UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

A decentralized-based blockchain framework for scientific manuscript peer-review

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Julio Rogers Cajas Guncay

Tutor:

Manuel Eugenio Morocho Cayamcela, Ph.D.

Urcuquí, Octubre de 2023

Autoría

Yo, **Julio Rogers Cajas Guncay**, con cédula de identidad **0931505994**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Octubre 2023.

Julio Rogers Cajas Guncay
CI: 0931505994

Autorización de publicación

Yo, **Julio Rogers Cajas Guncay**, con cédula de identidad **0931505994**, cedo a la Universidad de investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Octubre 2023.

Firmado electrónicamente por:
JULIO ROGERS CAJAS GUNCAY

Julio Rogers Cajas Guncay
CI: 0931505994

Dedication

This work is dedicated to the child I once was, the one who dreamt big and viewed the future with eyes full of hope. To the teenager who, with a mix of nervousness and excitement, first stepped through the gates of Yachay Tech, brimming with dreams and aspirations. And to my future self, for whom I hope to meet the expectations and make all the people I mentioned in the Acknowledgments section proud.

Acknowledgments

I would like to express my gratitude to my father, a sir for whom I hold great admiration. He has continually provided everything a son needs during his university journey: love, support, trust, and opportunities. My mother Rosa also deserves my heartfelt thanks for her unwavering care and love, and for nurturing my academic growth from an early age. I also appreciate my siblings, Shirley, Alex, and Lurdes, for their valuable advice, encouragement, and confidence. To my cherished nephews, Lussiana, Alyssa, and Julian, you are the source of my daily joy, motivation and love. My heartfelt gratitude extends to the close-knit circle of friends I formed during this phase of my life, namely Harvey, Franz, Genesis, Ricardo, Maily, Kevin, Steven, Danna, Anthony, Hector, Ivan, Dayana, and Esther. Your friendship and support have been invaluable. However, there are dozens of other people whose friendships have been vital to my journey. They have enriched my experience with their advice, support, and companionship. I am grateful to my peers from my generation, my housemates, especially those from I-33, my companions from Memes Yachay Tech who brought smiles to my face and occasionally led me into trouble. I also appreciate the people and friends I met through the clubs I joined, especially the "Club de Ciencias Computacionales". My heartfelt appreciation goes to "Los Bandidos de la 27." With them, I found a sense of belonging and deep affection; Mariu, Tati, Salpi, Emilio, Sam, Carlos, Nando, and Maithe - your companionship made our shared moments truly memorable. Nevertheless, several people are missing, people I met through other circumstances, I mean those friends with whom I shared very few moments, but they were significant. On the other hand, I haven't forgotten those I once considered special, with whom I shared memorable moments. Due to various circumstances, they're no longer part of my life. Yet, their influence and contribution to my life remain. I would also like to thank the person who appeared at the end of my university journey, but who has quickly earned a very special place in my life and in these acknowledgments, Diana, thank you for your trust and support. Now, I would like to express my gratitude to those who have been instrumental in my academic development—my teachers. I hold a special appreciation for those who guided me, believed in me, and sparked my interest in the subjects they taught. This includes Israel Pineda, Alexandra Jima, Manuel Morocho, Freddy Cuenca, Juan Mayorga, Simone Belli, Erick Cuenca, Zenaida Castillo, Juan Lobos, and Kamil Makowski. Despite my shortcomings as a student, in one way or another they helped me a lot. I conclude this section by once again acknowledging Manuel Morocho, my thesis advisor, Juan Riofrio, my unofficial co-advisor, and Alexandra Jima, Franz Guzman and Nicolas Serrano. Their guidance, shared knowledge, and insightful recommendations have been pivotal in the completion of my degree.

PS: I also extend my gratitude to my lifelong love, the Barcelona Sporting Club.

Abstract

The peer-review is the traditional process that a scientific journal implements prior to the publication of a research. The purpose of this procedure is for other experts to carry out a rigorous and consistent review of the submitted manuscript, and in this way they verify the veracity of the results. However, this review technique has received various criticisms throughout its existence. Including concerns about the low quality of the reviews, the high cost of publication, the lack of transparency of the review, the non-remuneration of reviewers, among others. Several alternatives proposals have been raised to provide a solution or improvement to these challenges. However, these solutions have not been transcendental. In recent years, proposals using blockchain technology and the benefits that a decentralized system offers have appeared. The problem with these proposals is that they tend to be theoretical, simple, or limited. This graduation project is presented as a better blockchain alternative that encompasses more qualities around the peer-review. This work propose a decentralized system that houses scientific journals that wish to use a peer-review infrastructure with different types of protocols. These protocols take advantage of the benefits of blockchain technology to solve or improve the challenges of the existing peer-review. Through the use of smart contracts, the interoperability of the proposal is evaluated. As well as the execution and transaction time, the computational costs and the real cost in ethers and dollars. These metrics are compared with the ones obtained by other proposals based on blockchain. The results show that the costs of this proposal tend to be higher due to the fact that it incorporates more functionalities, which allows a better operability and quality of the process of each magazine.

Keywords: Peer-review, blockchain, decentralized science, decentralized systems, decentralized journals.

Resumen

La revisión por pares, es el proceso tradicional que una revista científica implementa previo a la publicación de una investigación. Este procedimiento tiene por objetivo que otros expertos realicen una revisión rigurosa y consistente del manuscrito presentado, y de esta manera puedan constatar la veracidad de los resultados. Sin embargo, esta tecnica de revisión ha recibido diversas críticas a lo largo de su existencia entre ellas están la preocupación por la baja calidad de las revisiones, el alto costo de publicación, la falta de transparencia de la revisión, la no remuneración a los revisores entre otros. Debido a esto, se han planteado varias propuestas de alternativas para dar una solución o mejora a estos retos. Sin embargo estas soluciones no han sido trascendentales. En los últimos años han aparecido propuestas usando la tecnología blockchain y los beneficios que un sistema descentralizado ofrece, pero estas propuestas tienden a ser teoricos, simples o limitadas. Es por ello que este proyecto de graduación se presenta como una mejor alternativa blockchain que engloba mas cualidades entorno a la revision por pares. Se propone un sistema descentralizado que aloja revistas científicas que deseen ocupar una infraestructura de revision por pares con diferentes tipos de protocolos que sacan provecho de los beneficios de la tecnología blockchain para mejorar y/o solucionar los retos que tiene la actual revision por pares. Mediante el uso de smart contracts, se evalúa la interoperabilidad de la propuesta, así como el tiempo de ejecución y transacción, los costos computacionales y costo real en ethers y dólares. Estos parámetros son comparados con otras propuestas basadas en blockchain, en donde se obtiene como resultados que los costos de esta propuesta tienden a ser mayores debido a que acopla más funcionalidades lo que permite una mejor operatividad y calidad del proceso de cada revista.

Palabras Clave: Cadena de bloques, revision por pares, ciencia descentralizada, sistemas descentralizados.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

In the academic area, scientific publications or literature are a great contribution for research advances in different topics of science. These contributions are generated either by students, proffesors or researchers which in most of the cases are published in academic books or scientific journals. To publish in this last one, the manuscripts written by researchers must go through a process to be considered a publishable scientific article. This process is a review of the manuscript content, which in most of the cases is a peer-to-peer-review method, which is a review by external fellow researchers who are close to the area covered by the research. They will be responsible for verifying that the manuscript meets certain journal requirements. Likewise, that the arguments presented in the investigation have consistency, reliability, replicability, reproducibly and scientific validity.For example, reviewers should ensure that the manuscript describes sufficient detail. In this manner, an independent researcher can replicate the experiment or calculation performed and thus verify the results obtained in the submitted research. In this process, journals typically do not pay or compensate authors or reviewers.

On the other hand, in the technological and internet-based world, blockchain technology has taken a great leap forward. It has captured the interest and enthusiasm of millions of people [1]. Therefore, a large number of applications have been developed, with those in the financial field having the greatest impact. This is due to the main properties that characterize this technology such as decentralization, immutability, transparency, high-availability, and anonymity.

1.2 Problem Statement

The system or process used by most scientific journals for the review of scientific articles has been used for around 3 centuries [2]. However, this system has received several criticisms. A part of the scientific community that has participated in the peer-review process, either as a reviewer or as an author, has considered that this process has certain weaknesses and problems. Among the problems that exist, and that often occur, is the bias that may

exist at the time of the review by the reviewers, either in favor of or against the author or research topic. An example of this is the bias that can be generated when an author and a reviewer are from the same country [3].

Other problems that is usually related to the current peer-review system, is the slowness of the process to receive a response from the review. Regardless of the journal, it is considered that the time that authors have to wait is extremely high [4]. This problem can generate negative consequences for the author, an example of this is the fact that while they wait for a response from their reviewers, research with similar results or processes may be published first by other researchers [5].

On the other hand, we must also talk about the cost of publishing an article, which has a term known as "article processing charge" (APC). This charge or cost is an amount used to finance the review process and research publication. However, the problem around this is generated due to the high cost that publishing represents in certain magazines, charging values close to 4000 US dollars [6]. In addition to this problem, there is another nearby one. Although journals or publishers charge authors a high fee for publishing their research, in the review process, fellow scientists who serve as reviewers do not receive any reward for their work. This can be indirectly associated with the fact that on several occasions the reviewers perform low quality reviews. Due to this mechanism that magazines use, it is estimated that the free work done by the reviewers is equivalent to a donation of one billion dollars [7]. In 2020, the time employed in this labor were around 100 Million hours. The monetary value of time spent reviewing manuscripts by US reviewers is estimated to be over $1.5 billion, Chinese reviewers are estimated to be over $600 million, and UK reviewers are estimated to be nearly $400 million [8].

1.3 Objectives

The objectives that are expected to be obtained from the completion of the project are presented below:

1.3.1 General Objective

The main objective of this thesis is to offer the scientific world a proposal that could improve the characteristics of the review processes of scientific articles. In this way, it is contemplated to provide a blockchain-based system which can be used as a basis to solve certain problems that currently occur in the peer-review process. In this way, it is sought that either current journals can implement this proposed base in their processes, or a totally decentralized system managed by the scientific community that uses this base can be created.

1.3.2 Specific Objectives

- Investigate and analyze the different proposals that have been created as an alternative solution for the current challenges of the traditional peer-review process.

- Structure the characteristics and main components of the proposed system, such as consensus protocols, system tokenization and connections between smart contracts.
- Present an infrastructure for the peer-review process that allows journals to customize that process according to the variety of protocols to choose from.
- Build and demonstrate the interoperability of the proposed system using smart contracts.
- Analyze the execution and transaction time, the computational and transactional cost in terms of gas, ethers and dollars.
- Perform a comparative analysis of this proposal with other similar proposals.
- Present proposals for implementations that could be added to this base proposal for future work.

Chapter 2

Theoretical Framework

In this section, we focus on two main areas, which are necessary knowledge to understand this project. The first is the environment that encompasses the traditional publication review system. In which the publishing companies are the regulatory entity. They are in charge of managing the process of reviewing and publishing scientific articles. Regarding this, we will analyze the process, as well as the key and most essential concepts for a good understanding of this system. While the second area is the environment of decentralized systems, focusing directly on blockchain technology and related. Blockchain has been one of the most outstanding technologies in the last decade and promises to be a great revolution for the Internet. Various projects and companies from areas such as financial services, games, supply chains, and others have implemented this technology in their development. Being the "Bitcoin" project in the financial services field, the most outstanding and the one that started the heyday of the Blockchain. This project deals with a decentralized and purely electronic cash system that allows making payments online without needing a financial institution [9]. Regarding this second field, we will exhaustively analyze the most relevant and necessary concepts for developing applications and projects using this technology.

2.1 Academic and Scientific Environment

The scientific environment is a vast network that involves researchers, scholars, professionals, and institutions. They work together to develop the advancement of knowledge. In addition, they understand and explain various factors regarding the different disciplines that science encompasses. The main elements of the scientific community may vary depending on the area or science subject field. However, several critical components are essential to the functioning of this community. These include:

- **Researchers and Scholars:** They are the most important component of this ecosystem. They conduct scientific research, develop new theories, and publish their results in academic journals. These individuals are commonly highly educated and specialize in a particular study field.

- **Academic Institutions:** This component includes universities and research institutions. They are essential in this community because they provide resources and support for researchers, as well as opportunities for collaboration and networking.
- **Peer-Review:** An essential component on which this thesis focuses. Peer-Review focuses on quality control, legitimization, and self-regulation of scientific research. Peer-Review involves the work evaluation by experts peers in the same field [10]. They review and provide feedback on the work prior to publication.
- **Scientific Journals:** They are a vital component of the scientific community. They are a means for researchers to publish and share their work with the world. This component also provides a platform for researchers to receive feedback and criticism of their work from their peers.
- **Scientific Conferences and Meetings:** These components provide researchers and scholars with a space to present their work, learn about the latest developments in their field, and network with other professionals. These events are essential for promoting collaboration and advancing knowledge within the community.
- **Funding Agencies:** This component includes government institutions and private foundations. They play an essential role in supporting scientific research and innovation. These agencies provide financial resources and support for research development.
- **Ethics and Integrity:** Crucial components are Ethics and integrity. In this ecosystem, it is necessary to conduct research responsibly and ethically. The community must follow ethical standards in all aspects that involve the development of research, as well as in the interaction and collaboration with their colleagues and institutions.

2.1.1 Scientific Article

A scientific article is a document that clearly and precisely presents new research results on a specific topic or field. A scientific article intends to contribute to the advancement of knowledge and science. Researchers, scientists, and scholars write these articles to present their research results, theories, or proposals to their peers and the larger scientific community. This manuscript must offer all the necessary information so readers and reviewers can understand and replicate the research and its results. Also, it mentions the contribution it represents, the improvement and comparison with other articles, among other qualities [11]. After creating and writing an article, the author(s) sends it to an academic journal, congress, conference, or another specialized publication media. These articles go through a rigorous peer-review process. In this way, the authors accomplish their goal of contributing to the advancement of science. Scientific articles generally follow a specific format, which typically includes the following sections:

1. **Abstract:** This is a summary of the article, usually 250 words or less. This provides an overview of the motivation, problem statement, approach, methodology, results, and discussions [12].

2. **Introduction:** This section provides the background and context of the problem and the question that the research answers [13]. In addition, we can find here the research objectives and hypotheses.

3. **Methodology:** This section describes the methods and procedures used to develop the research. It must include all rigorous details in order to allow peers to reproduce the results [14].

4. **Results:** This section presents the research findings. The results must be quickly, clearly visible, and understandable [15]. Therefore, they are typically presented through tables, graphs, or other visual aids.

5. **Discussion:** This section analyzes and interprets the results. It also discusses any limitations or implications of the research. In addition, it suggests areas for future research [16].

6. **Conclusion:** This section provides a summary of the essential findings and their implications, and emphasizes the significance of the research.

2.1.2 Scientific Journals, Congresses, Conferences.

Journals, congresses, and conferences are all critical components in the academic and scientific environment since they are how articles are shared with the community. Each serves a different purpose in advancing knowledge and promoting collaboration among scholars and researchers. Since centuries ago, they have been in charge of the management and organization of the process that involves the publication of scientific articles. They differ in the demand standards and processes each of them has. Most of these entities charge the research authors a certain amount of money to publish their articles. In the same way, they charge the readers of said articles. While most of the scientists who carry out the review are not paid any economic value, their work is altruistic.

- **Academic Journals:** Academic journals are publication means that focus on specific subjects or scientific areas. Their goal is to disseminate the latest research findings and insights in their focus fields. These journals often require rigorous peer-review of submitted manuscripts to verify the quality of the article to be published. Examples of well-known academic journals include Nature, Science, and The Lancet.

- **Congresses:** A scientific congress gathers scholars, researchers, and professionals in a particular academic field. These events are usually held once or twice a year. These provide a space for their participants to present their latest research findings. In addition, we can find discussion panels, presentations of academic works, and networking opportunities at congresses.

- **Conferences:** Conferences are similar to congresses but are often smaller in scale and more focused on specific topics. Universities, professional associations, or private organizations may hold them. Like congresses, conferences provide an opportunity for the scientific community to share their work, learn about the latest developments in their field of interest and collaborate with peers.

In summary, academic journals, congresses, and conferences play an essential role in advancing knowledge and promoting collaboration among individuals in the scientific community. Each has a different purpose and offers unique benefits to the academic community.

2.1.3 Academic Publishing Process

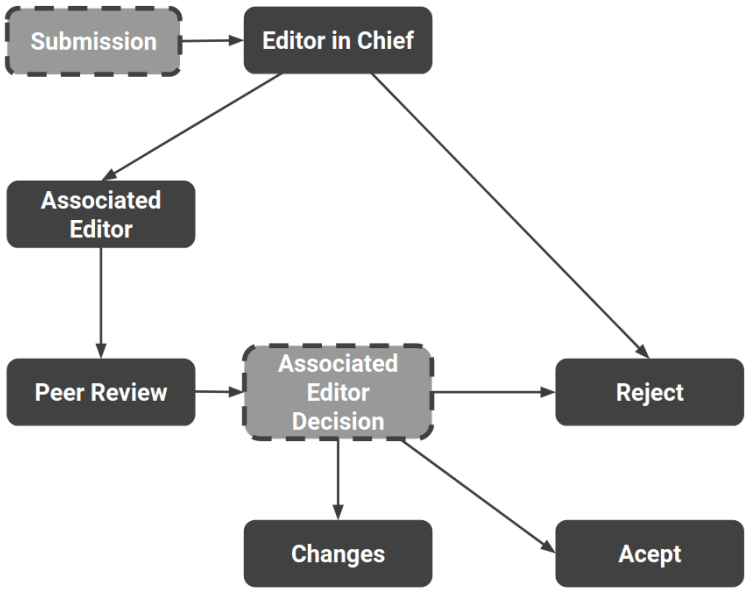

Figure 2.1: Traditional academic publishing process.

The traditional process of publishing an article is a strict system that guarantees that research results are analyzed and evaluated by experts in the research field. The process consists of several stages: manuscript preparation, submission, peer-review, revisions, and publication. It tends to have a structure like the one seen in Figure 2.1.

1. **Manuscript preparation:** The first step in the academic publishing process is the preparation of a manuscript. Researchers usually write everything that pertains to their research in a formal paper. They should follow the structure described in subsection 2.1.1. Likewise, they must comply with the guidelines set by the specific journal.

2. **Submission:** Once the manuscript is complete, it is submitted to a considered academic publisher. The author must follow the submission guidelines, including formatting requirements, word limits, and instructions for submitting supplementary material [17]. After this step, The Editor-in-Chief may consider manuscript rejection in case of detection of a significant or large number of errors in a prior verification stage. If the above does not happen, the editor-in-chief proceeds to send the paper to an associated editor with the required knowledge in the manuscript area. The associated editor is in charge of finding a team of experts who will perform the role of peer-reviewers.

3. **Peer-Review:** This is the step where the peer-reviewers evaluate the manuscript based on the quality, clarity, originality, and rigor of the research [10]. The reviewers provide feedback and criticism about the manuscript to the author, who then has the opportunity to revise and improve the manuscript. Finally, they send the results of their review to the associate editor. There are four well-known peer-review types: a single-blind, double-blind, open, and post-publication review. [18].

4. **Associated Editor Decision:** After receiving the results from the peer-reviewers, the associated editor decides to reject the manuscript, approve it, or reject it until the next submission, where the author considers and makes the changes that the reviewers suggested.

    - **Feedback Revision** If the manuscript was not accepted but received feedback, the author should revise the manuscript accordingly. This may involve changing the writing style, research design, or data analysis. Once the revision and the improvements are complete, the author can resubmit the manuscript to the journal or publisher for further review.
    - **Acceptance:** If the manuscript meets the quality parameters and guidelines requested by the journal, The publisher will accept the research for publication. The editor will work with the author to finalize the manuscript, including formatting, editing, and proofreading.

5. **Publication:** Once all corrections, enhancements, and additions of details are finished, the academic journal will publish the manuscript. The publication may be available in print or online and may be subject to copyright restrictions.

Ethics is a fundamental characteristic of this process. Authors, reviewers, and editors must adhere to ethical standards at every process step. Researchers must ensure that their work is original, authentic, accurate, and well-cited.

2.2 The Decentralized System

A decentralized system does not depend on a central unit as the control and management authority of the network, as opposed to centralized networks. A decentralized network replaces the central unit with a set of units that exercise the coordination and control of the network. However, there is one more type of network, the distributed system. It is different from the conventional (centralized) system. Because it is characterized by not having control units for the network, but rather all the computers or nodes are interconnected [19]. Since control does not depend on any node in particular, it eliminates the existing problems in the other two types of networks where if the control node fails, its dependents also fail. On the contrary, in a distributed system, if a node fails, it does not affect the rest of the network. Figure 2.2 shows the differences in the structures of the recently mentioned models.

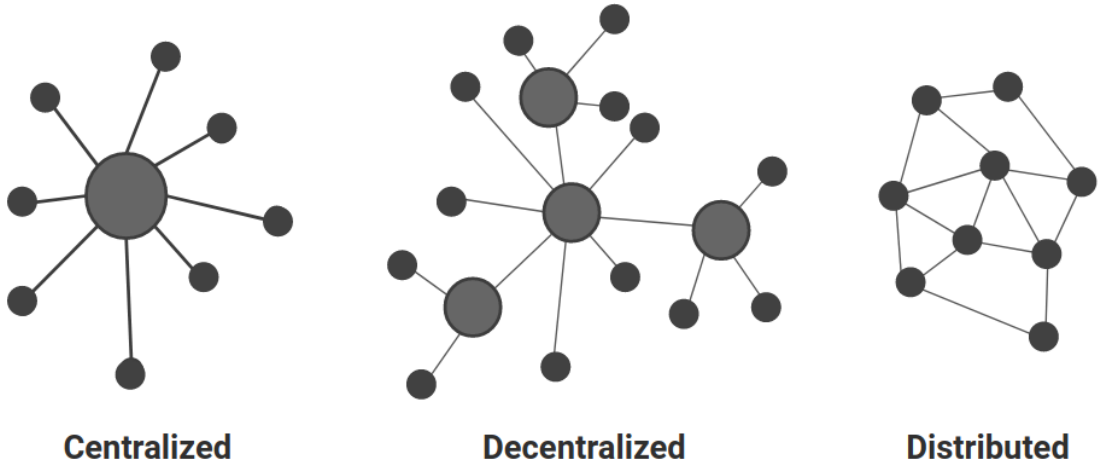

Figure 2.2: Comparison of the structure of centralized, decentralized and distributed systems

2.2.1 Distributed Ledger Technology

A distributed ledger technology (DLT) is a system based on a distributed system used for storing, exchanging, registering, and distributing data between the nodes that belong to the network. This enables us to have a complete history of all the changes in the data.

2.2.2 The Blockchain System

The blockchain system could be considered a type of DLT since it is an extensive list of information blocks connected through cryptographic hashes. Blocks are a set of data about transactions within the blockchain. Each block contains information about the cryptographic hash of the previous block; it also contains a timestamp and the stored data. The number of transactions stored in a block is usually based on a set time. In this way, a connected chain of blocks is formed whose information cannot be modified. If there is an alteration in the block information, its cryptographic hash will be altered, and therefore it will not match with the one registered in the successor block. This feature makes the system cryptographically secure and immutable, allowing participants to trust the stored information. On the other hand, this system has consensus rules to maintain consistency in the information in each network node. From another point of view, the blockchain system is also considered a linked list that uses hash pointers that refer to the previous block [20].

All these registers or blocks are stored and distributed to the network nodes. Therefore, each participant with a node has a copy of the complete ledger.

These data will be updated or changed through transactions, representing any movement, operation, or change of state of data in the network. Each node can carry out a transaction. However, the rest of the nodes must validate and accept this transaction. Blockchain uses a consensus mechanism that allows credibility and consistency in the information stored in the network. Consensus mechanisms allow the system to ensure that

the new blocks added are legitimate and do not contain falsified information. Some of these mechanisms are proof-of-work (PoW), proof-of-stake (PoS), and delegated proof-of-stake (DPOS), among others. The work of the nodes will be to generate a cryptographic hash for the legitimate block. Consensus protocols focus on the steps before hash generation. PoW is essentially about reaching certain computational effort requirements to generate a cryptographic hash [21]. Consequently, once a node generates a hash, the rest of the nodes check if the generated hash is correct and approve it. Since all the nodes compete simultaneously to generate the hash, the system is not energy-efficient.

Due to the inefficient use of energy that characterizes PoW, other alternatives were created, such as Proof of Stake (PoS). Which requires less energy since there are no limitations that hinder the creation of the hash; therefore, it is easier to generate one [22]. The node selected to generate the hash is randomly chosen from the nodes with the highest number of coins. This is because if the system suffers an attack, this will cause a devaluation of the currency. Consequently, wealthy stakeholders do not want any attack or vulnerability of the network. On the other hand, the DPOS is similar to the previous proof; the difference is that the stakeholders with more currencies delegate the mining work to another chosen node [22].

These consensus algorithms offer cryptocurrency as a reward to the nodes that have validated the blocks. Besides, charge a fee to the nodes that have sent a transaction to be validated.

Types of Blockchain

Four main types of blockchain are known: public, private, consortium and hybrid.

1. Public
   A public blockchain is a decentralized network open to anyone who wants to participate. This means no access restrictions exist; anyone can have a node and mine. This Permissionless blockchain usually offers economic incentives for those who work on its development and security. Public blockchains are highly transparent, and all transactions are recorded on a public ledger that anyone can access. In addition, public blockchains rely on a distributed network of nodes to validate transactions rather than a central authority [23]. Furthermore, public blockchains are highly secure, but it also makes them slower and more resource-intensive than other types of blockchains. The best-known and most outstanding blockchain projects are of this type, such as Bitcoin [9] or Ethereum [24].

2. Private
   A private blockchain is a closed system with access restrictions to join or interact with the network. A single entity or a group of entities controls the system. Due to these features, the system becomes more secure than public blockchains but less transparent [25]. Private blockchains can be faster and more efficient than public blockchains as they do not require as much computational power to validate transactions. Generally, users need to receive an invitation from the network administrators to participate.

3. Consortium
   A consortium blockchain is also called a federated blockchain. It is a kind of private

blockchain where multiple individuals work together to control the network [23]. Depending on the needs of the participants, consortium blockchain can be permissioned or unpermissioned. They offer a balance between the security of private blockchains and the transparency of public blockchains.

4. Hybrid
   A hybrid blockchain is a system that combines the features of public and private networks [25]. The blockchain features depend on the authorities or participants. Therefore, this mixed blockchain is customized depending on the specific needs and the purpose or reason why the blockchain is created.

2.2.3 Ethereum

Ethereum is a decentralized platform based on blockchain technology created by Vitalik Buterin [26]. Unlike Bitcoin, Ethereum was born as a disruptive perspective to create an instrument for the generation of decentralized applications that are managed through smart contracts.

- **Ethereum Virtual Machine**
  An essential piece in the Ethereum blockchain is its virtual machine, 'Ethereum Virtual Machine' (EVM), which stores all Ethereum accounts and transactions [24]. It also executes smart contracts to improve or incorporate functionalities or projects into the platform. Each participating node in the ethereum network has a copy of the state of this machine. Participants can send a request for the use of this machine to perform certain computations according to their interests. Given this, the rest of the community of participants will verify, validate and execute the request. When the computation is performed, a change in the state of the EVM is generated, which is updated throughout the network. These computational requests are known as transaction requests; they are stored on the blockcha in along with the state of the machine [27]. The EVM uses its own programming language called Solidity.

- **Ether** Ether (ETH) is the native cryptocurrency of Ethereum. Ether is used to pay transaction fees on the Ethereum network, as well as for the execution of smart contracts and gas fees. Any participant who advertises a group action request must supply a certain amount of ETH to the network as a bounty. The network can offer this bounty to anyone who eventually performs the task of substantiating the group action request, imposing corporal punishment, committing it to the blockchain, and transmitting it to the network [28]. The amount of ETH paid corresponds to the resources needed to try and do the computation. These bounties additionally forestall malicious participants from deliberately clogging the network by requesting the execution of infinite computation or different resource-intensive scripts, as these participants should obtain computation resources. Ether is similar to Bitcoin in that it is a digital currency that is stored and transferred electronically. Therefore, another use of Ether is as a medium of exchange or investment. Like other cryptocurrencies, the price of Ether is determined by supply and demand on cryptocurrency exchanges. Investors can buy and hold Ether in the hopes that its value will increase over time,

or they can trade Ether for other cryptocurrencies or fiat currencies. Its versatility and potential for use in a wide range of applications have made it a popular cryptocurrency among developers and investors.

- **Gas**
  Gas is an essential component in the ethereum network. It is considered the fuel needed for the operability of the network. Gas becomes a unit of measurement that helps us to count the amount of computational effort that the Ethereum network requires to execute a transaction or a smart contract [29]. Gas is used as a fee to pay for three things on the Ethereum network:

  – Smart contract execution: When the EVM executes a smart contract, the contract developer has to pay a gas fee to compensate the miners who execute the transaction. The cost of gas depends on the complexity of the smart contract and the amount of computational effort required to execute it.

  – Transaction processing: When a user sends a transaction on the network, he or she must also pay a gas fee to the miners who validate and process the transaction. Like the execution of a contract, the complexity of the transaction and the amount of computational effort required determines the gas fee.

  – Storage: In addition to smart contract execution and transaction processing, storage on the Ethereum network requires a gas fee. Each piece of data (bytecode) stored on the EVM requires a payment for the computational space and effort.

2.2.4 Smart Contracts

A smart Contract is a self-executing programming code containing the program logic and state data. It is executed as part of the transaction, assigned an address, and displayed on the blockchain [30]. Essentially, it is a computer program that automatically enforces the terms of a contract, eliminating the need for intermediaries [31]. Smart contracts are an integral part of the blockchain ecosystem, as they enable the creation of decentralized applications (dApps); this term will be explained in detail later. Smart contracts are typically written in a programming language designed explicitly for creating them. The best known is Solidity for the Ethereum network.

Smart contracts are automatically executed when certain conditions are met. For example, if a seller and buyer enter into a smart contract that stipulates that the sale of an artifact will be released once certain conditions for sale are met. Therefore, the payment and delivery of the device will be released automatically once compliance with the conditions is verified [32]. This eliminates the need for a third-party intermediary to oversee the transaction.

Smart contracts, like the information stored on the network, are immutable. Therefore, no one can perform corrections or improvements to the code. For this reason, before deploying a smart contract on the ethereum network, it is first tested on the test net, which are test networks that simulate the execution of the network for free.

Besides, participants do not write new code whenever they need to request a computation process on the EVM. Instead, application developers transfer programs (reusable

snippets of code) into the EVM state, and users build requests to execute these code snippets with varied parameters.

Writing codes using smart contracts allows these programs to have the features that a smart contract represents [32], such as the following:

- **Decentralization:** Since a decentralized network stores smart contracts, their features can be inherited towards smart contracts. Among these characteristics is decentralization, where a central entity does not execute smart contracts but instead by blockchain.
- **Transparency:** Smart contracts are visible to all participants on the blockchain network, meaning they are fully transparent. This feature makes it easy to track the progress of a contract.
- **Immutability:** Once a blockchain executes or stores a smart contract, it cannot be changed. This feature ensures that the contract terms are followed as written and eliminates the risk of fraud or manipulation.
- **Self-executing:** Smart contracts are programmed to execute automatically when the conditions are fulfilled, eliminating the need for intermediaries to supervise the fulfillment of the conditions.
- **Programmability:** Smart contracts are codes made by using a specific programming language. Depending on the project, smart contracts can be adaptable for different purposes. This feature makes them very versatile in various real-life situations.
- **Trustlessness:** Smart contracts are designed to offer the correct fulfillment of conditions without the need for trust between the participants. This feature means that they can be used to carry out transactions between two or more parties that do not know or trust each other.
- **Security:** Smart contracts are very secure as they have protection by advanced cryptography and decentralized blockchain network features. Consequently, this allows them to be highly resistant to hacking and cyberattack threats. In addition, as mentioned above, they are immutable. However, security also depends on the correct writing of the smart contract code.

2.2.5 DApp

A decentralized application (Dapp) is a type of software application established in a decentralized network such as Ethereum [33]. In other words, it is an application that does not depend on central servers like conventional applications. Its users interact with each other directly without the need for intermediaries such as servers or middlemen. On the front-end, Dapps use the same technologies as traditional Apps. However, on the back-end, they differ by using smart contracts.

Since decentralized apps are developed through smart contracts, dapps inherit its characteristics, such as decentralization, open-source, trustless, and transparent.

- **Decentralized:** There is no central authority that controls the application. Instead, the network is maintained and validated by a distributed network of nodes. This feature ensures that the app is highly resistant to censorship and hacking.
- **Open-source:** dApps are usually open source, which means their source code is publicly available for anyone to inspect, modify, or contribute. This feature generates a collaborative and transparent development community where anyone can participate in the development, contribute an improvement to the application, or solve challenges that the app presents [34].
- **Trustless:** dApps are designed to be trustless, meaning that users can interact with the app and each other directly without the need for trusted third parties. This is possible thanks to the self-executability of smart contracts that automate the execution of transactions on the blockchain.
- **Transparent:** Transparency is due to the fact that all transactions and data stored in the blockchain are publicly visible and auditable. This fact generates a high level of responsibility and trust among users since all actions are recorded on the blockchain and cannot be altered.

Some examples of dApps include:

- **Decentralized Finance (DeFi) dApps:** These are applications that provide financial services, such as lending and trading, on a decentralized blockchain network. Examples of DeFi dApps include Aave, Oasis, PWN, Uniswap, and Compound.
- **Gaming dApps:** These are applications that enable decentralized gaming experiences. They use blockchain technology to ensure fairness, transparency, and in-game asset ownership. Examples of gaming dApps include decentraland, cryptovoxels, dark forest, and Axie Infinity.
- **Supply Chain Management dApps:** These applications provide end-to-end visibility and transparency for supply chain processes. They use blockchain technology to track the journey of goods from production to distribution, ensuring authenticity and preventing possible information loss. Examples of supply chain management dApps include Provenance, VeChain, and OriginTrail.
- **Social Media dApps:** This kind of apps give users more control over their data and content. These dApps aim to remove the centralized control of traditional social media platforms and provide greater privacy and security to their users. Examples of social media dApps include Steemit, Minds, and Memo.
- **Identity Management dApps:** These are applications that aim to provide secure, decentralized identity verification solutions. These dApps use blockchain to create immutable identity records, ensuring greater privacy and security for users. Examples of identity management dApps include uPort, Civic, and SelfKey.

These are just a few examples of the most popular types of dApps out there. However, blockchain technology continues to evolve, so we can expect many dApps to emerge across various industries and use cases.

2.2.6 Tokens

In the blockchain ecosystem, a token is a widely used and very confusing term. Since it is commonly confused with the term cryptocurrency, we can start by simplifying the term token as a representation of an exchangeable digital asset. This asset can represent a particular value or utility in a decentralized network. Tokens are created and managed within a blockchain, allowing for secure and transparent tracking of ownership and transactions. The use given to this asset may vary depending on the purpose of each project.

Several different types of tokens can exist on a blockchain network, including:

- **Currency tokens:** These are tokens that serve as a digital currency. They allow users to transact and store value on a blockchain network. Examples of these tokens include Bitcoin, Ethereum, and Litecoin.
- **Security tokens:** These are tokens representing ownership or investment in an underlying asset, such as a company or real estate. Securities laws regulate security tokens and are subject to compliance with securities regulations.
- **Utility tokens:** These are tokens that provide access to a particular product or service on a blockchain network. Utility tokens are not designed to be used as currency but to provide a specific function or utility within the network.
- **Non-fungible tokens (NFTs):** These are tokens that represent a unique, one-of-a-kind asset, such as artwork, collectibles, or virtual real estate. NFTs are often used in gaming and digital art applications [35].

These are just a few definitions of known token types; however, many others exist, either derivatives or similar to those above or entirely different. Tokenization is the process of token creation, which is about creating a digital asset backed by a particular security or utility. Tokens can be transferred between users on a decentralized network through transactions; they are validated and recorded on the blockchain. This fact ensures that the ownership and transaction history of the tokens are secure and transparent. In recent years, tokens have become increasingly popular due to their ability to enable new forms of value exchange and economic gain. In the coming years, more tokens belonging to different types of decentralized projects may emerge.

Chapter 3

State of the Art

The first academic journal, called Journal des Scavans and Philosophical Transactions, appeared in January 1665 [36]. Since then, the emergence of means for the publication of scientific research began. In 1733, Medical Essays and Observations belonging to the Medical Society of Edinburgh appeared, which became the first journal to be peer-reviewed [2]. Over time, many other journals have emerged with different standards and focused on various areas, some of which had a significant impact, such as Nature or Science. At the beginning of the 90s, electronic journals emerged, like the Adonis journal that appeared in 1991 [37]. In those times, the movement of open-access journals also began, allowing the public to read scientific articles without paying for them. Therefore, since its inception, the environment of scientific publications has gone through different essential moments, such as the use of peer-review, digitization, and open access. However, despite hundreds of years, the system still in use is basically the same as the one implemented hundreds of years ago. Due to its lack of updating, several problems characterize it, including slowness, costs, inconsistencies, and biases, among others [38]. Faced with this scenario, various proposals for improvement have emerged over the years; Next, we will present and analyze a few recent and interesting proposals.

In 2020 a work was presented that focuses on combating three challenges. Among them is that reviewers seek to be assigned to particular articles to provide positive or negative reviews. For this, they have created a (randomized) algorithm for the assignment of reviewers, which optimally solves the problem of assignment of reviewers, taking into account the probability of assignment for any pair of reviewer-paper [39]. Secondly, in 2021, Researchers from Carnegie Mellon University proposed the Peerreview4all system, which they describe as an assignment algorithm based on an incremental max-ow procedure. This project focuses on retrieving manuscripts that should have been accepted [40]. In the same year, another proposal seeks to improve the performance and effectiveness of the process using a parallel model built via the Monte Carlo method in a distributed manner [41].

Distancing ourselves a bit from this recent discussion in the previous paragraph. In the last decade, several proposals, companies, and systems have emerged that use new technology to solve particular problems in their environments; this technology is the blockchain. The field that has used this technology the most is the financial field, with bitcoin [9] as an outstanding project. The scientific ecosystem is no exception because, in recent years, several projects have emerged focused on solving challenges or proposing improvements in

the scientific area through blockchain technology. Including the environment for scientific article reviews.

In 2019, Leible et al. [42] researched open-science projects that use blockchain technology. This paper contains a specific section around projects focused on the publication process. In this review, we found that the first projects of this kind have been carried out since 2017, with proposals such as CryptSubmit, among other proposals such as those presented by Spearpoint and Avital.

CryptSubmit is a system that addresses problems occurring during article reviews, such as plagiarism and dissemination, due to technical problems. Their solution is a decentralized system that automatically creates verifiable timestamps around the submitted manuscript [43]. To accomplish this, they use the Bitcoin blockchain and tools such as OriginStamp. Gipp et al. tell us that their project does not prevent academic plagiarism but that their system may deter some plagiarists. In their writing, they do not provide us with a results section. Likewise, they mention that they integrated their conceptual proposal into an open-source manuscript submission system. However, they do not provide further information about the code, pseudocode, or repository link.

The proposal by Spearpoint focuses directly on an economic system that charges a price to submit an article and review it[44]. This paper only has a theoretical proposal since it does not present a prototype. However, it presents exciting ideas, such as using ORCID for the registration of authors and reviewers. Spearpoint also mentions other ideas, like a payment for the reviewer's work. The payment amount depends on the review's delivery time—likewise, he proposes a payment to the authors according to the citations of their articles. In a certain way, this article focuses on the financial aspect of the system and the qualities that a system should have around the use of the token it has.

Avital's proposal is similar to the previous one because it uses tokens to develop a market-based peer-review payment system [45]. This paper considers that this proposal can solve the congestion problem in the article review pipeline. In addition, the author considers that the market regulation mechanisms allow controlling the quality and fairness of the reviews. This system contemplates that the author pays with the system token, which would be redistributed to the reviewers and to the system, where the reviewers receive the highest amount for the work done.

As of 2019, a more significant number of proposals appear. They contain the same pattern, but they differ in details. Among the proposals, we have PubChain [46], which deals with a decentralized open-access platform for scientific publications which uses Blockchain and IPFS technologies to share files between peers. In this way, they seek to solve a severe problem, such as paid access to scientific research. They also believed peer-review tends to be ineffective because reviewers do not receive a considerably adequate incentive for high-quality reviews. This project has among its features to encourage authors, readers, and reviewers through credits and rewards. This proposal performs a simulation with which they evaluate the soundness of the incentive structure. In addition, they implemented a prototype to demonstrate the critical characteristics proposed.

There is also a proposal [47], which focuses on the problems considered to have of significant impact on the peer-review system. One of these problems is the slow and expensive system, in addition to the fact that there is an unresolved bias. To solve this problem, the proposal deals with a double-blind review system to preserve the anonymity of authors and reviewers. It also addresses issues such as paying reviewers and inconsistent

and biased review metrics. That is why this project proposes a system that uses the Hiperledger Fabric Blockchain and the interplanetary file system(IPFS). This proposal presents a cryptocurrency called AcadCoin, which is used for financial transactions between editors and reviewers. At the same time, access control serves to keep the reviews double-blind. This system architecture distributes nodes between the blockchain and the IPFS. This project used the Hyperledger Composer Playground environment to implement the system.

Another exciting proposal [48] criticizes the current academic publication procedure due to a lack of transparency in the review process—and no reviewers incentives. There is also criticism that the scientific community has to pay to access scientific articles while the same community performs the reviewing of the manuscripts for free. That is why they propose Open-Pub, a decentralized academic publication scheme that is transparent but at the same time preserves privacy using Blockchain technology. This proposal uses a double-blind review to protect the identity of authors and reviewers. This project implements the system in the Ethereum blockchain, with which the authors carried out performance and efficiency tests.

In the same way as some previous proposals, we have another article [49] that also presents an idea about using the token as an incentive for reviewers. They present their proposal as an autonomous incentive system through the use of tokens, adding economic and cryptographic mechanisms. They use a concept called TCR (Token-curated registry), which is based on the idea of the free market and the material interest of individuals. With this mechanism, they encourage reliable work and discourage unreliable work of reviewers. The proposed system comprises 3 participants: the author, the reviewers, and the readers. Where there is an incentivization for reviewers by being paid with tokens. The value of these tokens depends directly on the journal's credibility, which is why they stipulate that if the work carried out by the reviewers is low-quality, then there would be less interest in said journal. Consequently, the token's value would be low, quite the opposite if the quality of the reviews is high, in which case the token's value would be high. So, through this mechanism, it is sought that the reviews are of good quality for the own interests of the reviewers.

One more proposal to consider is a blockchain application that empowers researchers, publishers, and others to integrate and access peer-review information in a decentralized data structure, such as blockchain [50]. In this way, they use the benefits of incorporating this technology, such as immutability, decentralization, cryptography, and autonomy for data security and privacy. This proposal presents an application integrating an open-access peer-review journal such as F1000Research.

Another proposal is this paper [51] that focuses on presenting a self-organizing peer-review in preprint. The authors added techniques, such as the concept of token economy and blockchain technology, to the peer-review process. The first technique was used in an incentive and penalty mechanism design for peer-reviewers; in this way, they seek to ensure the quality of the publications. Regarding blockchain technology, they use its decentralization characteristics for its implementation. However, they detail that the proposal cannot guarantee a successful future due to negative traits such as problems with scalability, which consequently has a lower performance of blockchain-based applications.

A proposal that encompasses more than scientific publications is the work presented in the article [52], which seeks to combat centralization around the current systems that

manage conferences and journals specialized in scientific research. Likewise, it aims to confront specific negative characteristics, such as the lack of transparency. This is the reason why they propose a decentralized alternative for the management of scientific conferences and publications through decentralized technologies such as Blockchain and IPFS. This proposal has a prototype portal where the authors can share articles with the organizers of a scientific research event. Also, they implemented symmetric key encryption of the articles to guarantee the confidentiality and authenticity of the intellectual property.

Among other exciting proposals, "Decentralized Science" (DecSci) [53] seeks to decentralize the system around scientific publications. Their system uses distributed technologies such as blockchain and IPFS to create a decentralized peer-review system based on an open-access infrastructure together with processes of transparent government. Unlike other proposals, this one does not seek to reward reviewers with cryptocurrency but instead enables a decentralized reviewer and review reputation system. Around this proposal, its authors have created a proof of concept prototype, a minimum viable product, and a cost analysis regarding the operations executed in the blockchain.

AntReview [54] is a project that offers a blockchain-based solution to address the drawbacks of the peer-review process, which is time-consuming, unrecognized, and unremunerated. To address these issues, Trovo and Massari propose an incentive protocol that rewards scientists for their contributions to the work of other scientists. Additionally, they implement a reputation system. This protocol allows authors to issue a call for peer-review, and if there is a fulfillment of the requirements, the reviews will be audited and paid for by the issuer. The system also implements a quadratic funding model to promote ethical behavior. Unlike the other projects, the authors include their project repository link in the paper.

Among the current proposals, one seeks to address the waiting time problem between the submission for review and the first resolution [55]. Likewise, it aims to increase the objectivity of the reviews, ensuring that there are no biases or human errors. To do so, they propose a framework in the cloud based on blockchain technology, the latest technology mentioned as a method to improve anonymity between authors and reviewers. This proposal takes the exact structure of the current submission system but adds a decentralized solution (blockchain) developed in a Java-based system.

An article [56] published in 2023 highlights problems with the traditional scientific publishing system, such as high costs, slow review processes, and a lack of rewards for contributors. To address these issues, the authors propose a blockchain-based decentralized platform that uses Ethereum smart contracts to speed up the publication process, reduce costs, and improve the quality of studies. The proposed system rewards cited editors, reviewers, and authors and makes scientific articles available worldwide for a small fee. The system was implemented using the Ethereum Virtual Machine and consists of a front-end, a middleware, and a back-end. The system automatically finds the appropriate editors and reviewers for related fields and rewards contributors with a token-based cryptocurrency. The proposed platform aims to make scientific publications more accessible, traceable, and decentralized.

Throughout this chapter, we review various investigations or system proposals with similar objectives to this thesis project. However, most of these scientific articles are only theoretical proposals, do not provide results, do not provide the code or project repository, or do not have a prototype. So this limits us to making qualitative comparisons regarding

beneficial ideas and proposals that focus on different challenges, except for the AntReview project [54], which provides the link to the project's GitHub repository. Also, the DeSci [53] project presents a code repository link and an evaluation table of the computational cost of executing the system's primary functions, expressing the cost in terms of gas, ethers, and US dollars. These last two projects allow us to make a comparison with our proposal.

For qualitative analysis, we will consider the last four proposals analyzed, which are the most recent. We will also take into account the traditional system for the comparison. The parameters are the network type, economic incentive or tokenization, code availability, IPFS usage, the type of system they have implemented, and their consensus mechanism.

System	Type of Network	Tokenization	Code Availability	IPFS Usage	System Type	Techniques/Protocols addition
Traditional	Centralized	No	No	No	Journal Peer-Review System	No
DeSci [53]	Ethereum	No	Yes	Yes	Journal Peer-Review System	Reputation System
Ants-Review [54]	Ethereum	No	Yes	Yes	Journal Peer-Review System	Standard Bounties
Gazis et al. [55] Proposal	Decentralised Cloud-based	No	No	No	Manuscript Submission System	Cloud-Based
Bestas et al. [56] Proposal	Ethereum	Yes	No	No	Journal Peer-Review System	Token Incentive
Our Proposal	Ethereum	Yes	Yes	Yes	Journals Peer-Review Ecosystem	Journals Network. Token Incentive. Reward and Penalty protocol. Reputation System.

Table 3.1: Peer-review systems proposal comparison

In the table 3.1, we can see how each project differs in terms of characteristics. However, all of them are a peer-review system for a journal except our proposal. This project contemplates a network of journals that use our peer-review infrastructure. Our infrastructure is characterized by offering various options of techniques and protocols for the consideration of each journal, such as the incentive through payment with tokens like [56], reward and penalty protocol that some oracle blockchain-based projects use like [57], or a reputation system like [53]. In addition, journals will also be able to customize other secondary features that are part of the peer-review system. In this way, our proposal differs significantly from other proposals. Since our proposal not only seeks to present an alternative system that combats the shortcomings of the traditional system but also creates an ecosystem for the scientific community. All of these features will be explained more specifically in the methodology section.

Chapter 4

Methodology

The development or methodology of this proposal has 3 phases, theoretical design, practical design, and testing. In the first segment, we will contemplate the initial idea and the development of its main characteristics, such as the system design, workflow, operability, interactivity scheme, and description of functions and protocols. On the other hand, in the practical development, we describe the mechanisms and tools necessary to implement the system proposed in the previous phase. Also, we specify the distribution of the functions in different smart contracts and explain the pseudocode for each primary function. In this way, the necessary process for the replication of the algorithm of this project is detailed. The final section of the methodology involves testing. In this last phase, we will develop unit tests to evaluate the correct functioning of the functions implemented in the previous phase. Figure 4.1 summarizes structurally the phases of the project development, where phase one simulates being the base of the creation. At the same time, the peak is the testing of the application.

4.1 Theoretical Design

4.1.1 Proof of Concept

Several successful cases of ideas/projects implemented with the blockchain infrastructure have generated significant results. Among them is the case of the most outstanding platform using Blockchain technology, bitcoin. Bitcoin [9], as mentioned above, is the first cryptocurrency created. Cryptocurrencies are digital assets used to exchange value just like traditional money; it differs by using cryptography as a means of security for exchange [58]. Bitcoin is currently the most used cryptocurrency with the greatest support globally. Among the reasons why this project was born is to present a better alternative for the finance environment. Where there is no central banking entity or trusted parties, thus presenting decentralization as the basis of the project. Likewise, it takes banks off the map, proposing a viable and scalable project for the community through proof-of-work protocols and consensus systems. After more than a decade of its creation, this project has become highly innovative and is an excellent alternative to traditional money; its acceptance is growing every year.

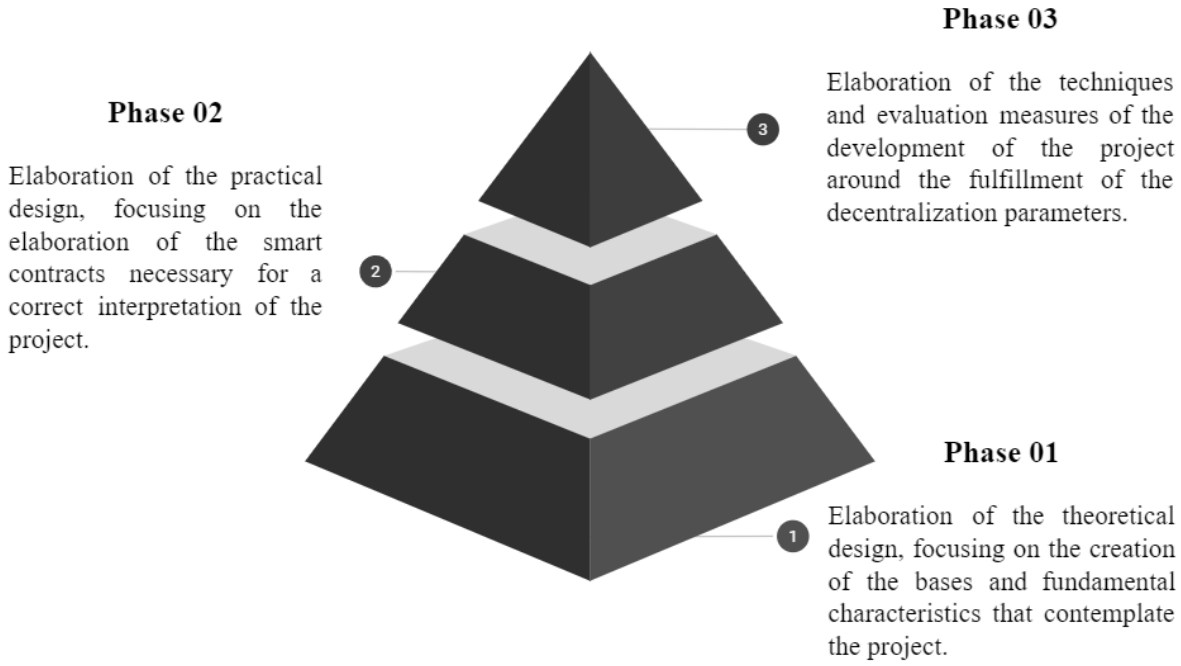

Figure 4.1: Phases that contemplate the development of the project.

Taking into account the situation of the Bitcoin project, which presented a disruptive alternative to the financial environment. This project proposal is similar in being a disruptive alternative in the scientific-academic environment. This project seeks to present a proposal using a decentralized infrastructure within the web3 where the scientific community can make use of the tools and benefits offered by blockchain technology. In that way, the process of publishing and reviewing scientific articles will be improved. This project expects that decentralized scientific journals will be managed by the scientific community and not by publishers. However, thanks to the system's design, existing journals with a centralized hierarchy can also be hosted. In this way, regardless of the type of journal, they will be able to use our peer-review mechanism.

4.1.2 Proposal

This project aims to offer a tool for peer-review, which different journals can use. We create a system for adding centralized or decentralized journals to accomplish this. The tool offers different protocols so journal developers can customize the peer-review model according to each journal's standards. Therefore, the project proposes the creation of a general system for the community where users (scientists) can register and, at the same time, journals can also join. When a journal is created, it will be able to choose the necessary features for the peer-review model they wish to use. In this way, journals and users become the prominent participants in the overall community system, which we will call the "community network". As for journals, they will have their own system for their peer-review process, which we

will call “journal peer-review infrastructure” or “JPRI”.

4.1.3 Community Network

This first system is not very complex; it is a simple system that aims to accommodate users belonging to the scientific community, who will take on the role of reviewers, research authors, or journal editors. It also seeks to host new or existing journals that wish to create a JPRI. Additionally, the system will have its own token, which the different journals will use for payment purposes in their peer-review processes. Therefore, for the design of this system, we will focus on the creation of the structures and functionalities of the three mentioned components in the following order:

1. Token
2. Journals
3. Users

Token

The system token is a transactional token; its functionality is essential since it will be used for the system's interoperability between participants and smart contracts, specifically for publication payment, peer-review, and journal creation. We call the token of this project “ThesisCoin” with its abbreviation “TSC”. We implement the standard functionalities that an ERC20 token must have [59], such as:

- TotalSupply(): provides information about the total token supply.
- BalanceOf(): provides the account balance of the owner's account.
- Transfer(): executes transfers of a specified number of tokens to a specified address.
- Approve(): allow a spender to withdraw a set number of tokens from a specified account
- TransferFrom(): executes transfers of a specified number of tokens from a specified address done by a spender.

These five functions will allow system users to transfer the token for payments, purchases, or subscriptions, among other functions, securely. In the practical design section, we can find a detail of each function, including the pseudocode for its implementation.

Journals

The journals are the most complex component of the Community Network section since, at the time of their registration, the system will grant a specific Smart Contract for their JPRI. In this section, we will discuss the functions prior to its creation, as well as the declaration of the characteristics of its JPRI.

- RequestJournalCreation(): Request the creation of a JPRI for a magazine.
- SeeRequests(): Provides the list of Journals creation requests.
- VoteRequest(): Allows voting regarding creating a journal.
- SeeRequestStatus(): Allows visualizing the status of a journal creation request.
- CreateJournal(): Allows the creation of a JPRI for a magazine.
- SeeJournals(): Provides the list of existing journals.

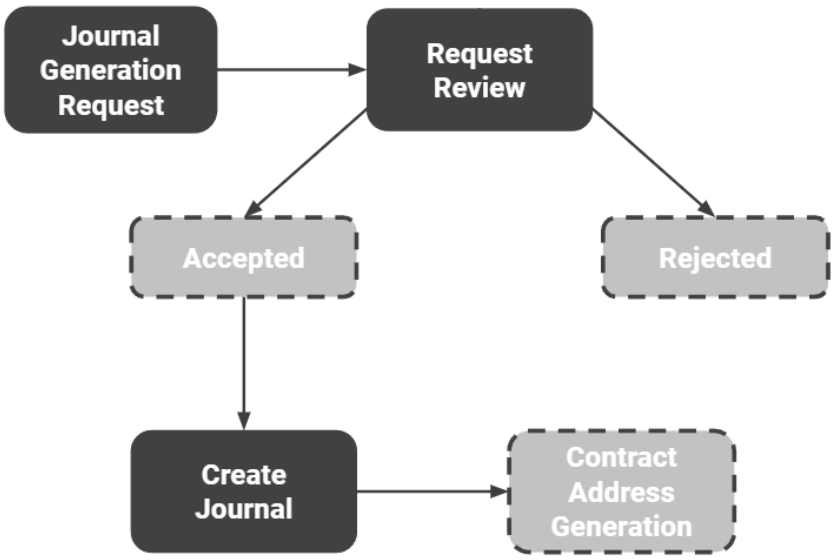

Figure 4.2: Process that has to pass a journal to be created.

These are the main functions that are necessary for the process of creating a journal. As can be seen in figure 4.2, to create one, journal developers must request permission for its creation. Subsequently, the community will decide to approve or reject the request. In consequence, a consensus protocol is necessary, which is described below:

In an established time, the consensus protocol used to approve or reject a request to create a journal is by voluntary vote. In this way, the community decides which journals could be created according to their analysis and perspectives. The journals with more than 50% of in favor votes will be approved. On the contrary, it will be rejected. In this way, it is sought that predators or pseudo-real journals cannot enter to be part of the system since the community will not allow it.

After the community has approved a journal for its creation, the journal's developers will be able to execute the proper functions for its creation, which also involves the creation of a JPRI, whose structure will be analyzed later. In addition, the details of each function belonging to this segment of creating a journal will be explained in the smart contracts section of practical design.

Users

Users are fundamental to this system since they will be the community that makes this project have a large scale and impact. This system will store the personal and academic information of the users, as well as the research or reviews carried out in the various journals indexed to the system. This information may be viewed or used by journals for their peer-review processes. The functions that a user can perform are:

- UserRegister(): Allow registering as a user in the system.
- UpdateData(): Allows the user to update their personal data.
- BuyTokens(): This function was already explained in the Token section.
- SeeJournals(): This function was explained in the Journals section.
- VoteRequest(): This function was already explained in the previous section.
- Interact in the Peer-Review process: As such, the user will be able to perform many more functions while interacting with the JPRI smart contract, where its functions explanation will be in later sections.

This segment is short in terms of functions since it only involves user registration and updating data functions. However, this does not mean that users have little relevance. On the contrary, they are the ones who will execute many of the public functions of this system. We already explained these functions in previous segments concerning tokens and journals. Likewise, users will execute certain functions corresponding to the JPRI segment.

4.1.4 Journal Peer-Review Infrastructure(JPRI)

The JPRI is the system designed for a journal to implement its Peer-Review process with blockchain technology's features. As mentioned before, in order for a journal to have its own peer-review tool, it must first request its creation in the Community Network system. After the acceptance of the application, the journal can implement the generated JPRI smart contract. The Peer-Review process proposed in this project consists of 4 general steps, which are found in figure 4.3. There are two peer-review steps. The first has the objective of being the review stage of the received manuscript, while the second has the objective of reviewing the revision made in the first peer-review. Besides, the characteristics of each stage may vary since the system offers to customize those stages according to the parameters considered by each journal.

Before starting the peer-review process, in other words, to upload an article to a journal or to be a reviewer for a journal, the user must first enroll in the journal. In order to enroll, the system will check the address of the function performer if he/she is part of the users of the Community Network system. Therefore, before the peer-review process, a journal has an enrollment function. In addition, there is another function to change the status of the reviewer role in the journal.

- Enrol(): Allows the user to enroll in a journal to participate in its peer-review process.

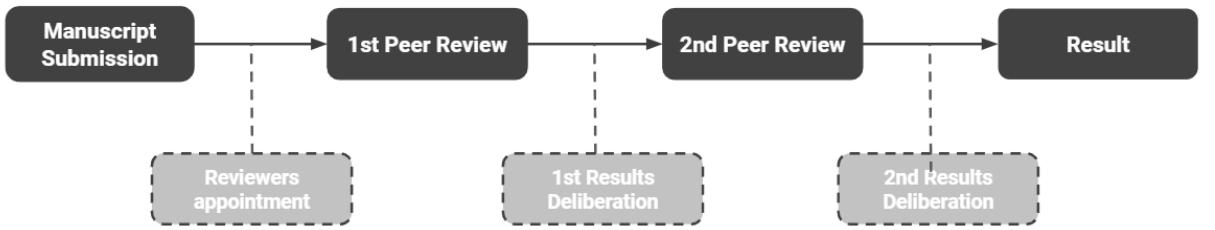

Figure 4.3: General steps of the peer-review process.

- ReviewerState(): Allows the user to update the reviewer role status.

On the other hand, the system must also consolidate certain functions around the journals, specifically functions regarding the protocols and characteristics that the journals chose at the time of their creation. Since the features of the JPRI model are according to the selected parameters of a journal, we must establish a good structure and distribution of those features in this step.

First Group Selection Protocols:

The system needs to establish the appropriate functions to choose the reviewers for the first peer-review group. The system offers four selection options:

- RandomSelection(): picks reviewers at random.
- DesignationSelection(): allows a journal editor to choose reviewers.
- PostulationSelection(): allows reviewers to apply to review a manuscript.
- RatingSelection(): picks reviewers according to reputation and availability.

In this way, the system will choose the group of reviewers according to the chosen protocol and the number of integrants established in the creation process of the journal.

Second Group Selection Protocols:

As in the previous selection segment, the system needs to establish the appropriate functions to choose the reviewers that will conform to the second group of peer-reviewers. However, there is also the possibility that this thread is performed directly by an editor and not by a group of reviewers. Therefore, we established four selection functions:

- RandomSelection(): picks reviewers at random.
- EditorSelection(): picks a editor from the editor list.
- PostulationSelection(): allows reviewers to apply to review a manuscript.
- RatingSelection(): picks reviewers according to reputation and availability.

Incentive Protocols:

Finally, the system provides extra protocol options such as incentives and penalties. These protocols are used in various real projects, such as financial projects or blockchain architectures where miners carry out proof of stake. If the miner does a good job, he or she receives an incentive, while if they seek to take advantage of the system dishonestly or by doing a lousy job, then he receives a penalty. In this project, we implement protocol options similar to those described above. The focus of these protocols is on the work performed by the reviewers. However, these protocols can be controversial in this area. That is why they are optional, and each journal will decide whether or not to add them according to their parameters and visions.

- **Token Incentive - Penalty Protocol:** This protocol seeks to optimize reviews, giving reviewers a greater incentive to do a good job. However, the question of having a monetary incentive can result in fraudulent actions by reviewers. In these cases, there is the counterpart of the incentive, which is the monetary penalty. The determination of a good review is subject to the second review. The reviewers of the second batch are specifically responsible for reviewing the first review. In this protocol, the monetary value of the system token is at stake because, in a hypothetical case where the reviewers carry out a dishonest job in the reviews, and despite this, they receive an economic incentive. This situation would result in the journal losing prestige and decreasing its users. Therefore the token's value would be affected.

  - TIPP(): receives payment and executes an incentive or penalty at the end of the process.

  A journal can implement this protocol in its JPRI infrastructure. However, to do so, it must make certain clarifications, such as the necessary parameters to consider a well-done review. In this way, there is no possibility of ambiguity.

- **Rating Incentive - Penalty Protocol:** This protocol is similar to the previous one. However, it does not involve tokens but rather the reviewer's reputation. In this way, the incentive is for the reviewer to have greater recognition through positioning in a ranking. At the same time, the penalty could also fall on the reputation, suspension, or even exclusion from the system. Like the previous protocol, the journals must provide the necessary clarifications for no ambiguities in the decisions.

  - RIPP(): increases or decreases a reviewer's rating according to the results of their work.

- **Payment Incentive Protocol:** The system allows journals to choose economic protocols, whether the journal charges a fee for the publication process or no fee. It may also be the case that it is optional; an example of its use would be that the process does not have a cost; however, there is an optional payment alternative for a faster review and publication process. Therefore the payment alternatives are:

  - PIP(): makes a payment to the reviewers.

Blind Peer-Review Protocol:

Blind peer-review is a common method used in academic publishing to ensure an impartial and unbiased evaluation of research papers. Among the existing methods, we have:

- Single-blind review: In a single-blind review, reviewers do not know the identity of the author(s) of the manuscript. Nevertheless, the reviewers' identities are known to the authors.
- Double-blind review: In a double-blind review, the authors and the reviewers conceal their identities from each other.
- Triple-blind review: In a triple-blind review, the identities of the authors, reviewers, and editors are all concealed from each other. This method is less common but is used in some journals to ensure maximum impartiality and minimize bias.

This last method, Triple-blind review, is used in this project to respect the objectives of this work about providing a system that allows for improving the characteristics of a peer-review process.

In this segment, we describe all the protocols that involve the system. The details of each function described in this segment will be specified in the smart contract development section of the practical design.

Manuscript Submission

This is the first step of the process, where a user uploads a manuscript of research they have carried out. We can assume that the primary function of this segment is only uploading a file and the respective data associated with the research. However, the truth is that in this segment, there are other necessary functions, such as those designed to execute the functions corresponding to the reviewer group selection protocols. These functions are the following:

- UploadManuscript(): allows the user to upload a manuscript.
- 1stReviewersSelection(): executes the function under the protocol that the journal stipulated for the first reviewers group.
- 2ndReviewersSelection(): executes the function under the protocol that the journal stipulated for the second reviewers group.
- IncentiveSelection(): executes the function under the protocol that the journal stipulated for the incentive to reviewers.

The process that transcends the first stage consists of an internal process such as the one seen in figure 4.4. The UploadManuscript function executes internally the rest of the functions corresponding to the protocols.

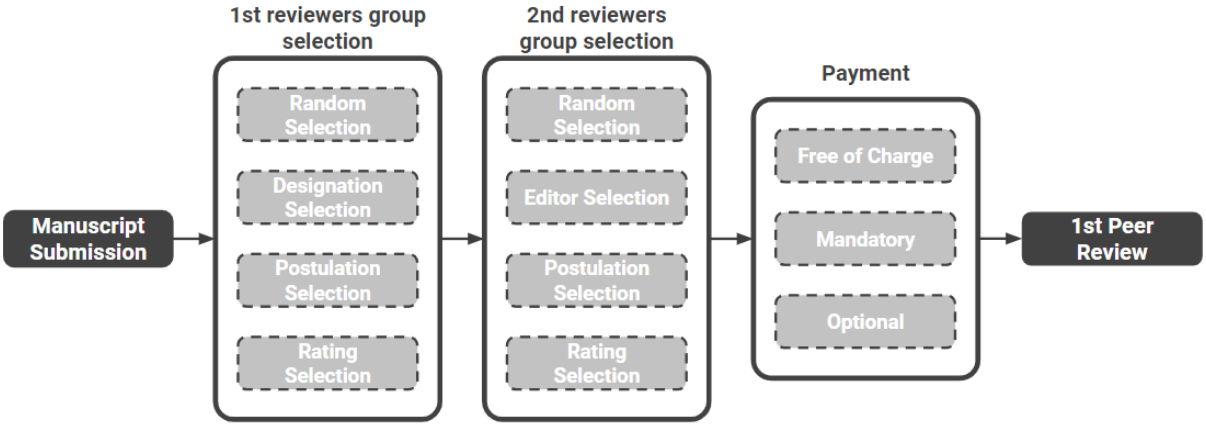

Figure 4.4: General steps of the manuscript submission.

First Peer-Review

This is the second station of the process, where the selected reviewers must accept or reject the assigned review; if accepted, the reviewer must continue to review the paper. In case the journal has a token incentive-penalty protocol (TIPP), the reviewer must stake tokens. After having carried out the review, the reviewer must upload the result and the respective feedback or justification. If a reviewer does not agree to perform the review, the system will select a replacement.

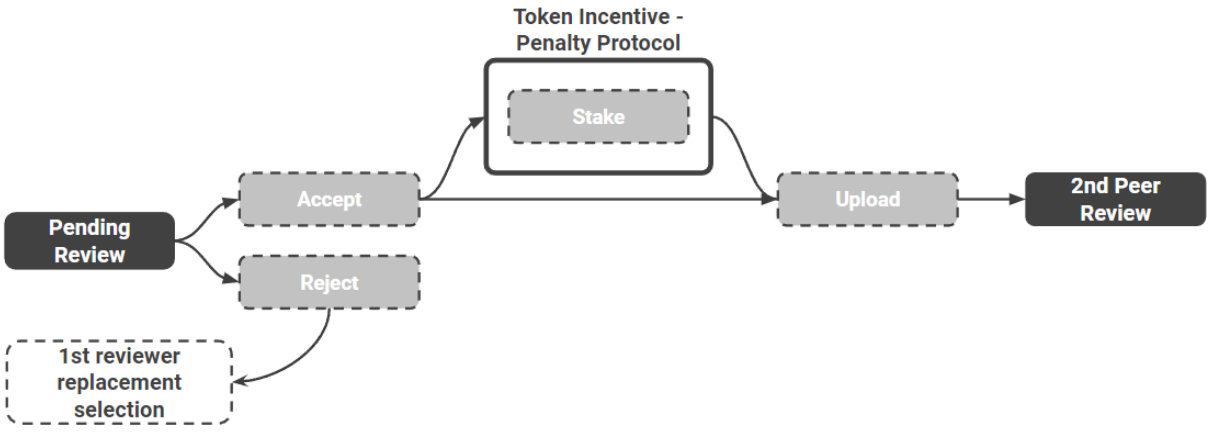

Figure 4.5: General steps of the 1st peer-review stage.

For the correct execution previously described, several functions will be needed, such as:

- CheckPendingsReviews(): allows a reviewer to see their pending reviews.
- AcReReview(): allows a reviewer to accept or reject a review.
- OtherReviewer(): choose a new reviewer for a review process.
- UploadReview(): allows uploading the review result and its attachment.

Second Peer-Review

In this third part of the process, the system addresses the second review round, where another selected reviewer group will evaluate the previous review. The objective of this stage is to there a quality process. For this, the group of selected reviewers or the selected editor must accept or reject the assigned work. In the editor's case, it will simply upload its result and the respective justification for its evaluation. The same happens for the group of reviewers. However, the result will be according to the calculation regarding the verdicts of each member of the group of reviewers.

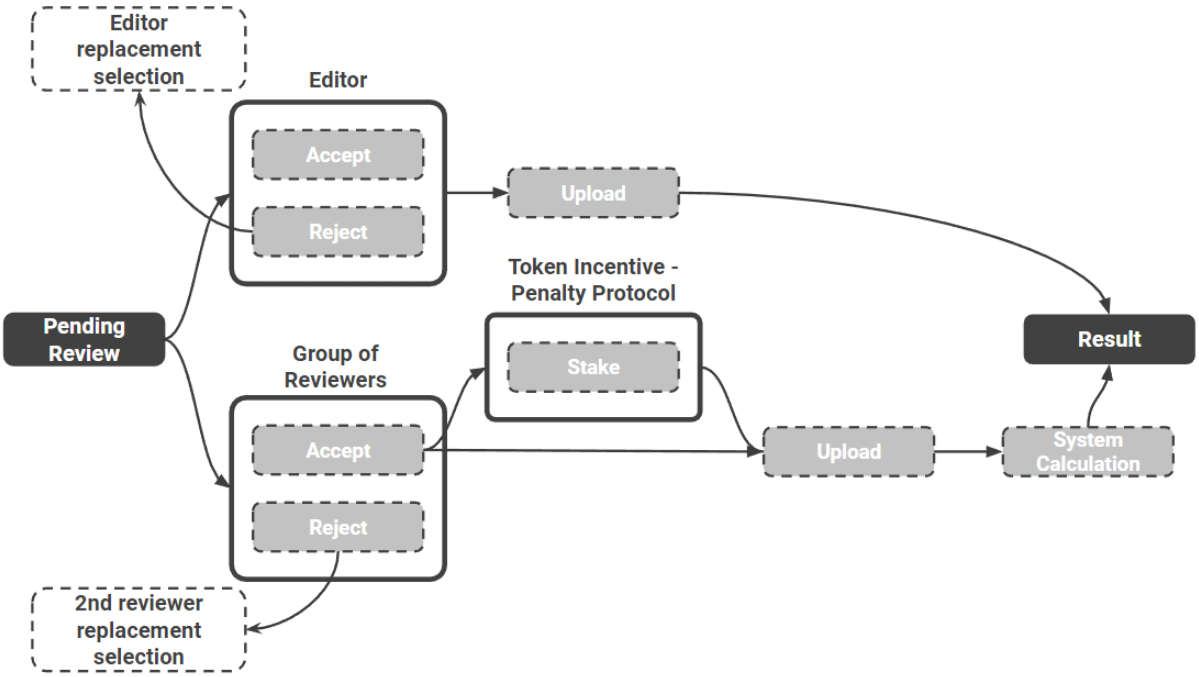

Figure 4.6: General steps of the 2nd peer-review stage.

- CheckPendingsReviews(): allows a reviewer to see their pending reviews.
- AcReReview(): allows a reviewer to accept or reject a review.
- OtherReviewer(): choose a new reviewer for a review process.
- OtherEditor(): choose a new editor for a review process.
- UploadReview(): allows uploading the review result and its attachment.
- See1PRResult(): allows the second reviewers to see the work done by the first reviewers group.

Result

This is the last step of the process, where the manuscript can have three types of results, either accepted, rejected, or needs changes. Therefore, the research author will be able to

review the result. If the manuscript neejeeds changes, the author can start the process by re-uploading the manuscript. Likewise, in this step, the execution of the corresponding incentive protocol is completed.

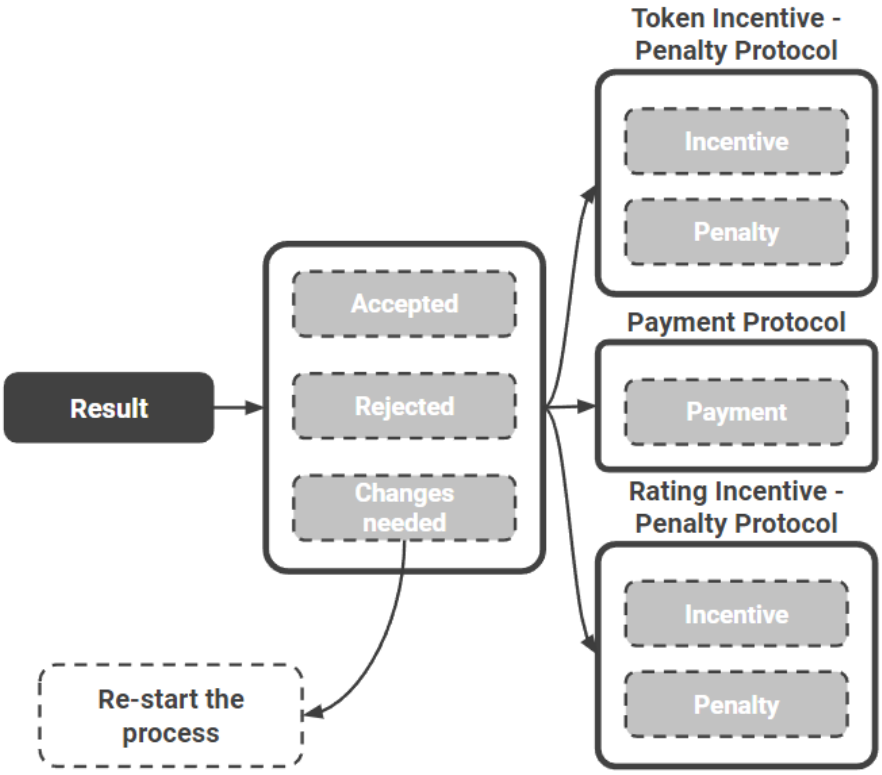

Figure 4.7: General steps of the final stage of the process.

- SeePRResults(): allows the author to see the peer-review result.
- ReuploadManuscript(): allows the author to reupload the manuscript.
- IncentiveProtocolExecution(): executes the incentive or penalty to the author according to the protocol used by the journal.

4.2 Practical Design

In this section of the project, the objective is to present the procedure and the necessary tools to implement a system that meets the fundamental characteristics of the project detailed in the theoretical design. To perform this, we use different types of tools and technologies to achieve the development of this project. We also present a segment where we explain and provide a pseudo-code.

4.2.1 Tools

The necessary tools for the development of the project vary between those that are focused on the backend and blockchain.

Backend Tools

- **Solidity** is the programming language that allows us to create smart contracts for the Ethereum network [60]. It is a high-level language whose main objective is to implement contracts simply and easily. Therefore, its syntax is very similar to that of other high-level languages, such as JavaScript. This language would carry out all the necessary logic for storage and operability within the blockchain. We use this language to implement our system in smart contracts.

Blockchain Tools

- **Remix** is an open-source web-based Integrated Development Environment by the Ethereum Foundation. Developers use Remix to write, test and deploy smart contracts on the Ethereum blockchain or in a TestNet. It has a user-friendly interface, a built-in Solidity compiler, a debugger, and a testing framework [61]. Remix has become an essential tool for Ethereum developers and has been used to build a wide range of decentralized applications (DApps) on the Ethereum blockchain. We will use this IDE to write the code and connect it with other tools, such as Ganache and Metamask.

- **TestNet** or Ethereum testnet is a network that imitates the Ethereum mainnet without using real Ether [28]. Developers use testnets to test their smart contracts without incurring the fees of the real Ethereum blockchain. Popular Ethereum testnets include Goerly, Sepolia, Ropsten, Rinkeby, and Kovan, providing a simulated environment that closely reaches the actual EVM features for testing purposes. Using testnets, developers can identify and fix bugs or issues before deploying their smart contracts on the real Ethereum mainnet. So, it allows developers to ensure that their smart contracts work as expected and minimizes the chance of losing real funds. Due to this, we will use a test network to simulate the operation of the smart contract without fee charges.

- **Etherscan** is a web-based tool that operates as a block explorer for the Ethereum network or Testnets block explorer [62]. It allows users to explore transactions, addresses, and smart contracts. Etherscan provides real-time updates on the latest blocks, transactions, gas prices, and recorded data on past transactions and blocks. Etherscan is widely used by Ethereum users and developers to explore and interact with the Ethereum blockchain, providing a wealth of details and data for monitoring and analyzing activity on the network. We use this tool to check the details of the smart contract transactions, such as the gas used and the running time.

- **Truffle Suite** is an open-source development framework for building decentralized applications and smart contracts. It simplifies the developing, testing, and deploying of smart contracts by providing a suite of tools, including a testing framework,

build pipeline, and automated contract deployment [63]. Truffle supports Solidity programming language and blockchain platforms like Hyperledger Fabric and Corda. We use the helpful Truffle tools to efficiently develop, test, and deploy the system's smart contracts.

- **Ganache** is a program belonging to the Truffle Suite. It is a personal blockchain that facilitates the development of a blockchain environment locally [63]. This local blockchain imitates the Ethereum network behavior. It deploys by default on the user's computer and provides a simple user interface for managing. It provides accounts addresses with fictitious ethers to simulate transactions and interact with smart contracts. It also supplies various testing tools and features that support developers in debugging and troubleshooting their smart contracts. We choose this Truffle Suite component to test and deploy the project's smart contracts locally.

- **MetaMask** is a digital wallet and a browser extension that allows netizens to connect personal accounts and securely interact with decentralized applications [63]. In this way, users integrate their browsers with the Ethereum blockchain technology. It allows users to access and manage their Ethereum-based assets(tokens). Therefore, they can easily send and receive cryptocurrency, swap, lend, borrow, and trade tokens. We will use this application to test the token system's operability using test accounts.

Complementary Tools

- **Github** is a web-based cloud service platform for version control using the Git version control system to let teamwork create and merge code branches [64]. It also enables multiple developers to store, share and work on the same codebase simultaneously. For this reason, it is a good place for open-source software projects. GitHub provides various features for collaboration, such as issue tracking, code reviews, and pull requests. We use GitHub to host our code and share it with project-interested people.

- **Visual Studio Code** or VS Code is a free and open-source code editor [65]. It supports many programming languages and provides several features and functionalities that help developers improve their work. It also has a large extension marketplace that allows developers to customize and enhance their development environment with additional tools and features. VS Code is known for its lightweight design, fast and efficient development, and built-in terminal that allows developers to execute commands and run scripts without leaving the editor. We use this code editor because it allows us to plug in blockchain tools.

- **InterPlanetary File System** or IPFS is a distributed file system for storing and sharing files in a peer-to-peer network [63]. A content-addressed hash identifies files, and no central server is required. IPFS uses a Distributed Hash Table (DHT) to manage the network and ensure that files are distributed and replicated across the network, making them available and accessible. IPFS offers improved efficiency, security, and resilience compared to traditional centralized file storage systems and has a wide range of potential applications, including hosting decentralized websites and providing a more secure and efficient way to share large files. We use this

platform to store large files such as manuscript pdfs. This method allows us to reduce the Ethereum gas fee for data storage. So, we only attach the large files' IPFS Content Identifier(CID) in the blockchain.

4.2.2 Smart Contracts

Using smart contracts, we implement all logic and necessary functions established in the theoretical design stage. Following that steps, we create three smart contracts.

1. Contract for the creation of the Project Token.
2. Contract for the Community Network.
3. Contract for the Journal Peer-Review Infrastructure.

Token Smart Contract

In the theoretical design section, we established the token as a component of the Community Network. At the time of code development, it is better to separate it into a specific smart contract and focus only on the token. In this way, we can improve the readability, consistency, and understanding of the distribution and development of our smart contracts.

For the development of the functionalities of the project token(THESIS), it is necessary to define specific structures, variables, and functions such as the following:

```
1  variable _totalSupply = 10000;
2
3 #Here the amount of tokens that each account(address) has will
      be saved.
4 dictionary _balances[address] = int;
5
6 #This dictionary will have as its key a list that will store
    two addresses, the first concerning the "owner" and the
    second for the "spender", while the dictionary key value is
    the amount the owner allows the spender to transfer to a
    third account.
7
8 dictionary _allowances[list]= int;
9
10 function Sender.Address(){
11   #This function returns the address of the account executing
       the function; in the case of solidity, the corresponding
       function is "msg.sender".
12  }
```

Now that we have declared some fundamental components, we can start with developing the pseudo-code of the functions detailed in the theoretical section.

1. **TotalSupply()** is a simple function that does not contain input parameters; it simply returns the value assigned to the variable "_totalSupply", as can be seen in the following pseudocode:

```
function TotalSupply(){
return _totalSupply};
}
```

---

2. **BalanceOf()** It is a function that receives an input argument which is "address" type; the objective of this function is to return the balance or number of tokens that an account has. So inside the function, it is searched in the "_balances" dictionary using the input argument as the key. In this way, the function returns the respective value to the given key.

```
function BalanceOf(address _account){
  return _balances[ _account];
}
```

---

3. **Transfer()**, this function takes two input arguments, an address, and an integer. The first is the address of the user who will receive the tokens. The second argument is the number of tokens to be transferred. For this function, it is necessary to check that the user has more tokens in his account than the amount indicated in the argument. Then the account tokens amount will be updated in the dictionary.

```
function Transfer(address _to, int _amount){
  address _From = Sender.Address();
  if (_balances[ _From] >= _amount){
              _balances[ _From] -= _amount;
              _balances[ _to] += _amount;
}
}
```

---

4. **Approve()**, This function takes two input arguments, an address, and an integer. The first is the address of a user(_spender) that will be allowed to use the tokens of another user(_owner) who runs the function. The second argument is the number of tokens to be approved. The dictionary called "_allowances" stores these two input arguments. The dictionary key is the list containing the addresses of the '_owner' and the '_spender', while the value is the '_amount'.

```
function Approve(address _spender, int _amount){
  address _Owner = Sender.Address();
  _allowances[_Owner, _spender]=_amount;
}
```

5. **TransferFrom()**, this function is similar to Transfer. The difference is that whoever executes the function does not directly participate in the transfer of the token but is a third party that was approved to transfer someone else's tokens. This function receives three parameters, the address of the owner of the tokens, the address of the user who will receive the tokens., and the number of tokens to be transferred. Within the function, there is a segment that verifies that whoever executes the function has the necessary permissions to carry out the transfer. Then the balances of the owner and receiver addresses are updated. At the same time, the function modifies the"_allowances" dictionary, decreasing the number of tokens approved for the performer(_Spender).

```
function TransferFrom(address _from, address _to, int
    _amount){
  address _Spender = Sender.Address();
  if (_allowances[_from, _Spender] >= _amount){
      if (_balances[_from] >= amount){
        _balances[_from] -= _amount;
        _balances[_to] += _amount;
        _allowances[_from, _Spender] -= _amount;
}}}
```

Community Network Smart Contract: Journals

The Smart Contract will have components that represent the journals and the users. In this section, we will focus on developing functions that correspond to journals. For the development of the journal functions, it is necessary to define specific structures, variables, and functions such as "journalrequest", which is a structure where its components are the necessary data for a request. We also have "_JournalRequests" which is a dictionary that stores the journal creation requests. In addition, we have "Journal", which is a structure that contains the necessary data for the journal creation. We continue with the "NewId" function, which aims to return an ID for a new request. Finally, we have the "TimeNow" function, which aims to return the value of the current time.

```
Structure journalrequest[address _Requester, string _Name,
    string _AboutLink, time _Time, string _State, int _UpVotes,
    int _DownVotes];

dict _JournalRequests[int] = journalrequest;
```

```
4
5 Structure journal[string _Name, string _ReviewersProtocol,
    string _ReviewProtocol, int _Cost, int _Reviewers1, int
    _Reviewers2];
6
7 dict  _Journals[address] = journal;
8
9 function NewId(){
10   if (length(_JournalRequests)>0){
11   return _JournalResquests[-1]._id + 1;
12   }else{
13   return 0;}
14 }
15
16 function TimeNow(){
17   #This function returns the value of the instantaneous time
       in which the function is executed.
18 }
```

Now that we have declared some fundamental components, we can start with developing the pseudo-code of the functions detailed in the theoretical section.

1. **RequestJournalCreation()**, this function receives as parameters the necessary data to generate a journal creation request structure and subsequently stores it in the "_JournalRequests" dictionary.

```
1 function RequestJournalCreation(string _name, string
    _aboutlink){
2 journalrequest _NewJournalRequest = journalrequest[Sender.
    Address(), _name, _aboutlink, TimeNow(),   in
    progress, 0, 0];
3 address _idrequest = NewId()
4 _JournalRequests[ _idrequest] = _NewJournalRequest;
5 return  _idrequest;
6 }
```

2. **SeeRequests()**, this is a simple function since it does not receive parameters; it simply returns the dictionary of existing requests.

```
1 function SeeRequests(){
2   return _JournalRequests;
3 }
```

3. **VoteRequest()**, this function allows community users to vote for or against the creation of a journal within an established time. Therefore, two parameters are received. First, the ID of the request to which a user wants to cast their vote, and the second one is the vote either "in favor" or "against". Before saving the vote, the function verifies that the function is running at the established voting time. If the condition is fulfilled, the function continues to save the vote received.

```
function VoteRequest(int _id, string _vote){
  if (TimeNow() - _JournalRequests[_id]._Time < 604800){
    if(_vote == in favor ){
      _JournalRequests[_id]._UpVotes+=1;
}else if (_vote == against ){
  _JournalRequests[_id]._DownVotes+=1;
}else{
return the time to vote is over. }
}
```

---

4. **SeeRequestStatus()**, the purpose of this function is to display the status of a request. For this, the parameter is the request identifier. Subsequently, the function checks if the voting time is over. If the time is up, the function returns the option with the most votes. If the voting time is not over yet, it returns that the voting is still in progress.

```
function SeeRequestStatus(int _id){
  if (TimeNow() - _JournalRequests[_id]._Time >= 604800
     and _JournalRequests[_id]._State == in progress
     ){
    if (_JournalRequests[_id]._UpVotes > _JournalRequests[
       _id]._DownVotes){
_JournalRequests[_id]._State = accepted ;
} else{
_JournalRequests[_id]._State = rejected ;}}
  return _JournalRequests[_id]._State;
}
```

---

5. **CreateJournal()**, to develop this function, a previous function is needed, which is "GenerateJournalContract", that will receive the JPRI template contract address as an argument.

```
function GenerateJournalContract(address _owner){
```

```
2   #This function aims to generate an address of a Smart
        Contract for the journal that requires it and meets
        the requirements. The address generated will be a
        template of the smart contract (we will call it ``
        JPRIContract) containing the journal's Peer-Review
         infrastructure. The details of this template are in
        the JPRI section.
3   return new JPRIContract(_owner);
4 }
```

CreateJournal function receives as an argument the characteristics of the JPRI. First, the function performs a verification of the journal creation approval. Subsequently, the function generates a "journal" structure with the features data. Then, the function generates the respective JPRI contract for this journal. Finally, the function stores all in the "_Journals" dictionary.

```
1 function CreateJournal(string _name, string
      _reviewersprotocol, string _reviewprotocol, int _cost,
      int _reviewers1, int _reviewers2l){
2   if (_JournalRequests[_id].Requester == Sender.Address()
        and _JournalRequests[_id]._State == accepted ){
3   journal _NewJournal = journal[_name, _reviewersprotocol,
        _reviewprotocol, _cost, _reviewers1, _reviewers2);
4   address _JournalAddress = GenerateJournalContract(sender
        .address());
5 _Journals[_JournalAddress] = _NewJournal;
6   return _JournalAddress;
7 }
8 }
```

---

6. **SeeJournals()**, this is a simple function that does not contain parameters; it simply aims to return the dictionary with the journals stored in the system.

```
1 function SeeJournals(){
2   return _Journals;
3 }
```

Community Network Smart Contract: Users

Users are the second component of this smart contract. In this section, we will focus specifically on the development of user-related functions. For the development of these functions, it is necessary to create a structure for the users where the essential data of the users would be stored. In the same way, a dictionary is necessary where we will store the created users.

```
1
2 Structure user[string _FullName, string _Email, string
      _University, string _AcademicField, string _CV, bool
      _RevisorRol, list _Journals, list _Articles, list _Reviews]
3
4 dict _Users[address]= user;
```

Now that we have declared some fundamental components, we can start with developing the pseudo-code of the functions detailed in the theoretical section.

1. **UserRegister()**, this function receives six parameters; four of them are personal information, another is the curriculum vitae, a link that redirects to the file stored in IPFS will be stored here, and the last parameter is about the decision to have a reviewer role. A "user" structure will store this information, and the "Users" dictionary will store the "user" structures.

```
1
2 function UserRegister(string _fullname, string _email,
      string _university, string _academicfield, string _cv,
      bool _revisorrol){
3    user _NewUser[ _fullname, _email, _university,
        _academicfield, _cv, _revisorrol, [empty],[empty], [
        empty]];
4    _Users[Sender.Address()]=_NewUser;
5 }
```

2. **UpdateData()**, this function receives five of the six parameters of the previous function since the objective of this function is to update the data if the user so requires it.

```
1 function UpdateData(string _email, string _university,
      string _academicfield, string _cv, bool _revisorrol){
2    _Users[Sender.Address()]._Email = _email;
3    _Users[Sender.Address()]._University = _university;
4    _Users[Sender.Address()]._AcademicField = _academicfield
        ;
5    _Users[Sender.Address()]._CV = _cv;
6    _Users[Sender.Address()]._RevisorRol = _revisorrol;
7 }
```

JPRI Smart Contract

Unlike the previous ones, this smart contract implements a more complex system, so it is necessary to distribute the code in the best way. This section will establish certain variables and structures necessary for constructing this smart contract. Subsequent sections will focus on the functions according to each step of the peer-review process.

As mentioned above, prior to the Peer-Review process, it is necessary to establish these two recently described functions. In addition, it is also necessary to establish a constructor of the smart contract that would receive the parameters that the journal creators established for the operation of the JPRI.

```
1 Constructor{ string _name, string _reviewersprotocol, string
    _reviewprotocol, int _cost, int _reviewers1, int _reviewers2
    };
2
3 Struct member[list _Articles,list _Reviews, bool _RevisorRol,
    list _ArticlesPending, list _ReviewsPending];
4
5 dict _Subscriptions[address]=member;
6
7 list _Reviewers[];
8
9 list _Editors[];
```

---

1. **Enroll()**, this function receives only one argument that deals with whether the user wants to be considered a reviewer in some peer-review process. The function creates a structure to store user information, such as their articles and reviews.

```
1
2 function enroll(bool _revisorrol){
3   if (addres.sender() in CommunityNetwork._Users){
4     if (address.sender() not in _Subscribers){
5     member _NewMember = member[[empty], [empty],
          _revisorrol, [empty], [empty]];
6     CommunityNetwork._Users[address.sender()]._Journals.
          push(journal.name);
7     _Subscribers[address.sender()]=_NewMember;}
8 }
```

---

2. **UpdateReviewerState()**, this is a simple function that receives an argument to update the reviewer role status of the user executing the function.

```
1
2 function UpdateReviewerState(bool _revisorrol){
```

```
    _Subscribers[address.sender()]._RevisorRol = _revisorrol
        ;
}
```

JPRI Smart Contract: Manuscript Submission

For a correct functioning of these functions, it is necessary to declare several lists, dictionaries, variables, and structures concerning the manuscript and reviewers.

```

Struct manuscript[address _autor, string _title, string
    _linkpaper, string _campo, list _reviewers1L, list
    _decisionstate1, list _decisionlink1, list _reviewers2L
    , list _decisionstate2, list _decisionlink2, string
    _State]

dict _Manuscripts[string]=manuscript;
```

1. **RandomSelection** this function will randomly choose a specific number of reviewers; the journal established this amount in its creation. The way to choose these reviewers is by the timestamp and a nonce.

```
int randNonce = 0;

 function RandomSelection(int _num){
            randNonce+= 1;
   int _modulus = length(_Reviewers);
   list _reviewersR;
   for(i=0; i<_num; i++){
_reviewersR.push(_Reviewers[uint(keccak256(abi.
    encodePacked(block.timestamp,msg.sender,randNonce))) %
    _modulus];)
}
return _reviewersR;
     }
```

2. **DesignationSelection** this function is part of a centralized protocol alternative since the system does not choose the reviewers but rather an editor who is part of the journal.

```
1
2 function DesignateEditors(address _editor){
3 if (address.sender() == address.owner()){
4 _Editors.push(_editor);
5 }
6 }
7
8 function DesginationSelection(list _Reviewers[]){
9   if (address.sender() in _Editors){
10   return _Reviewers;
11 }
12 }
13
14 function DesginationSelectionEditor(){
15   randNonce+= 1;
16   int _modulus = length(_Editors);
17 return _Reviewers[uint(keccak256(abi.encodePacked(block.
     timestamp,msg.sender,randNonce))) % _modulus];     }
```

3. **PostulationSelection** this function allows a user to apply as a reviewer for the review of a manuscript that does not have reviewers.

```
1 function PostulationSelection(_id){
2   if (Manuscripts[_id]._State == Pending) {
3 if (length(Manuscripts[_id]._reviewers1L)<_reviewers1){
4   Manuscripts[_id]._reviewers1L.push(address.sender());
5   if (length(Manuscripts[_id]._reviewers1L)==_reviewers1){
6   Manuscripts[_id]._State = In Process;
7   }
8 }
9 }
10 }
```

4. **ReviewerSelection**this function redirects to the function corresponding to the protocol selected in the creation of the journal.

```
1
2 function Reviewers1Election(){
3   if (_reviewersprotocol ==      RandomSelection      ){
4 return RandomSelection(_num)}
5 else if (_reviewersprotocol ==    Designation     Selection
    ){
6 return DesignationSelection()}
```

```
7 }
8
9 function Reviewers2Selection(int _num){
10   if (_reviewersprotocol ==      RandomElection      ){
11 return RandomElection(_num)}
12 else if (_reviewersprotocol ==    Designation     Election )
     {
13 return DesignationElectionEditor()}
14 }
```

---

5. **UploadManuscript()** this function takes the manuscript data as a parameter and stores it in the respective variables. It also ensures that the attached protocols are complied with and initialized.

```
1
2
3 function NewIdM(){
4   if (length(_Manuscripts)>0){
5   return _Manuscripts[-1]._id + 1;
6   }else{
7   return 0;}
8 }
9
10
11 function UploadManuscript(string _title, string _linkpaper
     , string _campo){
12   _autor = address.sender()
13   list _reviewers1L = reviewersElection1();
14 list _reviewers2L = reviewersElection2();
15 bool _pago = 0;
16 int _idM = NewIdM();
17 for(int i=0; i< length(_reviewers1L); i++){
18   _subscribers[_reviewers1L[i]]._pendingreviews.push((_id
       ,1));
19   _subscribers[_reviewers2L[i]]._pendingreviews.push((_id
       ,2));
20 }
21 _state =     Pending     ;
22 if balanceof(address.sender())>=_cost){
23   transfer(contract.address(), _cost);
24   _pago = 1;
25 }
26 if (_pago and length(_reviewers1L)== _reviewers1 and
     length(_reviewers2L)== _reviewers2){
27   _state =   In   process ;
```

```
28 }
29    manuscript _NewManuscript = manuscript[_autor, _title,
          _linkpaper, _campo, _reviewers1L, [empty], [empty],
          _reviewers2L, [empty], [empty], _state];
30    _Manuscripts[_idM]=_NewManuscript;
31 }
```

JPRI Smart Contract: 1st Peer-Review

In this section, the necessary functions were constructed to assign reviewers, as well as the functions required by reviewers to upload their findings.

1. **CheckPendingsReviews()** this function is straightforward as it only returns the number of pending reviews for an individual.

```
1 CheckPendingsReviews(){
2 return _subscribers[address.sender()]._pendingreviews}
```

2. **AcReReview()** this function is used for a reviewer to present the decision made regarding the review conducted.

```
1 AcReReview(int _id, string _decision){
2    if (address.sender() in _Manuscripts[_id].reviewers1L){
3 if _decision ==      reject     {
4    OtherReviewer(_id, address.sender());
5 } else{
6 if (_reviewprotocol ==     TIPP     ){
7    transfer(contract.address(), _stakecost);}
8 return _Manuscripts[_id]._linkpaper;
9 }
10 }}
```

3. **OtherReviewer()** this function is employed to assign another reviewer in the event that a previously chosen reviewer is unable to perform their task.

```
1 OtherReviewer(int _id, address _noreviewer){
2    _Manuscripts[_id].reviewers1L.pop(_noreviewer);
3    _Manuscripts[_id].reviewers1L += reviewersElection1(1);
4 }
```

4. **UploadReviewer()** this function allows reviewers to upload their decision and the corresponding report that supports the decision.

```
UploadReview(int _id, string _decision, string
    _decisionlink){
if (address.sender() in _Manuscripts[_id].reviewers1L){
manuscripts[_id]._decisionstate1.push(_decision);
manuscripts[_id]._decisionlink1.push(_decisionlink);
}}
```

JPRI Smart Contract: 2nd Peer-Review

This section repeats several functions used in the previous section; however, certain functions have specific differences, as in this segment, the decision can be made by the same group of reviewers or by an editor of the journal.

1. **CheckPendingsReviews()** this function is straightforward as it only returns the number of pending reviews for an individual.

```
CheckPendingsReviews(){
return _subscribers[address.sender()]._pendingreviews}
```

---

2. **AcReReview2()** this function is used for a reviewer or editor to present the decision made regarding the review conducted.

```

AcReReview2(int _id, string _decision){
  if (address.sender() in _Manuscripts[_id].reviewers2L){
if _decision ==     reject     {
  OtherReviewer(_id, address.sender());
} else{
if (_reviewprotocol ==     TIPP    ){
  transfer(contract.address(), _stakecost);}
return _Manuscripts[_id]._linkpaper;
}
}}
```

---

3. **OtherReviewer()** function is employed to assign another reviewer in the event that a previously chosen reviewer is unable to perform their task.

```
function OtherReviewer(int _id, address _noreviewer){
    _Manuscripts[_id].reviewers2L.pop(_noreviewer);
    _Manuscripts[_id].reviewers2L += reviewersElection2(1);
}
```

4. **OtherEditor()** this function is employed to assign another editor in the event that a previously chosen editor is unable to perform their task.

```
function OtherEditor(int _id, address _noeditor){
    _Manuscripts[_id].reviewers2L = editorElection();
}
```

5. **UploadReview()** this function allows reviewers or editors to upload their decision and the corresponding report that supports the decision.

```
function UploadReview(int _id, string _decision, string
    _decisionlink){
if (address.sender() in _Manuscripts[_id].reviewers2L){
manuscripts[_id]._decisionstate2.push(_decision);
manuscripts[_id]._decisionlink2.push(_decisionlink);
}}
```

JPRI Smart Contract: Result

In this segment, the author of a research study receives the outcomes of the review process of their manuscript.

1. **seePRResults()** this function assists an author in verifying the status of the review of the submitted manuscript.

```
function seePRResults(){
if (address.sender() == _Manuscripts[_id]._autor){
return manuscripts[_id]._State}}
```

2. **ReuploadManuscript()** this function allows an author to re-upload a corrected manuscript after having received suggestions for changes.

```
function ReuploadManuscript(int _id, string _linkpaper){
if (address.sender() == _Manuscripts[_id]._autor){
if (manuscripts[_id]._State = 'Changes'){
manuscripts[_id]._linkpaper = _linkpaper;}}}
```

4.3 Testing

After the project's development, we must evaluate the functionalities that are part of the smart contracts, which, as mentioned above, are autonomous codes executed on the blockchain. This section is of great importance for this project. The tests will verify that the contracts meet their objectives and have a correct operation, thus guaranteeing security before their execution in the blockchain. Unlike traditional software, smart contracts are immutable after being implemented in a public blockchain; that is, they are complexly upgradeable or improvable after their launch, so it is indispensable to carry out stringent tests for optimal implementation in a network like Ethereum. For this, we need to rigorously analyze and evaluate the quality of the written source code. Through testing, we can identify errors and vulnerabilities, thereby reducing the possibility that the software presents errors that could lead to costly consumption of Ethereum gas or massive and irrecoverable losses for system users.

There are two categories for these tests: automated and manual.

4.3.1 Manual Testing

Programmers or auditors carry out this kind of testing. They will execute the steps to follow in the program operation manually. This section includes audits where the developer or an external expert in the subject evaluates or audits each line of the contract code to find a possible vulnerability [66]. That is why this type of testing requires excellent skill and experience on the part of the auditor, as well as time and effort. However, this does not offer us plenty of security since it is susceptible to human error. On the other hand, among the positive points of these tests is that human intelligence could contribute by finding defects in the code that could go unnoticed in an automated testing process. Likewise, this type of testing can find vulnerabilities outside the code, problems, or errors that could arise in the user and program interaction.

4.3.2 Automated Testing

This type of testing involves automated tools focusing on executing written code tests repeatedly to find defects in smart contracts. This test category is efficient since it uses a few resources and offers more coverage than manual analysis. In addition, test data can be used in these tests, allowing us to visualize behaviors close to or equal to those expected to have in the real use of the project.

Among the different types of automated tests, we can consider the following:

1. **Functional Testing:** This type of test verifies the contract's functionality and ensures the correct operation of each function part of the code. For this, it is necessary to understand the correct behavior of the contract in different conditions. Subsequently, the functions' evaluation consists of comparing the expected values with the output received. Functional testing has three methods: Unit Test, Integration Test, and System Test.

    - **Unit Testing:** This test explicitly implies the correct functioning of each of the components [67] of the smart contract. It consists of testing the written

functions one by one. This test gives us a clear and concise idea of where an error can be located. Since a developer can check that the behavior of each function execution is as expected. For this type of test, the auditors make assertions, thereby verifying that the output is as expected.

- **Integration Testing:** This testing, unlike unit testing, focuses on running a test of all components together [68]. This test seeks to find errors or vulnerabilities during the interactions between the different functions of the contract or between multiple contracts.

4.3.3 Testing Tools

Various tools and platforms are available for automated testing on smart contracts, such as Truffle, Mocha, or Chai. These tools allow developers to write specific test cases covering different scenarios and situations where a user would use the smart contract. In addition, these tools also provide detailed reports on the performance and functionality of the smart contract, making it easy to identify and resolve potential bugs or vulnerabilities. These test environments are run separately from the main blockchain, allowing developers to test their contracts without worrying about the costs associated with transactions on the blockchain. Among the most popular tools we have:

- **Testing frameworks:** such as Mocha, Chai, and Truffle, which provide a framework and a set of functions for writing and running tests.
- **Automated testing tools:** such as Ganache, which allows you to create a test network and simulate transactions in real-time.
- **Code review tools:** such as Mythril, Listener, and Securify, which analyze contract code to identify potential vulnerabilities and bugs.
- **Real-time monitoring tools:** such as Remix Debugger, which allows you to trace and debug contracts while they are running on a network.

Different testing tools will be used for testing the current project, prioritizing the tools Truffle and Remix offer.

4.3.4 Testing Process

Functional Testing Process

We performed the functional testing process following the next steps:

1. Write unit tests for each function of smart contracts to verify the expected behavior of the contract in different situations.
2. Set up a test environment, in our case Ganache, to run the tests.
3. Run the unit tests in the test environment and verify that the results are as expected.

4. Analyze the tests' results to proceed with a correction or improvement in case of identifying any error or problem in the code.

5. Write and perform integration tests to verify the interaction between the different contracts and the implemented inheritance.

It is important to note that this testing process is iterative. Performing several multi-step iterations is necessary until the contract is error-free. We carry out the test cases in this project using the javascript programming language. For a specific example of the type of test we perform, here is the test pseudocode for the UserRegistration() function:

```
Initialize CommunityNetworkContract and chai.expect

Create a contract test for CommunityNetworkContract with accounts
    Declare variable CommunityNetworkContractInstance
    Declare variable userAccount as accounts[1]

    Create a beforeEach block
        Deploy a new instance of CommunityNetworkContract and assign it to CommunityNetworkContractInstance

    Create a test case "correct registration test"
        Define user details
            fullname <- "Harvey Marin"
            email <- "harvey.marin@yachaytech.edu.ec"
            university <- "Yachay Tech University"
            academicfield <- "Computer Science"
            cv <- "https://ipfs.filebase.io/ipfs/exampleCID"
            revisorrol <- true

        Call UserRegister function with user details and userAccount

        Retrieve registeredUser from the contract using _Users(userAccount)

        Verify user details
            Check if registeredUser._fullname is equal to fullname
            Check if registeredUser._email is equal to email
            Check if registeredUser._university is equal to university
            Check if registeredUser._academicfield is equal to academicfield
            Check if registeredUser._cv is equal to cv
```

```
29          Check if registeredUser._revisorrol is equal to
               revisorrol
30          Check if registeredUser._journals.length is equal
               to 0
31          Check if registeredUser._articles.length is equal
               to 0
32          Check if registeredUser._reviews.length is equal
               to 0
```

Chapter 5

Results and Discussion

This chapter presents the gas cost results of each smart contract and its primary functions. We focus on the gas cost of each function for different reasons. such as each operation executed in the blockchain has an associated cost. The user who executes the function pays the transaction fee, so the transaction cost must be as cheap as possible. Furthermore, the Ethereum blockchain has established gas limits. So the amount of gas required by a function must be below the limit. For this study, we use Remix VM (merge fork) to analyze the gas that each function requires. Even though we use a work environment replicating the main environment, it is still a separate network with its own computational resources and limitations. Moreover, the gas limit is determined by the nodes in the network. So, while the gas cost for a transaction may be similar between the testnet and the mainnet, the gas limit on each network may differ. That is why the results obtained in this section should be considered indicative and serve only to establish references.

We have decided that the parameters to evaluate are the values that Remix IDE gives us when executing a function. These values are transaction gas and execution gas in terms of Gwei. Gwei is equal to 0.000000001 ETH or 10^9 Wei; Wei is the smallest unit of Ethereum. One Ether (ETH) is equal to 10^{18} Wei. The price of gas can vary depending on network congestion and other factors. For this reason, in the following gas analysis tables, we will find columns referring to the dollar price, which varies around the gas price. For example, the price of gas in the Ethereum network is 31 Gwei at the time of the data gathering, whose date was March 12, 2023. Likewise, on that day, the price of Ether, the currency of Ethereum, was 1590.83 US dollars. These gas and Ether prices vary depending on the date, so the values present in the tables will be different depending on the day of value analysis.

5.1 Gas Cost Analysis

In table 5.2, we can see the analysis of the six primary functions of the CommunityNetwork smart contract. The first five functions have fair values because the process within each function is nothing more than the creation of structures, data statements, and a few conditionals. However, in the case of the CreateJournal function, its amount of gas is much higher than the other functions, 20 times higher than the UserRegister function. This hap-

pens because the CreateJournal function deploys a JPRI smart contract, which requires a high computational cost. The cost of gas for these two smart contracts is analyzed in the table 5.3.

Function	**Transaction Gas**	**Execution Gas**	**Dollar (1 Gwei)**	**Dollar (31 Gwei - Ethereum)**	**Dollar (3 Gwei - BSC)**	**Dollar (0.1 Gwei - Arbitrum One)**
UserRegister:	124828	101680	0,20	6,14	0,60	0,02
Update Data:	39202	16798	0,06	1,93	0,19	0,01
RequestJournalCreation:	211683	188739	0,34	10,41	1,01	0,03
VoteRequest:	72105	50409	0,11	3,54	0,34	0,01
SeeRequestStatus:	37847	16655	0,06	1,86	0,18	0,01
CreateJournal:	2607182	2583590	4,15	128,16	12,44	0,41

Table 5.1: Transaction and execution cost of the CommunityNetwork smart contract functions in terms of gas and dollars.

Table 5.2 analyzes 15 functions that we consider to be the main ones in the JPRI smart contract. Where seven functions are of low computational cost, these are:

- UpdateReviewerState
- Postulation
- EditorSelection
- Assignation
- GetManuscriptFile
- Get1gReviewFile

While the other five functions present higher gas values, unlike the previous seven functions, these here have a more complex process. As is the use of for loops, however, the executed iterations are few, which causes only a slight increase in gas. These functions are:

- RandomSelection
- AcceptOrRejectReview
- SubmitFirstGroupReview
- SubmitSecondGroupReview
- AnalyzeSGDecisions
- FinalizeReview

Finally, the RatingSelection and UploadManuscript functions have high gas costs. In the case of the UploadManuscript function, this is because it presents the execution of other functions inside it, such as the one that assigns reviewers to a manuscript. In comparison, the Rating Selection function presents a for loop that iterates through the list of reviewers belonging to the journal. This list can be extensive, which would generate a high cost in gas consumption. That is why the value found in the table is an average value obtained from the different gas costs that this function produced. In the tables 5.4, **??** and 5.6, we can find descriptive analyzes of the executions of the functions presented by for loops.

Function	**Transaction Gas**	**Execution Gas**	**Dollar (1 Gwei)**	**Dollar (31 Gwei - Ethereum)**	**Dollar (3 Gwei - BSC)**	**Dollar (0.1 Gwei - Arbitrum One)**
UpdateReviewerState	65862	44658	0,10	3,24	0,31	0,01
addEditor	90385	68953	0,14	4,44	0,43	0,01
RandomSelection	228137	206656	0,36	11,21	1,09	0,04
Postulate	77094	55387	0,12	3,79	0,37	0,01
RatingSelection	946887	925425	1,51	46,55	4,52	0,15
EditorSelection	98300	76459	0,16	4,83	0,47	0,02
Designation	77609	55941	0,12	3,81	0,37	0,01
UploadManuscrpt	589953	568664	0,94	29,00	2,82	0,09
AcceptOrRejectReview	240184	218737	0,38	11,81	1,15	0,04
SubmitFirstGroupReview	173462	151968	0,28	8,53	0,83	0,03
GetManuscriptFile	40735	18887	0,06	2,00	0,19	0,01
SubmitSecondGroupReview	168572	147115	0,27	8,29	0,80	0,03
Get1gReviewFile	41433	19848	0,07	2,04	0,20	0,01
AnalyzeSGDecisions	233340	211850	0,37	11,47	1,11	0,04
FinalizeReview	392870	371356	0,62	19,31	1,87	0,06

Table 5.2: Transaction and execution cost of the JPRI smart contract functions in terms of gas and dollars.

Apart from the functions, the cost of gas presented by the smart contract JPRI and CommunityNetwork was also analyzed. In the table 5.3, we can see that the smart contract community network is twice as expensive as the jpri smart contract. This is because the community network inherits the JPRI smart contract for the creation of the magazines and uses the Openzepellin ERC20 smart contract for the creation and use of a token for the system. This leads to the cost of gas being relatively high. On the other hand, the JPRI smart contract has a high gas value because it has a large number of functions. This is because a transaction for the deployment of contracts has a fixed cost of 31,000 gas units, to which 200 gas units must also be added for each octet that makes up the bytecode. Therefore, the greater the length of the contract, the greater its bytecode will be when compiled, and therefore, it will need more gas for its deployment.

Function	**Transaction Gas**	**Execution Gas**	**Dollar (1 Gwei)**	**Dollar (31 Gwei - Ethereum)**	**Dollar (3 Gwei - BSC)**	**Dollar (0.1 Gwei - Arbitrum One)**
Community Network	5219363	4793063	8,30	256,57	24,91	0,83
JPRI	2434056	2204332	3,87	119,65	11,62	0,39

Table 5.3: Transaction and execution cost of the CommunityNetwork and JPRI smart contracts in terms of gas and dollars.

	RandomSelection		RatingSelection	
	Transaction Gas	**Execution Gas**	**Transaction Gas**	**Execution Gas**
Mean:	228137,14	206656,02	946887,56	925425,49
Std:	33991,47321	34003,35702	270570,1309	270575,4899
CV:	14,89957891	16,45408492	28,5746843	29,23795517
Min:	167871	146691	475798	454701
25%:	198784,25	177187,5	726343,5	705039,5
50%:	230677,5	209137,5	960154,5	938609
75%:	259251,5	237918,75	1167363	1146137,25
Max:	283061	261138	1390721	1369181

Table 5.4: Descriptive statistics of RandomSelection and RatingSelection functions.

5.2 Statistical Data on the Functions Gas Cost

We previously mentioned that our code has functions whose computational cost may vary due to the number of iterations that the function performs. Given this, we take a sample of 100 executions for each function. The objective was to obtain a range of gas consumption for each function. In addition, obtain statistical data such as the mean, the standard deviation, the coefficient of variation, minimums, quartiles, and maximums. In this way, to have a more exact vision of the consumption that the execution of each function can have.

The first functions to be analyzed were selection by randomness and rating. The first function has an average of 228137 gas consumption and a coefficient of variation of 14.89%, which allows us to interpret that there is not a significant difference regarding consumption. This is because this function iterates low numbers relative to the number of reviewers a specific review process has. In comparison, the RatingSelection function has a coefficient of variation of 28.57%. This function is alarming because it iterates the list of reviewers associated with a journal, which can be a long list that would cause a lot of gas consumption. The number of registered reviewers was not exaggerated in our sample for this function. However, if it were, alternatives would have to be sought, such as using better ordering algorithms or using other off-chain means to carry out the respective calculation. The rest of the descriptive values of these functions are described in the table 5.4.

In the 5.5 table, we find three functions that reviewers execute. The first is the function that allows a reviewer to accept or reject the assigned review. This function has a high coefficient of variation because the gas consumption when the reviewer accepts is low, but when the review is rejected, the gas consumption is high. This is because if the review is rejected, the function executes an internal function to assign a replacement reviewer. The Submission functions for the first and second groups have a for loop that iterates through the list of reviewers assigned in a peer review process, which is why their computational cost also varies. However, the coefficient of variation is low.

	AcceptOrRejectReview		SubmitFirstGroupReview		SubmitSecondGroupReview	
	Transaction Gas	**Execution Gas**	**Transaction Gas**	**Execution Gas**	**Transaction Gas**	**Execution Gas**
Mean:	240184,9	218737,47	233340,89	211850,33	168572,46	147115,02
Std:	87587,11539	87434,88228	11899,07832	11920,15996	11357,19509	11310,54722
CV:	36,46653698	39,97252153	5,099439845	5,626689351	6,737277899	7,688234162
Min:	92258	75423	213882	192614	147836	126588
25%:	168591	146785,75	223281,75	201741,25	158893,75	137859,25
50%:	242070,5	220495	234239,5	212778	167694	146402,5
75%:	318677,5	297336,75	243569,5	221956,25	177869,25	156084,75
Max:	379000	357647	253604	231864	188128	166558

Table 5.5: Descriptive statistics of AcceptOrRejectReview and Submition functions of the JPRI.

	AnalyzeSGDecisions		FinalizeReview	
	Transaction Gas	**Execution Gas**	**Transaction Gas**	**Execution Gas**
Mean:	233340,89	211850,33	392870,06	371356,04
Std:	11899,07832	11920,15996	21540,07309	21534,05315
CV:	5,099439845	5,626689351	5,482747422	5,798762059
Min:	213882	192614	355407	333649
25%:	223281,75	201741,25	375805,5	354075,5
50%:	234239,5	212778	389012	367666
75%:	243569,5	221956,25	410458,75	388747,25
Max:	253604	231864	431831	410812

Table 5.6: Descriptive statistics of AnalyzeSGDecisions and FinalizeReview functions of the JPRI.

The last two functions to be analyzed belong to the final peer review process. The AnalyzeSGDecision function determines whether a manuscript is accepted, rejected, or requires changes. This function varies in its computational cost because it must iterate between the decisions given by the reviewers. In comparison, the FinalizeReview function contains the AnalyzeSGDecision function and other operations to be performed.

In the table 5.7, we can see how the computational costs of the functions of this proposal have a significant difference concerning the others. This is because this proposal presents functions that involve complex processes, such as the execution of internal functions. As is the case with "SendPaper", in our case, it would be "UploadManuscript" which executes the assignment of the reviewers automatically and does not have an external function that must be executed by a third party to perform an assignment. In addition, our functions present require statements as security protocols, which consume more gas but offer greater security for the system. Therefore, the other proposals have functions with lower computational costs. However, their functions are simpler, basic, and limited, in addition to the fact that they need security standards. Compared to our smart contract that presents security standards and a complete process.

Function	**DeSci**	**AntsReview**	**BloxBerg**	**Proposal**
SendPaper()	114812	98393	-	568664
AssignReviewers()	58707	-	-	76469
AcceptForReview()	23971	24239	21403	218737
UploadReview()	149760	132938	110599	151968
Publish()	-	19380	19843	-
UnPublish()	-	19,292	19,829	-
UploadChanges()	-	97349	-	44658
SeeReview()	-	14,593	17,329	18,384
Contract Deployment	-	1630257	1469931	2204332

Table 5.7: Comparison table of the gas cost of similar functions by each proposal.

Chapter 6

Conclusions

Scientific publications have a highly relevant mechanism such as the peer-review process, serving as a quality control mechanism by allowing peers to scrutinize and critique work before it is published. However, traditional peer-review processes are fraught with several challenges. These include issues of bias, low quality, lack of transparency, time consumption, expensive costs, potential for manipulation, and the recognition of the work involved. However, the advent of blockchain technology presents a unique opportunity to transform this space.

As a distributed ledger technology, blockchain offers a potential solution to these challenges. Its features of immutability, transparency, security and decentralization make it a suitable candidate to enhance the peer-review process. The immutability of the blockchain ensures that once data is stored on the blockchain, it cannot be changed. This provides a robust mechanism for protecting intellectual property and ensuring the integrity of the review, with a transparency feature that enhances the accountability and openness of the review process. The security of blockchain technology also prevents tampering or fraud, while its decentralization enables a more democratic and fair process, leading to better recognition of reviewers' work. By setting specific processes and protocols, we can create the right environment for a robust and reliable peer review process.

In this proposal, we presented a peer-review system that leverages blockchain technology to revamp the traditional peer-review process. This system introduces a novel, decentralized approach to academic publishing processes. This system offers an environment for the scientific community, with its participants being individual academics and academic journals. A central element of this system is a framework for the peer review process, called JPRI (Journal Peer Review Infrastructure). This infrastructure is versatile and customizable providing a flexible solution that caters to a wide range of needs, by offering different protocols. The objective achieved is that the journals can adapt and personalize their own review process according to the agreements, guidelines and fundamentals that each journal has. In addition, this system has its own token that is used for publication payment and remuneration for reviewers.

This decentralized system reduces the potential for collective bias, manipulation, or coercion, as it adheres to the principles of anonymity and immutability. By doing so, it can be sure to maintain the integrity of the review process and keep the quality of published content. Proposed features include a transparent review process, security against

fraud, immutable comments, and approval of reviewers. The system also allows for a more democratic review process, with power distributed among peers rather than concentrated in the hands of a few. Alternatively, the use of smart contracts can automate several aspects of the process, reducing administrative burden and speeding up publication time. By implementing solid programming, the system has robust features such as manuscript submission, reviewer selection and decision-making. These features are designed in a way that mirrors the real-world peer review process.

After the development of the system's smart contracts, we conducted an in-depth analysis of the system's computational cost, focusing on the gas cost of each function, and its equivalent in ethers, and dollars. In this way, providing a comprehensive overview of the system's performance metrics. On comparing our proposal with existing alternatives, it emerges that our system incurs higher costs as expected. However, this is a direct consequence of the system's complex and extensive feature set, which offers a more complete and refined peer-review process compared to other proposals. With the information provided from performance metrics of each function, the implementation of the smart contract was further optimized and refined.

This project, despite its relative novelty, stands poised to make a substantial and far-reaching impact. It presents a viable solution to the challenges that plague traditional peer-review processes. Therefore it has the potential to revolutionize the peer-review process. By offering a transparent, decentralized, and secure solution, we can reshape the landscape of scientific publications, facilitating a peer-review process that truly upholds the principles of scholarly integrity and robust academic discourse.

However, the system is not without challenges. The complexity of some features can present challenges. In addition, due to usage dependencies of each journal, the value of the system token may fluctuate, which may cause disagreements among the community. Ethereum's transaction costs and reliance on network speed affect the efficiency of the system. Furthermore, widespread acceptance and use of such systems requires a paradigm shift in academia towards blockchain technology. These are important factors that must be considered to ensure the viability and sustainability of the system. To alleviate these limitations, we recommend implementing some logic code on the frontend to reduce gas costs and blockchain dependencies. We also recommend exploring off-chain alternatives and external databases to store large amounts of data, thereby optimizing the efficiency of the system. Another recommendation would be to allow each magazine to generate its own token for its own purposes and independence. Future work includes continuing to refine and optimize the system, especially presenting this project to the Ethereum community for further enhancement. We aim to encourage the community to contribute by adding new functions, protocols, and paradigms. In addition, we hope to scale up to a comprehensive system that integrates front-end and back-end components, along with databases, to make this system more robust and efficient. Another avenue we're considering is to present the project for grants offered by various blockchain network projects. This approach would allow us to work with mentors and experts in the field, thereby promoting the project more effectively. In closing, this project serves as a pioneering step towards harnessing the power of blockchain technology in the realm of academic publishing, opening up new possibilities for the future of the peer-review process.

Bibliography

[1] A. Thakur and R. Verma, "An empirical three phase analysis of crypto market," 03 2022.

[2] B. Mudrak, "Scholarly publishing: A brief history," *AJE Expert Edge*, 2020.

[3] M. Thelwall, L. Allen, E.-R. Papas, Z. Nyakoojo, and V. Weigert, "Does the use of open, non-anonymous peer review in scholarly publishing introduce bias? evidence from the f1000research post-publication open peer review publishing model," *Journal of Information Science*, vol. 47, no. 6, pp. 809–820, 2021. [Online]. Available: https://doi.org/10.1177/0165551520938678

[4] V. M. Nguyen, N. R. Haddaway, L. F. G. Gutowsky, A. D. M. Wilson, A. J. Gallagher, M. R. Donaldson, N. Hammerschlag, and S. J. Cooke, "How long is too long in contemporary peer review? perspectives from authors publishing in conservation biology journals," *PLOS ONE*, vol. 10, no. 8, pp. 1–20, 08 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0132557

[5] J. Velterop, "Peer review–issues, limitations, and future development," *ScienceOpen Research*, 2015.

[6] D. J. Solomon and B.-C. Björk, "A study of open access journals using article processing charges," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 8, pp. 1485–1495, 2012. [Online]. Available: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.22673

[7] B. Aczel, B. Szászi, and A. Holcombe, "A billion-dollar donation: estimating the cost of researchers' time spent on peer review," *Research Integrity and Peer Review*, vol. 6, 11 2021.

[8] B. Aczel, B. Szaszi, and A. O. Holcombe, "A billion-dollar donation: estimating the cost of researchers' time spent on peer review," *Research Integrity and Peer Review*, vol. 6, no. 1, p. 14, Nov 2021. [Online]. Available: https://doi.org/10.1186/s41073-021-00118-2

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[10] J. P. Tennant, "The state of the art in peer review," *FEMS Microbiology letters*, vol. 365, no. 19, p. fny204, 2018.

[11] V. B. Shidham, M. B. Pitman, and R. M. DeMay, "How to write an article: Preparing a publishable manuscript!" *Cytojournal*, vol. 9, 2012.

[12] P. Koopman, "How to write an abstract," 1997.

[13] I. D. Cooper, "How to write an original research paper (and get it published)," *Journal of the Medical Library Association: JMLA*, vol. 103, no. 2, p. 67, 2015.

[14] V. B. Shidham, M. B. Pitman, and R. M. DeMay, "How to write an article: Preparing a publishable manuscript!" *Cytojournal*, vol. 9, 2012.

[15] L. M. Arrom, J. Huguet, C. Errando, A. Breda, and J. Palou, "How to write an original article," *Actas Urológicas Españolas (English Edition)*, vol. 42, no. 9, pp. 545–550, 2018.

[16] I. C. of Medical Journal Editors *et al.*, "Uniform requirements for manuscripts submitted to biomedical journals: writing and editing for biomedical publication," 2004.

[17] K. Mullane, S. Enna, J. Piette, and M. Williams, "Guidelines for manuscript submission in the peer-reviewed pharmacological literature," pp. 225–235, 2015.

[18] P. A. Ali and R. Watson, "Peer review and the publication process," *Nursing Open*, vol. 3, no. 4, pp. 193–202, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nop2.51

[19] J. Vergne, "Decentralized vs. distributed organization: Blockchain, machine learning and the future of the digital platform," *Organization Theory*, vol. 1, no. 4, 2020. [Online]. Available: https://doi.org/10.1177/2631787720977052

[20] I. Bashir, *Mastering blockchain*. Packt Publishing Ltd, 2017.

[21] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology—CRYPTO'92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings 12*. Springer, 1993, pp. 139–147.

[22] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. IEEE, 2017, pp. 557–564.

[23] H. Sheth and J. Dattani, "Overview of blockchain technology," *Asian Journal For Convergence In Technology (AJCT) ISSN -2350-1146*, Apr. 2019. [Online]. Available: https://asianssr.org/index.php/ajct/article/view/728

[24] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[25] S. S. Sarmah, "Understanding blockchain technology," *Computer Science and Engineering*, vol. 8, no. 2, pp. 23–29, 2018.

[26] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform."

[27] "Ethereum virtual machine (evm)." [Online]. Available: https://ethereum.org/en/developers/docs/evm/

[28] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[29] "Gas and fees." [Online]. Available: https://ethereum.org/en/developers/docs/gas/

[30] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117 782–117 801, 2020.

[31] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19316280

[32] P. Catchlove, "Smart contracts: a new era of contract use," *Available at SSRN 3090226*, 2017.

[33] "Introduction to dapps." [Online]. Available: https://ethereum.org/en/developers/docs/dapps/

[34] C. Gacek and B. Arief, "The many meanings of open source," *IEEE Software*, vol. 21, no. 1, pp. 34–40, 2004.

[35] U. W. Chohan, "Non-fungible tokens: Blockchains, scarcity, and value," *Critical Blockchain Research Initiative (CBRI) Working Papers*, 2021.

[36] D. Banks, "Starting science in the vernacular. notes on some early issues of the philosophical transactions and the journal des sçavans, 1665-1700," *ASp. la revue du GERAS*, no. 55, pp. 5–22, 2009.

[37] A. Keefer, "Electronic journals, scholarly communication and libraries," *BiD: textos universitaris de biblioteconomia i documentació, 2001, junio, núm. 6*, 2001.

[38] R. Smith, "Peer review: a flawed process at the heart of science and journals," *Journal of the royal society of medicine*, vol. 99, no. 4, pp. 178–182, 2006.

[39] S. Jecmen, H. Zhang, R. Liu, N. B. Shah, V. Conitzer, and F. Fang, "Mitigating manipulation in peer review via randomized reviewer assignments," in *Advances in Neural Information Processing Systems*, vol. 2020-December, 2020, cited By :11. [Online]. Available: www.scopus.com

[40] I. Stelmakh, N. Shah, and A. Singh, "Peerreview4all: Fair and accurate reviewer assignment in peer review," *Journal of Machine Learning Research*, vol. 22, 2021, cited By :5. [Online]. Available: www.scopus.com

[41] L. Liu, Z. . Tan, C. DIao, and N. Cai, "Parallel analysis on novel peer review system for academic journals," in *Proceedings of the 33rd Chinese Control and Decision Conference, CCDC 2021*, 2021, pp. 2514–2519. [Online]. Available: www.scopus.com

[42] S. Leible, S. Schlager, M. Schubotz, and B. Gipp, "A review on blockchain technology and blockchain projects fostering open science," *Frontiers in Blockchain*, p. 16, 2019.

[43] B. Gipp, C. Breitinger, N. Meuschke, and J. Beel, "Cryptsubmit: Introducing securely timestamped manuscript submission and peer review feedback using the blockchain," in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2017, pp. 1–4.

[44] M. Spearpoint, "A proposed currency system for academic peer review payments using the blockchain technology," *Publications*, vol. 5, no. 3, p. 19, 2017.

[45] M. Avital, "Peer review: Toward a blockchain-enabled market-based ecosystem," *Communications of the Association for Information Systems*, vol. 42, no. 1, p. 28, 2018.

[46] T. Wang, S. C. Liew, and S. Zhang, "Pubchain: A decentralized open-access publication platform with participants incentivized by blockchain technology," 2019. [Online]. Available: https://arxiv.org/abs/1910.00580

[47] I. Zhou, I. Makhdoom, M. Abolhasan, J. Lipman, and N. Shariati, "A blockchain-based file-sharing system for academic paper review," in *2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2019, pp. 1–10.

[48] Y. Zhou, Z. Wan, and Z. Guan, "Open-pub: A transparent yet privacy-preserving academic publication system based on blockchain," 2020. [Online]. Available: https://arxiv.org/abs/2007.03915

[49] A. Kosmarski and N. Gordiychuk, "Token-curated registry in a scholarly journal: Can blockchain support journal communities?" *Learned Publishing*, vol. 33, no. 3, pp. 333–339, 2020.

[50] J. Lawton, K. Uzdoğan, and P. Cox, "Peer review aggregation utilizing blockchain technology," in *2021 3rd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, 2021, pp. 8–11.

[51] Y. He, K. Tian, and J.-R. Fu, "An incentive mechanism-based framework to assure the quality of self-organizing peer review in preprint," *Data Technol. Appl.*, vol. 55, pp. 609–621, 2021.

[52] L. Medury and S. Ghosh, "Decentralized peer-review research solution," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2021, pp. 1–7.

[53] Ámbar Tenorio-Fornés, E. P. Tirador, A. A. Sánchez-Ruiz, and S. Hassan, "Decentralizing science: Towards an interoperable open peer review ecosystem using blockchain," *Information Processing Management*, vol. 58, no. 6, p. 102724, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457321002089

[54] B. Trovò and N. Massari, "Ants-review: A privacy-oriented protocol for incentivized open peer reviews on ethereum," in *European Conference on Parallel Processing*. Springer, 2021, pp. 18–29.

[55] A. Gazis, G. Anagnostakis, S. Kourmpetis, and E. Katsiri, "A blockchain cloud computing middleware for academic manuscript submission," *WSEAS TRANSACTIONS ON BUSINESS AND ECONOMICS*, vol. 19, pp. 562–572, feb 2022. [Online]. Available: https://doi.org/10.373942F23207.2022.19.51

[56] M. Beştaş, R. Taş, E. Akin, M. Ozkan-Okay, Aslan, and S. S. Aktug, "A novel blockchain-based scientific publishing system," *Sustainability*, vol. 15, no. 4, p. 3354, Feb 2023. [Online]. Available: http://dx.doi.org/10.3390/su15043354

[57] S. Woo, J. Song, and S. Park, "A distributed oracle using intel sgx for blockchain-based iot applications," *Sensors*, vol. 20, no. 9, p. 2725, 2020.

[58] L. Tredinnick, "Cryptocurrencies and the blockchain," *Business Information Review*, vol. 36, no. 1, pp. 39–44, 2019.

[59] OpenZeppelin, "Erc20." [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/erc20

[60] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 1.

[61] X. B. Wu, Z. Zou, and D. Song, *Learn ethereum: build your own decentralized applications with ethereum and smart contracts*. Packt Publishing Ltd, 2019.

[62] Etherscan, "Etherscan—the ethereum blockchain explorer," 2023. [Online]. Available: https://etherscan.io/

[63] D. Mohanty, "Ethereum for architects and developers," *Apress Media LLC, California*, pp. 14–15, 2018.

[64] GitHub, "Github," Website, 2023, accessed: January 10, 2023. [Online]. Available: https://github.com/

[65] Microsoft, "Visual studio code," Website, 2023, accessed: January 10, 2023. [Online]. Available: https://code.visualstudio.com/

[66] J. Itkonen, M. V. Mantyla, and C. Lassenius, "How do testers do it? an exploratory study on manual testing practices," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 494–497.

[67] P. Runeson, "A survey of unit testing practices," *IEEE Software*, vol. 23, no. 4, pp. 22–29, 2006.

[68] S. K. Singh and A. Singh, *Software testing*. Vandana Publications, 2012.