Degree in Telecommunication Engineering

Bachelor's final work

Funding Rounds with Blockchain

Author

Álvaro Lastra Aragoneses

Supervised by

Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

May 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Funding Rounds with Blockchain en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Álvaro Lastra Aragoneses Fecha: 1/6/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO

Firmado digitalmente por 71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO
Fecha: 2023.06.04 15:33:25 +02'00'

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: 04/06/2023

Degree in Telecommunication Engineering

Bachelor's final work

Funding Rounds with Blockchain

Author

Álvaro Lastra Aragoneses

Supervised by

Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

May 2023

Rondas de Financiación con Blockchain

Autor: Lastra Aragoneses, Álvaro

Director: Fernández-Pacheco Sánchez-Migallón, Atilano

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

Resumen del Proyecto

El trabajo pretende desarrollar una plataforma web en donde inversores y emprendedores puedan llevar a cabo rondas de financiación para las empresas. Para el desarrollo de la plataforma se usará la tecnología blockchain y Smart contracts que nos brindarán una gestión descentralizada, pública y segura de la red. La tecnología blockchain reduce los costes de terceras partes que gestionan las rondas de financiación y da información anónima, veraz y transparente accesible para todo el mundo. El fin de este trabajo es crear una plataforma que pueda ser usada por todo el mundo y que mejores las relaciones fiduciarias en los negocios.

Palabras Clave: Blockchain, Rondas de Financiación, Ethereum, IPFS, The-Graph, React

Introducción

Vivimos en un mundo capitalista en constante evolución y desarrollo gracias a la tecnología. Actualmente, es más fácil que nunca crear y destruir riqueza; el tiempo medio y el capital para crear una empresa se han desplomado[1]. La evolución de la tecnología en los últimos 20 años ha hecho que sea más fácil registrar una empresa en línea, intercomunicarse con el mundo casi sin costes, operar sin una ubicación física y la comunidad de empresarios está creciendo rápidamente. Todos estos

cambios han dado lugar a un aumento de la creación de start-ups en todos los países[2].

Las rondas de financiación tradicionales son largas y costosas, con procesos complejos y un acceso limitado a los pequeños inversores. Los emprendedores en las primeras etapas de la empresa buscan inversiones rápidas para entrar en un mercado competitivo en el que el tiempo es crucial para el éxito. Blockchain es una tecnología transparente, anónima y fiable que nos ayuda a resolver muchos de los problemas actuales de la financiación de start-ups.

Este proyecto final pretende explorar el uso de blockchain en las rondas de financiación, especialmente en las primeras etapas de una empresa, donde el acceso a un mayor grupo de inversores y costes reducidos son fundamentales. Este trabajo también explorará las formas de resolver los problemas fundamentales de inversión mediante contratos inteligentes. Además, el proyecto determinará el potencial de blockchain para democratizar la red para pequeños inversores y determinará otros beneficios del uso de redes Blockchain.

Metodología

En primer lugar, investigamos sobre Blockchain utilizando la metodología deductiva para adquirir conocimientos en este campo. Después, exploramos Ethereum y los Smart Contracts para aprender cómo aplicar esta tecnología. En segundo lugar, realizamos una revisión bibliográfica de las rondas de financiación, determinando sus principales problemas y cómo las actuales rondas de financiación se enfrentan a estos retos. Esto nos proporcionó una visión completa del panorama actual y nos ayudó a identificar las áreas de mejora que se pueden resolver con blockchain. Por último, utilizamos Agile, la metodología más común en el desarrollo de software, para desarrollar la aplicación.

Resultados

El sistema desarrollado ha demostrado ser eficaz para rondas de financiación usando la tecnología blockchain. Los Smart Contracts han permitido crear rondas de financiación de manera transparente y anónima. Además hemos conseguido desarrollar una Dapp con React que permite a los usuarios normales usar todas las funcionalidades de una forma rápida e intuitiva.

La Dapp posee varias vistas con las que se puede interactuar con los smart contracts, además de poder encontrar información guardada mediante IPFS relacionada con las rondas de financiación. En las imagenes 1 y 2 podemos ver respectivamente las vistas de la lista con todas las rondas de financiación y la vista de la página de una BitRound.

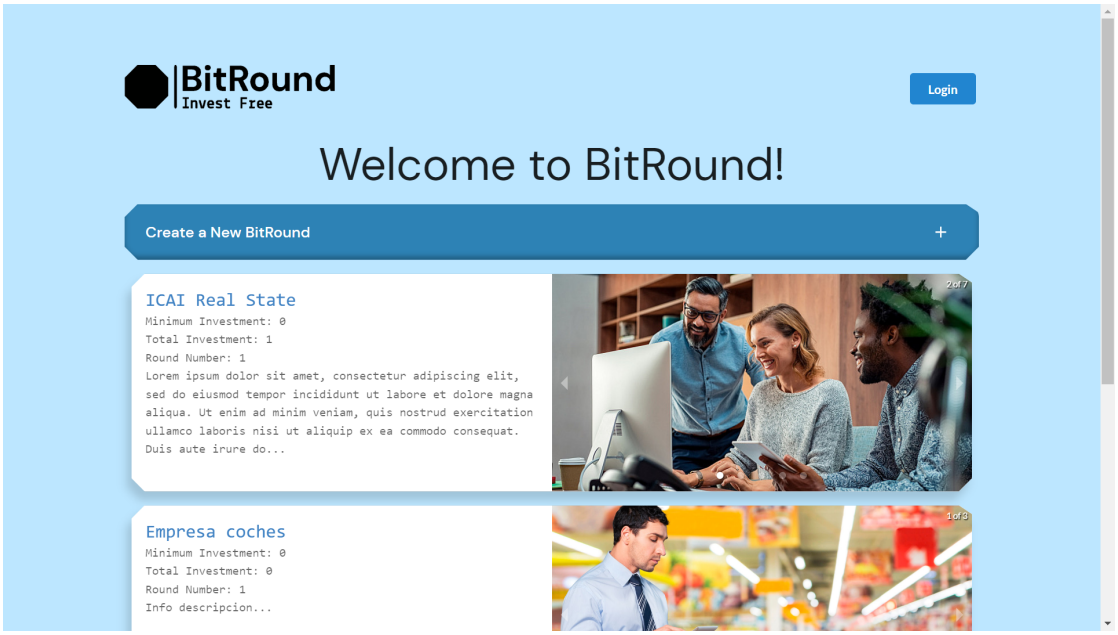

Figure 1: Home Page Dapp

En términos de costes, el sistema ha demostrado ser rentable en algunas ocasiones

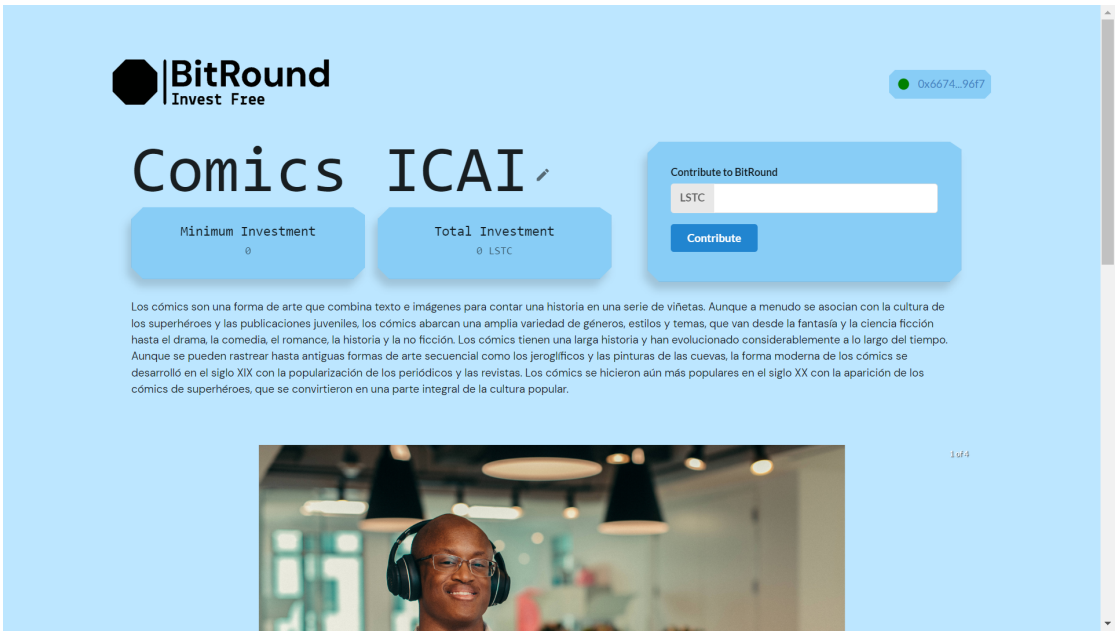

Figure 2: BitRound Page

en comparación con las rondas de financiación tradicionales.

En términos de accesibilidad, el sistema ha demostrado ser accesible para un amplio rango de inversores. La naturaleza descentralizada de la blockchain permite a cualquier persona con acceso a internet participar en las rondas de financiación, lo que democratiza el proceso de inversión.

Conclusiones

El proyecto ha demostrado que la tecnología blockchain puede ser una solución eficaz para facilitar las rondas de financiación, especialmente en las primeras etapas de una empresa. Ha cumplido el objetivo de crear una red descentralizada, transparente, anónima, que mejora las relaciones fuduciarias y que reduce los costes en algunos casos.

En términos de perspectivas futuras, hay varias áreas que podrían ser exploradas para mejorar aún más el sistema. Estas incluyen la implementación de más funcionalidades en los contratos inteligentes, la mejora de la interfaz de usuario y, principalmente, la exploración de otras tecnologías blockchain.

En resumen, este proyecto ha demostrado que la tecnología blockchain tiene un gran potencial para revolucionar las rondas de financiación y se espera que su uso se generalice en el futuro.

Funding Rounds with Blockchain

Author: Lastra Aragoneses, Álvaro

Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano

Collaborating Entity: ICAI – Universidad Pontificia Comillas

Abstract

The work aims to develop a web platform where investors and entrepreneurs can conduct company financing rounds. Blockchain technology and Smart contracts will provide decentralized, public, and secure network management for the platform's development. Blockchain technology reduces the costs of third parties managing the financing rounds and gives anonymous, truthful, and transparent information accessible to everyone. This work aims to create a platform for everyone to use and improve business fiduciary relationships.

Keywords: Blockchain, Funding Rounds, Ethereum, IPFS, TheGraph, React

Introduction

We live in a capitalist world in constant evolution and development due to technology. Currently, it is easier than ever to create and destroy wealth; the average time and capital to create a company have plummeted[1]. The evolution of technology in the past 20 years has made it easier to register a business online, intercommunicate with the world with almost no costs, operate without a physical location, and the entrepreneur community is growing fast. All these changes have led to a growth in the creation of start-ups in every country[2].

Traditional funding rounds are lengthy and costly, with complex processes and limited access to small investors. Entrepreneurs seek rapid investment to enter a competitive market where time is crucial to success. Blockchain is a transparent, anonymous, and reliable technology that helps us to solve many of the current problems in start-up funding.

This final project aims to explore the use of blockchain in funding rounds, especially in implementing the technology in the early stages of a company where access to a broader pool of investors and reduced costs is critical. This work will also explore ways to solve fundamental investment issues through smart contracts that secure investment. Additionally, the project will determine the potential for blockchain to democratize the network for small investors and other benefits of using Blockchain networks.

Methodology

First, we researched about Blockchain using the deductive methodology to gain knowledge in this field. Then, we explored Ethereum and Smart Contracts to learn how to apply this technology. Second, we conducted a literature review of funding rounds, determining their main issues and how the current funding rounds face these challenges. This provided us with a comprehensive view of the current landscape and help us identify improvement areas that can be solved with blockchain. Finally, we used Agile, the most common methodology in software development, to develop the application.

Results

The developed system has proven effective for financing rounds using blockchain technology. Smart Contracts have allowed creating of funding rounds transparently

and anonymously. In addition, we have developed a Dapp with React that allows regular users to use all the functionalities quickly and intuitively.

The Dapp has several views with which we can interact with the smart contracts and find information stored via IPFS related to the financing rounds. In the images 1 and 2, we can see the list views with all the funding rounds and the page view of a BitRound.

In terms of cost, the system has proven to be cost-effective at times compared to traditional funding rounds.

In terms of accessibility, the system has proven to be accessible to a wide range of investors. The decentralized nature of blockchain allows anyone with internet access to participate in funding rounds, which democratizes the investment process.

Conclusions

The project has demonstrated that blockchain technology can be an effective solution to facilitate financing rounds, especially in the early stages of a company. It has fulfilled the objective of creating a decentralized, transparent, anonymous network that improves fiduciary relationships and reduces costs in some cases.

Regarding prospects, several areas could be explored to improve the system further. These include implementing more functionality in smart contracts, improving the user interface, and exploring other blockchain technologies.

In summary, this project has demonstrated that blockchain technology has great potential to revolutionize funding rounds, and its use is expected to become more widespread.

Contents

1 Introduction **17**

2 Description of the technologies **19**

2.1 Ethereum 19

2.2 Solidity 20

2.3 React 20

2.4 Web3 JS 21

2.5 Infura 21

2.6 Metamask 21

2.7 Remix IDE 22

2.8 Visual Studio Code 22

2.9 GitHub 22

2.10 IPFS 23

2.11 Graph Protocol 23

3 State of the art **24**

3.1 Cryptography 24

3.1.1 Hash Function 24

3.1.2 Asymmetric cryptography 25

3.1.3 Private Key 26

3.1.4 Public Key 26

3.1.5 Elliptic Curves 27

3.2 Blockchain 29

3.2.1 Addresses 30

3.2.2 Transactions 30

3.2.3 Block 31

3.3 Consensus Protocol . . . 32
3.3.1 Proof of Work (PoW) . . . 32
3.3.2 Proof of Stake (PoS) . . . 32
3.3.3 Proof of Capacity (PoC) . . . 33
3.4 Smart Contracts . . . 33
3.4.1 Bitcoin Scripts . . . 33
3.4.2 Ethereum Smart Contracts . . . 35
3.4.3 Ethereum Tokens (ERC20) . . . 35
3.5 Funding Rounds . . . 36

4 Work Definition **40**
4.1 Motivation . . . 40
4.2 Objectives . . . 40
4.3 Methodology . . . 42
4.4 Economic Analysis . . . 43
4.4.1 Technology Resources . . . 43
4.4.2 Human Resources . . . 45
4.4.3 Break Even . . . 45

5 System development **49**
5.1 Architecture design . . . 49
5.1.1 Users . . . 50
5.1.2 Customers Flows . . . 51
5.1.3 UML diagram . . . 53
5.2 Implementation . . . 53
5.2.1 Contract Design . . . 53
5.2.2 The Graph Indexer . . . 62
5.2.3 IPFS . . . 63

6 Results 65

7 Conclusions and future scope 68

A Alignment with the Sustainable Development Goals (SDGs) 70

B Installation instructions 72

C User Manual 75

D Queries API 81

References 82

List of Figures

1 Home Page Dapp 4

2 BitRound Page 5

3 Time and cost of creating a business in economies covered by Doing Business report [2] 17

4 Ethereum Logo [7] 19

5 Solodity Logo [9] 20

6 Metamask Logo [14] 22

7 Diffie-Hellman key exchange concept with mixing paints [21] 26

8 Two-dimensional elliptic curve plot [22] 28

9 Scatter Plot of Elliptic curve mod2503 [19] 29

10 Address creation [19] 30

11 Transaction Diagram [27] 31

12 Standard scripts stack structure [19] 34

13 Million of euros raised by startups in Spain since 2014 [39] 38

14 Kanban board 43

15 Fee vs. BitRounds for break even 47

16 Architecture design Diagram 50

17 User flows [48] 52

18 User flows with the contracts [49] 54

19 GraphQL APIs developed with GraphQL Voyager 62

20 Home Page Dapp 66

21 Ethereum Energy Consumption Index, Source: digiconomist.net 70

22 Metamask connection wiht the Dapp 75

23 Metamask connected 75

24 Create BitRound Page 76

25 Add Information view 76
26 BitRound Page 77
27 Successful Contribution 78
28 New Round Card 79
29 New Request view 80
30 Request Card 80

List of Code Blocks

5.1 BitRound Factory 56
5.2 BitRound variables 58
5.3 BitRound contract Functions 60
5.4 BitRound contract Functions 2 61
5.5 JSON CIDs 64
B.1 deploy.js 73
B.2 factory.js 74
D.1 GraphQL Queries 81

1 Introduction

We live in a capitalist world in constant evolution and development due to technology. Currently, it is easier than ever to create and destroy wealth; the average time and capital to create a company have plummeted[1]. The evolution of technology in the past 20 years has made it easier to register a business online, intercommunicate with the world with almost no costs, operate without a physical location, and the entrepreneur community is growing fast. All these changes have led to a growth in the creation of start-ups in every country[2]. We are going to focus on Spain and U.S. primarily.

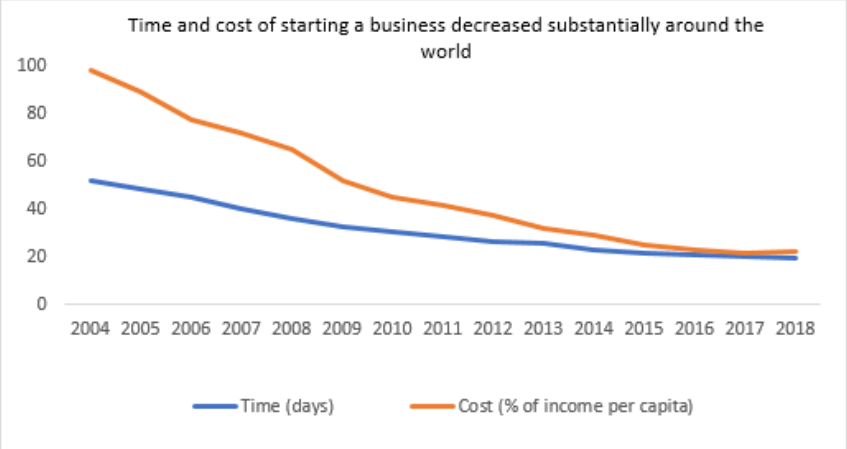

Figure 3: Time and cost of creating a business in economies covered by Doing Business report[2]

Traditional funding rounds are lengthy and costly, with complex processes and limited access to small investors. Entrepreneurs seek rapid investment to enter a competitive market where time is crucial to success. Although new ways of funding a start-up have emerged, such as crowdfunding and ICOs[3], they have many

fraud cases that keep funders away from investing[4]. Blockchain is a transparent, anonymous, and reliable technology that helps us to solve many of the current problems in start-up funding.

This final project aims to explore the use of blockchain in funding rounds, especially in implementing the technology in the early stages of a company where access to a broader pool of investors and reduced costs is critical. This work will also explore ways to solve fundamental investment issues through smart contracts that secure investment. Additionally, the project will determine the potential for blockchain to democratize the network for small investors and other benefits of using Blockchain networks.

Blockchain is an emerging technology that has snowballed in recent years. We know blockchain from cryptocurrency, but many fields, such as finance, cyber-security, and real estate, already use this technology. Funding rounds must be transparent and cheaper for all investors, and blockchain solves these problems. Smart contracts properly manage all funding rounds and meet the requirements of the entrepreneurs.

2 Description of the technologies

For the development of our work, we will use several technologies. We will launch our project in the Ethereum Network and code our smart contracts in Solidity. The blockchain network will provide transparent data, anonymity, and efficiency to our platform. Later, we will need to develop a decentralized application (Dapp) where the users can launch funding rounds and invest in startups. For the front end of our application, we will use React.

2.1 Ethereum

Ethereum is the primary tool for our work; it is a decentralized, open-source network that allows us to launch Smart Contracts through Blockchain methodology [5]. Vitalik Buterin, a Russian-Canadian programmer, conceived and co-founded Ethereum in 2013 to decentralize the web. Ethereum network replicates Bitcoin, created in 2008 by the inventor of blockchain Satoshi Nakamoto, and adds Smart Contracts that allow programmers to launch decentralized apps (Dapps). The cryptocurrency of the network is Ether (ETH). Because it is open-source, there are several Ethereum networks, which means we can fork the code and create our private blockchain network that allows access only to specific users. [6]

Figure 4: Ethereum Logo [7]

People use Ethereum over Bitcoin because of the velocity of transactions and smart contracts. Bitcoin mines the blocks through Proof of Work, while Ethereum uses

Proof of Stake, a more efficient and economical way of mining. Ethereum, in the beginning, used Proof of Work but changed its consensus mechanism architecture in 2022 to make the network more secure, less energy-sensitive, and better for scaling.

2.2 Solidity

Solidity is an object-oriented programming language in which we will code Smart Contracts. Several blockchain platforms use Solidity to code their smart contracts, making it the most popular programming language for blockchain. Solidity is familiar to web developers because it uses a syntax similar to ECMAScript. Therefore, its syntax is similar to languages like javascript, although there are some differences, such as Solidity has static types and variadic return types.[8]

Figure 5: Solodity Logo[9]

Solidity is a new growing programming language; consequently, it is constantly changing and needs consolidated libraries, making it more challenging to maintain the code and update deprecated libraries. Another key of Solidity is its immutability, meaning we cannot modify an executed contract.[8]

2.3 React

React is an open-source Java scripting library that allows the development of user interfaces for web applications. It aims to facilitate the development of single-page applications. Facebook and the community maintain the code.[10] The benefits of using this programming language are the fast learning and development of web

applications, which I needed for this project. It also has compatible libraries such as semantic ui and material ui, which are frameworks that allow us to improve the aesthetics of our app.

2.4 Web3 JS

Web3 JS is an open-source JavaScript library that allows us to interact with contracts as JS objects. Web3 JS has other features, such as listening for on-chain events and built-in utilities that make it easier to interact with contracts.[11] We chose this library because it is the gateway between our React App and the Ethereum network.

2.5 Infura

Infura is a high-availability cloud service suite that provides infrastructure nodes for developers working on projects based on the Ethereum blockchain. In essence, we use Infura because it allows for a simple interface and API for developers to connect to the Ethereum network without setting up and maintaining their infrastructure.[12] Infura is the blockchain node we connect with to deploy contacts, call functions, and send transactions. Infura saves time for developers because it handles the complexity of running an entire Ethereum node.

2.6 Metamask

Metamask is a wallet extension for your navigator that allows users to interact with web Dapps based on the Ethereum blockchain. Metamask, like every digital wallet, allows users to send, swap, receive, and store cryptocurrencies and Ethereum tokens. It is the most popular digital wallet.[13]

Figure 6: Metamask Logo[14]

2.7 Remix IDE

Ethereum IDE is a toolset that allows developers to code, compile, debug, and deploy contracts in the Ethereum network or public remix networks forked from Ethereum.[15] This public test networks allow developers fast deployment and contract interactions. In this development, I have used the web version of Remix IDE, which requires no setup and has an intuitive user interface for users of any knowledge level.

2.8 Visual Studio Code

Visual Studio Code is a free and open-source code editor developed by Microsoft. It is one of the most popular and widely used code editors today due to its flexibility, extensibility, and ease of use. VSC supports many programming languages, including Solidity and JavaScript, and its plugins allow users to personalize the code editor. VSC fits this project because it is a light and fast program.

2.9 GitHub

GitHub is a collaborative development platform where developers push their coding projects. GitHub uses the Git version control system, which means developers can track changes to source code over time, make backups and revert changes if necessary.[16]

2.10 IPFS

InterPlanetary File System (IPFS) is a protocol and peer-to-peer network designed for storing and sharing data in a distributed filesystem[17]. Juan Benet founded IPFS in 2015 to replace the traditional Hyper Text Transfer Protocol (HTTP) with a more decentralized, secure, and efficient system. IPFS leverages the same concepts underpinning blockchain technology: decentralization and data distribution. Each file and all its blocks has a unique fingerprint called a cryptographic hash. Since it is decentralized, it is robust against attacks and avoids the risk of central points of failure. Moreover, IPFS minimizes bandwidth use by fetching data from the nearest node available in the network rather than from a centralized location.

2.11 Graph Protocol

The Graph Protocol is a decentralized indexing protocol for querying networks like Ethereum and IPFS[18]. Anyone can build and publish open APIs, named subgraphs, that applications can query using GraphQL. The Graph enables developers to access Blockchain data efficiently without centralized servers. It aims to create decentralized applications that interact with each other without intermediaries, providing a truly open and trustless environment for users.

3 State of the art

3.1 Cryptography

Before we begin explaining blockchain, we must understand the cryptography behind it and how it is implemented. Blockchain and all other networks leverage current cryptography knowledge to create solid and secure networks.

3.1.1 Hash Function

A hash function is any function that maps a given input into a fixed-length result. The input size can be any size, but the output will always have the same length. Given an input, the output is always the same. Thus, you can use the hash output as a digital fingerprint. The three main properties of a hash function are:[19]

1. Working out the original from the output is impossible.
2. The same input always returns the same output.
3. No collisions: different inputs return different outputs.

The three main uses of hashes in blockchain are:

1. Transaction hash: Creates a unique identifier for each transaction.
2. Block hashes: Create a unique identifier for each header block and, in Bitcoin, due to its randomness, allows the mechanism of mining. Each hash block header is included in the following block header to securitize the blockchain.
3. Addresses: The public key is hashed with SHA-160 to create the address in Bitcoin. It provides an extra security layer and makes the address shorter because the output length is shorter than the public key.

3.1.2 Asymmetric cryptography

Asymmetric cryptography, also known as Public-key cryptography, is the field of cryptography that uses a pair of keys to encrypt and decrypt data. One-way cryptographic functions generate the private and public keys, and the security of the encryption, as always in cryptography, is based on the secrecy of the private key.[20]

Asymmetric cryptographic system was created because of the need for a rapid key exchange, the lack of secure channels, and the increased number of users. Before public key cryptography, there was only symmetric cryptography which uses the same key to encrypt and decrypt[20]. Symmetric cryptography requires a prior exchange of the keys in a secure channel; by contrast, in the Diffie-Hellman key exchange, the public keys are spread in the network, and the secrecy is not compromised. The shared secret is computed offline based on the shared public key of the partner and your private key; see key exchange concept with paints figure 7.[21]

Anyone with the public key can encrypt a ciphertext, but it can only be decrypted with the private key, known as public key encryption. It also can be used for digital signature; the private key owner can generate a message's signature, and anyone with the public key can verify if the message is signed with the original private key[20]. This signature exchange is implemented in Bitcoin when someone wants to unlock Bitcoins that belong to an address. An address in Bitcoin is the public key (see 3.2.1), and someone demonstrates that an address belongs to him by providing the signature. The owner does not provide the secret key and demonstrates the ownership of the address to the network.[19]

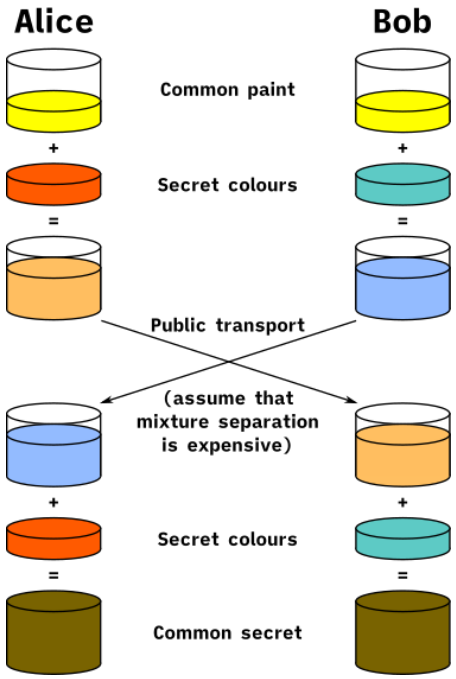

Figure 7: Diffie-Hellman key exchange concept with mixing paints[21]

3.1.3 Private Key

A private key is just a random number. To create a private key, you need a reliable source of randomness. In the case of bitcoin, it is a 256-bit number, meaning there are more private keys than atoms in the universe (2^{256}). Therefore, the probability of two people choosing the same private key is negligible. You should never share your private key because it is the seed to participate in the network, and somebody can replace you if he has your private key.[19]

3.1.4 Public Key

The public key is your identifier in the network, and it is like a mailbox where anybody can send you whatever they want. You unlock whatever they send to you with your private key. The public key is created with your private key. There are different methods to create a public key from a private key. Bitcoin uses ECDSA

(Elliptic curve digital signature algorithm), which implements DSA (Digital Signature Algorithm). DSA was released by the National Institute of Standards and Technology (NIST) as the standard for digital signatures.[19]

To create a pair of keys, we use elliptic curve multiplication:

$$d\ G = Q$$

$$d \textit{ is private key}$$

$$G \textit{ is the geneartor point}$$

$$Q \textit{ is the public key}$$

A private key is an integer number, while Q and d are points in the elliptic curve. Getting the private key from the public key is impossible, but it is easy to do the elliptic multiplication to get the public key.[22]

3.1.5 Elliptic Curves

Elliptic curves are described by the set of solutions (x,y) for $y^2 = x^3 + ax + b$; for some coefficients a and b.[22] For instance, a=0 and b=7 are the coefficients specified in secp256k1[19], the elliptic curve implemented in bitcoin. The elliptic curve also needs to define prime modulus, and any number will be within that range when we perform mathematical operations. The fact that a modulus is a prime number is crucial for cryptography, mainly based on number theory and the hardness of prime factorization of large numbers.[23]

As we mentioned before, every elliptic curve has a generator point G, which determines the starting point for all the operations in the curve, such as elliptic

multiplication, to calculate the public key.

The elliptic curve function plotted in a two-dimensional diagram looks like figure 8. However, in cryptography, we want to use a finite number of points of whole numbers. Therefore, the elliptic function used looks like a scatter plot. Secp256k1 looks more like the scatter plot 9, which scatters an elliptic curve of mod 2503 (the elliptic curve scatter plot of bitcoin would have 2^{256} points).

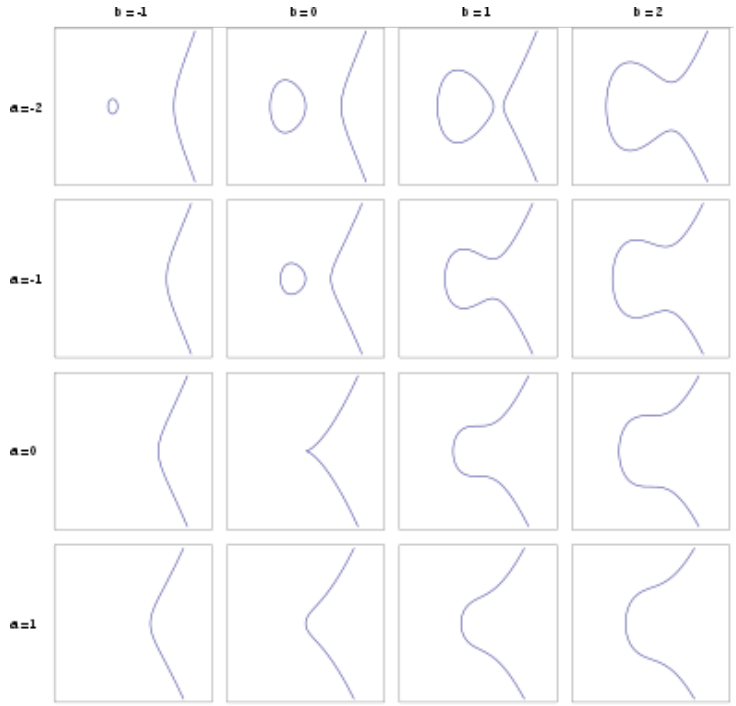

Figure 8: Two-dimensional elliptic curve plot [22]

For computing, using whole numbers in a finite field is easier than real numbers in a continuous area. The accuracy when you work with decimal numbers is lower than working with whole numbers. Therefore, whole numbers suit better in cryptography. [19]

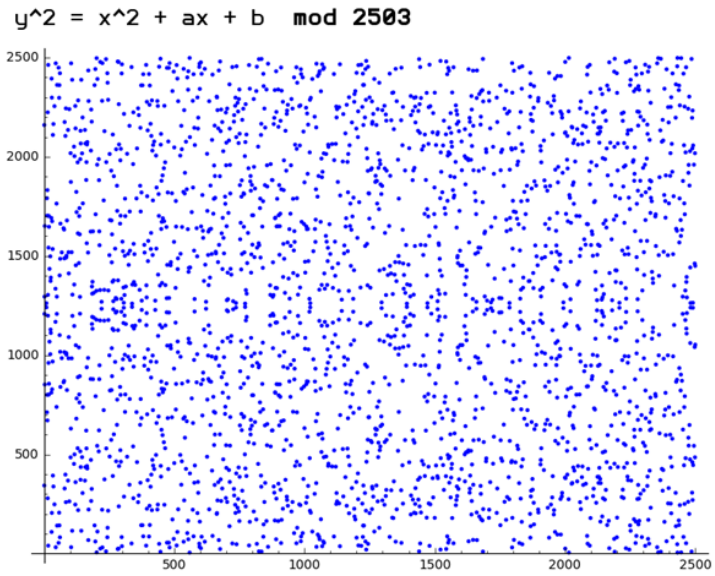

Figure 9: Scatter Plot of Elliptic curve mod2503[19]

3.2 Blockchain

Blockchain emerged in 2008 with the release of Bitcoin by Satoshi Nakamoto[24]. At the beginning of Blockchain, it was only used for transacting cryptocurrencies, and the goal of Satoshi, as he states in Bitcoin's paper, was to create a monetary system where banks were no longer necessary. Years later, thanks to the release of smart contracts, blockchain was introduced in many fields, such as finance, identity providers, certifications, bureaucracy processing, etc. Although blockchain technology seems complex, I will summarize it, explain its purpose, and how it works.

Blockchain is an advanced decentralized, transparent database that allows everyone in the network to interact with its information. Blockchain creates an immutable log that is governed by the users of the network, who are responsible for the verification of the transactions.[25]

Although many features vary in each blockchain, I will explain the most common characteristics of blockchain with a particular focus on Bitcoin (the first blockchain) and Ethereum (the first smart contract blockchain and the one I use for this project).

3.2.1 Addresses

Addresses are used to avoid sharing the full public key with the network. "An address is a shorthand way of writing blocking scripts in a human-readable form." - echeveria (on IRC). Each blockchain network uses its transformation to obtain the address, and hash functions are used to reduce the size of the public key. In Bitcoin, we create the address by adding a prefix and a checksum to the end of the public key hash (Figure 10). It is then converted to Base58, which makes the address more human-readable. With these changes, we avoid errors when sharing our addresses.

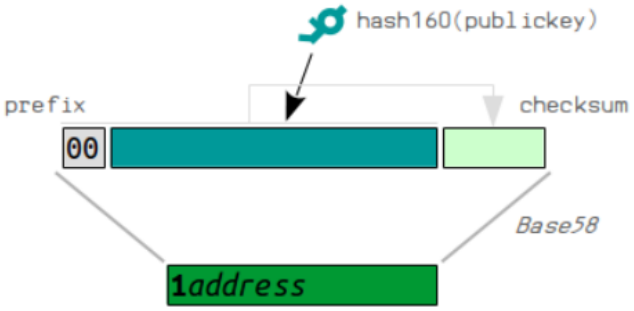

Figure 10: Address creation[19]

3.2.2 Transactions

A transaction is a transfer of value in the Blockchain[26], in Bitcoin, is the transfer of the cryptocurrency between two accounts. In other Blockchains like Ethereum,

the concept of the transaction is broader because the transfer of value can be between accounts and contracts. The unique identifier of a transaction is the hash of the transaction's data.

In Bitcoin, the transaction has at least one input and one output. All the bitcoins transferred are from outputs of previous transactions, and all the inputs included in a transaction must be spent. In other words, if you have a previous output assigned to your address and want to send the money to another account, the remaining input you do not send must be sent to your account again [19]; otherwise, those bitcoins would be lost. See figure 11 for the transaction structure.

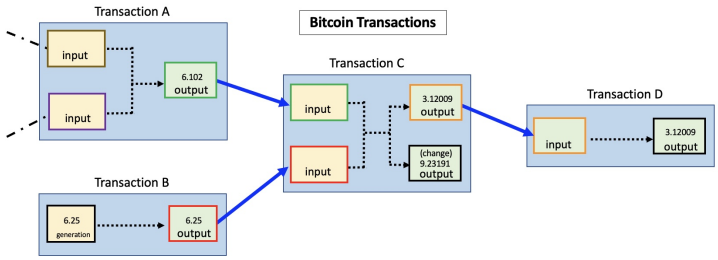

Figure 11: Transaction Diagram [27]

3.2.3 Block

The blocks are the data stored in the Blockchain. The block content varies depending on the Blockchain implementation. Usually, the data stored is a log of all transactions.

When a user publishes a transaction, it is sent to a network node, and the node shares the transaction with the rest of the nodes. The transaction is stored in a memory pool containing all the transactions not in the Blockchain yet. Every node selects transactions from the memory pool and tries to mine the block. If the mining is successful, the candidate block is added to the network, and the node

receives a reward[19]. There are different ways to mine a block; see section 3.3.

3.3 Consensus Protocol

The consensus protocol is the mechanism of the Blockchain network to achieve the ledger's consensus. Blockchain consensus is much more efficient than human verifiers and audits, and what is more important, the consensus is decentralized [28]. The blockchain avoids fraud and makes the data immutable thanks to consensus protocol. Some of the most essential consensus protocols are described below.

3.3.1 Proof of Work (PoW)

This consensus mechanism, used by Bitcoin and other cryptocurrencies, is the most primitive and consists of nodes solving complex mathematical problems to validate transactions and create new blocks. PoW was created as a technique to avoid email spam.[29] The first node to solve the problem is rewarded with newly minted cryptocurrency. This process is energy-intensive and requires significant computational power.[29]

3.3.2 Proof of Stake (PoS)

Proof of Stake (PoS) is an alternative consensus mechanism to PoW, designed to address the high energy consumption issue. In PoS, validators are chosen to create new blocks based on the number of coins they hold and other factors.[28]

This process requires much less computational power and is more energy-efficient.[28] Examples of PoS-based blockchains include Cardano and the newer versions of Ethereum.

There is a variation of PoS named delegated proof of stake, in which network participants vote for a limited number of nodes (delegates) to validate transactions

and create new blocks. This approach is more scalable and efficient than traditional PoS systems.[30]

3.3.3 Proof of Capacity (PoC)

Proof of Capacity (PoC) is another consensus mechanism that leverages available storage space instead of computational power or coin ownership. In PoC, miners allocate disk space to store solutions to mathematical problems, and the probability of mining a block depends on the amount of storage space dedicated.[31]

This method is more energy-efficient than PoW, but it may lead to increased storage costs for miners[31]. Burstcoin is an example of a blockchain that uses PoC.

3.4 Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They are designed to automate transactions and enable trustless, decentralized applications on blockchain networks.

Smart contracts provide increased transparency, security, and efficiency compared to traditional contracts.

3.4.1 Bitcoin Scripts

Although Bitcoin does not allow smart contracts, it incorporates a scripting language to facilitate transactions within its blockchain network. Bitcoin scripts can be considered the beginning of smart contracts. Although Bitcoin scripts might be less flexible and more basic than Ethereum's smart contracts, they still play a vital role in the Bitcoin network's functionality, enabling features like multi-signature transactions, timelocks, and conditional payments.[19]

A Bitcoin script is included in every output, and a Bitcoin script must always be provided to unlock an output. The data structure is a LIFO stack, not a Turing complete code. Therefore, it has limited capabilities. However, its simplicity makes the transactions more secure because it reduces the likelihood of vulnerabilities[19]. There are many standard scripts; the most common is P2PKH (Pay to public key hash), for which you must provide your public key and a signature to unlock the Bitcoin output. There are many standards like P2PK (Pay To pubkey), P2MS (Pay To Multisig), P2SH (Pay To Script Hash), and NULL DATA[19], you can see their stack structure in Figure 12.

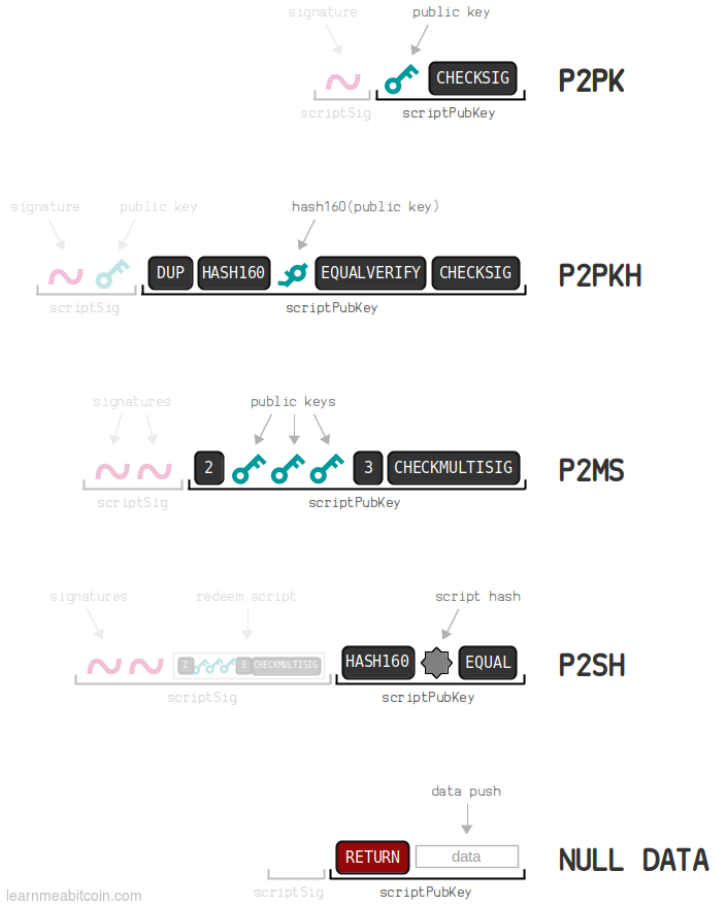

Figure 12: Standard scripts stack structure[19]

3.4.2 Ethereum Smart Contracts

Ethereum smart contracts are self-executing contracts that are stored within the Ethereum blockchain. Contracts are defined solely through code, so there is no need for intermediaries[32]. Smart contracts are executed by the Ethereum Virtual Machine (EVM), which runs from multiple nodes but it is a single entity for the entire network. Thanks to this, the Ethereum network is continuous and immutable[33]. In short, the EVM is a decentralized computer that supports the Ethereum network and is also used to execute code.

Smart contracts can be programmed in several languages, the most common is Solidity, but there are others, such as Vyper, Yul, and Fe, each with pros and cons[34]. All of them, unlike Bitcoin script, are Turing complete, which offers complex logic and allows developers to create decentralized applications (Dapps).

3.4.3 Ethereum Tokens (ERC20)

Ethereum smart contracts also offer the possibility to develop your token, which provides the same functionality as Ethereum's primary token Ether. The most adopted standard for developing tokens is ERC20, defining a common set of rules for tokens created on the platform. These tokens, commonly known as ERC20 tokens, follow the same protocol, ensuring compatibility between different tokens and decentralized applications (dApps), helping developers exploit all their functionalities without learning each token feature separately. The ERC20 standard outlines a series of functions, including how tokens can be transferred, queried for their total supply, approved to be spent by other addresses, and accessed for individual balances.[35]

By adhering to the ERC20 standard, developers can easily create tokens, as they share a standardized interface. This simplifies integration with various platforms,

such as wallets and exchanges, making it more convenient for users and developers. Overall, the ERC20 standard has been crucial in facilitating the creation of numerous tokens for various purposes.

3.5 Funding Rounds

A funding round is when a company or entrepreneur wants to raise money from one or more investors. Since 2008 the most common way to raise money for startups is funding rounds. Different rounds depend on the nature of the investors, the quantity, and the stage of the business.[36]

"Friends and family" or seed rounds create the startup. The seed investors are the founders themselves or "friends and family." Then, the second funding round for a startup is named the angel round. Angel rounds are early investments from investors not part of the company or related to the founders. It is when the company is in an early stage, and the risk is highest because the startup will probably not get positive cash flow in the beginning years. Investors or 'angels' are looking for high returns on projects when they are just an idea, prototype, or proof of concept. Most of the time, seed and angel round is the same.[36]

Venture capital firms lead the venture rounds. In this stage, the company is growing and has positive prospects for the future. They raise a large amount of money, between $1M and $30M, and letters name the series of stock sold, e.g., "round A," "round B," and so on. If the company reaches "D" or "E," it indicates it is not going as well as expected.[36]

Before going public or a significant merger or acquisition, the last stage for a startup is the mezzanine round, which is the last private funding.[37]

As described above, the traditional way of funding a startup is through funding

rounds. However, since 2017, initial coin offering (ICO) has become popular. An ICO is a new way of raising money wherein the company sells a digital cryptocurrency or token; it is the cryptocurrency's equivalent to an initial public offering (IPO). The first ICO dates from July 2013, and it is a common way of raising money for new blockchain startups. Sometimes this token does not represent a stake in the company; it may have a utility related to its product or service.[3]

People have heavily criticized the ICOs because of the many cases of fraud.[3] In fact, ICO activity began to decrease in 2019 because of the need for a legal framework that regulates this area. The ICOs are legal, but we might never recover funds lost due to fraud or incompetence because they are not regulated. The Securities Exchange Commission (SEC) warns investors of scammers that use "pump and dump" schemes[3], in which the scammers talk up the value of a cryptocurrency to pump up the price and then dump the value to get a profit. There are many cases in which the SEC has intervened an ICO when they consider it necessary, e.g., Telegram in 2019. We are still determining how long the legality of ICOs is going to last; for instance, the People's Bank of China banned ICOs in 2017.[3]

Another popular way of funding a company is via crowdfunding, which raises small amounts of money from a large number of investors[38]. It has become a popular way of entrepreneurship because of easy access to a large pool of investors and its popularization in media. Crowdfunding usually has a small minimum investment and is very popular for startups. Kickstarted, Seedrs, and Indiegogo are some of the most popular sites for crowdfunding.

In conclusion, there are five ways of investing in a company: ICO, crowdfunding, Angel Investors, Venture Capital Funds and Corporate Investors. ICO and crowdfunding are the newest and they are used for startups to raise rapidly money

and access to a broader pool of investors. Angel investors are individuals who seek for high returns in the early stages of a company. Venture Capital are funds that are looking for new companies with high potential, they invest on behalf of their owners. Finally, corporate investors represent the company which buys other companies, however, the companies invest also through venture capital funds.

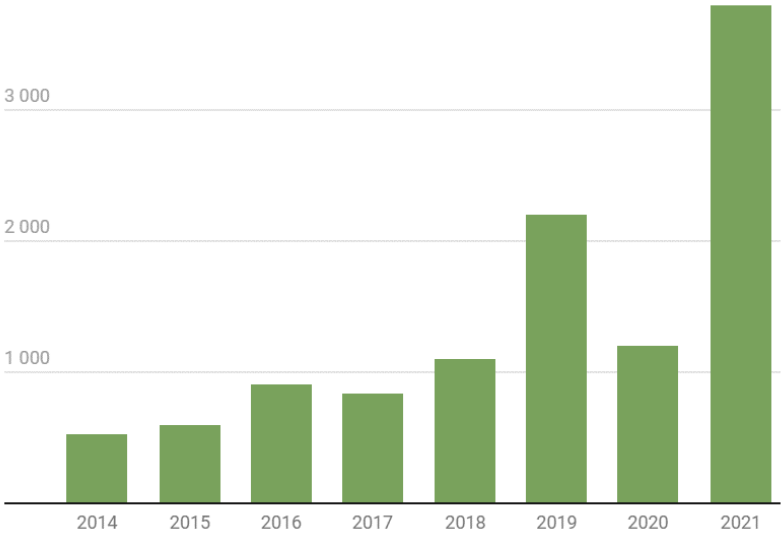

Figure 13: Million of euros raised by startups in Spain since 2014[39]

Nowadays, funding rounds are more popular than ever in Spain. Domestic startups raised more money than ever in history thanks to the boost of foreign capital. Funding rounds for startups boomed and are exponentially increasing over the last few years, as shown in Figure 1. These operations raised a record 3.8 billion euros in 2021. It is the highest data recorded in a single year and represents a 250% increase compared to 2020. The average amount of a round in Spain rose from 1.7 million in 2020 to 6.3 million in 2021.[39]

There is currently no platform for blockchain financing rounds. The development of blockchain applications is booming, and we can extract broad knowledge from the current open-source applications. Blockchain has given rise to many decentralized

applications, and many are looking for funding. There is no better place to fund a blockchain company than in a blockchain company, and there is where we take action. Therefore, we have a growing technology and a market which has yet to introduce this technology.

4 Work Definition

4.1 Motivation

The funding rounds need an update to the market's new necessities to meet the requirements of investors and entrepreneurs. Entrepreneurs seek lower commissions than current ones, investors seek transparency, and both seek security. There are a few things we want to change about the process:

Firstly, the fees are too high for the company. A startup needs to raise as much money as possible, and the costs of funding rounds reduce profitability and increase opportunity costs. Optimizing costs is one of the top priorities for a startup. Secondly, the minimum investment is a problem; for example, young investors with low capital cannot invest. Finally, need for more transparency; many startups falsify their accounts to get more investors. Blockchain creates an immutable log that shows all the investors and how much they have invested, avoiding fraud.

The application scalability is also attractive; with blockchain, the platform could add new features that help the startup manage its fiduciary relationship with shareholders. Future development of the application could add new functionalities ranging from shareholder voting to dividend distribution.

4.2 Objectives

We aim to develop a decentralized application that enhances the current funding round services. The three main objectives of this work are:

- Reduce funding costs.
- Transparent and anonymous platform.

- Improve fiduciary relationships.

One way of improving the services is to reduce the costs of the funding rounds; we can see that the commissions of the current platforms for financing rounds for startups in Spain are very high. Sego finance, the rebranding of the old financing platform Socios Inversores and one of the biggest funding rounds platforms, has a fixed cost of 1500€ and a commission of 6% over the raised funds[40]. Seedrs, a significant international funding platform, has fees of $2.500 fixed and 7.5% over the raised funds[41]. We can see that the commissions of the current companies are similar.

The minimum investment is also an issue for these platforms; low-income investors cannot be part of a financing round. Blockchain technology can reduce costs and let low-income investors participate, fostering a more inclusive investment environment. Through our final work degree, we aim to create a platform that eliminates entry barriers, empowering individuals to contribute to the growth of innovative startups.

Another objective is to make the financing round transparent and anonymous to provide reliability. With this, we avoid the recent cases of fraud in corporate accounts and ensure a secure environment for all parties involved. Our final work degree will focus on developing a blockchain-based solution that enables the purchase and sale of company shares, providing liquidity in the small business market and facilitating the startup valuation process.

Finally, we will improve fiduciary relationships between investors and entrepreneurs, protecting investors' capital from moral risk decisions and trying to align investors and entrepreneurs' objectives. By implementing blockchain technology, our final work degree aims to foster trust and collaboration, promoting a more sustainable

and successful startup ecosystem for all stakeholders involved.

4.3 Methodology

To develop our work, we will follow the steps described below.

First, we will research Blockchain using the deductive methodology to gain knowledge in this field. Then, we will explore Ethereum and Smart Contracts to learn how to apply this technology. After the research, we will deduct how to implement this technology specifically for this work.

Second, we will conduct a literature review of funding rounds, determining their main issues and how the current funding rounds face these challenges. This will provide us with a comprehensive view of the current landscape and help us identify improvement areas that can be solved with blockchain.

We will use Agile, the most common methodology in software development, to develop the application. Agile is a set of principles that uses an iterative scope to develop applications. We will use the Kanban framework, which gives us the rules to follow during application development. We visualize the work to see the application's process and progress. A Kanban board would look like in Figure 21.

After developing the prototype, we will evaluate and test its performance. We will consider its effectiveness, the problems it solves, and new problems that arise.

Finally, we will consider future advancements and opportunities to expand the platform, considering new challenges in funding rounds.

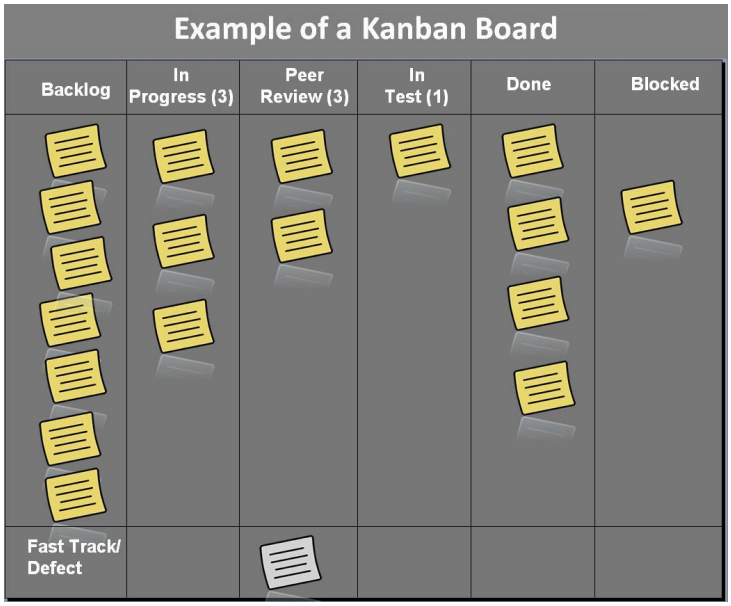

Figure 14: Kanban board

4.4 Economic Analysis

We will analyze the budget for this project, which is comprised of human resources and technology costs. After estimating the budget of this project, we will create a scenario in which we monetize the Dapp and calculate the company's break-even.

4.4.1 Technology Resources

The computer used to code the entire application is the HP Pavilion x360 14-cd0011ns, which cost 900€ 5 years ago. The specifications of the laptop are:

- CPU: Intel Core i5-8250U
- Graphic Card: NVIDIA GeForce MX130 (DDR3 2 GB)
- RAM: SDRAM DDR4-2400 12 GB (1 x 4 GB, 1 x 8 GB)

The Agencia Tributaria in Spain stipulates that the computer has a 4-year fiscal

life (35,040 h.). We have not used the computer exclusively to develop this app. Therefore we will estimate the cost of the computer based on the total work hours for this work.

Table 1: Breakdown of hours by task

Actividad	**Horas**
Organization, planning, meetings and consultations	5
Analysis, study and testing of technologies	100
Development	400
Testing	30
Documentation	300
TOTAL	**835**

$$\text{Coste por hora (Portátil)} = \frac{900€}{35040\,\text{horas}} \approx 0'0257\,€/\text{hora} \tag{1}$$

$$\text{Coste total (Portátil)} = 0'02576\,€/\text{hora} \times 835\,\text{horas} \approx 21'45\,€ \tag{2}$$

All the software used to develop this work is free and open source. Therefore, there is no extra cost related to the software.

The cost of deploying the contract is 2684603 gas, which is around 290€ [42]. Our Dapp needs an Infura node for hosting IPFS. This expense is the only maintenance cost we have. The pricing of the Infura node is 0.08$/GB/month for data storage and 0.12$/GB/month for data transfer [43]. Thus, the maintenance cost is variable.

Table 2: Breakdown of Development Costs

Cost	**Price**
Computer	21'45 €
Deploy Contract	290 €
TOTAL	**311'45 €**

Table 3: Breakdown of Variable Maintenance Costs

Cost	**Price**
Data Storage	0.08 €/GB/month
Data Transfer	0.12 €/GB/month

To make a profit, we can apply a fixed fee above the average space consumption or a percentage fee depending on the storage consumption of each user. However, the costs of infura are very cheap compared to a regular database because we estimate a storage of 10MB per BitRound. Thus, the price of creating a BitRound will be 0.002 €, $(0.08 + 0.12)\,GB/month * 0.01\,GB$.

Since it is a decentralized application, the entrepreneur will incur the rest of the costs of creating a BitRound.

4.4.2 Human Resources

For the development of this application, we needed a software developer whose salary is 29 344 € on average in Spain[44]. Application's development time is six months; thus, the human resources cost is 14 672 €.

Furthermore, we also need human maintenance costs for the application. The most logical way to address maintenance is by hiring a software developer for the year. Therefore, total maintenance costs account for 29 344 €, the salary of a software developer.

4.4.3 Break Even

Total development costs are 14 983,45 €. This amount is the initial investment needed for creating this project. Once we have developed this project, we need to earn money to make a profit. The fixed costs of each year are 29.344 €, and the variable costs are 0.002 € per BitRound.

The price of creating a BitRound in the Blockchain is around 2 million Gwei, around 200€[42]. We can add a fee of 10€, a fixed fee of 5%, making a profit of 10€ for each BitRound created. Thus, our break-even equation is:

$$0 = (Fee - Variable\,Costs) * BitRounds - Fixed\,Costs$$

$$0 = (10 - 0.002) * BitRounds - 29.344$$

$$BitRounds = 2935$$

The total number of BitRounds needed to break even is 2935. Kickstarter has 593.000 campaigns created since April 2009[45]. That is 42.357 campaigns a year. Therefore, we would need 14.43% of Kickstarter campaigns to break even. Kickstarter market share is 21%[46], so we would be competing approximately for $14.43/100 * 21 = 3\%$ of market share. It is essential to mention that, as stated in state of the art, this is a growing market, which favors us for an entrance.

Although 10 euros seems reasonable compared to competitors' fees, we can lower the price more and attract more clients without increasing the BitRounds needed to break even. We are going to plot the relationship between BitRounds and Fee given by the break-even equation:

$$0 = (Fee - 0.002) * BitRounds - 29.344$$

$$BitRound = \frac{29.344}{Fee - 0.002}$$

The x-axis is the Fee per BitRound in euros, and the y-axis is the BitRounds needed to break even. As we can see, between 2.5€ and 10€, the slope decreases slower than between 0 and 2.5€. Therefore, it might be interesting to decrease

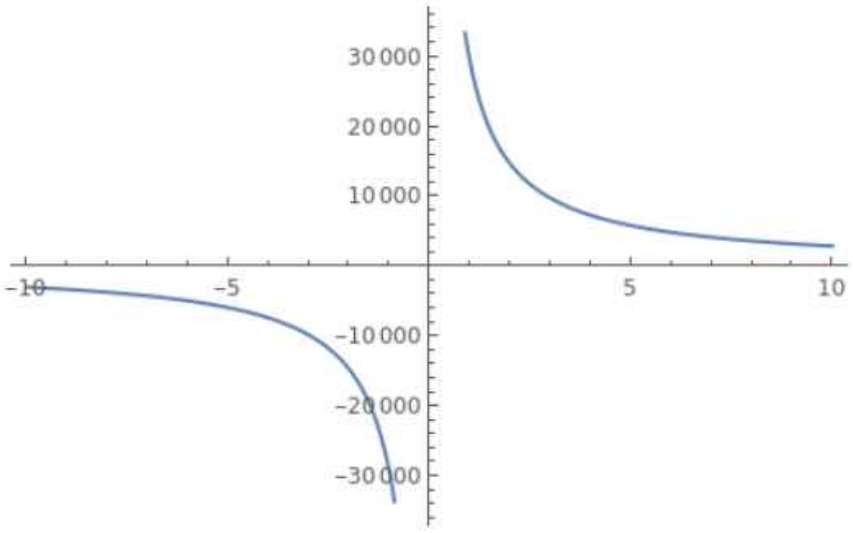

Figure 15: Fee vs. BitRounds for break even

even more the fee if we attract more clients. We need to carry out further analysis of the demand and supply curves to unveil which is the optimum fee. We also have to take into account that entrepreneurs are already paying 200 euros to deploy the BitRound in Ethereum Blockchain; there is little difference between 202.5 and 210 euros.

Our platform has high fixed fees at BitRound's creation that might keep some entrepreneurs away. We will compare ourselves to competitors like Kickstarter, where we only pay fees on the amount raised (around 5% in Spain[47]). Thus, our funding rounds are profitable compared to Kickstarter if the BitRound raises more than 4200€ (210/0.05). The investors in Kickstarter also pay fees when they invest in a campaign, between 3% and 5% in Spain[47]. In our platform, the investors pay around 230.000 of gas, which today is around 22€. Therefore, an investor is profitable to invest in our platform if they invest more than 550€ (22/0.04).

Other competitors like Sego Finance have fixed costs of 1.500 €; however, they provide more services which might be interesting for some entrepreneurs. We offer fewer services for less price. The rest of the competitors, not crowdfunding sites, are generally more expensive. However, they offer more financial services.

5 System development

In this section, we will talk about the development of our system, which is fully decentralized. For the development of the Dapp, we have used several technologies: IPFS, Ethereum Blockchain, The Graph Blockchain, and React. As it is not a centralized Web App, the user needs an Ethereum wallet like Metamask to log into it.

We chose this technology because it suits our app's necessities well. IPFS is a decentralized storage that makes our data more secure and resistant to censorship. Ethereum is a public log that gives our app transparency and anonymity. The Graph is an Ethereum indexer that helps us to organize and query information in the blockchain to show it to the final user. Finally, React allows us to develop an easy and fast front end for the final user.

In the rest of the section, we will explain the architecture and the structure of our code and data. Finally, we will explain the implementation in the front end of our Dapp.

5.1 Architecture design

Our application has a front-end and a back-end. The front-end is developed in React and creates an intuitive user interface that supports wallet connectivity to interact with Blockchain. With the react app we can interact with the contract and use all flows described below.

On the other hand, the backend comprises the following technologies: Ethereum smart contract, IPFS, and TheGraph. Each of them has specific functionality for our application; the smart contract records all the information of the financing

rounds and creates an immutable log, IPFS allows us to add information that complements the essential information of the financing round and that is not efficient to store in the contract and, finally, TheGraph allows us to create an API to search for information about the BitRound efficiently.

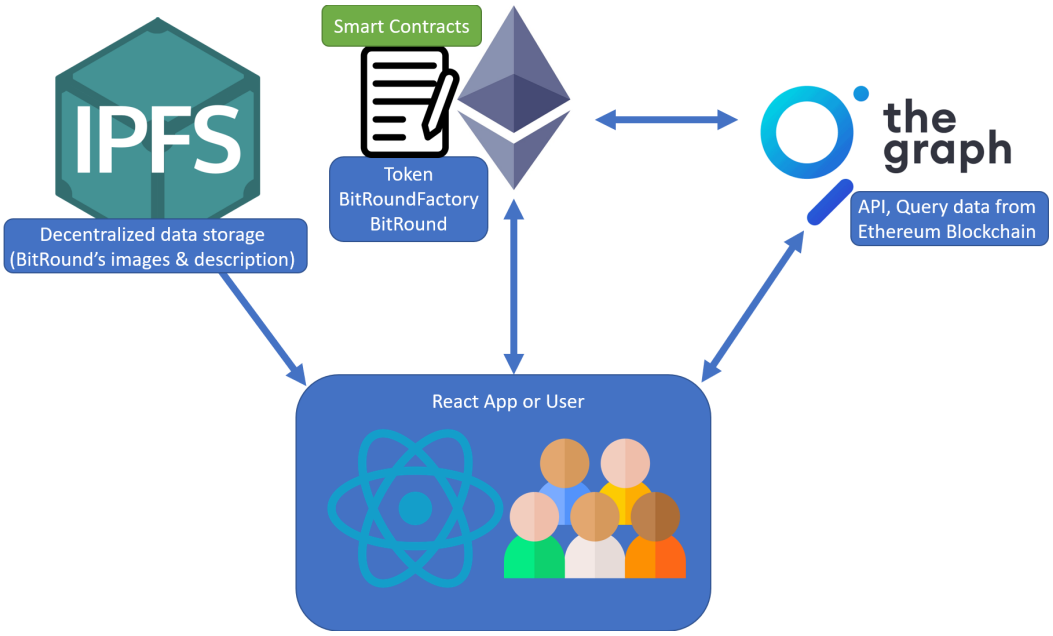

Figure 16: Architecture design Diagram

5.1.1 Users

There are two kinds of customers in our Dapp, the same customers a usual funding round web has. A customer can be an investor, entrepreneur, or both. The investors deposit money in the campaigns, and the entrepreneurs create the campaigns to raise money. In a founding web for start-ups like Kickstarter, the entrepreneur posts his company or project, and if he can raise enough money, Kickstarter sends all the money to the entrepreneur with a 6% fee and no control over the money. Our application with blockchain proposes an investment manner for the early stages of a company with lower fees and more control over the money

raised; it is a win-win for entrepreneurs and investors.

A third kind of user exists, which is a potential customer. Anyone can visit the Dapp on the internet and explore the different campaigns.

5.1.2 Customers Flows

In our applications, several flows for our customers exist. We plot them with the backend components involved, see Figure 17 diagram.

If we are an entrepreneur:

- Create a new BitRound. A BitRound represents a company for our Dapp.
- Modify BitRound. Each BitRound has a description and images to describe the company.
- Create a new Round. A company has many rounds to raise money, as we explained in state of the art section. The entrepreneur can decide when to start a Round and its duration.
- Create Request. In order to protect the investors, we have created a methodology based on requests to withdraw money. The investors approve or deny the spending of the entrepreneur.
- Finalize Request. Any participant in the blockchain can do this action; it costs money. Therefore, the entrepreneur will probably execute the function. If the Request has the necessary approved votes, this transaction will withdraw the amount requested from the contract.
- View BitRounds created.

If we are an investor:

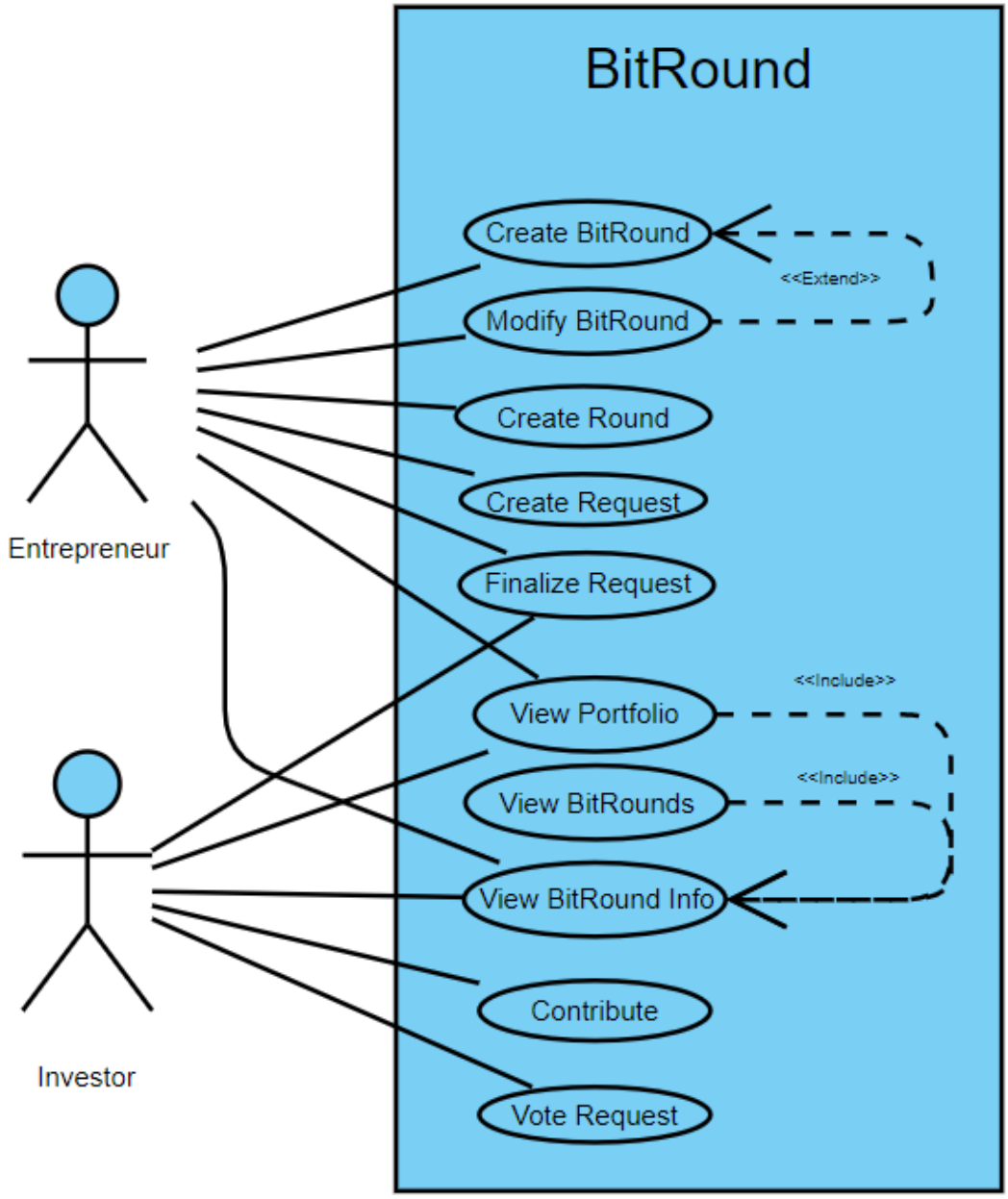

Figure 17: User flows[48]

- Contribute to a Round of a BitRound.
- Vote a request. Our vote is equivalent to your stake in the company.

If we are a connected user, which means we connected our wallet:

- We can access our portfolio view to find all the BitRounds we invested in and created.

Even though we are not connected, all users can view all the BitRounds created and their information.

5.1.3 UML diagram

The user can interact with the BitRoundFactory, BitRound, and Token contracts following the diagram in Figure 18. Through these interactions with the contract, the user can perform the functionalities described in section 5.1.2.

The diagram describes the interactions with the contracts for the User Flows of Create BitRound, Start Round, Contribution, Request, Approval, and Withdrawal. The rest of the interactions are view, which means is a simple call instructions.

5.2 Implementation

5.2.1 Contract Design

The smart contract coded in solidity allows us to manage the previous functionalities with anonymity and transparency and creates an immutable log. We have designed two contracts, a BitRound contract, which represents a company in the Blockchain, and a BitRound Factory contract, which creates and keeps a record of our contracts in the Blockchain. We implement the Factory for various reasons:

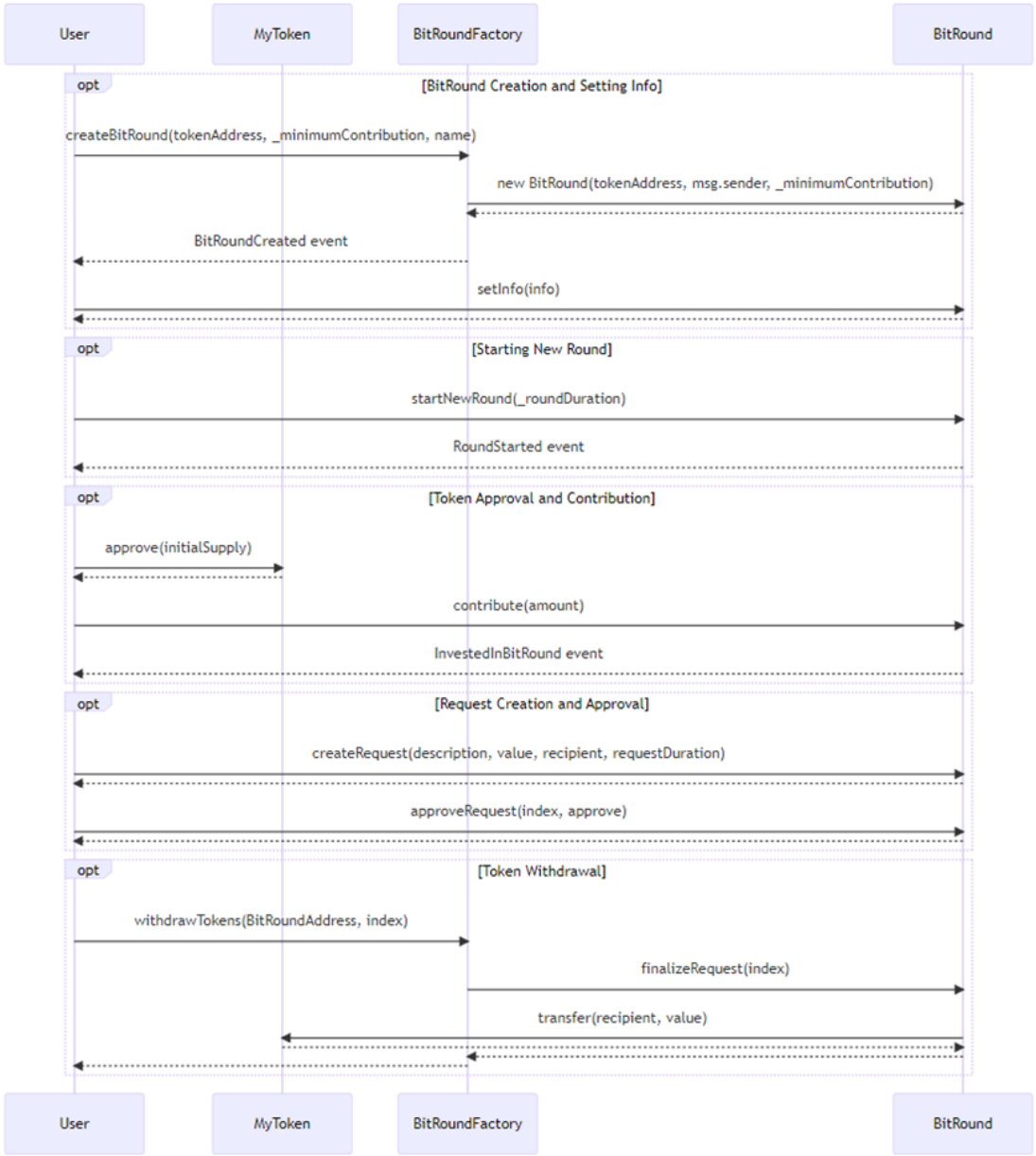

Figure 18: User flows with the contracts[49]

- Security: The customers know that every BitRound contract deployed in the Blockchain by our Factory will be the BitRound contract designed for this project. Otherwise, a malicious entity can deploy a BitRound contract, for instance, without security restrictions for withdrawal, which will lead to cases of fraud. By implementing the Factory, investors can rely on our contract because every BitRound created by the Factory is the original contract.

- Keep a record. The Factory, combined with The Graph technology, helps us keep a record of all the BitRounds created in our Dapp, making it easy to show them to the users.

- Creating a Factory for our contracts is a usual pattern to manage contracts in the Blockchain. Best implementations allow developers to update contracts. [50]

Code Block 5.1: BitRound Factory

```
contract BitRoundFactory {

    event BitRoundCreated(address indexed creator, address indexed bitRound,
    ↪  string name);

    function createBitRound(
        address tokenAddress,
        uint256 _minimumContribution,
        string memory name
    ) public {
        address newBitRound = address(new BitRound(tokenAddress, msg.sender,
    ↪  _minimumContribution));
        emit BitRoundCreated(msg.sender, newBitRound, name);
     }

    function withdrawTokens(address BitRoundAddress, uint256 index) public {
        IBitRound token = IBitRound(BitRoundAddress);
        token.finalizeRequest(index);
    }
 }
```

In the code of the factory 5.1, we observe that the factory contract is straightforward. It only has no variable, it is not necessary an array that keeps a record of all the created campaigns because we use The Graph indexer to store of all created campaigns. One function interacts with the BitRound contract: withdrawTokens. The function accesses the BitRound function finalizeRequest, which is ownable, which means that if our factory deploys a BitRound, our factory is the only one that can execute these two functions of the BitRound. This implementation provides an extra security layer for investors to avoid fraud through contract factories.

Every time a BitRound is created, the contract emits an event that will help us to query information later with The Graph.

The BitRound contract has the variables shown in code block 5.2. The first variable is the Token Address, which refers to the contract address of the token (currency) we want to use in our funding round. One of the problems we found when we developed the Dapp is that the investors have to invest in Ethers if we want to keep the money stored in our contract. This is an issue because cryptocurrencies are very volatile, and the price of the ethers can fluctuate too much, which is unsafe for investors and entrepreneurs. However, Ethereum allows the creation of tokens, which are cryptocurrencies, and deploys them through smart contracts. Some tokens, such as USDT, are pegged to the dollar. Our contract can use any personalized token: a pegged token to a FIAT currency, any token in the Ethereum network, or even the token company's design. The first option will solve the problem of the volatility of cryptocurrencies. The last option will be similar to IPOs, providing our contract's extra security and reliability layer.

Code Block 5.2: BitRound variables

```
1  contract BitRound is Ownable {
2      IERC20 public token;
3      uint256 public roundNumber;
4      uint256 public roundEndTime;
5      uint256 public roundDuration;
6      address public manager;
7      uint256 public minimumContribution;
8      uint256 public totalInvestment;
9      Request[] public requests;
10     string public BitInfo;
11
12     struct Request {
13          string description;
14          uint value;
15          address recipient;
16          bool complete;
17          uint approvalCount;
18          uint refuseCount;
19          uint256 roundEndTime;
20          mapping(address => bool) approvals;
21      }
22
23     struct Round {
24         uint256 totalContribution;
25         uint256 totalParticipants;
26     }
27
28     mapping(uint256 => Round) public rounds;
29     mapping(uint256 => mapping(address => uint256)) public participants;
30     mapping(address => uint256) public shareholders;
```

From lines 3 to 5 of code block 5.2, we find the variables related to the current round of the company, the number of the round, when the current round finishes, and its total duration in days. The manager variable in line 6 refers to the entrepreneur, the wallet address who created the BitRound. It has nothing to do with the owner,

who is in the factory contract because the owner is who deploys the contract. Therefore, if the owner of a BitRound is the factory contract's address, we know that BitRound is reliable. MinimumContribution variable defines the minimum contribution to our company; entrepreneurs might not want a too fragmented company. Line 10 is a string variable that is the CID in IPFS. That CID refers to a JSON in IPFS containing information about our company: text and images.

Line 9 is an array with all withdrawal requests; we created a Request structure containing all the relevant information. The request has a mapping of all the voters to check if someone has already voted, the amount, a short description, end time, and a counter of approvals and refusals. The Request structure also has a boolean parameter to mark the completion of a request.

There is a Round structure defined, which contains the total contribution and number of participants in each round. The evolution of the rounds in a company is vital to know its valuation by investors. We store all rounds in a mapping and create a hashmap of the participants of each round and their stake in each round. Finally, there is a shareholders mapping that stores each shareholder's stake in the company, which is relevant for the voting system and checking the stakes individually.

Code Block 5.3: BitRound contract Functions

```
constructor(address tokenAddress, address creator, uint256 _minimumContribution) {
        token = IERC20(tokenAddress);
        manager = creator;
        minimumContribution = _minimumContribution;
    }

    function startNewRound(uint256 _roundDuration) public restricted {
        require(roundEndTime < block.timestamp, "Previous round still ongoing");
        roundNumber++;
        roundDuration = _roundDuration;
        roundEndTime = block.timestamp + roundDuration * 1 days;
        emit RoundStarted(roundNumber, block.timestamp, roundEndTime);
    }

    function contribute(uint256 amount) public {
        require(block.timestamp < roundEndTime, "No ongoing funding round");
        require(amount >= minimumContribution, "Amount below minimum contribution");

        token.transferFrom(msg.sender, address(this), amount);
        if (participants[roundNumber][msg.sender] == 0) {
            rounds[roundNumber].totalParticipants++;
        }
        shareholders[msg.sender] += amount;
        participants[roundNumber][msg.sender] += amount;
        totalInvestment += amount;
        rounds[roundNumber].totalContribution += amount;
        emit InvestedInBitRound(msg.sender, address(this), amount);
    }

    function createRequest(string memory description, uint value, address recipient, uint256 requestDuration) public restricted {
        Request storage newRequest = requests.push();
        newRequest.description = description;
        newRequest.value = value;
        newRequest.recipient = recipient;
        newRequest.complete = false;
        newRequest.refuseCount = 0;
        newRequest.approvalCount = 0;
        newRequest.roundEndTime = block.timestamp + requestDuration * 1 days;
    }
```

In code block 5.3, we show some of the most relevant functions in our code. The function of contributing is to invest in the company. First, it checks the minimum contribution, and if the invested amount is higher than the minimum contribution, it transfers the token amount from the sender to the contract. The BitRound contract uses an IERC interface token to execute the transaction. Notice that the BitRound contract is executing this transaction. Therefore, the investor has to pre-approve the contract to spend the token on his behalf. We have to take this consideration into account in the front-end implementation. Furthermore, contribute controls the shareholder stake in each round and total.

Code Block 5.4: BitRound contract Functions 2

```
1     function finalizeRequest(uint index) public onlyOwner {
2         Request storage request = requests[index];
3
4         require(!request.complete, "Request is already finalized");
5         require(request.approvalCount > (totalInvestment / 2) ||
6
7                 (request.approvalCount > request.refuseCount) &&
8                 ((request.approvalCount + request.refuseCount) >
  ↪  (totalInvestment * 70 / 100)),
9                 "You need more votes!");
10
11        token.transfer(request.recipient, request.value);
12        request.complete = true;
13    }
14
15    function setInfo(string memory info) public restricted {
16        BitInfo = info;
17    }
```

Notice that finalizeRequest is Ownable, which means they can only be executed by factory contract in our Dapp. We implemented the Ownable restriction in finalizeRequest because if tokens are withdrawn, it is verified by our contract.

The function finalizeRequest checks if a minimum number of votes approves the request. If the request is approved, the contract transfers the amount from its balance to the recipient.

SetInfo and createRequest functions are restricted. Restricted functions are called without a factory contract and only by the manager.

Notice also that in contribute function, in the end, it emits an event with all the relevant information we want to store in our indexer with The Graph.

5.2.2 The Graph Indexer

First, we will explain the structure of all data we want to collect for our factory contract. The structure is similar to a usual SQL database. The Gaph helps us to structure the data chained in the Blockchain. Chain structure is helpful for immutability and security, but it is only efficient to search for information about contracts if we use an indexer.

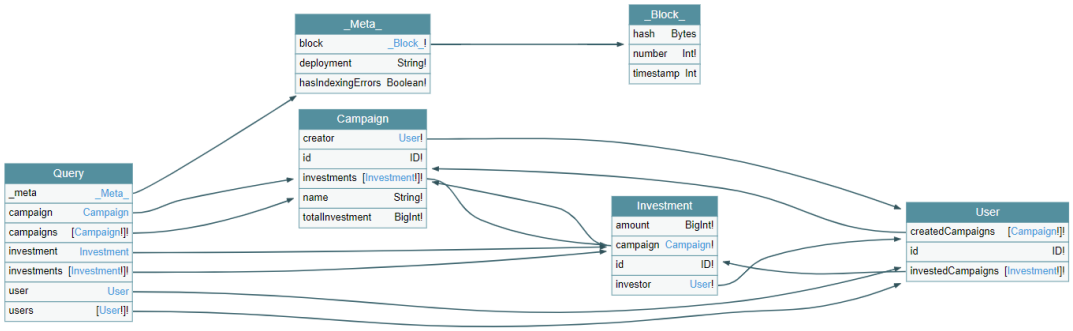

Figure 19: GraphQL APIs developed with GraphQL Voyager

Figure 19 visually represents all the endpoints we can query with GraphQL. There are three main tables of information: Campaign, Investment, and User. The table

named _Meta_ also collects all the metadata from the Ethereum Blockchain. The table _Meta_ has a collection of _Block_, which has the metadata of one block. On the left, we have the Query table from which we obtain all data and are in charge of querying data.

Notice the relationships between data; the User, for example, has a collection of createdCampaigns and a collection of investments. Thus, we can query for a user and get all its investments and created campaigns.

With structured data, we can make valuable queries. On the home page, we query the top 10 total invested BitRounds. We can only do this query in the Blockchain if we store this data specifically for each User, which is not cost-efficient in Blockchain. In our React application, we query using Apollo client; see code block D.1.

5.2.3 IPFS

For the final user, we wanted to implement a friendly user interface that shows information about the company, so we decided to add a description and images for each BitRound. Storing this data in the blockchain is costly. Therefore we needed another technology to store this data. We chose IPFS because it is a decentralized protocol, cheap, and many decentralized applications such as Uniswap and NFT Dapps use it.

To store the minimum data in the blockchain, we store just one CID, an address in IPFS, which stores a JSON with the CIDs of the description and images. Storing a CID in the contract and a JSON in IPFS with all the CIDs is the most efficient manner of storing data.

Therefore, the CID string variable will refer to a JSON stored in IPFS that looks

like code block 5.5. There are two fields, one description CID address where description text is stored and an image array with all the CIDs addresses where the images are stored.

Code Block 5.5: JSON CIDs

```
{
  "images": [
    "QmZ8mkNkfAu4J3pzc86ZhsHDUo5K9N1QNBcDyyhDU731xD",
    "QmWNuQkgt89Ai1HKGzxyY9kmkBth8bMdRjHt7mcxitEpXH",
    "QmdJR6aRLwCCXZbQvYZ5TX89tT2Qof4cA7gjH8Ed6ZXZM3"
  ],
  "description": "QmTLLwMY89G7UndTJEaoQa9LnHbJj43WnrCs9QBVhKcZRr"
}
```

Since IPFS is a decentralized protocol, anybody can edit this field and personalize its application individually. However, we also add a page in our application where anybody can edit these fields and personalize the BitRound page. Anybody can add as many images as they want, and the text is as long as the entrepreneur wants.

6 Results

We will review the objectives stated in section 4.2 of this work.

- The main objective was to create a decentralized contract that manages the company funding process. We have successfully achieved this objective by creating a secure contract that properly manages the whole process of funding a company.

- Reducing costs: the current costs of our Dapp are fixed compared to variable costs of crowdfunding sites, obove a minimum investment exposed in section 4.4.3 our Dapp is cheaper. Our application provides fewer services than current competitors; however, it is an interesting choice for the niche of start-ups who want to raise money fast and cheaply. Gas prices are increasing and fluctuate every day. Thus, prices of deployment also fluctuate. Ethereum is the most expensive network, 200 times more expensive than the second most expensive network. This goal is partially achieved.

- Minimum contribution barriers: This objective is fulfilled and democratizes the network to allow the entry of small investors.

- Transparency and anonymity: Blockchain technology implementation provides us with the tools to fulfill this goal. Ethereum creates an immutable anonymous log that registers all transactions of our application.

- Fully decentralized app: We have created an application where intermediaries are unnecessary, and we need an Ethereum wallet to raise and invest money. We achieve this objective by implementing blockchain, IPFS, and The Graph.

- Improve fiduciary relationships: Apart from transparency, immutability, and

anonymity, we have developed a smart contract specialized for a secure investment. We achieve this objective by implementing a secure smart contract and providing tools like withdrawal requests [30] [29] that protect the investors from fraud. However, there is still work to be done in this area since we can implement more tools that increase the security of the investments.

In this section, we show the final Dapp for the user, explaining its flows and how to use it. As we said, we developed the user interface with React.

Figure 20: Home Page Dapp

The results shown in the user manual [C] are screenshots of our front-end application. However, it is essential to note that anybody can execute all functionalities without our front-end application since it is decentralized. Even they can personalize their posts in our Dapp without using it. Therefore the main purpose of this work is fulfilled, since we have created a fully decentralized app that does not depend on an intermediary. Executing all the functionalities from our Dapp

is recommended since it is more intuitive.

7 Conclusions and future scope

Overall, we have achieved the main goals since we created a decentralized web application that appropriately manages funding rounds with blockchain. We had to research for new technologies to deploy a fully decentralized Dapp that is user-friendly and do not reduce user experience due to its decentralization.

In terms of cost, the system has proven to be cost-effective at times compared to traditional funding rounds, as exposed in section 4.4.3. Our application is cheaper sometimes, however the entrepreneur needs to decide if it is interesting to pay less fees and get less services.

In terms of accessibility, the system has proven to be accessible to a wide range of investors. The decentralized nature of blockchain allows anyone with internet access to participate in funding rounds, which democratizes the investment process.

All the crucial BitRound information is public, anonymous and secure in our network. The fraud is waived with the request of withdrawal system that provides an extra security layer. We can say that we achieved the security and anonymity of the platform by implementing smart contracts.

We have achieved the main objectives of this work, but there are still areas of improvement and new scopes.

The most important issues of the Dapp taking into account feedback from people working in finance:

- Time value of money: The money invested in a company is locked until a successful request occurs. That money could be staked in an asset and produce a profit while it is locked in the BitRound. Finding ways to invest

that money to get returns while it is locked might be interesting.

- Updating the smart contract: It would be interesting to implement a contract that can be updated so it does not become deprecated. The immutability of the blockchain has the counterpart of updating the contracts.

- Legal environment: Every company needs a legal environment to participate in our society. Our application needs a legal framework which is vital for a company.

- Although the objective of this work is not to make our Dapp profitable, It is interesting to find ways of making a profit with this Dapp. Although everybody can post its BitRound in a decentralized manner, the IPFS node, which is implemented in our React app, is hosted by us, which means it costs money to us. We use IPFS so entrepreneurs can personalize their BitRounds with a description and photos. We can ask for a fee for personalizing since hosting an IPFS node costs us money. It is important to note that any entrepreneur could still personalize from outside our React app their BitRound via IPFS and Blockchain, so it is optional to use our React app to create a complete BitRound.

- We have implemented our smart contract in Ethereum since it is the Blockchain with more information for coding smart contracts. We can explore another Blockchain that might be cheaper or provides us with more capabilities. In fact, Ethereum is the network with higher cost, there are alternatives with 200 times lower gas prices like Binance Smart Chain. In BSC we can deploy contracts coded in Solidity, therefore, our smart contract would be compatible and we reduce the costs by 200 times.

A Alignment with the Sustainable Development Goals (SDGs)

Ethereum moved from Proof of Work to Proof of Stake, which makes the network cleaner and aligned with goal number 13: climate action. Mining the blocks in Ethereum was very polluting; however, the advances in blockchain have led to a cleaner mining method. Proof of Stake, introduced by other blockchains such as Cardano, is 99% cleaner than Proof of Work. Ethereum announced this change many years ago by the name of 'The Merge', and in 2022, they made it real.

The Merge has an actual impact on the world; Ethereum's electricity footprint was around 8.5TWh per year, similar to electricity consumption in Bangladesh, and overnight it dropped to 0.01TWh, see Figure 5. This fact clearly shows the intention of blockchain to be sustainable and clean technology.

Figure 21: Ethereum Energy Consumption Index, Source: digiconomist.net

Proof of Stake enables all the users to participate in the governance of the network. Before The Merge, there was a capital barrier entry if we wanted to participate in

the mining of blocks. Now, we can participate with as much capital as we wish to and receive the yield in return. Ethereum has removed the minimum investment for mining, which aligns with goal number 10: reducing inequality.

The efficient use of resources is also a principal matter in the industry, and we should achieve this goal through technological progress. It is crucial to develop an industry that is accessible to anyone and sustainable. We want to achieve this through an open-source code that enables everyone to participate in the development of the industry, allowing specialists and experts to be part of this project and add value to society with their knowledge. This way, we achieve goal number 9: 'Industry innovation and infrastructure.'

Ethereum is constantly updating its code not only to bring the latest technology to the market but also to achieve sustainable goals that leave the world better than we found it.

B Installation instructions

Copy the repository from: https://github.com/lastriita/BitRound. You need to have installed Node.js. Then go to the home folder and run 'npm install' to install all the libraries.

Then deploy your own contract BitRound Factory. First, you compile the contract by running 'npx hardhat compile'. Then you need to deploy the contract in the blockchain by using the js file deploy B.1. After you create deploy.js, run 'npm deploy.js'.

Create factory.js B.2 and add the address of the contract factory you deployed.

Run 'npm run dev' to initialize the Dapp in local. You need to have metamask installed to interact with the contracts in your navigator.

Code Block B.1: deploy.js

```
const HDWalletProvider = require("@truffle/hdwallet-provider");
const Web3 = require("web3");
const compiledFactory =
↪  require("./artifacts/contracts/BitRound.sol/BitRoundFactory.json");
const compiledToken =
↪  require("./artifacts/contracts/BitRound.sol/MyTestToken.json");

const provider = new HDWalletProvider(
  'your word seeds',
  'your infura node'
);
const web3 = new Web3(provider);

const deploy = async () => {
  try {
    const accounts = await web3.eth.getAccounts();

    console.log("Attempting to deploy from account", accounts[0]);

    const result = await new web3.eth.Contract(compiledFactory.abi)
    .deploy({ data: compiledFactory.bytecode })
    .send({ gas: "3000000", from: accounts[0] });
    //, gasPrice: web3.utils.toWei("10", "gwei")

    console.log("Contract deployed to", result.options.address);

    const result2 = await new web3.eth.Contract(compiledToken.abi)
    .deploy({ data: compiledToken.bytecode,  arguments: [100] })
    .send({ gas: "3000000", from: accounts[0] });

    console.log("Token deployed to", result2.options.address);

    provider.engine.stop();
  } catch (error) {
    console.error("Error during deployment:", error);
    provider.engine.stop();
  }
};

deploy();
```

Code Block B.2: factory.js

```
import web3 from "./web3";
import campaignFactory from
↪  './artifacts/contracts/BitRound.sol/BitRoundFactory.json';

const instance = new web3.eth.Contract(
    campaignFactory.abi,
    //your contract address
);

export default instance;
```

C User Manual

A user can visit our Dapp whether logged in or not. There is only a one-page view that a not logged user cannot visit: the portfolio page. A not logged user can visit the rest of the pages, but they obviously cannot modify any data; they can only view the content.

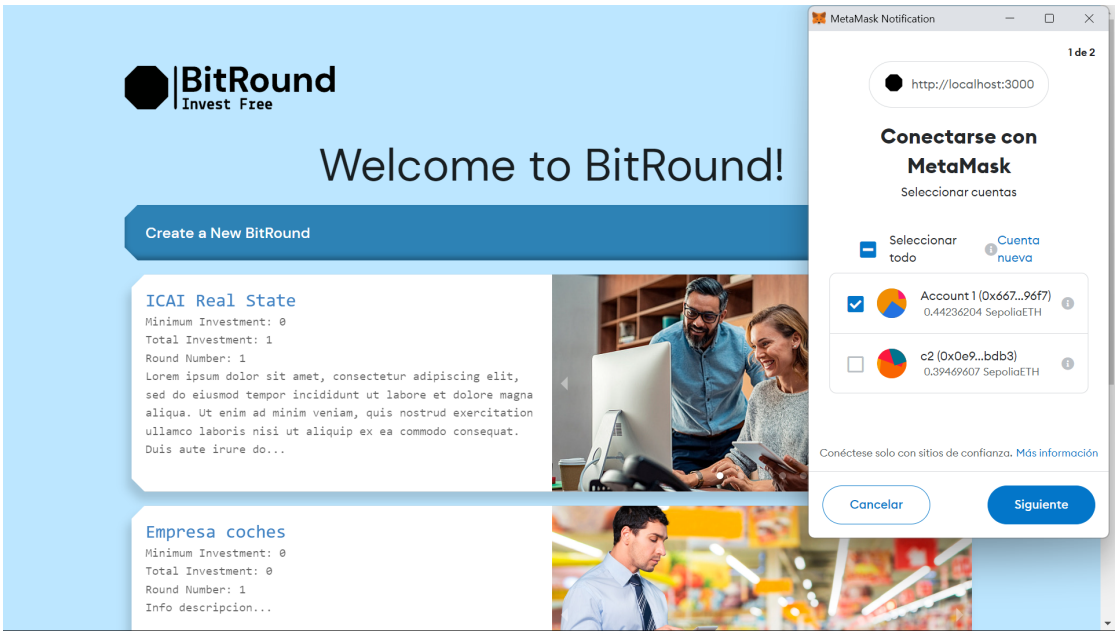

Figure 22: Metamask connection wiht the Dapp

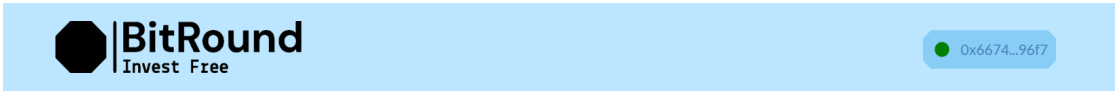

Figure 23: Metamask connected

In Figure 20, we see a screenshot of the homepage of our Dapp. This page shows all the BitRounds listed, a button to create a new BitRound, and a login button.

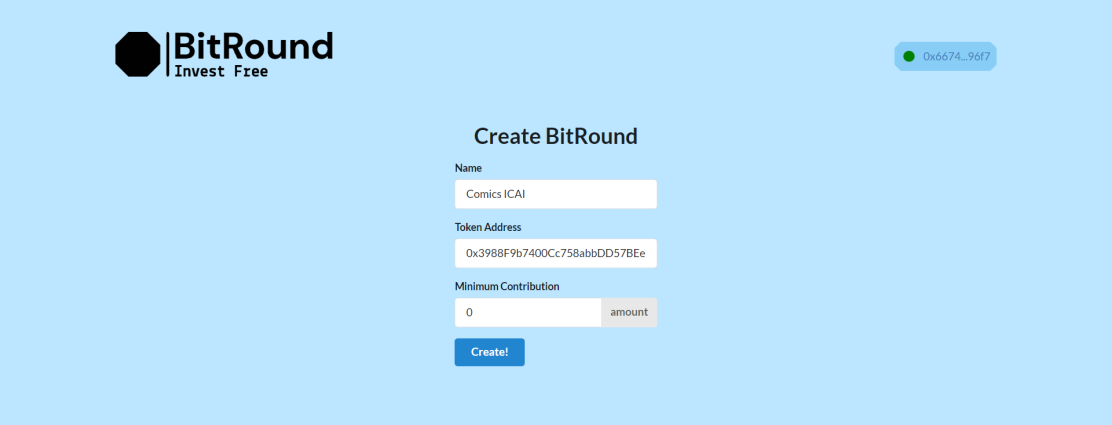

Figure 24: Create BitRound Page

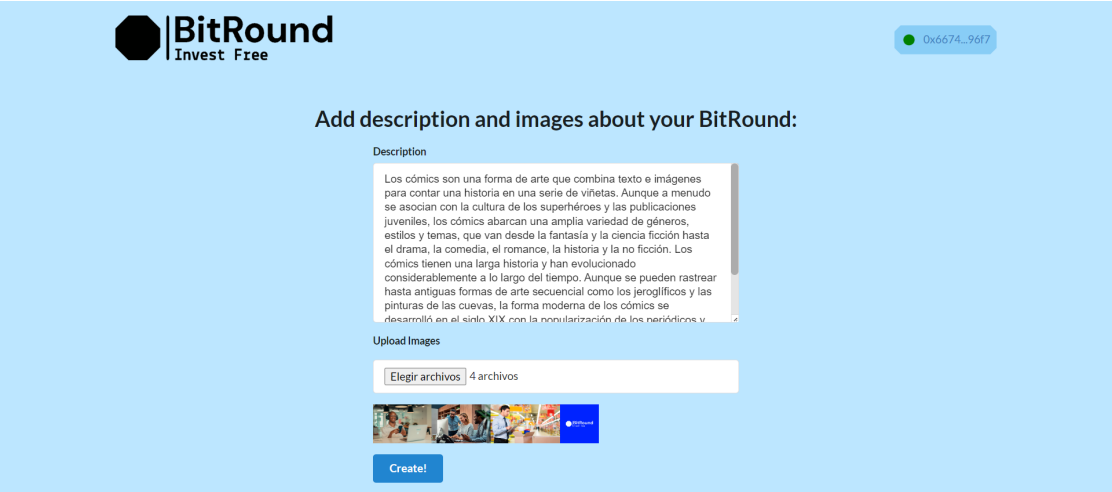

Figure 25: Add Information view

If we click on a BitRound, we will be redirected to the BitRound view. If we click on the login button, we will be asked to connect our wallet with the page. See Figure 22. If the connection succeeds, we will see Figure 23, showing our Ethereum public address. This flow is similar to the login flow in decentralized applications like Uniswap.

If we click on the button '+' of our home page to create a new BitRound, we will be

redirected to the Figure 24 view. Here we fill the form with the company's name, the token's address we want to use for funding, and the minimum contribution in the token's minimum unit of value. In Figure 24, we use a token I created in the testnet named LSTC.

When we complete the form, we are redirected to the second step of BitRound creation, Figure 25. This page has information about BitRound, which we can edit in the future. We can even leave this form empty and complete it after.

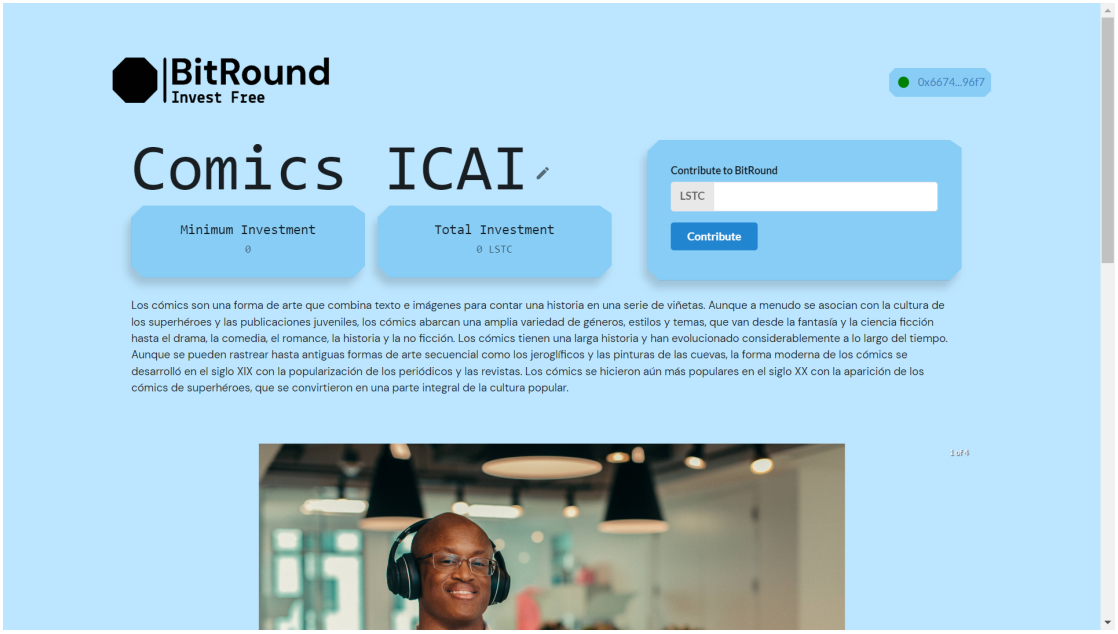

Figure 26: BitRound Page

If we have paid all the gas fees and completed all fields of the first step, we have successfully created a new BitRound. We will be redirected to our BitRound page (Figure 26), where we can find all the information about the BitRound. There is a description of the company, the title of the company, and a carousel of images.

In the top right, we find the contribute card where we invest in the token currency

the amount we desire.

Contribution is the only step that requires two steps in the blockchain, as we have to allow the contract to spend tokens on our behalf. The first step will approve this spending, and the second step is to invest the money in the BitRound. Since we are investing tokens and the contract always executes this transfer of tokens, we have to do this action even if we invest in the BitRound from outside our Dapp.

After we have completed the two steps, we see the success message that confirms our investment in the BitRound; see Figure 27

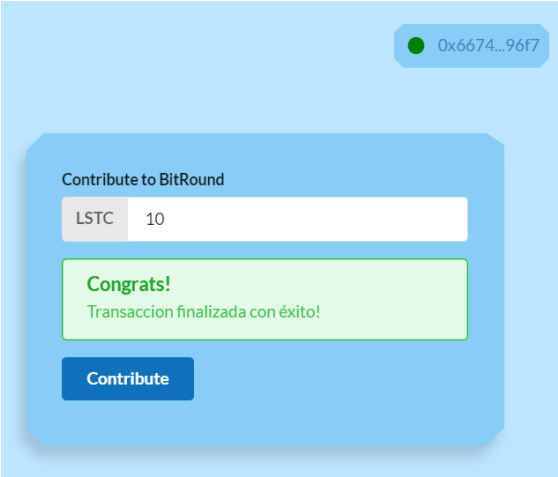

Figure 27: Successful Contribution

With the BitRound created, the entrepreneur can initiate as many rounds as he wants, filling the duration of each round. There is only one rule: the entrepreneur cannot initiate another round if there is one ongoing. Our front end manages this rule appropriately; the contract also checks this restriction. In our front-end application, the new round action (Figure 28) is only shown if the user is logged in with the manager account since it is a function restricted for managers.

Managers also can create new spending requests. We create them below the Bi-

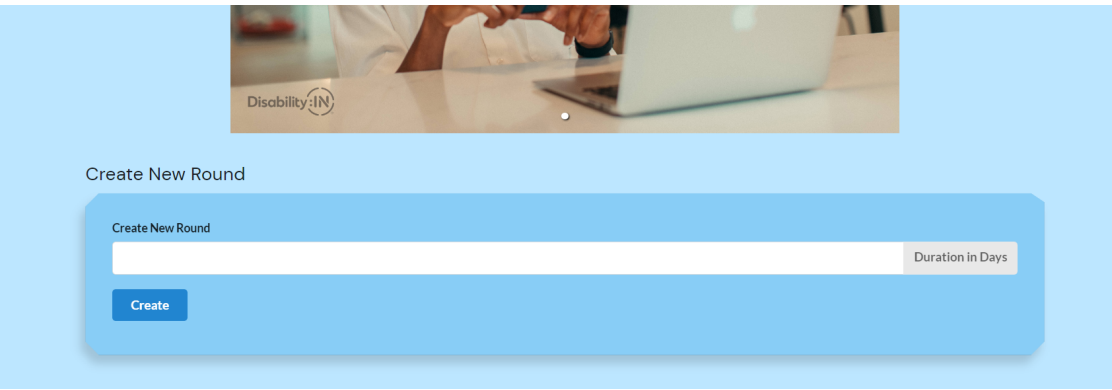

Figure 28: New Round Card

tRound page, where the user can find all the finalized and ongoing requests ordered by creation time.

In each request card, an investor can approve or refuse the request. On the right of the card, we can see the total number of votes and the percentage of approvals and refusals based on the total stake. If the minimum number of approvals is reached, any user can finalize the request, and the transaction will be executed. The request card has the interface shown in Figure 30.

If the request is approved, the card will be green; if the request is not approved within the request time, the card color will be red.

Finally, the portfolio page can only be accessed if logged in. This page has two tabs: invested BitRounds and created BitRounds. This page shows all the related information of our contract with an address—the page query for the user's information to the BitRound subgraph.

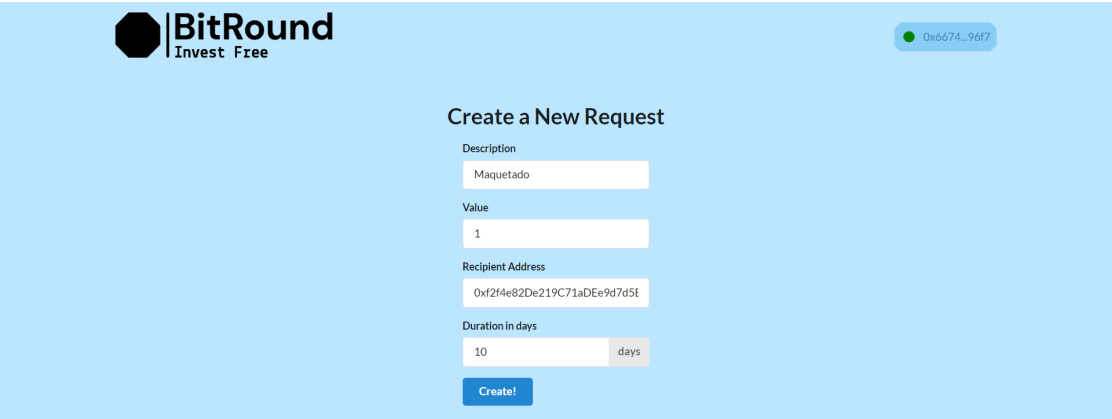

Figure 29: New Request view

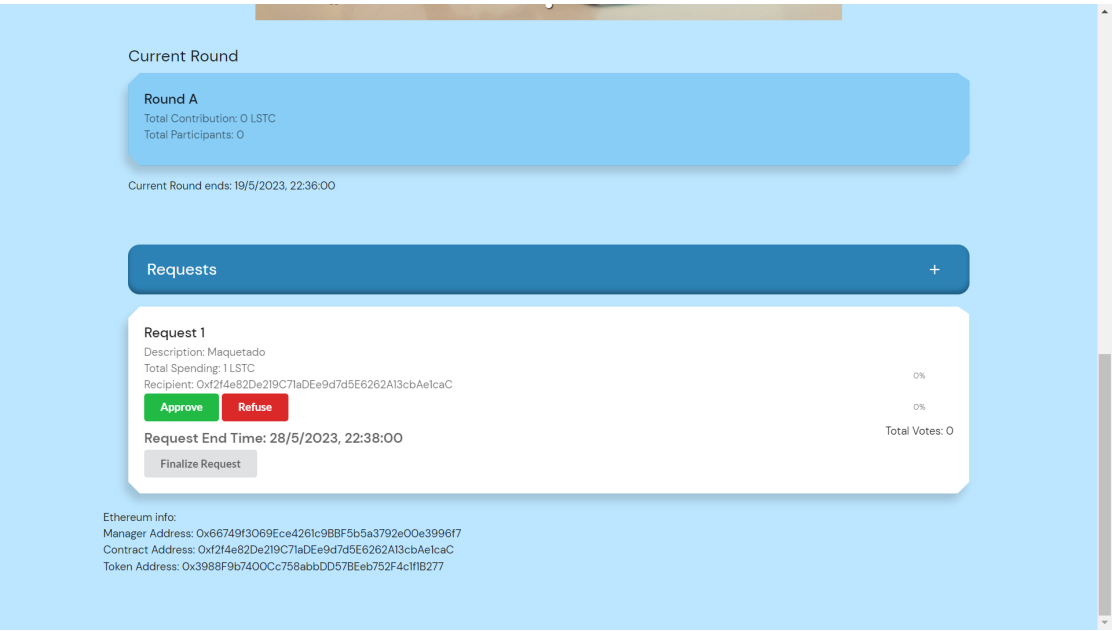

Figure 30: Request Card

D Queries API

Code Block D.1: GraphQL Queries

```
const PORTFOLIO_QUERY = gql`
  query Portfolio($userId: ID!) {
    user(id: $userId) {
      id
      investedCampaigns(first: 10, orderBy: amount, orderDirection: desc) {
        amount
        campaign {
          name
          id
        }
      }
      createdCampaigns(first: 10, orderBy: totalInvestment, orderDirection:
↪  desc) {
        id
        name
        totalInvestment
      }
    }
  }
`;

const TOP_INVESTMENTS_QUERY = gql`
  query TopInvestments {
    campaigns(first: 10, orderBy: totalInvestment, orderDirection: desc) {
        id
        name
        creator {
          id
        }
        totalInvestment
    }
  }
`;
```

References

[1] COSTEERICK & TJONG FREDERIC MEUNIERNADIA NOVIKCYRIANE. *How did starting a business become easier than ever?* 2017. URL: https://blogs.worldbank.org/developmenttalk/how-did-starting-business-become-easier-ever.

[2] *Doing Business Legacy*. https://www.worldbank.org/en/businessready/doing-business-legacy. Accessed: 2023-04-25.

[3] JAKE FRANKENFIELD. *Initial Coin Offering (ICO): Coin Launch Defined, with Examples*. 2022. URL: https://www.investopedia.com/terms/i/initial-coin-offering-ico.aspr (visited on 2022).

[4] Christian Rauch Kimberly Gleason Yezen H. Kannan. "Fraud in startups: what stakeholders need to know". In: (1984).

[5] *Welcome to Ethereum*. https://ethereum.org/en/. Accessed: 2023-04-26.

[6] *Ethereum*. https://es.wikipedia.org/wiki/Ethereum. Accessed: 2023-04-26.

[7] *Ethereum-Logo*. https://artistsatrisk.org/donations/ethereum-logo/?lang=es. Accessed: 2023-04-26.

[8] *Solidity*. https://es.wikipedia.org/wiki/Solidity. Accessed: 2023-04-26.

[9] *Solidity Logo*. https://blog.knoldus.com/structure-of-a-contract-in-solidity/. Accessed: 2023-04-26.

[10] *React*. https://es.wikipedia.org/wiki/React. Accessed: 2023-04-26.

[11] *Web3 JS*. https://web3js.org/. Accessed: 2023-04-27.

[12] *Infura*. https://www.infura.io. Accessed: 2023-04-27.

[13] *Metamask*. https://metamask.io. Accessed: 2023-04-27.

[14] *MetaMask Cryptocurrency Wallet Review.* https://www.investopedia.com/metamask-cryptocurrency-wallet-review-5235562. Accessed: 2023-05-1.

[15] *Remix IDE.* https://remix-project.org. Accessed: 2023-04-27.

[16] *GitHub.* https://es.wikipedia.org/wiki/GitHub. Accessed: 2023-04-27.

[17] *Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System.* https://arxiv.org/abs/1407.3561. Accessed: 2023-05-16.

[18] *About The Graph.* https://thegraph.com/docs/en/about/. Accessed: 2023-05-16.

[19] *Learn Me a Bitcoin.* https://learnmeabitcoin.com/technical/. Accessed: 2023-04-25.

[20] *Public-key cryptography.* https://en.wikipedia.org/wiki/Public-key_cryptography. Accessed: 2023-04-25.

[21] *Diffie–Hellman key exchange.* https://en.wikipedia.org/wiki/DiffieHellman_key_exchange. Accessed: 2023-04-25.

[22] *Elliptic Curve.* en.wikipedia.org/wiki/Elliptic_curve. Accessed: 2023-03-14.

[23] *Integer factorization.* https://en.wikipedia.org/wiki/Integer_factorization. Accessed: 2023-03-14.

[24] *Bitcoin: A Peer-to-Peer Electronic Cash System.* https://bitcoin.org/bitcoin.pdf. Accessed: 2023-04-25.

[25] *Bitcoin.* https://es.wikipedia.org/wiki/Bitcoin. Accessed: 2023-04-25.

[26] *How Does a Blockchain Transaction Work?* https://www.ledger.com/academy/how-does-a-blockchain-transaction-work. Accessed: 2023-04-26.

[27] *Transaction.* https://en.bitcoin.it/wiki/Transaction. Accessed: 2023-04-26.

[28] *What Are Consensus Mechanisms in Blockchain and Cryptocurrency?* https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp. Accessed: 2023-04-26.

[29] *What Is Proof-of-work (PoW)? All You Need to Know.* https://blockworks.co/news/what-is-proof-of-work. Accessed: 2023-05-1.

[30] *Proof Of Stake Vs Delegated Proof Of Stake.* https://101blockchains.com/proof-of-stake-vs-delegated-proof-of-stake. Accessed: 2023-05-1.

[31] *Proof of Capacity (Cryptocurrency) Overview.* https://www.investopedia.com/terms/p/proof-capacity-cryptocurrency.asp. Accessed: 2023-05-1.

[32] *INTRODUCTION TO SMART CONTRACTS.* https://ethereum.org/en/developers/docs/smart-contracts/. Accessed: 2023-05-1.

[33] *ETHEREUM VIRTUAL MACHINE (EVM).* https://ethereum.org/en/developers/docs/evm/. Accessed: 2023-05-1.

[34] *SMART CONTRACT LANGUAGES.* https://ethereum.org/en/developers/docs/smart-contracts/languages/. Accessed: 2023-05-1.

[35] *ERC-20 TOKEN STANDARD.* https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Accessed: 2023-05-1.

[36] *Doing Business Legacy.* https://www.investopedia.com/articles/personal-finance/102015/series-b-c-funding-what-it-all-means-and-how-it-works.asp. Accessed: 2023-04-25.

[37] *Mezzanine Capital.* https://en.wikipedia.org/wiki/Mezzanine_capital. Accessed: 2023-04-26.

[38] *Tim Smith (2022). Crowdfunding: What It Is, How It Works, Popular Websites.* https://www.investopedia.com/terms/c/crowdfunding.asp. Accessed: 2023-05-16.

[39] *2021, EL AÑO DE LAS MEGA RONDAS*. https://lab.expansion.com/record-startup/. Accessed: 2023-04-25.

[40] *Grupo Sego Finance: plataformas y características — Mayo 2023*. https://finanzas.roams.es/entidades-financieras/segofinance/. Accessed: 2023-05-2.

[41] *Guía básica de crowdfunding 15: Los costes del crowdfunding*. https://vanacco.com/articulo/costes-crowdfunding/. Accessed: 2023-05-2.

[42] *Gas fees Calculator*. https://www.cryptoneur.xyz/en/gas-fees-calculator?gas-input=46834&gas-price-option=on. Accessed: 2023-05-24.

[43] *Infura Pricing*. https://www.infura.io/pricing. Accessed: 2023-05-24.

[44] *Indeed Software Developer Salary in Spain*. https://es.indeed.com/career/desarrollador-de-software/salaries. Accessed: 2023-05-24.

[45] *Kickstarter: Estadísticas*. https://www.kickstarter.com/help/stats. Accessed: 2023-05-24.

[46] *6sense: Kickstarter*. https://6sense.com/tech/crowdfunding/kickstarter-market-share. Accessed: 2023-05-24.

[47] *Kickstarter: Comisión para España*. https://www.kickstarter.com/help/fees. Accessed: 2023-05-24.

[48] *Visual Paradigm*. https://online.visual-paradigm.com/diagrams/features/use-case-diagram-software/. Accessed: 2023-05-25.

[49] *Mermaid*. https://mermaid.live. Accessed: 2023-05-25.

[50] *Cloning Solidity smart contracts using the factory pattern*. https://blog.logrocket.com/cloning-solidity-smart-contracts-factory-pattern/. Accessed: 2023-05-16.