «FTX,» [En línea]. Available: <https://ftx.com/es>.
- [14] «Coinbase,» [En línea]. Available: <https://www.coinbase.com/es>.
- [15] Uniswap, «Uniswap Interface,» [En línea]. Available: <https://app.uniswap.org/#/swap?chain=mainnet>.
- [16] Pancake, «Home | Pancake,» [En línea]. Available: <https://pancakeswap.finance/>.
- [17] «Aave - Open Source Liquidity Protocol,» [En línea]. Available: <https://aave.com/>.
- [18] Crypto20, «CRYPTO20 - First Tokenized Cryptocurrency Index Fund,» [En línea]. Available: <https://www.crypto20.com/en/>.
- [19] Crypto20, «Crypto20 Whitepaper,» [En línea]. Available: <https://cdn.crypto20.com/pdf/c20-whitepaper.pdf>.
- [20] C. 10, «Crypto 10,» [En línea]. Available: <https://invictuscapital.com/es/crypto10hedged>.
- [21] «Home | Uniswap Protocol,» [En línea]. Available: <https://uniswap.org/>.
- [22] G. a. W. A.M. and Wood, Mastering Ethereum: Building Smart Contracts and DApps, O'Reilly Media, Inc., 2018.
- [23] «Ataque DAO 2016,» 2016. [En línea]. Available: <https://medium.com/coinmonks/reentrancy-exploit-ac5417086750>.
- [24] Balsamiq , «Balsamiq Wireframes - Industry Standar Low-Fidelity Wireframing Software | Balsamiq,» [En línea]. Available: <https://balsamiq.com/wireframes/>.
- [25] Paxos, «Home - Paxos,» [En línea]. Available: <https://paxos.com/>.
- [26] Stoic, «Crypto powered bot for Binance powered by AI - Stoic,» [En línea]. Available: <https://stoic.ai/>.
- [27] Metamask, «The crypto wallet for Defi Web3 Daps and NFT | MetaMask,» [En línea]. Available: <https://metamask.io/>.
- [28] Curve, «Curve.fi,» [En línea]. Available: <https://curve.fi/>.
- [29] BlackRock, «Planificación Financiera y Gestión de Inversiones | BlackRock,» [En línea]. Available: <https://www.blackrock.com/es>.

- [30] Interactive Brokers, «Inicio | Interactive Brokers U.K. Limited,» [En línea]. Available: <https://www.interactivebrokers.co.uk/es/home.php>.
- [31] G. Wood, «Ethereum Yellow Paper: a formal specification of Ethereum, a programmable blockchain,» 6 5 2022. [En línea]. Available: <https://ethereum.github.io/yellowpaper/paper.pdf>.
- [32] @wackerow y @julian-st, «Opcodes for the EVM | ethereum.org,» [En línea]. Available: <https://ethereum.org/en/developers/docs/evm/opcodes/>.
- [33] wolflo, «Appendix - Dynamic Gas Costs,» [En línea]. Available: <https://github.com/wolflo/evm-opcodes/blob/main/gas.md>.