UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO

Allan Farias Fávaro

Uma Solução Baseada em Blockchain Para Revisão Por Pares de Artigos Científicos

Araranguá
2022

Allan Farias Fávaro

Uma Solução Baseada em Blockchain Para Revisão Por Pares de Artigos Científicos

Dissertação submetida ao Programa de Pós-Graduação em Tecnologias da Informação e Comunicação da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Tecnologias da Informação e Comunicação..
Orientador: Prof. Cristian Cechinel, Dr.
Coorientador: Prof. Roderval Marcelino, Dr.

Araranguá
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Fávaro, Allan Farias
Uma Solução Baseada em Blockchain Para Revisão Por Pares de Artigos Científicos / Allan Farias Fávaro ; orientador, Cristian Cechinel, coorientador, Roderval Marcelino, 2022.
99 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Campus Araranguá, Programa de Pós-Graduação em Tecnologias da Informação e Comunicação, Araranguá, 2022.

Inclui referências.

1. Tecnologias da Informação e Comunicação. 2. Tecnologias da Informação e Comunicação. 3. Blockchain. 4. Revisão por Pares. 5. Smart Contracts. I. Cechinel, Cristian . II. Marcelino, Roderval. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Tecnologias da Informação e Comunicação. IV. Título.

Allan Farias Fávaro

Uma Solução Baseada em Blockchain Para Revisão Por Pares de Artigos Científicos

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Documento assinado digitalmente
Martin Augusto Gagliotti Vigil
Data: 18/12/2022 08:55:17-0300
CPF: ***.000.789-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Martin Augusto Gagliotti Vigil, Dr.
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Antonio Carlos Sobieranski
Data: 18/12/2022 17:33:03-0300
CPF: ***.305.809-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Antonio Carlos Sobieranski, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Tecnologias da Informação e Comunicação.

Documento assinado digitalmente
Giovani Mendonca Lunardi
Data: 18/12/2022 19:06:04-0300
CPF: ***.394.559-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Giovani Mendonça Lunardi, Dr.
Coordenador do Programa

Documento assinado digitalmente
Cristian Cechinel
Data: 18/12/2022 12:51:59-0300
CPF: ***.628.449-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Cristian Cechinel, Dr.
Orientador

Araranguá, 25 de Outubro de 2022.

Este trabalho é dedicado à minha família, por todo o suporte que me deram para a conclusão do trabalho.

AGRADECIMENTOS

Agradeço à minha família, pelo apoio e suporte durante todo o tempo do curso. Aos professores Cristian Cechinel e Roderval Marcelino, orientador e coorientador, pela paciência e pelos ensinamentos durante o curso. À Universidade Federal de Santa Catarina, pelas pelas oportunidades que me foram concedidas. À todos os professores que conheci durante o curso, pelo conhecimentos e experiências compartilhados. Finalmente, agradeço à FAPESC pelo financiamento deste trabalho.

“You can,
you should,
and if you’re brave enough to start,
you will.”
(Stephen King)

RESUMO

O surgimento do *Bitcoin* em 2008 apresentou a primeira aplicação do *blockchain*, desde então sua aplicação tem se tornado popular tanto na academia quanto em variados setores da economia. Ele oferece características importantes como descentralização, transparência, integridade dos dados e pseudo-anonimato. Estas características fornecem um meio para a execução de *smart contracts*, trechos de código autônomos e auto executáveis. O *Ethereum* foi o primeiro *blockchain* a implementar um protocolo para execução de *smart contracts*, o que permitiu a criação de aplicações descentralizadas. O processo de revisão por pares é muito importante no meio acadêmico, pois pode ser aplicado à revisão de artigos científicos, tanto para revistas científicas quanto para conferências, assim como à revisão de propostas de projeto de pesquisa. Estas atividades são de grande importância para o desenvolvimento da ciência e disseminação do conhecimento. Contudo sofrem questionamentos devido à falta de transparência, lentidão, viés nas revisões e falta de incentivo à participação dos revisores e editores, que normalmente atuam de forma voluntária. Deste modo, este trabalho propõe um modelo de revisão de artigos científicos e projetos de pesquisa baseado no *Ethereum*. Ele busca resolver ou mitigar os principais problemas relacionados ao processo, apresentando um mecanismo de incentivos aos revisores e editores e buscando oferecer anonimato durante o processo, ao mesmo tempo em que permite aplicação de diferentes tipos de revisão aberta. Um protótipo foi implementado utilizando a linguagem *Solidity* e foi utilizado na execução de experimentos para cinco cenários de testes diferentes, através dos quais os resultados foram obtidos. O custo de execução nos diferentes cenários é apresentado e as limitações do modelo são discutidas, assim como futuros trabalhos são sugeridos. Apesar das limitações, fica evidente que o *blockchain* pode ser utilizado para melhorar o processo de revisão de artigos.

Palavras-chave: Revisão por Pares. Revisão de Artigos. Smart Contracts. Ethereum.

ABSTRACT

The emergence of Bitcoin in 2008 presented the first application of blockchain, since then its application has become popular both in academia and in various sectors of the economy. It offers important features like decentralization, transparency, data integrity and pseudo-anonymity. These features provide a way for executing smart contracts, standalone and self-executing code snippets. Ethereum was the first blockchain to implement a protocol for executing smart contracts, which allowed the creation of decentralized applications. The peer review process is very important in academia, as it can be applied to the review of scientific papers, both for scientific journals and conferences, as well as the review of research project proposals. These activities are of great importance for the development of science and the dissemination of knowledge. However, they are questioned due to the lack of transparency, slowness, bias in the reviews and lack of incentive to the participation of reviewers and editors, who usually work on a voluntary basis. Thus, this work proposes a model for reviewing scientific articles and research projects based on the Ethereum blockchain. It seeks to solve or mitigate the main problems related to the process, presenting an incentive mechanism for reviewers and editors and seeking to provide anonymity during the process, while allowing the application of different types of open review. A prototype was implemented using the Solidity language and was used in the execution of experiments for five different test scenarios, through which the results were obtained. The cost of execution in the different scenarios is presented and the limitations of the model are discussed, as well as future works are suggested. Despite the limitations, it is evident that blockchain can be used to improve the paper review process.

Keywords: Peer Review. Paper Review. Smart Contracts. Ethereum.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E PROBLEMATIZAÇÃO

O *blockchain* foi a primeira implementação de uma tecnologia de registros distribuídos, DLT (do inglês *Distributed Ledger Tecnologies*), introduzido por Nakamoto (2008) como base para o *Bitcoin*, a principal criptomoeda da atualidade. Por conta disso, muitas vezes os termos DLT e *blockchain* são confundidos (BASHIR, 2018).

Ele é composto por uma cadeia de blocos de dados, armazenados de maneira distribuída pelos dispositivos participantes de uma rede *peer-to-peer*, P2P. Toda alteração feita no conjunto de dados será refletida em todos os participantes. Porém para que as alterações sejam aceitas elas devem ser aprovadas por algoritmos de consenso, onde a maioria dos dispositivos deve validá-las (BASHIR, 2018; ROSA, 2019).

Normalmente as criptomoedas utilizam redes P2P públicas, ou seja, qualquer pessoa pode conectar dispositivos e fazer parte da rede. Esta abordagem contribui para a segurança da rede como um todo, pois para controlar o algoritmo de consenso é necessário controlar a maioria do poder computacional disponível na rede. Contudo, a execução dos algoritmos de consenso demanda grande quantidade de recursos computacionais e energia (NAKAMOTO, 2008; SCHUEFFEL, 2018).

Com o tempo algumas alternativas que dispensam criptomoedas foram desenvolvidas através de redes privadas, como o projeto *Hyperledger*[1]. Neste caso o custo computacional das transações é baixo quando comparado com as redes públicas, pois os algoritmos de consenso são executados por dispositivos confiáveis mantidos pelos responsáveis pelo projeto.

Estes aspectos do *blockchain* eliminam a necessidade de um órgão regulador, como os necessários em aplicações financeiras, o que viabilizou a implementação do *smart contract*. Sua ideia foi teorizada na década de 1990 por Szabo (1997), que o definiu como um protocolo de transações eletrônicas capaz de executar os termos de um contrato. Sua aplicação é capaz de reduzir custos e o tempo necessário para o cumprimento dos termos de um contrato. A principal alternativa para execução de *smart contracts* atualmente é o *Ethereum*, protocolo desenvolvido para a execução de *smart contracts* em uma rede *blockchain* de maneira segura e autônoma (WOOD, 2014).

O *blockchain* permite o armazenamento de informações de maneira segura e confiável. Por isso, vem tendo cada vez mais aplicações estudadas e desenvolvidas por pesquisadores e empresas das mais variadas áreas. Assim, algumas soluções desenvolvidas utilizando *blockchain* vão desde aplicações do mercado financeiro, seguros, armazenamento distribuído, Internet das Coisas (*IoT*, do inglês *Internet of Things*), emissão e armazenamento seguro de diplomas, como o apresentado por Palma *et al.*

[1] Disponível em: https://www.hyperledger.org/

(2019), e soluções propostas para o processo de revisão por pares de artigos científicos (CROSBY *et al.*, 2016; ROSSUM, 2017).

O processo de revisão por pares pode ser aplicado a diferentes atividades acadêmicas como, publicação de monografias, geração de diplomas, revisão de artigos para publicação em revistas científicas e conferências, assim como a avaliação de propostas de projeto de pesquisa para financiamento. Suas principais funções quando aplicado à publicação de artigos em revistas e conferências incluem a disseminação do conhecimento mais atual, o arquivamento do conhecimento canônico, o controle de qualidade das informações publicadas e a atribuição de crédito aos autores pelo trabalho desenvolvido em suas pesquisas (ROWLAND, 2002). Quando aplicado à avaliação de projetos de pesquisa, a revisão por pares serve de base para a escolha de quais pesquisas receberão financiamento, influenciando diretamente o desenvolvimento do conhecimento.

Desta maneira, ele pode ser considerado o principal meio para controle de qualidade da maioria do conhecimento científico e para sua evolução. Ele promove a autorregulação do conhecimento através da seleção crítica, resultando assim na melhoria da qualidade das pesquisas científicas. Através da avaliação da qualidade das pesquisas, ele auxilia a decidir quais devem receber financiamento e determina quais serão publicadas. Desse modo, ele deve ser imparcial para garantir a idoneidade do conhecimento. Um dos meios utilizados atualmente para evitar avaliações tendenciosas é conhecido como *double blind peer review* (revisão por pares duplo-cega), onde os autores não conhecem a identidade dos revisores e *vice-versa* (BORNMANN, 2011; ROWLAND, 2002).

Neste contexto, este trabalho apresentará um modelo de revisão baseado em *blockchain* e *smart contracts*, buscando aumentar sua transparência, manter o anonimato dos participantes, quando necessário, e oferecer incentivos aos revisores e editores. Buscando através do desenvolvimento do trabalho responder as seguintes perguntas de pesquisa:

- P1: Como *blockchain* e *smart contracts* podem contribuir para a transparência do processo de revisão por pares?
- P2: Como o *blockchain* pode contribuir para o oferecimento de incentivos aos revisores e editores?
- P3: Como o *blockchain* pode garantir a propriedade intelectual dos artigos?

1.2 JUSTIFICATIVA

O processo de revisão por pares é utilizado na revisão de artigos científicos desde o início dos periódicos acadêmicos no século 17. Inicialmente somente os

revisores conheciam a identidade dos autores, o que acabou gerando problemas, pois aconteciam manipulações na avaliação com muita facilidade e frequência. Com o tempo outros modelos foram adotados na tentativa de evitá-los, como a revisão aberta, onde as identidades de autores e revisores são conhecidas por ambos e a revisão *double blind*, citada anteriormente (ROWLAND, 2002).

Contudo, mesmo considerando o advento de novos padrões, ele ainda apresenta problemas. Mesmo no modelo *double blind* pode ser possível que os revisores conheçam a identidade dos autores. Este fato muitas vezes torna a análise dos revisores enviesada, influenciada por atributos pessoais dos autores ou dos próprios revisores, e não baseada somente em critérios científicos, prejudicando a justiça do processo. Ele sofre também com falta de transparência que acarreta em falta de confiabilidade, podem ocorrer fraudes por parte dos autores e até mesmo apropriação de ideias por parte de revisores ou editores (ROSSUM, 2017; DAS, 2016).

Outro problema importante é a demora para execução da revisão, que demanda muito trabalho dos participantes, atrasando as publicações e o desenvolvimento de novas ideias inovadoras e disruptivas (BORNMANN, 2011; DAS, 2016).

Com o avanço das tecnologias *blockchain* nos últimos anos, elas passaram a ser aplicadas nas mais diversas áreas que exigem segurança, confiabilidade, transparência e imutabilidade nas operações.

Desse modo, este trabalho visa sugerir uma solução baseada na tecnologia *blockchain* e *smart contracts* para contribuir com a transparência do processo de revisão por pares de artigos científicos, assim como uma maneira segura de garantir a propriedade intelectual sobre os artigos aos autores. A transparência do processo e o anonimato sobre a identidade das partes envolvidas são questões fundamentais para sua adoção por parte de periódicos acadêmicos. Assim como o oferecimento de incentivos aos revisores e editores em conjunto com a garantia da propriedade intelectual são questões relevantes para os participantes dos processo.

1.3 OBJETIVOS

Esta seção descreve os objetivos, geral e específicos, que busca-se alcançar durante o desenvolvimento deste trabalho.

1.3.1 Objetivo Geral

Criar um modelo de revisão por pares de artigos científicos e propostas de projeto de pesquisa baseado em *blockchain* e *smart contracts*.

1.3.2 Objetivos Específicos

- Estudar os fundamentos sobre a tecnologia *blockchain*.

- Estudar os fundamentos sobre *smart contracts*.
- Identificar os principais problemas relacionados ao processo de revisão por pares de artigos científicos.
- Com base nas informações encontradas, propor um modelo de revisão por pares baseado em *blockchain*.
- Modelar e desenvolver um protótipo do sistema.
- Testar o funcionamento da rede *blockchain* e dos *smart contracts* aplicados à revisão por pares de artigos científicos.

1.4 INTERDISCIPLINARIDADE E ADERÊNCIA AO PPGTIC

O Programa de Pós-Graduação em Tecnologias da Informação e Comunicação (PPGTIC) é um programa interdisciplinar que abriga três linhas de pesquisa: Tecnologia, Gestão e Inovação; Tecnologia Educacional e Tecnologia Computacional. Este trabalho está centrado na linha de pesquisa de Tecnologia Computacional, cujo principal objetivo, segundo "é desenvolver modelos, técnicas e ferramentas computacionais auxiliando na resolução de problemas de natureza interdisciplinar" (PPGTIC, 2020).

A tecnologia escolhida para o desenvolvimento deste trabalho já foi utilizada anteriormente no programa por Rosa (2019), aplicada à logística reversa. A tecnologia *blockchain* está em ascensão nos últimos anos, podendo ser utilizada na solução de diversos problemas, assim este trabalho visa aplicá-la ao processo de revisão por pares de artigos científicos, problema este relacionado à área educacional.

Portanto, pode-se perceber que tanto a tecnologia utilizada, quanto a área na qual será aplicada se enquadram na linha de pesquisa de Tecnologia Computacional. O trabalho caracteriza-se pela sua interdisciplinaridade ao utilizar tecnologias inovadoras na solução de problemas relacionados à educação.

1.5 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado da seguinte maneira. O capítulo 2 apresenta o estado da arte sobre a aplicação do *blockchain* ao processo de revisão por pares de artigos científicos, através de uma revisão sistemática de literatura. A fundamentação teórica sobre a tecnologia *blockchain* e *smart contracts*, assim como o processo de revisão por pares são apresentados no capítulo 3. Os procedimentos metodológicos adotados para a realização do trabalho, assim como o próprio modelo proposto e o seu protótipo são apresentados no capítulo 4. O capítulo 5 apresenta os resultados obtidos através da execução dos experimentos, assim como uma discussão sobre o modelo e

suas limitações. O capítulo 6 apresenta as conclusões obtidas através dos resultados, assim como propostas de trabalhos futuros buscando solucionar suas limitações.

2 ESTADO DA ARTE

Este capítulo apresenta a revisão sistemática de literatura (RSL) sobre o tema estudado, conduzida para auxiliar o desenvolvimento da pesquisa. São apresentados os estudos encontrados aplicando *blockchain* e *smart contracts* ao processo de revisão por pares de artigos científicos. Estes estudos servem de base para o desenvolvimento deste trabalho.

2.1 METODOLOGIA

A Revisão Sistemática de Literatura (RSL) foi conduzida utilizando a metodologia proposta por Kitchenham (2004), seguindo três etapas: (1) Planejamento da revisão, (2) Execução da revisão e (3) Relato dos resultados. Elas são descritas em detalhes a seguir.

2.1.1 Planejamento da Revisão

Nesta etapa foi definida a *string* de busca utilizada para encontrar os artigos nas bases de dados:

(“peer-review” OR “paper review”) AND (“blockchain” OR “smart contract” OR “ethereum”)

A *string* utilizada leva em consideração os termos “*peer-review*” e “*paper review*”, conectados pelo operador “OR”, pois em uma busca exploratória inicial ambos foram encontrados referindo-se ao processo de revisão por pares de artigos científicos. Os termos “*blockchain*”, “*smart contract*” e “*ehtereum*” foram conectados pelo operador lógico “OR” pois são as tecnologias de interesse para a RSL. Finalmente, as duas expressões foram conectadas utilizando o operador lógico “AND”, pois buscam-se artigos que tratem da aplicação de *blockchain* e *smart contracts* ao processo de revisão de artigos científicos.

As bases de dados selecionadas foram *Scopus*, *IEEEXplore* e *Web of Science*. A revista *Frontiers in Blockchain* foi incluída na RSL, pois é reconhecida pela publicação de pesquisas aplicadas sobre *blockchain* e não foram encontrados artigos publicados por ela nas bases pesquisadas. A busca foi realizada no dia 31 de Maio de 2021. No caso da revista *Frontiers in Blockchain* foram analisados todos os artigos publicados até a data da busca.

2.1.2 Execução da Revisão

A seleção dos estudos foi realizada em quatro passos:

1. Execução da *string* de busca nas bases;

2. Leitura do título e resumo para seleção dos estudos candidatos, quando necessário foi efetuada a leitura do texto completo;

3. Aplicação dos critérios de inclusão e exclusão (quadro 1);

4. Seleção dos estudos.

Quadro 1 – Critérios de inclusão e exclusão. Fonte: Elaborado pelo autor.

Inclusão	**Exclusão**
Estudos que aplicam *blockchain* e *smart contracts* ao processo de revisão por pares Artigos publicados em revistas e conferências *Fulltext* disponível	Estudos que não são aplicados Artigos duplicados

Os passos 2, 3 e 4 foram executadas simultaneamente em cada rodada de seleção dos artigos. A busca resultou em 1376 trabalhos nas três bases de dados, mais todos os 120 artigos da revista *Frontiers in Blockchain* verificados manualmente. Através do título e da leitura do resumo foram excluídos os que não satisfaziam os critérios de inclusão (quadro 1). Nesta etapa foram selecionados 38 artigos para a leitura de todo o texto, dos quais 16 foram excluídos. Entre os 22 artigos restantes foram excluídos os duplicados, resultando em 17 selecionados para o desenvolvimento da pesquisa, como mostra a tabela 1. Os resultados são apresentados e discutidos nas próximas seções.

Tabela 1 – Quantidade total de artigos obtidos durante a RSL. Fonte: Elaborado pelo autor.

Base de Dados	**Número de Resultados**
IEEEXplore	349
Scopus	333
Web of Science	694
Total (antes da exclusão)	**1376**
Total (depois da exclusão)	**20**
Frontiers in Blockchain (buscado manualmente)	120
Frontiers in Blockchain (depois da exclusão)	2
Total (incluindo duplicados)	**22**
Total (excluindo duplicados)	**17**

2.2 RESULTADOS

Todos os 17 artigos selecionados para o estudo foram publicados a partir de 2017, o que mostra que esta é uma área de pesquisa muito recente. Do mesmo modo, todos são escritos em Inglês.

A figura 1 apresenta a quantidade de artigos por ano de publicação, divididos por tipo: revistas e conferências. Percebe-se que a quantidade dos artigos publicados apresenta uma tendência de aumento. Esta tendência não se mantém em 2020, onde a pandemia de covid-19 pode ter exercido influência sobre a quantidade de publicações na área.

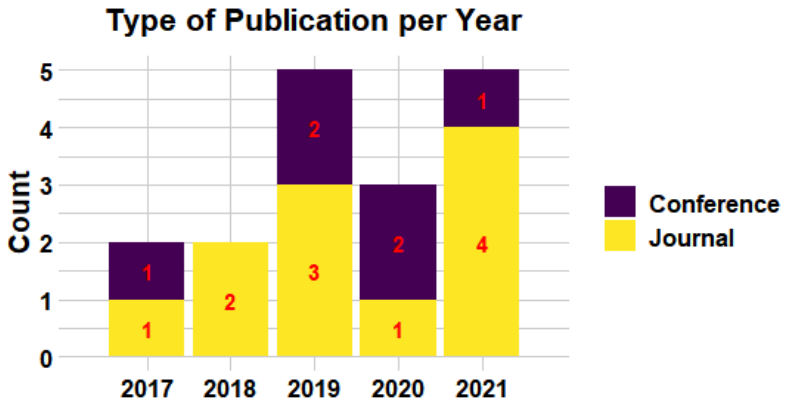

Figura 1 – Tipos de artigos publicados por ano. Fonte: Elaborado pelo autor.

A figura 2 apresenta a distribuição dos artigos publicados por países. Estados Unidos, França, Coreia do Sul e China possuem dois artigos publicados cada, enquanto Brasil, Reino Unido, Dinamarca, Alemanha, Espanha, Eslovênia, Rússia, Austrália e Nova Zelândia possuem somente um artigo publicado cada.

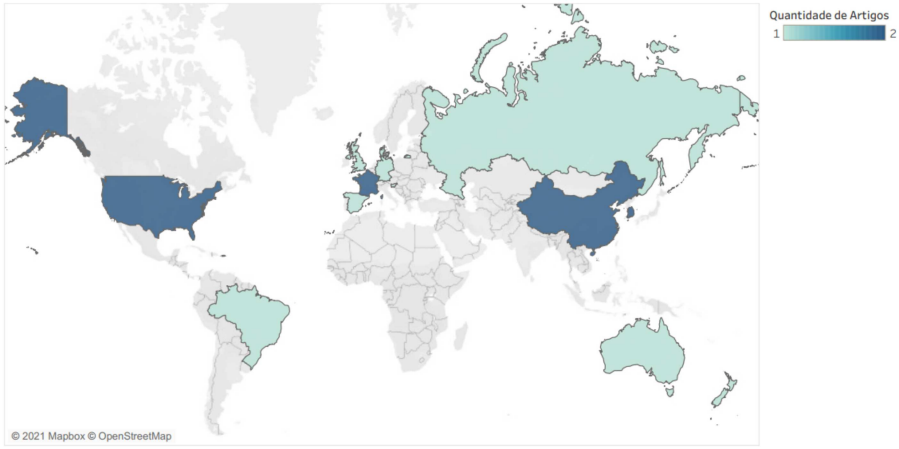

Figura 2 – Quantidade de artigos publicados por países. Fonte: Elaborado pelo autor.

2.2.1 Categorização

Para facilitar uma análise mais profunda foram selecionadas dimensões de interesse com base nas características encontradas nos estudos selecionados. Com base nessas dimensões os artigos foram classificados e analisados, para auxiliar a compreensão desta área de pesquisa. As dimensões selecionadas são descritas detalhadamente a seguir.

2.2.1.1 Business Model

Esta dimensão diz respeito ao modelo de negócios adotados pelos sistemas propostos nos estudos. Durante a leitura e análise dos artigos foram identificados cinco tipos de modelo de negócios, sendo eles:

- *Publishing Ecosystem*: um ecossistema de publicações, referindo-se à sistemas que suportam mais do que uma revista científica ou conferência pertencentes a diferentes editoras. Estes sistemas permitem que as editoras compartilhem um sistema integrado que possibilita a troca de informações entre elas.
- *Publishing System*: um sistema de publicações. Diferente dos ecossistemas, eles suportam revistas ou conferências de somente uma editora.
- *Peer-Review Payment Ecosystem*: um ecossistema voltado ao pagamento dos participantes do processo de revisão por pares, principalmente os revisores, como forma de incentivá-los a participar do processo.
- *Timestamp Proof*: o principal objetivo deste modelo de negócios é oferecer uma prova de existência ao manuscrito a ser submetido para revisão. Ela busca reduzir a possibilidade de fraudes e roubo de ideias, pois consegue atestar de maneira confiável em que momento o manuscrito foi submetido. Para tal, utiliza-se a imutabilidade dos dados no *blockchain*, em conjunto com o *timestamp* de cada transação, que indica o momento em que ela foi criada.
- *Preprint Publishing Ecosystem*: semelhante ao *Publishing Ecosystem*, porém voltado para a disponibilização dos artigos antes da revisão por pares (*preprint*).

Observando a figura 3 percebe-se que o modelo de negócios mais adotado é o *Publishing Ecosystem*, seguido por *Publishing System*, evidenciando que a principal preocupação dos trabalhos é melhorar o processo de revisão e a publicação científica. Logo após aparece o modelo *Peer-Review Payment Ecosystem*, focado na realização de pagamentos, principalmente aos revisores. Os outros modelos, *Timestamp Proof* e *Preprint Publishing Ecosystem*, são pouco pesquisados.

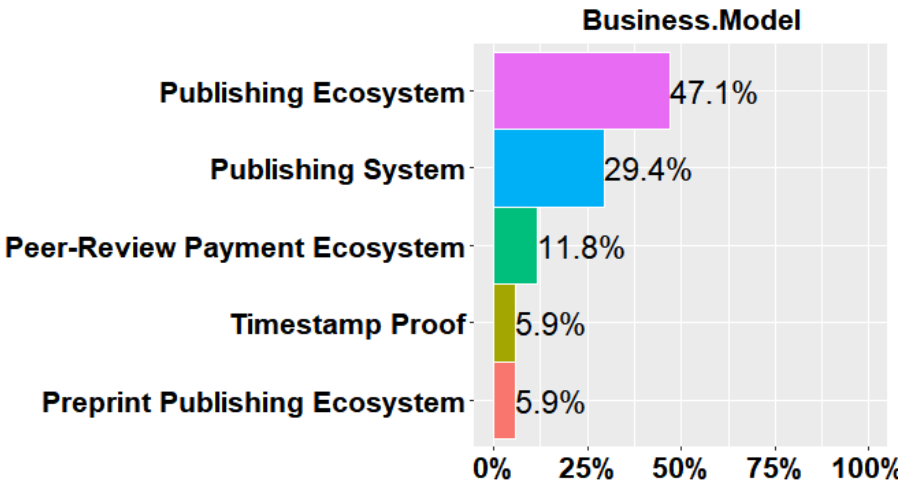

Figura 3 – Modelos de negócios. Fonte: Elaborado pelo autor.

2.2.1.2 Open Access

A dimensão *Open Access* avalia os sistemas de acordo com a definição de *Open Access Journals* apresentada por Bailey Jr. (2005). Os artigos devem ser disponibilizados na plataforma de maneira gratuita e possibilitar o acesso à qualquer pessoa, ao mesmo tempo em que permite que os autores mantenham a propriedade intelectual sobre os artigos. Neste caso, os custos de publicação normalmente são pagos pelos autores. Com base leitura dos artigos, eles foram classificados da seguinte maneira:

- *Yes*: os sistemas disponibilizam os artigos publicados como *Open Access*.
- *Optional*: esta característica é opcional, podendo ser utilizada ou não pela editora.
- *Not Mentioned*: os estudos não mencionam explicitamente ou não deixam claro na descrição do sistema se oferecem esta possibilidade.
- *Not Applicable*: refere-se aos estudos que propõem sistemas de pagamento, não tratando da publicação de fato dos artigos.

A maioria dos trabalhos não menciona se utilizam *Open Acces* (figura 4), contudo percebe-se que uma quantidade significativa adota esta estratégia, ou pelo menos oferece essa possibilidade. O que mostra que há uma preocupação com a democratização do conhecimento científico.

2.2.1.3 Review Type

A dimensão *Review Type* avalia os tipos de revisão utilizados pelos sistemas de acordo com as definições apresentadas na seção 4.2.6. Elas são classificadas da seguinte maneira:

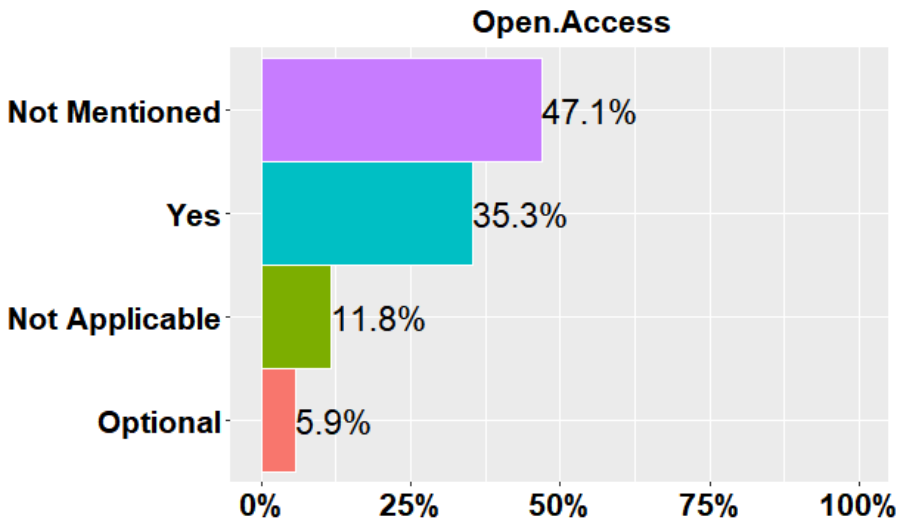

Figura 4 – Open Access. Fonte: Elaborado pelo autor.

- *Open*: quando o processo de revisão contém pelo menos uma das características de revisão aberta apresentadas na seção 4.2.6.
- *Single-Blind*: os autores não conhecem a identidade dos revisores, mas os revisores conhecem a identidade dos autores.
- *Double-Blind*: autores e revisores não conhecem a identidade uns dos outros.
- *Not Applicable*: refere-se aos trabalhos com modelo de negócios *Peer-Review Payment Ecosystem* e *Timestamp Proof*, pois não implementam a revisão por pares.
- *Not Mentioned*: trabalhos que não mencionam explicitamente e não deixam claro no texto qual o modelo de revisão utilizado.

Analisando os resultados na figura 5 percebe-se que a revisão aberta é a mais adotada, mostrando uma preocupação com a transparência do processo de revisão. Por outro lado a revisão *Double-Blind* também é oferecida com frequência. Vale ressaltar que os trabalhos de Mackey *et al.* (2019) e Coelho e Brandão (2019) oferecem os três tipos de revisão, portanto estão contabilizados nas três estatísticas. Assim como Choi e Seo (2021) oferece as revisões *Open* e *Double-Blind*, portanto é contabilizado nas duas estatísticas.

2.2.1.4 Reviewers Incentive

Esta dimensão analisa quais os tipos de incentivos que são oferecidos aos revisores. Foram identificadas quatro tipos de incentivos oferecidos (figura 6). A maioria dos trabalhos utiliza *tokens* como uma forma de incentivar os revisores. Outra forma

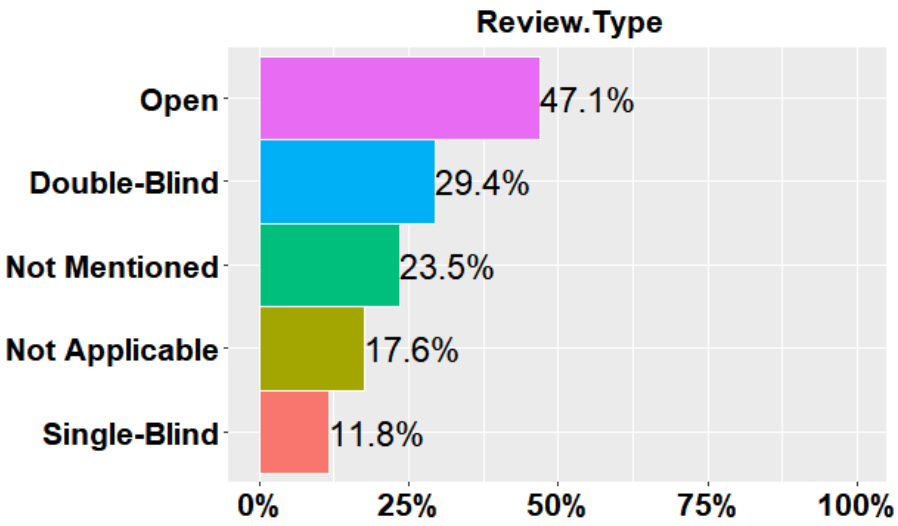

Figura 5 – Review Type. Fonte: Elaborado pelo autor.

bastante utilizada é a atribuição de reputação aos revisores (*Reputation*), seja somente dentro da plataforma, influenciando na quantidade de *tokens* recebidos, ou através dos relatórios de revisão, permitindo que eles sejam referenciados.

Nos trabalhos propostos por Trovò e Massari (2021), Duh *et al.* (2019), He, Tian e Fu (2021), Coelho e Brandão (2019) e Kosmarski e Gordiychuk (2020) a reputação está associada aos *tokens* como forma de incentivo, portanto eles estão contabilizados nas duas estatísticas. De maneira semelhante em Khan e Shahaab (2021) a reputação está associada ao compartilhamento da autoria do artigo (*Authorship*), assim ele está contabilizado nas duas estatísticas.

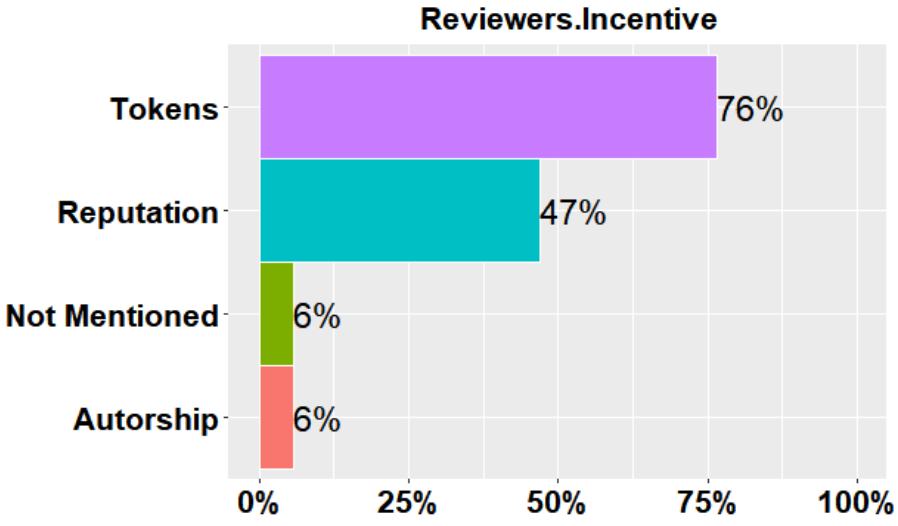

Figura 6 – Reviewers Incentive. Fonte: Elaborado pelo autor.

2.2.1.5 Token Economy

Token Economy refere-se à natureza dos *tokens* utilizados nos sistemas. Segundo Yoo (2020) os *tokens* podem ser classificados de duas maneiras: *token* IOU (*I Owe You*) ou moeda nativa. Os *tokens* IOU possuem uma natureza centralizada, pois necessitam que o fiador (quem fornece os *tokens*) implemente a troca de *tokens*. A moeda nativa é o cerne da *Token Economy*, pois pode ser garantida sem a necessidade de um fiador, de acordo com as regras do *blockchain*. As classificações definidas a seguir baseiam-se nestas duas definições:

- *Monetary*: os *tokens* são de natureza moeda nativa ou IOU com implementação da troca de *tokens*.
- *Non-monetary*: os *tokens* são de natureza IOU, porém sem a implentação da troca de *tokens*. Portanto, podem ser utilizados somente dentro da plataforma, não possuem valor real fora dela.
- *Not Mentioned*: quando não é mencionado explicitamente ou não fica claro na descrição do sistema qual método foi adotado.
- *Not Applicable*: refere-se aos sistemas que não utilizam *tokens*.

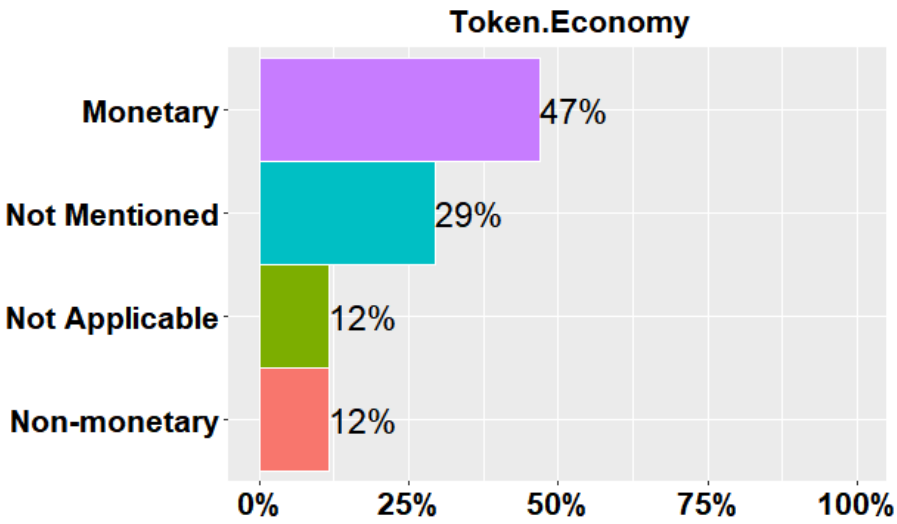

Figura 7 – Token Economy. Fonte: Elaborado pelo autor.

Analisando a figura 7 percebe-se que existe uma grande preocupação entre os estudos em oferecer incentivos através de *tokens* aos revisores, sejam eles *Monetary*, principalmente, ou *Non-Monetary*. Isto mostra que existe uma certa insatisfação com o fato de o trabalho de revisão de artigos normalmente ser voluntário, buscando ao menos oferecer aos revisores alguma redução nos custos de publicação, no caso *Non-monetary*.

2.2.1.6 Blockchain Access

A dimensão *Blockchain Access* avalia se existe controle de acesso aos *blockchains* utilizados nos trabalhos. Segundo Oliveira *et al.* (2019) as redes públicas não possuem nenhum tipo de controle de acesso, ou seja, não há restrições sobre quem pode conectar-se à rede. Enquanto as redes privadas possuem algum mecanismo para controle de acesso, restringindo a entrada de participantes, isso ocorre pois elas são controladas por alguma instituição ou um conjunto de instituições. As classificações utilizadas baseiam-se nestes conceitos:

- *Public*: utiliza *blockchains* públicos.
- *Private*: utiliza *blockchains* privados.
- *Not Mentioned*: não são mencionados explicitamente se os *blockchains* utilizados são públicos ou privados.

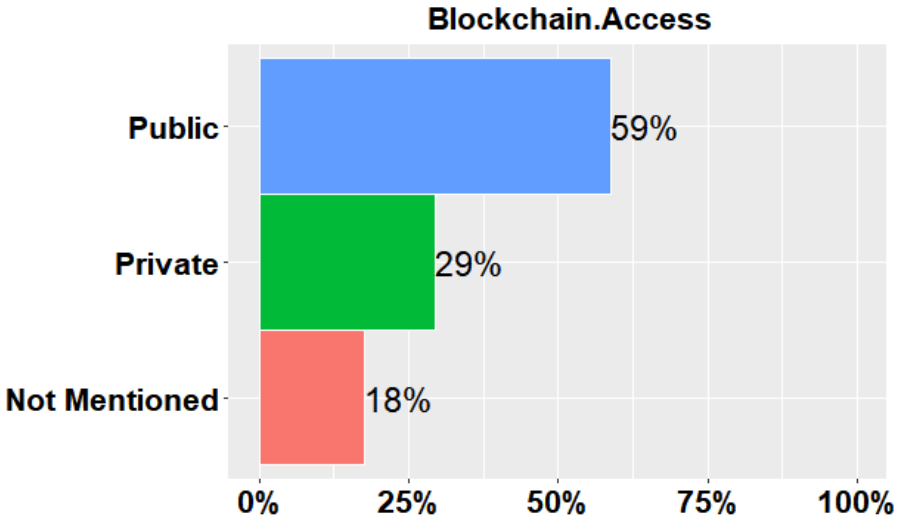

Figura 8 – Blockchain Access. Fonte: Elaborado pelo autor.

A maioria dos sistemas utiliza *blockchains* públicos (figura 8), enquanto poucos modelos propostos não mencionam se utilizam públicos ou privados. Cabe salientar que neste caso o trabalho proposto por Mackey *et al.* (2019) utiliza dois tipos de *blockchain*, sendo um público e outro privado, portanto ele é contabilizado nas duas estatísticas.

2.2.1.7 Blockchain Identification

Esta dimensão analisa se existe identificação dos participantes dos *blockchains* utilizados nos trabalhos. Existem duas classificações quanto à identificação dos participantes de uma rede *blockchain*: permissionada, onde os participantes são identificados

e seus papéis na rede podem ser diferenciados, e não permissionada onde não há identificação dos participantes e tampouco diferenciação dos seus respectivos papéis (XU *et al.*, 2017; OLIVEIRA *et al.*, 2019). As classificações utilizadas baseiam-se nestas definições:

- *Permissioned*: utiliza *blockchains* permissionados.
- *Permissionless*: utiliza *blockchains* não permissionados.
- *Not Mentioned*: não são mencionados explicitamente se os *blockchains* utilizados são permissionados ou não permissionados.

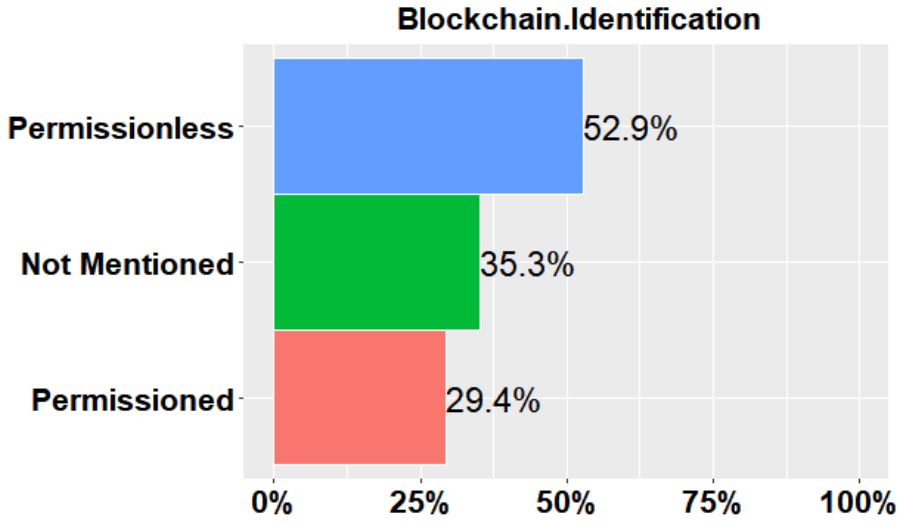

Figura 9 – Blockchain Identification. Fonte: Elaborado pelo autor.

Percebe-se pela figura 9, que existe uma preferência entre os autores em utilizar *blockchains* não permissionados, enquanto os permissionados são minoria. Vale ressaltar que, os trabalhos propostos por Avital (2018) e He, Tian e Fu (2021) são compatíveis tanto com *blockchains* permissionados quanto com não permissionados, enquanto Mackey *et al.* (2019) utiliza os dois tipos em etapas diferentes de sua solução, assim, estes trabalhos são contabilizados nas duas estatísticas.

2.2.1.8 Blockchain Used

Aqui são avaliados quais *blockchains* são ou planejam ser utilizados nos trabalhos (figura 10). O *blockchain* mais utilizado é o *Ethereum*, seguido pelo *Hyperledger*. O *Bitcoin*, *Corda* e *Pubchain* são os menos utilizados, com o último sendo proposto na solução de Wang, Chang Liew e Zhang (2020) enquanto utiliza o *Ethereum* no desenvolvimento do protótipo do sistema. Portanto, ele é contabilizado nas duas estatísticas, assim como Khan e Shahaab (2021) que propõe um sistema compatível com *Hyperledger* e *Corda*.

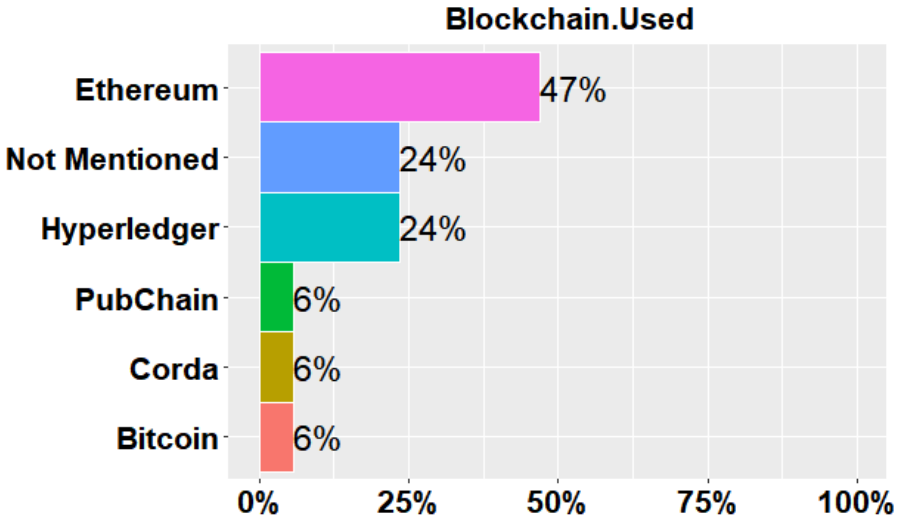

Figura 10 – Blockchain Used. Fonte: Elaborado pelo autor.

2.2.1.9 Paper Storage

Aqui é analisada a forma de armazenamento dos artigos nos trabalhos (figura 11). Percebe-se uma preferência pelo sistema de arquivos distribuído *IPFS* (*InterPlanetary File System*), seguido pelo uso de bancos de dados. Enquanto alguns trabalhos não mencionam qual a forma de armazenamento, existem também os trabalhos que não lidam com o armazenamento de artigos. Outros meios, como *Preprint Servers*, *Self-archiving* e armazenamento *On-chain* são menos utilizados. Vale destacar que, o trabalho de Mackey *et al.* (2019) oferece a possibilidade de utilizar diferentes meios, como IPFS, *Preprint Servers* e *Self-archiving*, portanto ele está contabilizado nestas três estatísticas.

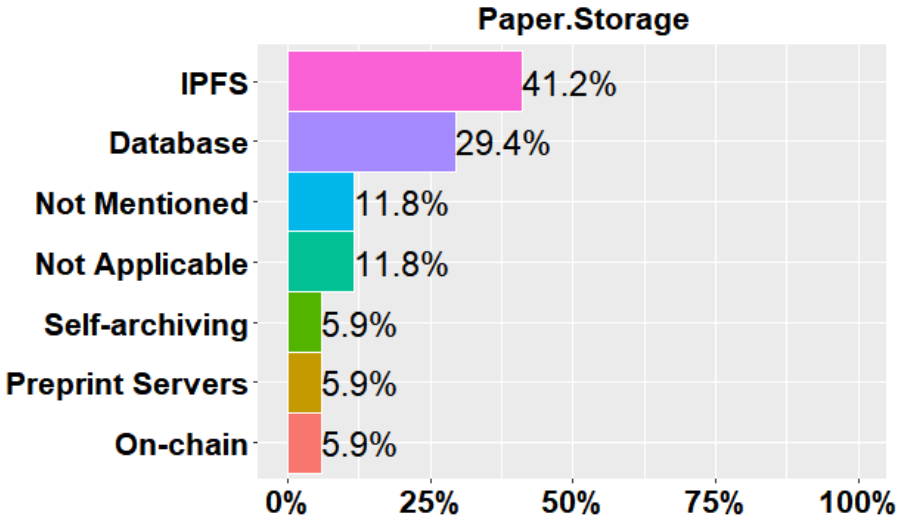

Figura 11 – Papers Storage. Fonte: Elaborado pelo autor.

2.2.1.10 Maturity

Esta dimensão analisa a maturidade dos trabalhos encontrados, elas são explicadas a seguir:

- *Prototype*: quando há algum protótipo do sistema em desenvolvimento e testes.
- *Model*: quando é proposto um modelo bem estruturado do sistema, contudo não é mencionado o desenvolvimento de um protótipo.
- *Framework*: semelhante ao *Model*, porém o termo *framework* é utilizado explicitamente pelos autores.
- *Service*: o sistema está em funcionamento.
- *Concept*: quando a ideia do sistema ainda não está bem estruturada, como no *Model*.

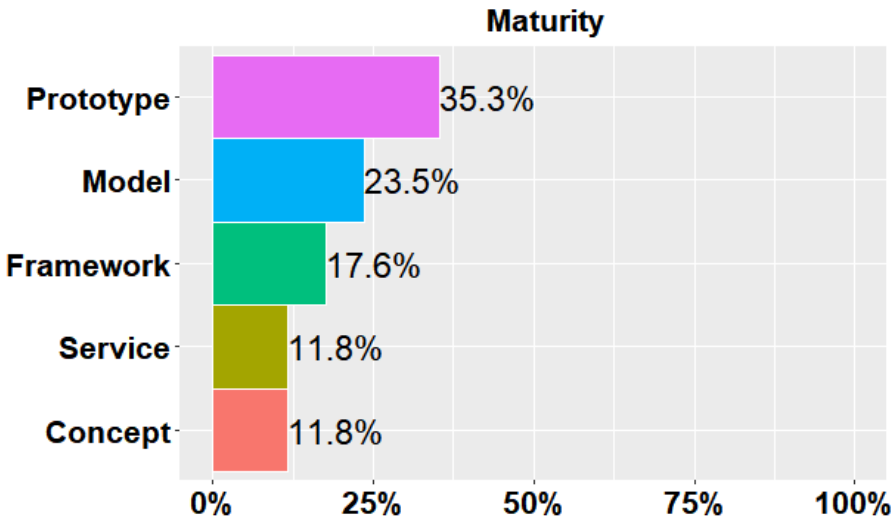

Figura 12 – Maturity. Fonte: Elaborado pelo autor.

A maturidade (figura 12) que mais foi alcançada pelos artigos é *Prototype*, seguida pelo *Model* e *Framework*. Existem também, trabalhos em estágio inicial, com maturidade *Concept*, assim como os trabalhos de Duh *et al.* (2019) e Gipp *et al.* (2017) já disponíveis como serviço.

2.2.2 Relação Entre Dimensões

A interseção entre dimensões ajuda a fornecer uma melhor compreensão sobre as particularidades dos artigos. A figura 13 apresenta as relações entre as dimensões *Blockchain Access* e *Blockchain Identification*. Percebe-se uma preferência por

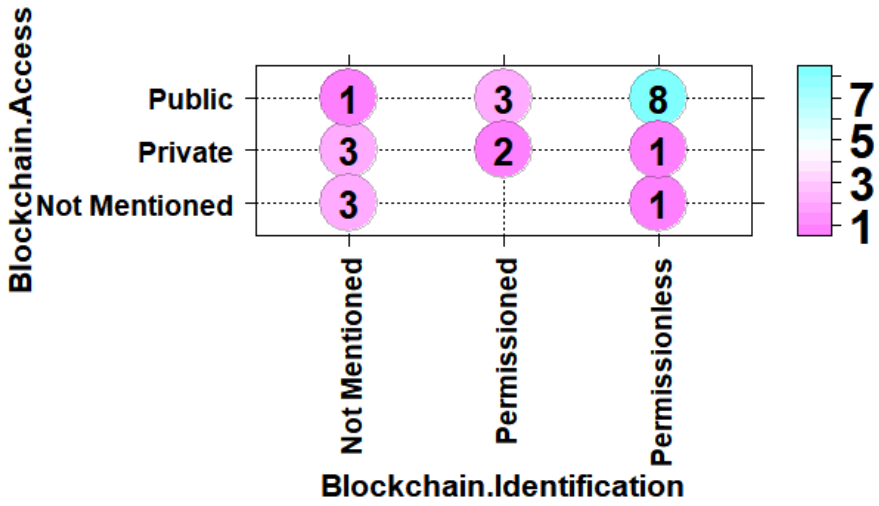

Figura 13 – Relações entre as dimensões *Blockchain Access* e *Blockchain Identification*. Fonte: Elaborado pelo autor.

blockchains públicos e não permissionados talvez por suas características capazes de oferecer maior transparência e confiabilidade aos dados Xu *et al.* (2017).

A figura 14 apresenta as relações entre as dimesões *Blockchain Access* e *Review Type*. Pela figura percebe-se que a revisão aberta é adotada com a mesma frequência em *blockchains* públicos (4) e privados (4). Enquanto a revisão *double blind* é aplicada com maior frequência a *blockchains* privados (3) do que a públicos (2). É evidente também a quantidade de trabalhos que utilizam *blockchains* públicos e não mencionam qual revisão foi adotada (3). Vale destacar que Mackey *et al.* (2019) utilizam um *blockchain* público e um privado em sua proposta, porém o processo de revisão em si é executado no *blockchain* privado. Portanto apesar do sistema utilizar os dois tipos, neste caso o trabalho está contabilizado somente na dimensão *Private*.

A figura 15 apresenta as relações entre as dimensões *Blockchain Used* e *Token Economy*. É possível perceber que há uma quantidade significativa de trabalhos com a *Token Economy Monetary* que não mencionam qual *blockchain* é utilizado (4), assim como trabalhos que utilizam o *Ethereum* e não mencionam qual o tipo de *Token Economy*. Em contrapartida três trabalhos utilizam o *Ethereum* e sua *Token Economy* é *Monetary*. Fica evidente uma concentração de casos nos eixos *Ethereum* e *Monetary*, o que mostra uma preferência por este *blockchain* no desenvolvimento de soluções e uma preocupação em oferecer algum tipo de incentivo monetário aos revisores.

A figura 16 apresenta as relações entre as dimensões *Business Model* e *Token Economy*. Percebe-se que para o modelo de negócios *Publishing Ecosystem* a *Token Economy* mais utilizada (4) é a *Monetary*, Assim como uma concentração de trabalhos nos eixos *Monetary*, *Publishing System* e *Publishing Ecosystem*.

Enquanto a figura 17 mostra as relações entre *Reviewers Incentive* e *Token*

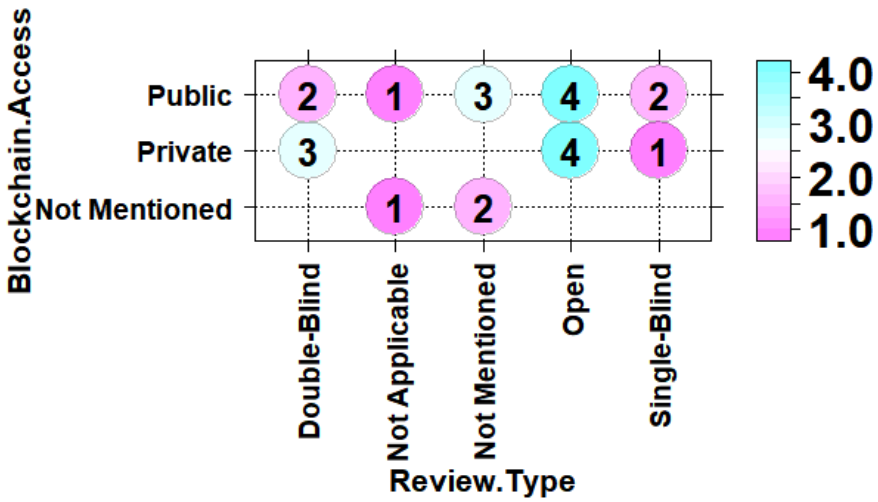

Figura 14 – Relações entre as dimensões *Blockchain Access* e *Review Type*. Fonte: Elaborado pelo autor.

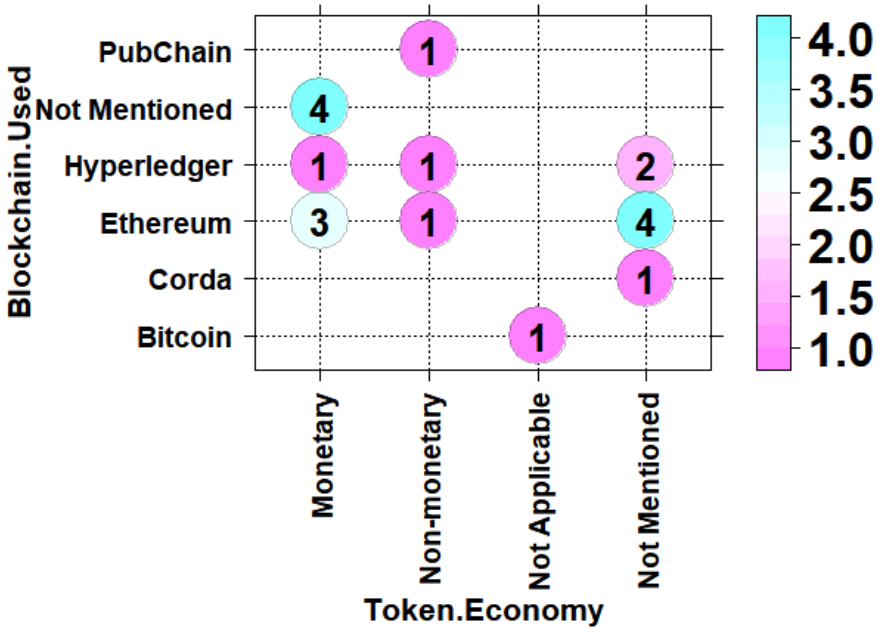

Figura 15 – Relações entre as dimensões *Blockchain Used* e *Token Enconomy*. Fonte: Elaborado pelo autor.

Economy. Percebe-se pela figura que a maioria dos trabalhos que utiliza *tokens* como incentivo aos revisores adota uma *Token Economy* do tipo *Monetary* (8), mostrando que eles são utilizados para permitir algum retorno financeiro aos revisores. Vale ressaltar que os trabalhos de Trovò e Massari (2021), Duh *et al.* (2019), He, Tian e Fu (2021), Coelho e Brandão (2019) e Kosmarski e Gordiychuk (2020) oferecem incentivos do tipo *Tokens* e *Reputation*, portanto estão contabilizados nas duas estatísticas. De

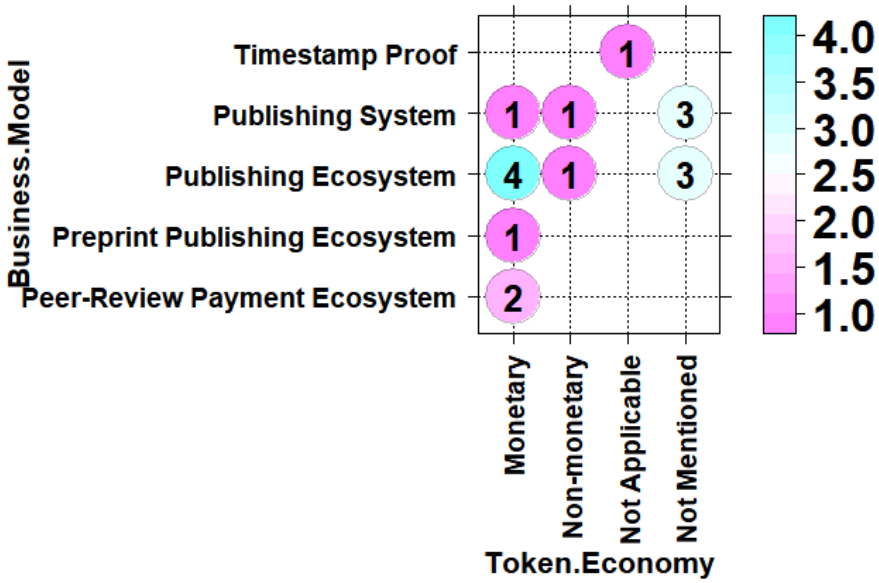

Figura 16 – Relações entre as dimensões *Business Model* e *Token Enconomy*. Fonte: Elaborado pelo autor.

maneira semelhante Khan e Shahaab (2021) oferecem incentivos do tipo *Authorship* e *Reputation* aos revisores, e está contabilizado nestas duas estatísticas.

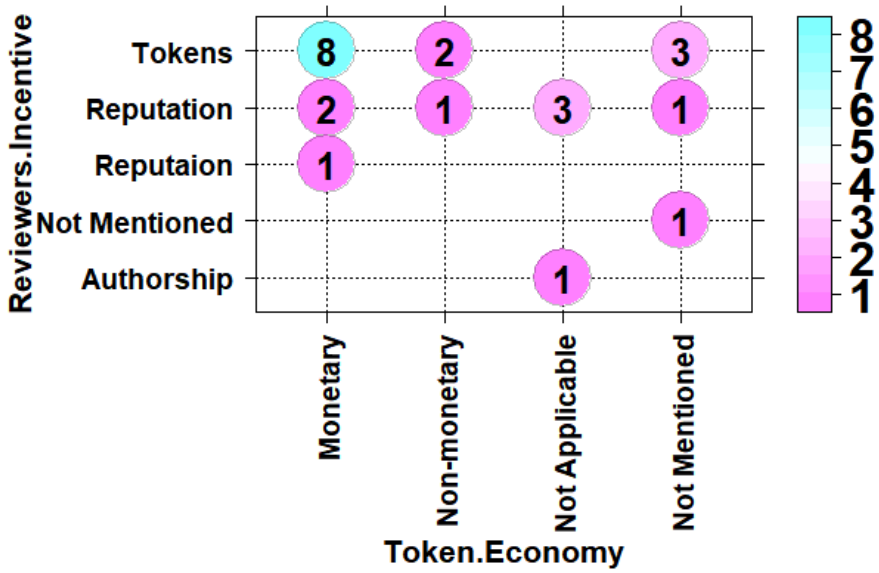

Figura 17 – Relações entre as dimensões *Reviewers Incentive* e *Token Enconomy*. Fonte: Elaborado pelo autor.

A figura 18 apresenta as relações entre as dimensões *Blockchain Used* e *Maturity*. É possível notar uma preferência pela utilização do Ethereum para o desenvol-

vimento de protótipos (4), assim como para propor modelos de sistemas (3). Cabe salientar que o trabalho de Wang, Chang Liew e Zhang (2020) propõem o *blockchain Pubchain* para seu sistema, mas utilizam o *Ethereum* para o desenvolvimento do protótipo, portanto ele está contabilizado em *Pubchain* e *Model*, assim como em *Ethereum* e *Prototype*. Similarmente, Khan e Shahaab (2021) apresentam um modelo compatível com *blockchains* privados, onde citam o *Hyperledger* e o *Corda*, portanto ele é contabilizado nas duas estatísticas.

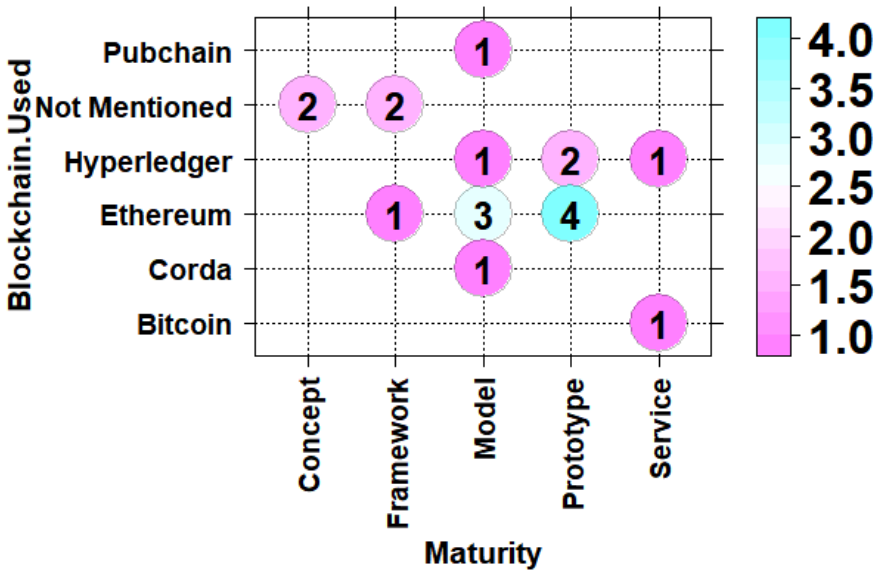

Figura 18 – Relações entre as dimensões *Blockchain Used* e *Maturity*. Fonte: Elaborado pelo autor.

Finalmente, a figura 19 apresenta as relações entre as dimensões *Blockchain Used* e *Paper Storage*. É possível perceber que a forma de armazenar os artigos está concentrada no IPFS, principalmente para o *Ethereum* (4) e para o *Hyperledger* (3), com bancos de dados como a segunda forma de armazenamento mais utilizada. Vale destacar que em Wang, Chang Liew e Zhang (2020) o IPFS está contabilizado tanto para o *Ethereum* quanto para o *Pubchain*, assim como em Khan e Shahaab (2021) ele está contabilizado para o *Hyperledger* e o *Corda*. Por outro lado em Duh *et al.* (2019) existem três possibilidades para o armazenamento dos artigos: *Preprint Servers*, *Self-archiving* e IPFS, portanto neste caso o *Hyperledger* é contabilizado para todos eles.

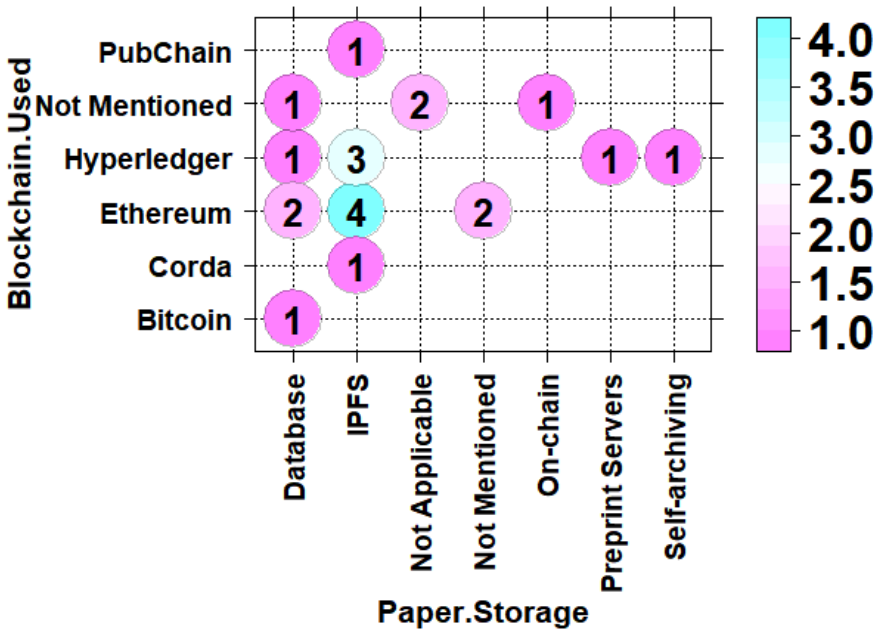

Figura 19 – Relações entre as dimensões *Blockchain Used* e *Paper Storage*. Fonte: Elaborado pelo autor.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os aspectos teóricos dos conceitos necessários para o entendimento deste trabalho. São apresentados alguns conceitos referentes a *blockchain*, *smart contracts* e o processo de revisão por pares de artigos científicos.

3.1 *BLOCKCHAIN*

A tecnologia *blockchain* oferece um livro-razão público, ordenado, imutável, compartilhado e confiável para registrar transações, ativos digitais e *smart contracts*. Os dados são armazenados em uma cadeia de blocos pelos participantes de uma rede P2P de maneira distribuída (BASHIR, 2018; ROSA, 2019).

O *blockchain* foi criado por Satoshi Nakamoto[1] para suportar o funcionamento do *Bitcoin*, principal criptomoeda da atualidade (NAKAMOTO, 2008). Sendo esta, a primeira implementação de uma tecnologia de registros distribuídos, DLT. Por conta disso, muitas vezes os termos DLT e *blockchain* são confundidos. Contudo o primeiro refere-se à bancos de dados distribuídos de qualquer tipo, enquanto o segundo refere-se especificamente a bancos de dados distribuídos compostos por blocos de transações. Em outras palavras, todos os *blockchains* são DLTs, porém nem todos os DLTs são, necessariamente, *blockchains* (BASHIR, 2018).

A cadeia de blocos (figura 20) é, basicamente, uma lista encadeada onde cada bloco contém os seus dados, o *hash* destes dados e o *hash* do bloco anterior. Neste caso os dados são as transações realizadas entre a criação do bloco atual e a criação do novo bloco. Esta abordagem garante a segurança e a imutabilidade dos dados, pois para qualquer alteração efetuada nos dados o valor dos *hashs* será alterado. O fato de armazenar o endereço do bloco anterior garante o acesso a todos os dados (BASHIR, 2018; ROSA, 2019; ZHENG; ZHU; SI, 2019).

Os registros distribuídos são criptograficamente seguros, o que significa que criptografia foi aplicada para fornecer serviços de segurança, tornando os registros protegidos contra adulteração. Esses serviços incluem integridade dos dados e autenticação de origem dos dados. Isso torna o *blockchain* um meio seguro e o acesso público faz com que seja transparente, para o armazenamento de dados que necessitem de sua integridade garantida (BASHIR, 2018; XU *et al.*, 2017).

Existem quatro camadas que processam as transações para que elas façam parte do *blockchain*, apresentadas na figura 21. A camada de transação (*Transaction Layer*) é responsável pela interação com os usuários, é através dela que eles geram novas transações. Enquanto isso, a camada de validação (*Validation Layer*) é responsável por validar as novas transações, verificando se elas obedecem as regras da rede, em caso positivo a transação é aceita, caso contrário é rejeitada. As transações são

[1] Pseudônimo utilizado pelo criador ou grupo de criadores do *Bitcoin*.

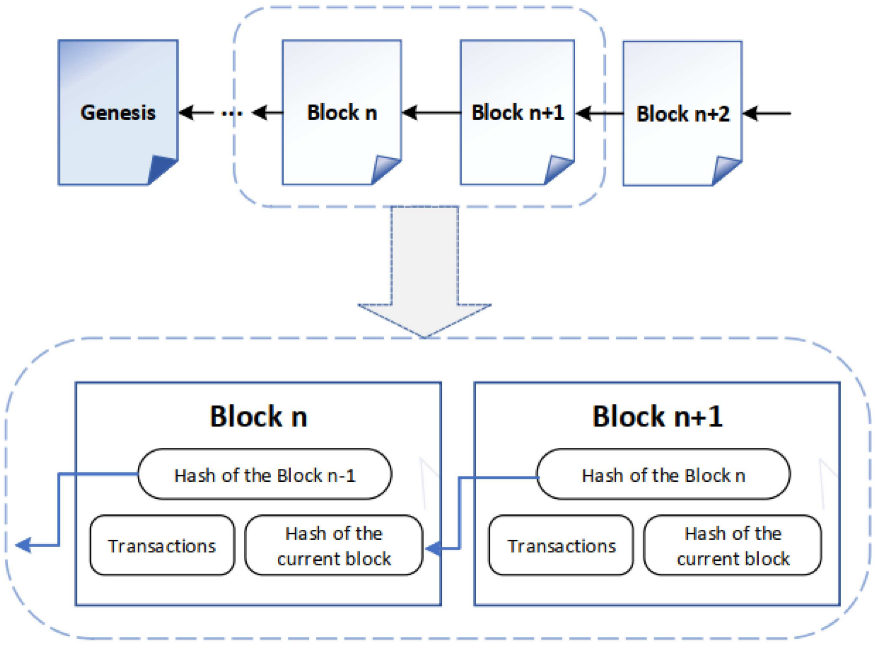

Figura 20 – Estrutura do *blockchain*. Fonte: (ZHENG; ZHU; SI, 2019).

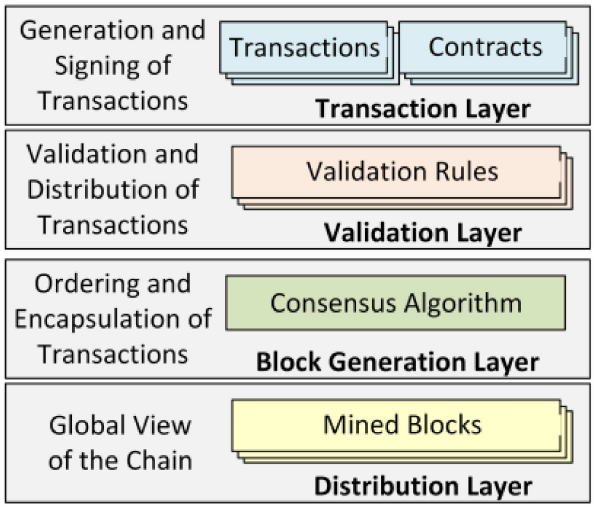

Figura 21 – Camadas que compõem o *blockchain*. Fonte: (OLIVEIRA *et al.*, 2019).

inseridas nos blocos pela camada de geração de blocos (*Block Generation Layer*), ela é responsável pelo processo conhecido como mineração, onde um nó da rede é escolhido para gerar um novo bloco, adicionando as transações aprovadas. A camada de distribuição (*Distribution Layer*) distribui o bloco minerado e validado para a estrutura do *blockchain* armazenado nos nós adjacentes (OLIVEIRA *et al.*, 2019).

Existem diferentes características relacionadas à disponibilidade dos dados no *blockchain*, o papel desempenhado por cada nó e o algoritmo de consenso da rede. Segundo Oliveira *et al.* (2019) eles podem ser classificados de acordo com os diferentes tipos:

- Públicos ou privados: nos *blockchains* públicos não existe controle de acesso à rede, ou seja, qualquer pessoa pode conectar um dispositivo e fazer parte dela contribuindo para o algoritmo de consenso, assim como acessar os dados armazenados. Enquanto os *blockchains* privados restringem a entrada de participantes na rede, eles normalmente são governados por uma instituição ou conjunto de instituições que determinam quem tem permissão para ser admito.
- Permissionados ou não permissionados: nos *blockchains* permissionados os participantes da rede são identificados, o que permite a atribuição de papéis diferenciados para cada participante. Esta abordagem permite, por exemplo, que somente alguns nós executem o algoritmo de consenso, reduzindo consumo de energia e recursos computacionais. Em contrapartida, os não permissionados não utilizam qualquer forma de identificação e, consequentemente, todos os dispositivos exercem o mesmo papel.

Os conjuntos de características público/privado e permissionado/não permissionado podem ser combinadas entre si para incrementar suas funcionalidades, aumentando a flexibilidade do *blockchain* e expandindo sua gama de aplicações. Pode-se utilizar um *blockchain* público não permissionado, onde não há restrição para a entrada de participantes, assim como não há diferenciação de papéis na rede, como o *Bitcoin* ou *Ethereum*. Assim como um privado não permissionado, onde a entrada é restrita, porém não há diferenciação de papéis. Um *blockchain* público permissionado não tem restrição sobre a entrada de participantes, contudo pode identificá-los e atribuir diferentes papéis, permitindo a execução de algoritmos de consenso menos custosos, como por exemplo o projeto *LACChain*. Enquanto um *blockchain* privado permissionado aplica ambas características, restringe a entrada de participantes ao mesmo tempo em que os identifica e atribui diferentes papéis, como o *Hyperledger*.

3.1.1 Algoritmos de Consenso

Devido à sua natureza descentralizada e não possuir um coordenador central, a adoção de algum mecanismo para manter a consistência da rede se faz necessário. Existem dois problemas que devem ser resolvidos na aplicação de *blockchains*, gasto duplo e *Byzantine Generals Problem*, *BGP*. O problema do gasto duplo refere-se a reutilizar a mesma moeda em duas transações ao mesmo tempo, nas moedas tradicionais este não é um problema, pois utiliza-se a moeda física. Em transações eletrônicas uma instituição centralizada confiável garante que o gasto duplo não ocorra, contudo em *blockchains* ele é resolvido fazendo com que muitos nós verifiquem as transações de forma distribuída. *BGP* é um problema conhecido em sistemas distribuídos, onde nós maliciosos na rede podem enviar informações erradas deliberadamente para confundir a sua execução. Neste caso os nós normais devem ser capazes de distinguir as infor-

mações maliciosas e obter as informações consistentes de outros nós. Os algoritmos de consenso são aplicados à solução destes problemas, existem diversos algoritmos diferentes que possuem características próprias e podem ser aplicados em variadas situações (JAIN; JAT, 2022; ZHENG; ZHU; SI, 2019).

O algoritmo *Proof of Work*, *PoW*, é utilizado no *Bitcoin*, ele consiste na resolução de um quebra-cabeça criptográfico, o nó que o resolve primeiro tem o direito de adicionar um novo bloco e recebe uma certa quantidade de *Bitcoin* como recompensa. Este processo demanda uma grande quantidade de energia e poder computacional para ser executado, pois o quebra-cabeça é uma função *hash*. Deste modo, nós com maior poder computacional têm mais chance de obter sucesso. Se alguém controlar mais de 50% do poder computacional da rede, pode controlar o seu funcionamento (JAIN; JAT, 2022; ZHENG; ZHU; SI, 2019).

No algoritmo *Proof of Stake*, *PoS*, o nó que deseja adicionar um novo bloco deve apostar uma quantidade de *tokens* nisso. Se ele for selecionado, adicionará o bloco e receberá uma quantidade de *tokens* como recompensa, proporcional ao valor que foi apostado. Esta abordagem reduz drasticamente o consumo de energia e poder computacional. O *Ethereum* migrou para este algoritmo a partir do dia 15 de Setembro de 2022 (ZHENG; ZHU; SI, 2019).

O algoritmo *Practical Byzantine Fault Tolerance*, *PBFT*, pode funcionar com até $\frac{1}{3}$ de nós maliciosos, buscando o consenso através da troca de informações entre os nós honestos. A princípio uma mensagem do tipo *Request* é enviada do cliente para o nó principal, que envia uma mensagem do tipo *Pre-prepare* para os outros nós. Cada nó que aceita a requisição transmite uma mensagem do tipo *Prepare* para todos os nós e a recebe de todos os outros, se a maioria aceita a requisição eles entram no modo *commit*. Neste modo cada nó envia uma mensagem do tipo *Commit* para todos os outro nós, assim cada um executa a requisição. Depois da execução cada os nós enviam uma mensagem do tipo *Reply* para o cliente. Ele funciona em ambientes com um número limitado de nós, devido à grande troca de mensagens entre eles, porém consome muito menos energia do que o *PoW*. Este algoritmo é utilizado pelo *blockchain* privado *Hyperledger* (JAIN; JAT, 2022; ZHENG; ZHU; SI, 2019).

3.1.2 *Smart Contracts*

Os *smart contracts* foram introduzidos por Szabo (1997). Eles são análogos aos contratos do mundo real, porém digitais e autoexecutáveis. Contratos geralmente envolvem dois participantes, mas para garantir o seu cumprimento no mundo real normalmente uma terceira parte confiável é necessária. Esta terceira parte pode ser um órgão governamental, um banco etc., porém os *smart contracts* foram pensados para eliminar esta terceira parte centralizada. Desse modo, para que sua implementação fosse possível era necessário um meio seguro e confiável para sua execução, este

meio surgiu com o *blockchain* (PANDA; SATAPATHY, 2021).

Smart contracts podem ser definidos como programas de computador determinísticos que executam de maneira autônoma e segura através de transações de um *blockchain*, capazes de manter seu estado, receber dados de entrada e interagir com criptomoedas (HILDENBRANDT *et al.*, 2018; BASHIR, 2018). Eles normalmente contém alguma lógica de negócios e uma quantidade limitada de dados (BASHIR, 2018; PALMA, 2020). Eles são executados pelos dispositivos participantes do *blockchain*, e são armazenados nos blocos de dados (PANDA; SATAPATHY, 2021).

Os *smart contracts* foram implementados de fato com a introdução do *Ethereum* por Buterin (2014). Ele foi criado com o objetivo fornecer um protocolo para a execução de aplicações descentralizadas. O *Ethereum* executa os *smart contracts* através da *Ethereum Virtual Machine* (*EVM*). O endereço que deseja que o *smart contract* seja executado deve pagar taxas de execução, as taxas são definidas em *gas*, que é considerado o custo computacional para executar a tarefa (HILDENBRANDT *et al.*, 2018; PANDA; SATAPATHY, 2021; PALMA, 2020). Quanto um *gas* custa em *ether* oscila de acordo com as condições da rede.

Segundo Bashir (2018) o *Ethereum* possui dois tipos de contas que permitem a execução de transações: *Externally Owned Accounts* (*EOA*), são associadas a um usuário humano, são capazes de enviar transações e não possuem código associado e; *Contract Accounts* (*CA*), são associadas a um código que é armazenado no *blockchain* e pode ser executado em resposta à transações. Ambas as contas são controladas por um par de chaves assimétricas, possuindo uma chave privada, a partir da qual é derivada uma chave pública. Normalmente as contas são identificadas na rede através de seu endereço, ele é derivado a partir da chave pública associada à conta.

3.1.3 *Tokens*

Segundo Ferreira (2020) um *token blockchain* constitui um título digital ao portador, onde a propriedade é determinada pelos dados incorporados no *blockchain*. Existem basicamente dois tipos de *token*: a moeda nativa, que é emitida de acordo com as regras do próprio *blockchain* e não necessita de um fiador; e o *IOU token*, que requer uma entidade centralizada para controlar sua emissão e atuar como fiador para implementar a troca de *tokens* (YOO, 2020).

No *Ethereum* existem padrões para o desenvolvimento de *tokens* do tipo *IOU*, eles existem para garantir a compatibilidade com diferentes aplicações. Os padrões relevantes para este trabalho são o *ERC-20*[2] e *ERC-721*[3]. O primeiro é o responsável pela padronização dos *tokens* fungíveis, ou seja, cada *token* do mesmo *smart contract* tem o mesmo valor monetário. Enquanto o segundo padroniza os *tokens* não fungíveis

[2] https://ethereum.org/pt/developers/docs/standards/tokens/erc-20/
[3] https://ethereum.org/pt/developers/docs/standards/tokens/erc-721/

(*NFT*), ou seja, cada unidade pode ter valor monetário diferente, inclusive quando geradas pelo mesmo *smart contract*. Os *NFT*s são únicos e podem representar um ativo, digital ou físico, no *blockchain* (VALEONTI *et al.*, 2021).

Para permitir a comercialização automatizada de um *token ERC-20* é necessário que ele seja disponibilizado em um *smart contract* conhecido como *liquidity pool*. Ele armazena uma quantidade de criptomoedas (*Ether*, por exemplo) e dos *tokens ERC-20*, permitindo que qualquer usuário compre-os e venda-os sem a necessidade de interação com o seu criador. Existem alguns *liquidity pools* para *tokens ERC-20*, como o *Uniswap* e o *SushiSwap* por exemplo, que são muito utilizados na rede *Ethereum*.

Para representar um ativo digital o *NFT* utiliza um arquivo de metadados no formato *json*, onde as informações sobre o ativo são armazenadas, incluindo o seu endereço de armazenamento original. O *smart contract* do *NFT* armazena então o endereço deste arquivo de metadados, associado-o ao endereço no *blockchain* de quem possui o *token*.

Os *NFT*s podem ser utilizados para se manter os direitos sobre a propriedade intelectual de diferentes tipos obras, como arte digital, por exemplo. Sobre este tema, Leonardo Barboza, Scherreier Ferneda e Beatriz Sass (2021) avaliam o uso jurídico dos *NFT*s para propriedade intelectual. Eles concluem que, apesar de ainda não ter uma clara abordagem jurídica no Brasil, autores podem utilizar os *NFT*s para certificar digitalmente a autoria das obras. Enquanto Valeonti *et al.* (2021) propõem a aplicação dos *NFT*s em galerias, bibliotecas, arquivos e museus para permitir a venda de cópias digitais das obras do seu acervo, mantendo os direitos sobre a propriedade intelectual.

O conceito de *Soulbound Tokens* (*SBT*) foi apresentado por Buterin (2022), eles são, basicamente *NFT*s não transferíveis. Portanto, eles podem representar ativos, tanto digitais quanto físicos assim como os *NFT*s. Porém, depois que o *SBT* é gerado e associado a determinado endereço não é possível transferi-lo para outro. Eles podem ser utilizados em contextos onde não busca-se valor monetário, permitindo a criação de um sistema de reputação, por exemplo, sendo utilizado como recompensa e como uma forma de registro de atividades realizadas.

3.1.4 *InterPlanetary File System*

O sistema de arquivos interplanetário *IPFS*[4], do inglês *InterPlanetary File System*, é um sistema capaz de armazenar e fornecer acesso a dados de maneira distribuída. Os dados são endereçados pelo seu conteúdo, através do *Content Identifier* (*CID*), que nada mais é do que um *hash* criptográfico do conteúdo. O *IPFS* utiliza o algoritmo *sha-256* para gerar o *CID*, isso garante um *hash* único para cada arquivo. Portanto, cada nova versão de um arquivo recebe um novo *CID*, tornando os arquivos armazenados no *IPFS* resistentes à adulteração e censura.

[4] Disponível em: https://ipfs.tech/

3.2 REVISÃO POR PARES DE ARTIGOS CIENTÍFICOS

O processo de revisão por pares é utilizado na revisão de artigos científicos desde o início dos periódicos acadêmicos no século 17. Inicialmente adotava-se um tipo de revisão fechada onde somente os revisores conheciam a identidade dos autores, o que acabou gerando problemas, pois aconteciam manipulações na avaliação com muita facilidade e frequência. Com o tempo outros modelos foram adotados na tentativa de evitá-los, como a revisão aberta e a revisão fechada *double blind* (ROWLAND, 2002).

Quando um manuscrito é submetido a alguma revista científica, ele é analisado primeiramente pelo editor responsável. Nesta etapa é avaliado se o escopo do trabalho é compatível com a revista, em caso negativo ele é rejeitado antes mesmo da revisão ser iniciada. Se passar por esta etapa então os revisores, especialistas no assunto, são escolhidos, normalmente são dois por manuscrito. Cada revisor pode classificar o artigo como publicável imediatamente, publicável com melhorias ou não publicável. Na maioria dos casos melhorias e correções são necessárias, então o parecer dos revisores é enviado para o autor, iniciando a segunda rodada de revisões. A escolha dos revisores é importante, pois até 80% dos artigos publicados passam por alguma revisão (ROWLAND, 2002; ALI; WATSON, 2016).

Portanto, a revisão por pares é o principal meio para controle de qualidade do conhecimento científico. A decisão final sobre a publicação cabe ao editor, mas o parecer dos revisores está fortemente relacionado com esta decisão (BORNMANN, 2011). Mesmo assim, normalmente os revisores exercem o seu trabalho de maneira voluntária. De maneira similar, os autores não recebem pagamento pelo seu artigo publicado, mesmo ele sendo vendido pela editora. Quando o artigo é publicado na modalidade *Open Access*, ele é disponibilizado de forma gratuita, porém os autores devem pagar altas taxas de publicação para a editora. Apesar das taxas cobradas, seja de autores ou dos leitores que pretendem adquirir o artigo, os custos de revisão são muito mais baixos do que os cobrados pelas editoras (ROWLAND, 2002).

A revisão por pares fechada é um modelo onde a identidade de pelo menos uma das partes envolvidas no processo de revisão, normalmente os revisores, não é conhecida. Existem dois tipos de revisão fechada: *single blind* e *double blind*. No modelo *single blind*, os autores não conhecem a identidade dos revisores enquanto os revisores conhecem a identidade dos autores. No modelo *double blind*, as identidades de autores e revisores são desconhecidas por ambos (ALI; WATSON, 2016).

Existem diferentes definições sobre a revisão aberta, como a apresentada por Ali e Watson (2016), onde a definem como um modelo onde a identidade de autores e revisores são conhecidas por ambas as partes. Neste caso ela busca evitar revisões desnecessariamente rudes, incentivando comentários mais construtivos para os autores, uma vez que os nomes dos revisores são publicados pela revista. Em ambos os

casos o artigo geralmente é publicado depois do processo de revisão, com a aprovação dos revisores.

Contudo, o conceito de revisão aberta pode ser muito mais amplo, como mostra Ross-Hellauer (2017), apresentando sete características diferentes de revisão aberta:

- *Open identities*, também conhecida como revisão não cega ou revisão por pares assinada, onde autores e revisores conhecem as identidades uns dos outros.
- *Open Reports*, em que os relatórios de revisão são publicados juntamente com o artigo.
- *Open participation*, também conhecida como revisão pública, permite uma participação maior da comunidade acadêmica, não somente os revisores específicos.
- *Open interaction*, onde é permitido o debate direto entre revisores e autores, e entre revisores.
- *Open pre-review manuscripts*, nesse caso os manuscritos são disponibilizados, através de servidores de *pre-print* por exemplo, antes do processo de revisão ser iniciado.
- *Open final-version commenting*, permite revisões e comentários às versões finais, já revisadas formalmente, dos artigos.
- *Open platforms*, onde a revisão é facilitada por alguma entidade externa à *venue*.

Um processo de revisão que possua qualquer uma destas características pode ser considerado como revisão aberta. Diferentes características podem ser combinadas para oferecer um processo de revisão mais aberto.

3.2.1 Problemas Apresentados na Revisão Por Pares

Mesmo sendo utilizado há muito tempo, o processo de revisão por pares apresenta alguns problemas, sendo alguns deles levantados de maneira muito clara por Ross-Hellauer (2017):

- Falta de confiabilidade e inconsistência: a confiabilidade do processo pode ser questionada, pois está sujeita ao julgamento humano que é subjetivo. Isso pode acarretar em inconsistências nas decisões de publicação ou rejeição, inclusive para detectar e prevenir erros e fraudes nas submissões.
- Lentidão para a publicação: muitas vezes há uma demora muito grande entre a submissão do artigo e sua publicação, acarretando em um grande atraso no desenvolvimento do conhecimento científico.

- Falta de responsabilidade e riscos de subversão: a falta de transparência do processo de revisão tradicional, considerado até como um sistema de caixa preta, permite que os participantes tenham poder para subverter o processo. Essa falta de transparência permite que editores rejeitem unilateralmente manuscritos sem explicações ou escolher deliberadamente revisores que discordam das ideias apresentadas. Assim como os revisores podem esconder conflitos de interesse, prejudicando autores considerados rivais ou inimigos e favorecendo amigos, ou até mesmo roubar as ideias dos autores e submeterem à outras revistas como suas próprias ideias.

- Falta de incentivo e reconhecimento: os revisores e editores exercem seu trabalho de forma voluntária e normalmente de forma anônima.

- Falta de revisores, diretamente relacionada à lentidão no processo de revisão, pois exige uma grande carga de trabalho sobre os revisores existentes.

- Viés social e intelectual: na revisão do tipo *single blind* pode haver, por parte dos revisores, discriminação cultural, por gênero ou prestígio da instituição. Assim como revisores podem discriminar resultados e métodos inovadores que contrariem o paradigma dominante.

- Desperdício: os comentários dos revisores frequentemente atribuem contexto ou apontam trabalhos futuros, assim como as discordâncias podem expor áreas de tensão em teorias e argumentos. Estas discussões e trocas de informações poderiam contribuir com o aprendizado de novos pesquisadores e revisores.

Estes problemas são corroborados por ROSSUM (2017), Das (2016), Bornmann (2011) e Rowland (2002). Devido à sua importância para o controle de qualidade do conhecimento científico, é importante o desenvolvimento de soluções capazes de resolver tais problemas, ou pelo menos mitigá-los.

O estudo de Kravitz *et al.* (2010), apresenta uma análise da relação entre as recomendações dos revisores e a decisão dos editores no *Journal of General Internal Medicine*. Eles perceberam que cerca de 20% dos manuscritos com recomendação para não rejeição, ou seja, para prosseguir com a revisão, foram rejeitados pelos editores. Assim como cerca de 48% do total de manuscritos submetidos foram rejeitados. Por isso um processo de revisão mais transparente é necessário, buscando evitar a tomada de decisões em desacordo com as recomendações dos revisores.

Através de um questionário aplicado a pesquisadores de várias partes do mundo das áreas de ciência, tecnologias, engenharia e matemática, Silbiger e Stubler (2019) perceberam que cerca de 58% dos participantes já receberam revisões não profissionais em seus manuscritos submetidos para publicação. Os autores entre eles que pertencem à grupos minoritários nas ciências relataram que isso afetou a confiança

no seu trabalho. Assim como, muitas vezes revisores tendem a avaliar a qualidade do trabalho baseado em questões como o gênero. A disponibilização dos relatórios de revisão pode acarretar em uma melhora na qualidade dos mesmos, incentivando avaliações mais construtivas e evitando este tipo de problema (BRAVO *et al.*, 2019).

4 PROCEDIMENTOS METODOLÓGICOS

Este capítulo apresenta a metodologia utilizada no desenvolvimento do estudo. É apresentado o modelo proposto no trabalho (figura 22), assim como a sua classificação de acordo com as características levantadas no capítulo 2. O protótipo desenvolvido é apresentado na seção 4.2.

4.1 SISTEMA DE REVISÃO POR PARES BASEADO EM *BLOCKCHAIN*

Os principais problemas relatados na literatura estão relacionados à falta de transparência e falta de incentivos aos participantes. A solução desenvolvida neste trabalho busca resolver ou mitigar tais problemas atacando estas duas características dos sistemas de revisão por pares.

Esta solução é desenvolvida para o *blockchain Ethereum*, implementando todas as funcionalidades do sistema em *smart contracts*. Os artigos, propostas de pesquisa e relatórios de revisão são armazenados no *IPFS*, de maneira distribuída e são endereçados através do seu *hash*. Incentivos são oferecidos aos revisores e editores através do *token PubToken*, criado para ser utilizado como moeda na plataforma. Através dele são efetuados os pagamentos das taxas exigidas por cada *venue*.

A Fig. 22 apresenta o modelo geral do sistema proposto. Ele consiste de um aplicativo descentralizado, *Dapp* (*Decentralized application*), pois é executado no *blockchain* do *Ethereum*. Dois módulos principais compõem o sistema, *front end* e *back end*. O *front end* é responsável pela interação com os usuários, desenvolvido em *JavaScript* ele utiliza a biblioteca *web3.js* para executar as funções dos *smart contracts*. A conexão entre os usuários e os *back end* é feita por meio do *Metamask*, responsável por fornecer os endereços dos mesmos, que são utilizados para controlar o acesso às informações contidas nos *smart contracts* e as permissões para execução de determinadas funções.

O *back end* é composto por sete *smart contracts* que são acessados diretamente pelo *front end*, representados pelos retângulos em azul, eles são implantados antes do sistema ser disponibilizado on-line. Esta abordagem foi necessária, pois o compilador da linguagem *Solidity* limita o tamanho máximo do código compilado a 24.576 *bytes*, desde o *hard fork Spurious Dragon*[1] do *Ethereum*. Durante o desenvolvimento constatou-se que a implementação de todas as funcionalidades em somente um *smart contract* ultrapassava muito este limite. *Paper* e *Venue*, representados pelos retângulos em verde, são gerados por *Submission* e *Publisher*, respectivamente.

Devido ao funcionamento do *blockchain* público e do *IPFS*, o sistema apresenta as características *Open Reports* e *Open Pre-review Manuscripts* por padrão. Pois todas as informações armazenadas no *blockchain* são públicas, de maneira semelhante,

[1] O *hard fork* ocorreu em 22 de Novembro de 2016

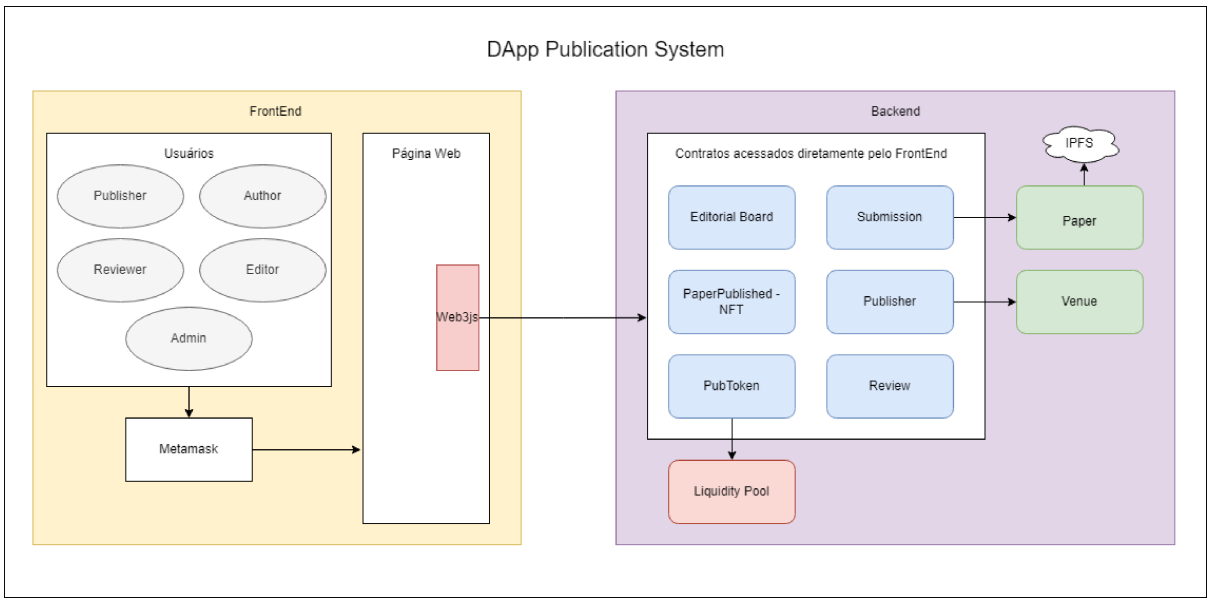

Figura 22 – Modelo geral do sistema proposto. Fonte: Elaborado pelo autor.

quem possui o *CID* do arquivo no *IPFS* obtém acesso a ele. Todos os artigos publicados por revistas e conferências são disponibilizados como *Open Access*. O *timestamp* do *blockchain* é armazenado no *smart contract* em conjunto com o *CID* do arquivo armazenado no *IPFS*. Este processo garante uma prova imutável do momento em que o artigo foi submetido à plataforma, conhecido como prova de existência, e permite a verificação manual da existência do documento (TROVÒ; MASSARI, 2021). Estas características oferecem transparência ao sistema, pois os *smart contracts* são executados publicamente, assim como os relatórios de revisão e manuscritos também são disponibilizados publicamente. A transparência busca coibir comportamentos inadequados, como viés durante a revisão e tentativas de fraude, ao mesmo tempo em que busca aumentar a confiança no processo.

Quando não se utiliza a revisão *Open Identities* as identidades dos participantes, tanto revisores como autores são mantidas em anonimato através do pseudo-anonimato. Neste caso, utiliza-se as contas de propriedade externa do *Ethereum*, para identificar os participantes durante o processo, semelhante à uma das técnicas utilizadas por Trovò e Massari (2021). Neste caso, somente os endereços das contas são utilizados no *blockchain* e nenhuma informação sobre a identidade dos participantes é armazenada durante a revisão. Somente quando um artigo é publicado as identidades dos autores são solicitadas. Em uma futura submissão é muito simples para um autor utilizar um novo endereço a fim de manter o anonimato. Desse modo, busca-se combater o viés durante a revisão.

Para controlar o avanço do manuscrito ou proposta de projeto no processo de revisão o seu *status* é mantido e atualizado internamente, de acordo com as decisões

do editor responsável. A tabela 2 apresenta de forma resumida os valores do *status*, uma breve descrição e sua aplicação. Utilizam-se valores de 1 até 8, indicando quais são aplicados para submissão de artigos e quais para propostas de projeto. O *status* 4 não é aplicado em projetos, pois normalmente avaliações de propostas de projeto não passam por uma segunda rodada de revisões.

Tabela 2 – *Status* dos artigos/projetos durante o processo de revisão. Fonte: Elaborado pelo autor.

Código	Descrição	Aplicação
1	Submetido	Artigo/Projeto
2	*Desk Reject*	Artigo/Projeto
3	Em revisão - 1ª Rodada	Artigo/Projeto
4	Em revisão - 2ª Rodada	Artigo
5	Aguardando versão final	Artigo/Projeto
6	Publicado/Aceito	Artigo/Projeto
7	Rejeitado	Artigo/Projeto
8	Revogado	Artigo/Projeto

4.1.1 Soluções Adotadas pelo Modelo

O quadro 2 apresenta de maneira concisa como o modelo aqui apresentado ataca os problemas relatados sobre o processo de revisão por pares. Sua implementação busca executar o processo de revisão de maneira transparente, assim como oferecer incentivos aos participantes. As seções a seguir descrevem de maneira detalhada as soluções adotadas.

4.1.2 *Tokens*

O *PubToken* foi criado para ser utilizado como moeda na plataforma. Ele serve para o pagamento das taxas de publicação, assim como para incentivo aos revisores e editores, desse modo busca-se resolver o problema da falta de incentivo. Espera-se também que este mecanismo atraia o interesse de mais potenciais participantes, reduzindo a carga de trabalho sobre cada indivíduo e, consequentemente, auxilie a reduzir o tempo de revisão.

Quando é utilizado como incentivo, ao término do processo de revisão, exceto para *desk reject*, *PubTokens* são cunhados e transferidos para editores e revisores. A quantidade de *tokens* gerada ainda não foi definida, pois estudos ainda são necessários para encontrar uma valor adequado. Esta definição ainda não ocorreu uma vez que, ela está relacionada ao valor monetário do *PubToken*. Para o protótipo desenvolvido neste trabalho foi gerado 1 *PubToken* para cada revisor e editor.

A taxa de publicação, quando exigida, é transferida para o *Paper* no momento da submissão. Se o autor tentar transferir um valor diferente da taxa cobrada pela

Quadro 2 – Soluções adotadas para os problemas relatados na literatura. Fonte: Elaborado pelo autor.

	Soluções Adotadas pelo Modelo Proposto
Falta de Confiabilidade e Inconsistência	Oferece um meio transparente para a execução do processo de revisão através de um *blockchain* público e não permissionado; Oferece diferentes tipos de revisão aberta.
Lentidão para a Publicação	Incentivos monetários são oferecidos aos revisores e editores, buscando atrair mais interessados a participar do processo de revisão; Futuramente um sistema de reputação será implementado visando complementar os incentivos; A revisão *Open Participation* é oferecida, permitindo que qualquer usuário da plataforma atue como revisor.
Falta de Responsabilidade e Riscos de Subversão	Novamente um meio transparente para a execução do processo de revisão é oferecido; Os *smart contracts* permitem a execução de maneira autônoma, onde todas as *venues* seguem as mesmas regras e o mesmo fluxo de execução.
Falta de Incentivo e Reconhecimento	Incentivos monetários são oferecidos, e futuramente um sistema de reputação será implementado.
Falta de Revisores	Busca-se contornar este problema oferecendo-se incentivos aos revisores participantes, com o objetivo de atrair mais interessados.
Viés Social e Intelectual	A revisão *double blind* é oferecida, através do pseudo-anonimato, visando impedir este tipo de comportamento; Diferentes tipos de revisão aberta são oferecidos, principalmente *Open Reports* e *Open Identities*, visando desencorajar este tipo de comportamento.
Desperdício	A revisão *Open Reports* é oferecida, tornando os relatórios de revisão públicos.

venue, a submissão não é efetuada. Se o artigo for publicado, a taxa é transferida para a *venue*, caso ele seja rejeitado, a taxa é transferida de volta para o autor.

Ao final do processo de revisão é gerado um *NFT* para o artigo, exceto para *Research Project Calls*, em caso de publicação. Seu arquivo de metadados armazena as informações dos autores, nomes e *orcid*, assim como do próprio artigo, título e palavras-chave. Ele é armazenado no *IPFS* como um arquivo do tipo *Json* e o seu

CID é salvo no *token PaperPublished*. Uma vez que o *NFT* é criado não é possível alterar suas informações, gerando um registro seguro capaz de associar os autores ao artigo original, sendo transferido ao primeiro autor. Este mecanismo é utilizado para manter a propriedade intelectual sobre o conteúdo do artigo, baseando-se no estudo de Leonardo Barboza, Scherreier Ferneda e Beatriz Sass (2021), é apresentada a possibilidade de utilizar-se *NFT*s para assegurar a propriedade intelectual sobre arte digital. Mesmo que seja possível transferir o *token* para outras pessoas, as informações contidas nos metadados do *NFT* continuam imutáveis e associadas aos autores.

Um *SBT* será implementado no sistema com base no *NFT* utilizado atualmente, pois sua aplicação é mais adequada para ativos que não possuem valor monetário agregado. Contudo a ideia de *SBT* foi sugerida somente em Janeiro de 2022 e o trabalho já estava em estágio avançado de desenvolvimento, portanto não foi possível aplicá-lo inicialmente. Como ele possui as mesmas funcionalidades de um *NFT*, excluindo a transferência de *tokens*, é possível tratá-lo como uma extensão do padrão *ERC-721*. Desse modo, além de assegurar a propriedade intelectual, um sistema de reputação será implementado com base no *SBT*, sendo estendido inclusive aos relatórios de revisão.

4.1.3 Participantes

É possível cadastrar três tipos de *venues*: *Journal*, *Conference* e *Research Project Call*. Cada uma pode ser configurada de acordo com suas particularidades como, taxa de publicação, número de revisores para cada artigo, prazo para envio dos relatórios de revisão e a adoção, ou não, de algum dos tipos de revisão aberta. Das definições apresentadas por Ross-Hellauer (2017), explicadas na seção 3.2, as seguintes são configuráveis no sistema: *Open Identities*, *Open Participation* e *Open Final-version Commenting*.

Existem três atores principais que atuam na plataforma: autores, revisores e editores. Os autores são responsáveis por submeter novos artigos ou propostas de projeto de pesquisa para revisão. Somente um autor, normalmente o primeiro autor, deve ficar responsável pela submissão do manuscrito e das novas versões. Os revisores são cadastrados nas *venues*, exceto quando a revisão é *Open Participation*, e são responsáveis por avaliar os manuscritos e propostas de projeto e enviar um relatório com o seu parecer. Os editores são os responsáveis pelo processo de revisão, internamente a plataforma trata todos os responsáveis pela revisão como editores. Eles decidem se o manuscrito será publicado, rejeitado ou se a proposta de projeto será aceita, assim como devem adicionar novos revisores e convidá-los a revisar um artigo. Editores e revisores recebem *PubTokens* como incentivo por participar de revisões.

Adicionalmente existem mais dois atores, *Publisher* e o administrador do sistema. *Publisher* refere-se às editoras, instituições que têm autorização para adicionar

novas *venues* e configurar suas características. O administrador implanta os *smart contracts* inicialmente, adiciona novos *Publishers* e é responsável por administrar os *Pubtokens*. Ele os transfere para o *liquidity pool* escolhido e pode acionar a função *burn* e eliminar alguns *tokens* para controlar o seu valor.

4.1.4 Tipos de Revisão

A plataforma disponibiliza a revisão de artigos científicos chamada de tradicional, pela definição de Ross-Hellauer (2017), seu funcionamento é explicado na seção 4.1.4.1. Assim como diferentes tipos de revisão aberta, que podem ser combinados ou adotados isoladamente, adicionando ao processo tradicional ou utilizando o *Open Participation*.

O quadro 3 apresenta de maneira resumida os tipos de revisão aberta que são disponibilizados e quando podem ser são utilizados. *Open Identities* acontece quando as identidades dos autores e revisores são publicadas durante a revisão, ela pode ser habilitada ou desabilitada de acordo com as necessidades das *venues*, esta abordagem baseia-se no trabalho de Choi e Seo (2021).

Quadro 3 – Tipos de Revisão Aberta oferecidos. Fonte: Elaborado pelo autor.

Tipos de Revisão Aberta	**Descrição**	**Opções**
Open Identities	As identidades dos autores e revisores são divulgadas	Configurável
Open Reports	Divulgação pública dos relatórios de revisão	Sempre
Open Participation	Qualquer um pode atuar como revisor	Configurável
Open Pre-review Manuscripts	O manuscrito é disponibilizado antes da revisão	Sempre
Open Final-version Commenting	Comentários dos leitores são permitidos depois que o artigo é publicado	Configurável

As características *Open Reports* e *Open Pre-review Manuscripts*, referem-se à disponibilização pública dos relatórios de revisão e do manuscrito, respectivamente, antes da publicação do artigo ou do projeto ser aceito. No estágio atual de desenvolvimento da plataforma esta opção está sempre ativada, isso acontece devido às características do *Ethereum* e do *IPFS*. Para permitir que estas opções sejam configuráveis a opção de armazenamento em bancos de dados pode ser oferecida, assim como a possibilidade de criptografar os arquivos antes do envio ao *IPFS*. Estas abordagens precisam ser implementadas *off-chain*, o que não é o foco do trabalho neste momento, aqui busca-se oferecer uma plataforma totalmente funcional que não exija nenhuma implementação local para as *venues*.

A revisão *Open Participation* tem um *workflow* diferente da tradicional, onde qualquer usuário da plataforma pode assumir o papel de revisor, seu funcionamento é descrito na seção 4.1.4.2. Esta é uma opção configurável, assim como a característica *Open Final-version Commenting* que permite o envio de comentários sobre os artigos, após a sua publicação.

4.1.4.1 Revisão Tradicional

A figura 23 apresenta o *workflow* do processo de revisão tradicional, ele segue as mesmas etapas tanto na revisão fechada, quanto na *Open Identities*, contudo a identidade dos autores é requerida em momentos diferentes. Este *workflow* é baseado no modelo de revisão baseado em *smart contracts* proposto por Yoo e Won (2018).

Inicialmente o primeiro autor do artigo efetua a submissão, neste momento uma nova instância do *smart contract Paper* é criada, as informações necessárias são adicionadas e o seu *status* é definido como 1 (tabela 2). Um editor então é designado aleatoriamente para ser o responsável pelo processo. Ele deve decidir, no primeiro momento, se o manuscrito será aceito para revisão. Caso sim, ele deve escolher e convidar os revisores, o *status* é atualizado para 3 e a primeira rodada de revisões é iniciada, caso contrário o manuscrito é rejeitado (*desk reject*), com *status* 2. Se a revisão for *Open Identities*, nesta etapa deve-se enviar as informações de identificação dos autores, da mesma forma elas devem constar no manuscrito. Senão, os autores devem permanecer anônimos neste momento.

A partir deste ponto cada revisor convidado deve decidir se aceita ou não revisar o manuscrito. Se aceitar, cada um pode enviar seu relatório de revisão, que será divulgado publicamente junto com o manuscrito. Senão, o editor deve convidar um novo revisor. Caso a revisão seja *Open Identities*, a identificação dos revisores deve constar nos relatório de revisão, caso contrário devem permanecer anônimos.

Com base nos relatórios de revisão, o editor deve decidir se o manuscrito deve ser publicado, se sim a revisão é finalizada, a versão final do artigo é solicitada e o *status* atualizado para 5. Caso contrário, o manuscrito pode ser rejeitado, se o for, o processo é encerrado com a rejeição e *status* 7. Caso a revisão prossiga, a segunda rodada de revisões é iniciada, o *status* é atualizado para 4 e as melhorias são solicitadas ao autor com base no parecer dos revisores.

Na segunda rodada de revisões, o autor deve submeter uma nova versão do manuscrito com correções e melhorias baseadas nas solicitações dos revisores. Neste ponto os revisores têm acesso à nova versão diretamente, e devem enviar novos relatórios de revisão. Como anteriormente, os relatórios devem conter sua identificação, em caso de revisão *Open Identities*, caso contrário eles devem permanecer anônimos. Em qualquer caso de rejeição, exceto *desk reject*, os *tokens* são gerados e enviados para o editor e os revisores.

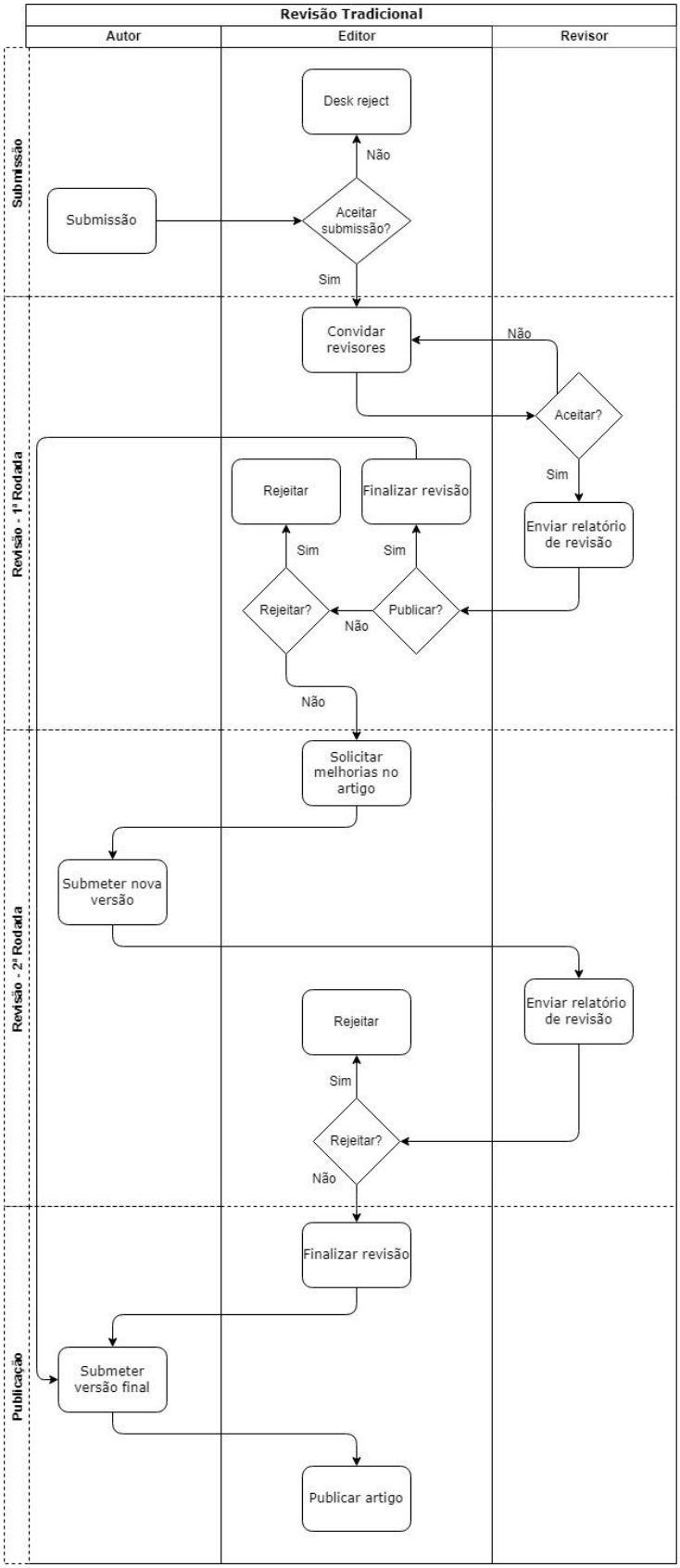

Figura 23 – *Workflow* Revisão Tradicional. Fonte: Elaborado pelo autor.

Novamente o editor deve avaliar o manuscrito com base nos relatórios de revisão, se for rejeitado o processo é encerrado com *status* 7, se for aceito para publicação a revisão é finalizada e a versão final é solicitada com *status* atualizado para 5. O autor então deve submeter a versão final devidamente corrigida e identificada, assim como as informações de identificação dos autores, nomes e orcid, também devem ser submetidas à plataforma. Caso a revisão seja *Open Identities* estas informações já constam no sistema, pois foram enviadas na submissão do manuscrito. Neste caso a solicitação da versão final pode não ser necessária, se novas correções não forem solicitadas. Isso acontece, pois a identidade dos autores já consta no manuscrito e na plataforma, portanto a ação Finalizar Revisão pode ser ignorada e o editor pode publicar o artigo diretamente.

Finalmente o artigo é publicado, seu *status* é atualizado para 6 e tanto revisores quanto o editor receberão *PubTokens*. Os *tokens* são gerados pelo sistema, assim como o *NFT PaperPublished* é gerado e transferido para o autor. Caso a *venue* seja do tipo *Research Project Call*, o *NFT* não é gerado para o projeto de pesquisa. A taxa de publicação é transferida do *Paper* para a *Venue*, em caso de rejeição a taxa é devolvida ao autor.

4.1.4.2 *Open Participation*

A figura 24 apresenta o *workflow* do processo de revisão *Open Participation*, ele segue as mesmas etapas utilizando, ou não, a característica *Open Identities*. Contudo as identidades dos autores e revisores são requeridas em momento diferentes. Este *workflow* baseia-se no modelo de revisão proposto por Trovò e Massari (2021).

Inicialmente o primeiro autor do artigo efetua a submissão, neste momento a plataforma gera uma nova instância do *smart contract Paper* com *status* 1. Então um editor é designado aleatoriamente para ser o responsável pelo processo. Ele deve decidir, no primeiro momento, se o manuscrito será aceito para revisão, caso sim, ele o disponibiliza para revisão atualizando seu *status* para 3. Caso contrário o manuscrito é rejeitado (*desk reject*) com *status* 2. Se a revisão for *Open Identities*, nesta etapa deve-se enviar as informações de identificação dos autores, assim como elas devem constar no manuscrito, senão, os autores devem permanecer anônimos neste momento.

A partir deste ponto cada participante do sistema pode atuar como um revisor, não é necessário estar cadastrado a nenhuma *venue*, assim como é desnecessário ser convidado para revisar. Desse modo, qualquer interessado pode enviar relatórios de revisão. Com base nestes relatórios o autor pode enviar novas versões do manuscrito a qualquer momento. Caso a revisão seja *Open Identities*, deve constar a identificação dos revisores nos relatório de revisão, caso contrário devem permanecer anônimos, o mesmo vale para as novas versões do manuscrito.

Quando novas versões do manuscrito são submetidas, o editor é informado,

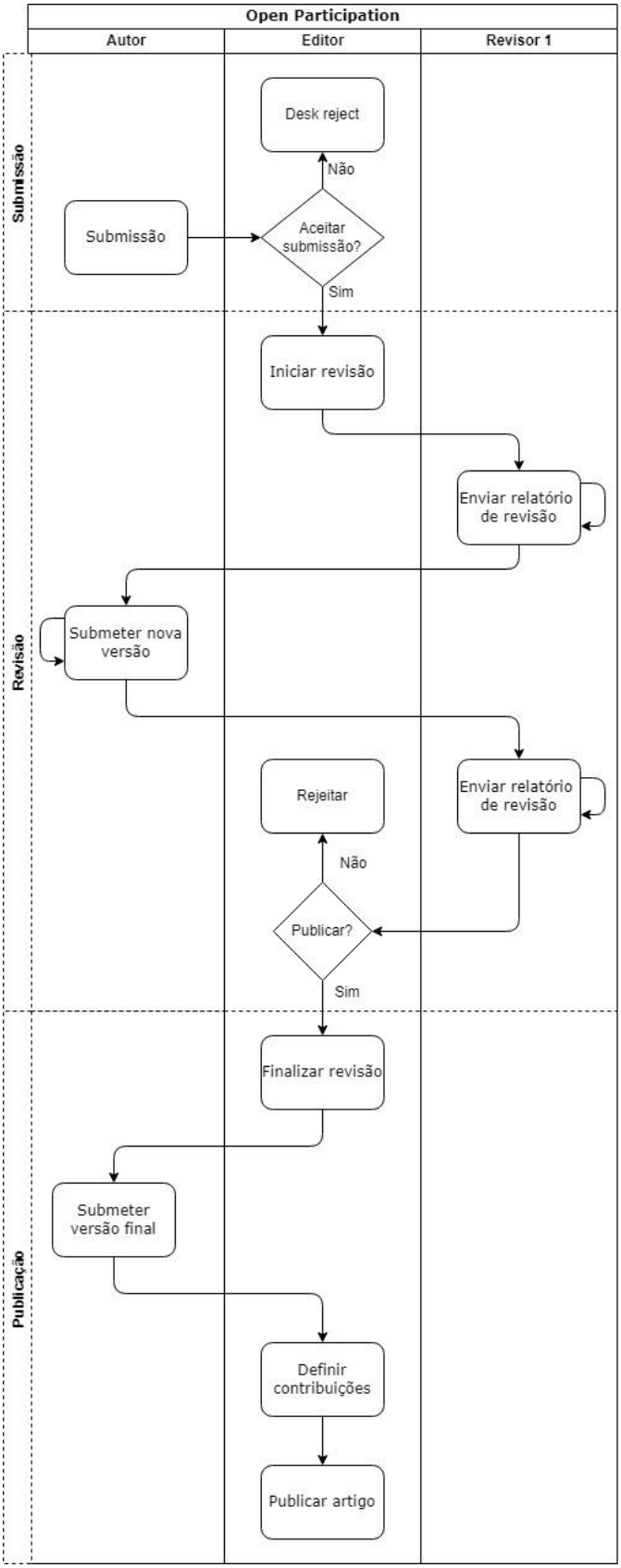

Figura 24 – *Workflow* Revisão *Open Participation*. Fonte: Elaborado pelo autor.

então ele pode avaliar se está adequado para publicação. As etapas Enviar relatório de revisão e Submeter nova versão podem ser executadas quantas vezes forem necessárias e em qualquer sequência. O editor pode rejeitar o manuscrito a qualquer momento, se assim julgar necessário. Caso ele decida publicar o artigo, a revisão é finalizada e, nesta etapa, a versão final do artigo é solicitada, atualizando seu *status* para 5.

Nesta etapa o autor submete a versão final do artigo, devidamente identificada, assim como as informações dos autores, nomes e orcid. Aqui o editor deve definir, com base nos relatórios de revisão, quais revisores contribuíram de fato para a revisão do artigo. Estes são os revisores que receberão *PubTokens* como recompensa. Esta abordagem visa coibir a prática indesejada de alguém enviar um texto qualquer como relatório de revisão, somente para receber *tokens* por isso. Caso o artigo seja publicado, a taxa de publicação é transferida para a *Venue*, caso seja rejeitado, a taxa é devolvida para o autor.

Finalmente o artigo é publicado, seu *status* atualizado para 6 e todos os revisores selecionados pelo editor, assim como o próprio editor recebem *PubTokens*. O *NFT PaperPublished* é gerado para a versão final do artigo e transferido para o endereço do autor. Em caso de revisão *Open Identities*, não é necessário solicitar a versão final do artigo, pois tanto o manuscrito está identificado, como as informações já constam na plataforma. Portanto as etapas Finalizar revisão e Solicitar versão final podem ser ignoradas e o artigo pode ser publicado diretamente.

4.1.5 Classificação

Quadro 4 – Classificação do modelo. Fonte: Elaborado pelo autor.

Categoria	**Classificação**
Business Model	Publishing Ecosystem
Open Access	Yes
Review Type	Open/Double-Blind
Reviewers Incentive	Tokens
Token Economy	Monetary
Blockchain Access	Public
Blockchain Identification	Permissionless
Blockchain Used	Ethereum
Paper Storage	IPFS

De acordo com as categorias apresentadas no capítulo 2, o modelo aqui apresentado pode ser classificado conforme mostra o quadro 4. O modelo de negócios (*Business Model*) adotado é o ecossistema de publicações (*Publishing Ecosystem*), enquanto disponibiliza os artigos como *Open Access* e oferece diferentes formas de revisão aberta (*Open*), ao mesmo tempo em que oferece a revisão fechada *double blind*.

Sobre os incentivos (*Reviewers Incentive*), ele pertence à categoria *Tokens*, enquanto possui uma *Token Economy* do tipo *Monetary*, ou seja, os *tokens* têm valor monetário. Em suas classificações sobre o *blockchain*, ele foi desenvolvido para o *Ethereum*, consequentemente o acesso (*Blockchain Access*) ao *blockchain* é público (*Public*) e a identificação (*Blockchain Identification*) é não permissionada (*Permissionless*).

4.1.6 Contribuições do Trabalho

Quadro 5 – Resumo das contribuições propostas neste trabalho. Fonte: Elaborado pelo autor.

	Trabalhos Encontrados	**Este Trabalho**
Transparência	A maioria utiliza blockchains públicos e não permissionados; Muitos oferecem a revisão aberta.	Utiliza blockchain público e não permissionado; Diferentes tipos de revisão aberta.
Incentivos	A maioria dos trabalhos oferecem tokens e/ou utilizam um sistema de reputação como incentivo.	Tokens são oferecidos como incentivo, incluindo um NFT visando manter a propriedade intelectual sobre os artigos com os autores, que servirá de base para um sistema de reputação.
Tokens	Muitos dos trabalhos que oferecem tokens permitem obter-se moeda fiduciária através deles; Alguns são utilizados somente paracontabilizar a reputação dos usuários; Nenhum trabalho encontrado utiliza NFTs ou SBTs.	Um token é utilizado para pagamentos na plataforma, inclusive para revisores e editores; Um NFT é associado ao artigo publicado, ele será a base de um SBT que implementaráum sistema de reputação.

De acordo com as características apresentadas pelos trabalhos encontrados o quadro 5 sumariza as contribuições propostas por este trabalho. O modelo aqui proposto avança na aplicação de *tokens* quando comparado à literatura encontrada, utilizando um *NFT* para manter a propriedade intelectual sobre os artigos em posse dos autores. O *NFT* será modificado para criar um *SBT*, que além da propriedade intelectual, funcionará como um sistema de reputação para a plataforma, sendo aplicado também para os relatórios de revisão estendendo-o aos revisores.

Diferentes tipos de revisão aberta são oferecidos, permitindo que cada *venue* selecione as características que deseja, somente o trabalho de Choi e Seo (2021)

aplica este tipo de solução. Os *PubTokens* são utilizados para pagamentos das taxas de publicação e dos incentivos monetários aos revisores e editores. O mesmo acontece para *venues* do tipo *Research Project Call*, oferecendo incentivos monetários para os revisores e responsáveis pela aprovação dos projetos. Nenhum dos trabalhos encontrados mencionam como são aplicados os incentivos neste caso.

4.2 PROTÓTIPO

Esta seção apresenta a implementação do protótipo do sistema, baseado no modelo apresentado anteriormente. Os *smart contracts* foram implementados na linguagem *Solidity*. Uma versão inicial da interface de usuário foi desenvolvida em *Javascript*, utilizando o *framework React.js* para permitir a execução dos experimentos, ela foi criada com o objetivo de ser funcional e interagir com os *smart contracts*. Não houve tempo hábil de desenvolver uma interface pensando na melhor experiência para os usuários. A interação entre a interface de usuário e os *smart contracts* é intermediada pela biblioteca *web3.js*. Os endereços dos usuários são fornecidos pelo *Metamask*. Para facilitar a compreensão tanto os artigos, quanto as propostas de projeto são tratados como artigo, pois ambos são representados pelo *smart contract Paper*.

4.2.1 Tecnologias Utilizadas

No protótipo desenvolvido são utilizadas as seguintes tecnologias:

- *Ganache*: executa um *blockchain* pessoal similar ao do *Ethereum*, permitindo a execução de *smart contracts* e realização de testes de maneira simples e eficiente.
- *Truffle*: ambiente de desenvolvimento e *framework* de testes que permite compilar e implantar *smart contracts* de maneira simplificada.
- *JavaScript* e *React*: linguagem de programação e biblioteca utilizadas para o desenvolvimento do *FrontEnd*.
- *Metamask*: carteira de criptomoedas desenvolvida para o *Ethereum*, ela funciona como uma extensão para o navegador ou aplicativo móvel e permite a interação com aplicativos descentralizados.
- *web3.js*: biblioteca de *JavaScript* que permite a conexão do *FrontEnd* com os *smart contracts* armazenados no *blockchain* e a execução das suas funções.
- *IPFS*: *InterPlanetary File System*, um protocolo de hipermídia P2P desenvolvido para armazenar conteúdo em um sistema de arquivos distribuído.

- *Solidity*: linguagem de programação orientada a objetos de alto nível para implementação de *smart contracts*, desenvolvida para a *Ethereum Virtual Machine*.

4.2.2 Paper

```
38  contract Paper
39  {
40      uint8 private status;
41      PubToken private token;
42
43      paper_data private paper;
44      uint256 private timestamp;
45
46      address private author_addr;
47      string[] private names;
48      string[] private orcids;
49
50      address[] private reviewers_invited;
51      address[] private reviewers_declined;
52      address[] private reviewers;
53      string[] private first_reviews;
54      string[] private second_reviews;
55      string[] private open_participation_reviews;
56      address[] private open_participation_reviewers;
57
58      address private editor;
59      Venue private venue;
60      uint8 private number_rev;
61
62      comment_data[] private comments;
```

Figura 25 – Trecho de código 1 *Paper*. Fonte: Elaborado pelo autor.

O *smart contract Paper* desempenha um papel fundamental no sistema, centralizando todas as informações necessárias para garantir o seu funcionamento. Ele limita o acesso e a manipulação dessas informações através das variáveis e funções apresentadas a seguir.

Suas variáveis (figura 25) armazenam todas as informações referentes aos artigos: *status*, título, palavras-chave e o *CID* do arquivo armazenado no *IPFS*. Para tal utiliza-se uma *struct* chamada de *paper_data* contendo os campos *title*, *keywords* e *ipfs_hash*. Assim como as informações dos autores, nome (*names*) e *orcid* (*orcids*). Os endereços dos revisores: os convidados (*reviewers_invited*), os que aceitaram (*reviewers*) e os que recusaram (*reviewers_declined*) revisar o artigo, assim como os participantes da revisão *Open Participation* (*open_participation_reviewers*). Da mesma forma que os relatórios de revisão, tanto para a primeira e segunda rodadas de revisão (*first_reviews*, *second_reviews*), quanto para a revisão *Open Participation* (*open_participation_reviews*), também são armazenados. Para os comentários utiliza-se a *struct comment_data* que possui os campos *comment* e *name*, para armazenar o *CID* do comentário no *IPFS* e nome do autor do comentário, respectivamente.

As estruturas do tipo *mapping* (figura 26) são utilizadas para acelerar a verificação de informações, pois funciona como uma *hash table* que armazena as informações

```
64      mapping(address => bool) private open_participation_reviewers_map;
65      mapping(address => string) private open_participation_names;
66      mapping(address => bool) private reviewers_map;
67      mapping(address => bool) private reviewers_invited_map;
68      mapping(address => uint256) private timestamp_accept;
69      mapping(address => bool) private reviewers_declined_map;
```

Figura 26 – Trecho de código 2 *Paper*. Fonte: Elaborado pelo autor.

através de um par chave-valor. Esta estrutura permite que cada item possa ser indexado através de sua chave, quando essa chave é única, eliminando a necessidade de iteração sobre todos os elementos. No *Paper* todas as chaves são os endereços dos revisores associados à ele de alguma forma, como mostrado a seguir:

- *open_participation_reviewers_map*, faz o mapeamento de um endereço para um valor do tipo *bool*, que indica se o endereço em questão participou da revisão do tipo *Open Participation*;
- *open_participation_names*, faz o mapeamento do endereço para uma *string*, que armazena o nome dos revisores de uma revisão do tipo *Open Participation*;
- *reviewers_map*, faz o mapeamento de um endereço para um *bool*, que indica se determinado endereço é um revisor do artigo, para o caso de a revisão não ser *Open Participation*;
- *reviewers_invited_map*, semelhante ao caso anterior, porém aplicado aos revisores que foram convidados pelo editor;
- *reviewers_declined_map*, também semelhante ao caso anterior, mas para os revisores que recusaram revisar o artigo;
- *timestamp_accept*, faz o mapeamento de um endereço para um valor do tipo *uint256*, que armazena o *timestamp* do momento em que o revisor aceitou revisar o artigo.

As funções presentes em *Paper* manipulam ou fornecem acesso às suas próprias variáveis, não interagem com outros *smart contracts*, com exceção de *PubToken*. Em contrapartida elas são executadas através de outros *smart contracts*, ou seja, a interface de usuário não tem acesso direto à elas.

O construtor (figura 27) recebe os seguintes parâmetros: o *CID* do arquivo no *IPFS* (*_hash*), o endereço do primeiro autor (*_author*), o endereço da *venue* a qual o artigo está sendo submetido (*_venue*), o endereço do editor designado para o artigo (*_editor*) e o endereço do *smart contrct* que implementa o *PubToken* (*_token*). Todos os valores são armazenados internamente e o *status* do artigo é definido como 1

```
71      constructor(string memory _hash, address _author, address _venue, address _editor, address _token)
72      {
73          paper.ipfs_hash.push(_hash);
74          timestamp = block.timestamp;
75          author_addr = _author;
76          editor = _editor;
77          venue = Venue(_venue);
78          token = PubToken(_token);
79          number_rev = venue.get_number_reviewers();
80          status = 1;
81      }
82
83      function add_paper_information(address _sender, string calldata _title, string[] calldata _keywords) external
84      {
85          require(_sender == author_addr, "You are not the Author");
86          paper.title = _title;
87          paper.keywords = _keywords;
88      }
89
90      function add_authors_information(address _sender, string[] calldata _names, string[] calldata _orcid) external
91      {
92          require(_sender == author_addr, "You are not the Author");
93          require(venue.get_open_identities() || status == 5, "You can't add this information now");
94
95          for(uint i = 0; i < _names.length; i++)
96          {
97              names.push(_names[i]);
98              orcids.push(_orcid[i]);
99          }
100     }
```

Figura 27 – Trecho de código 3 *Paper*. Fonte: Elaborado pelo autor.

(Submetido). As funções *add_paper_information* e *add_authors_information* são responsáveis por receber e armazenar as informações sobre os artigos, título (*_title*) e palavras-chave (*keywords*); e sobre os autores, nome (*_names*) e orcid (*_orcid*). Ambas funções têm sua execução restrita ao primeiro autor, com a segunda sendo restrita também à revisão *Open Identities* ou ao *status* 5 (Aguardando versão final). Não foi possível adicionar as informações do artigo no construtor do *smart contract*, devido a uma limitação na linguagem *Solidity*, que não consegue trabalhar com muitas variáveis passadas por parâmetro nas funções.

A função *add_reviewer_invited* (figura 28) é executada pelo editor e responsável por adicionar um revisor à lista de convidados a revisar o artigo. O revisor pode aceitar através de *reviewer_accept* ou recusar executando *reviewer_decline*, ambas podem ser executadas somente por um revisor convidado. Os relatórios de revisão são enviados através de *add_review* (figura 29). Eles podem ser enviados somente por revisores que já aceitaram revisar o artigo, ou em caso de revisão *Open Participation*. O envio também é limitado à quando o *status* é 3 (Em revisão: 1ª rodada) ou 4 (Em revisão - 2ª rodada), e se o prazo para o envio ainda não foi atingido. Na revisão *Open Participation* o editor deve definir quais relatórios de revisão foram relevantes para a versão final do artigo, esta etapa fica a cargo da função *add_reviewer_open_participation*.

Somente o primeiro autor pode enviar uma nova versão do artigo utilizando a função *add_new_version* (figura 30). Sua execução é limitada a quando o *status* é 4

```
247     function add_reviewer_invited(address _sender, address _rev) external
248     {
249         require(_sender == editor);
250         require(_rev != editor);
251 
252         reviewers_invited.push(_rev);
253         reviewers_invited_map[_rev] = true;
254     }
255 
256     function reviewer_accept(address _rev) external
257     {
258         require(reviewers_invited_map[_rev]);
259         require(!venue.get_open_participation());
260         require(reviewers.length < venue.get_number_reviewers());
261 
262         timestamp_accept[_rev] = block.timestamp;
263         reviewers.push(_rev);
264         reviewers_map[_rev] = true;
265     }
266 
267     function reviewer_decline(address _rev) external
268     {
269         require(reviewers_invited_map[_rev]);
270         require(!venue.get_open_participation());
271 
272         reviewers_declined.push(_rev);
273         reviewers_declined_map[_rev] = true;
274     }
```

Figura 28 – Trecho de código 4a *Paper*. Fonte: Elaborado pelo autor.

(Em revisão - 2ª rodada) ou 5 (Aguardando versão final), ou quando a revisão é *Open Participation* e o *status* é 3 (Em revisão - 1ª rodada). Ao finalizar a revisão, a função *transfer_tokens* é executada pelo editor para gerar e transferir *PubTokens* aos revisores, ao mesmo tempo em que transfere a taxa de publicação para a *Venue* correspondente. Caso a opção *Open Final-version Commenting* esteja habilitada a função *comment* adiciona um comentário, enviado por qualquer usuário ao artigo publicado.

4.2.3 Venue

Uma nova instância do *smart contract Venue*, figura 31, é criada toda vez que um local de publicação é adicionado por alguma editora. Ele é responsável por armazenar os endereços dos seus editores (*editors*), revisores (*reviewers*) e artigos submetidos (*submitted_papers*), publicados (*published_papers*), rejeitados (*rejected_papers*) e revogados (*revoked_papers*). A variável *owner* armazena o endereço da editora que cadastrou a *venue* na plataforma, com a taxa de revisão definida em *fee*. Assim como as suas informações, como qual o tipo de *venue* (*venue_type*), seu nome (*name*), o número de revisores para cada artigo (*number_rev*), o prazo para enviar uma revisão

```
276     function add_review(address _sender, string calldata _hash) external
277     {
278         require(reviewers_map[_sender] || (venue.get_open_participation()), "You are not a Reviewer");
279         require(status == 3 || status == 4, "This Paper is not under revision");
280         require(block.timestamp <= timestamp_accept[_sender] + venue.get_deadline_review(), "Deadline Exceeded");
281 
282         if(venue.get_open_participation())
283         {
284             require(_sender != author_addr && _sender != editor);
285             open_participation_reviews.push(_hash);
286             open_participation_reviewers.push(_sender);
287             open_participation_reviewers_map[_sender] = true;
288         }
289         else if(status == 3)
290             first_reviews.push(_hash);
291         else if(status == 4)
292             second_reviews.push(_hash);
293     }
294 
295     function add_reviewer_open_participation(address _sender, address _rev) external
296     {
297         require(_sender == editor, "Only Editor");
298         require(open_participation_reviewers_map[_rev]);
299 
300         reviewers.push(_rev);
301         reviewers_map[_rev] = true;
302     }
```

Figura 29 – Trecho de código 4b *Paper*. Fonte: Elaborado pelo autor.

```
196     function add_new_version(string memory _new_version) external
197     {
198         require(status == 4 || status == 5 || (venue.get_open_participation() && status == 3));
199         paper.ipfs_hash.push(_new_version);
200     }
201 
202     function transfer_tokens(address _editor) external
203     {
204         require(editor == _editor);
205         require(status == 6 || status == 7);
206 
207         for(uint8 i = 0; i < reviewers.length; i++)
208         {
209             token.mint(reviewers[i], _editor, 1 ether);
210         }
211 
212         token.mint(_editor, _editor, 1 ether);
213         require(token.transfer(address(venue), token.balanceOf(address(this))), "Error in transfer for Venue");
214     }
215 
216     function comment(string calldata _comment, string calldata _name) external
217     {
218         require(venue.get_open_fv_commenting(), "This Venue don't allows commenting");
219         require(status == 6, "This Paper isn't published");
220 
221         comment_data memory c;
222         c.comment = _comment;
223         c.name = _name;
224         comments.push(c);
225     }
```

Figura 30 – Trecho de código 5 *Paper*. Fonte: Elaborado pelo autor.

(*deadline_review*) e a variável que indica se a submissão está habilitada (*submission*) são armazenadas. Uma *struct* chamada de *review_type* foi criada para definir os tipos de revisão configuráveis na plataforma, ela possui os seguintes campos do

```
21  contract Venue
22  {
23      address private owner;
24
25      PubToken private token;
26
27      address[] private submited_papers;
28      address[] private published_papers;
29      address[] private revoked_papers;
30      address[] private rejected_papers;
31      address[] private reviewers;
32      address[] private editors;
33
34      uint256 fee;
35
36      review_type private review;
37      uint8 private venue_type;
38      uint8 private number_rev;
39      uint private deadline_review;
40      string private name;
41      bool private submission;
```

Figura 31 – Trecho de código 1 *Venue*. Fonte: Elaborado pelo autor.

tipo *bool*: *open_identities*, *open_participation* e *open_fv_commenting*. Os campos de *review_type* permitem habilitar, quando atribuído valor *true*, ou desabilitar, quando atribuído *false*, as revisões *Open Identities*, *Open Participation* e habilita comentários no artigo publicado, respectivamente.

```
43      mapping(address => bool) private submited_papers_map;
44      mapping(address => bool) private published_papers_map;
45      mapping(address => bool) private rejected_papers_map;
46      mapping(address => bool) private revoked_papers_map;
47
48      mapping(address => bool) private reviewers_map;
49      mapping(address => bool) private editors_map;
50      mapping(address => data) private reviewers_data;
51      mapping(address => data) private editors_data;
```

Figura 32 – Trecho de código 2 *Venue*. Fonte: Elaborado pelo autor.

Estruturas do tipo *mapping* (figura 32) também são utilizadas para acelerar a verificação de informações. Todas as chaves são endereços de revisores, editores ou artigos, como mostrado a seguir:

- *submitted_papers_map*, faz o mapeamento de um endereço para um valor do tipo *bool*, que indica se o artigo foi submetido à venue;
- *published_papers_map*, também faz o mapeamento do endereço de um artigo para um *bool*, que indica se o artigo foi publicado;

- *rejected_papers_map*, semelhante aos casos anteriores, porém indica se o artigo foi rejeitado;
- *revoked_papers_map*, semelhante aos casos anteriores, porém indica se o artigo foi revogado;
- *reviewers_map*, faz o mapeamento de um endereço para um *bool*, que indica se o endereço é de um revisor cadastrado na *venue*;
- *editors_map*, semelhante ao caso anterior, mas indica se o endereço é de um editor cadastrado;
- *reviewers_data*, faz o mapeamento de um endereço para uma *struct data*, que armazena nome e orcid dos revisores;
- *editors_data*, igual ao caso anterior, porém para os editores.

As funções presentes em *Venue* manipulam ou fornecem acesso às suas próprias variáveis, não interagem com outros *smart contracts*. Em contrapartida elas são executadas através de outros *samrt contracts*, ou seja, a interface de usuário não tem acesso direto à elas.

```
51      constructor(address _owner, string memory _name, uint8 _type, int _deadline_review,
52                  uint8 _number_rev, address _token, uint256 _fee)
53      {
54          owner = _owner;
55          name = _name;
56          venue_type = _type;
57          deadline_review = uint(_deadline_review);
58          number_rev = _number_rev;
59          token = PubToken(_token);
60          submission = true;
61          fee = _fee;
62      }
```

Figura 33 – Trecho de código 3 *Venue*. Fonte: Elaborado pelo autor.

O construtor (figura 33) recebe os seguintes parâmetros: o endereço da editora que cadastrou a *venue* (*_owner*), o nome da *venue* (*_name*), o tipo (*_type*), o prazo que cada revisor tem para enviar seu relatório de revisão (*_deadline_review*), o número de revisores por artigo (*_number_rev*), o endereço do *PubToken* (*_token*) e o valor da taxa de publicação (*_fee*). Todos os valores são armazenados internamente.

As funções *add_reviewer* e *add_reviewer_blind* (figura 34) são responsáveis por adicionar revisores à *venue*. A a primeira armazena as informações dos revisores, endereço (*_rev*), nome (*_name*) e orcid (*_orcid*) para *venues* que adotem *Open Identities*. Enquanto a segunda adiciona somente o endereço do revisor, utilizada em

```
82      function add_reviewer(address _rev, address _editor, string calldata _name, string calldata _orcid) external
83      {
84          require(editors_map[_editor], "You are not a Editor of this Venue");
85          require(review.open_identities, "You can't add a reviewer to this Venue");
86          require(!review.open_participation);
87          require(!reviewers_map[_rev]);
88
89          reviewers.push(_rev);
90          reviewers_map[_rev] = true;
91
92          reviewers_data[_rev].name = _name;
93          reviewers_data[_rev].orcid = _orcid;
94      }
95
96      function add_reviewer_blind(address _rev, address _editor) external
97      {
98          require(editors_map[_editor], "You are not a Editor of this Venue");
99          require(!review.open_identities, "You can't add a reviewer to this Venue");
100         require(!review.open_participation);
101         require(!reviewers_map[_rev]);
102
103         reviewers.push(_rev);
104         reviewers_map[_rev] = true;
105     }
```

Figura 34 – Trecho de código 4 *Venue*. Fonte: Elaborado pelo autor.

```
64      function submit_paper(address _paper) external
65      {
66          submited_papers.push(_paper);
67          submited_papers_map[_paper] = true;
68      }
69
70      function publish_paper(address _paper) external
71      {
72          published_papers.push(_paper);
73          published_papers_map[_paper] = true;
74      }
75
76      function revoke_paper(address _paper) external
77      {
78          revoked_papers.push(_paper);
79          revoked_papers_map[_paper] = true;
80      }
81
82      function open_submission(address _owner) external
83      {
84          require(_owner == owner, "You are not the Owner of this Venue");
85          submission = true;
86      }
87
88      function close_submission(address _owner) external
89      {
90          require(_owner == owner, "You are not the Owner of this Venue");
91          submission = false;
92      }
```

Figura 35 – Trecho de código 5 *Venue*. Fonte: Elaborado pelo autor.

venues que não adotam *Open Identities*. Ambas funções tem sua execução restrita a algum editor já cadastrado. De maneira similar, a função *add_editor* adiciona à *venue* um novo editor e suas informações, nome (*_name*) e orcid (*_orcid*), somente a editora pode executá-la.

As funções *submit_paper*, *publish_paper* e *revoke_paper* (figura 35) são responsáveis por adicionar o endereço do artigo passado por parâmetro (*_paper*) aos vetores e *mappings* correspondentes. Enquanto *open_submission* e *close_submission* permitem que a editora libere ou encerre a submissão de novos artigos.

```
147     function set_open_identities_true(address _publisher) external
148     {
149         require(_publisher == owner, "This is not the Publisher of this Venue");
150         review.open_identities = true;
151     }
152
153     function set_open_identities_false(address _publisher) external
154     {
155         require(_publisher == owner, "This is not the Publisher of this Venue");
156         review.open_identities = false;
157     }
158
159     function set_open_participation_true(address _publisher) external
160     {
161         require(_publisher == owner, "This is not the Publisher of this Venue");
162         review.open_participation = true;
163     }
164
165     function set_open_participation_false(address _publisher) external
166     {
167         require(_publisher == owner, "This is not the Publisher of this Venue");
168         review.open_participation = false;
169     }
```

Figura 36 – Trecho de código 6 *Venue*. Fonte: Elaborado pelo autor.

```
171     function set_open_fv_commenting_true(address _publisher) external
172     {
173         require(_publisher == owner, "This is not the Publisher of this Venue");
174         review.open_fv_commenting = true;
175     }
176
177     function set_open_fv_commenting_false(address _publisher) external
178     {
179         require(_publisher == owner, "This is not the Publisher of this Venue");
180         review.open_fv_commenting = false;
181     }
```

Figura 37 – Trecho de código 7 *Venue*. Fonte: Elaborado pelo autor.

Existem também as funções que configuram os tipos de revisão adotadas pela

venue (figuras 36 e 37): *set_open_identities_true*, *set_open_identities_false*, *set_open_participation_true*, *set_open_participation_false*, *set_open_fv_commenting_true* e *set_open_fv_commenting_false*. Todas são executadas somente pela editora.

4.2.4 Submission

O *smart contract Submission* (figura 38) é implantado pelo administrador do sistema, ele interage com a interface de usuário e permite o acesso às funcionalidades referentes à submissão de artigos presentes em *Venue* e *Paper*. Ele armazena uma lista com as *venues* cadastradas e uma instância de *PubToken* (*token*). O construtor recebe o endereço da instância do *PubToken* e armazena internamente.

```
11  contract Submission
12  {
13      address[] public venues;
14
15      PubToken private token;
16
17      event submited(address venue, address editor, address paper, uint timestamp);
18      event invite_editor(address editor, address paper, uint timestamp);
19      event improvement_submited(address paper, address reviewer, uint timestamp);
20      event commented(address paper, address comment_author, uint timestamp);
21      event authors_data_submited(address paper, address editor, uint timestamp);
22      event final_version_submited(address paper, address editor, uint timestamp);
23
24      constructor(address _token)
25      {
26          token = PubToken(_token);
27      }
```

Figura 38 – Trecho de código 1 *Submission*. Fonte: Elaborado pelo autor.

Eventos são utilizados para comunicar a interface de usuário que determinadas ações foram executadas no *blockchain*, eles são apresentados a seguir:

- *submitted*, indica que um novo artigo foi submetido, informando o endereço da *venue* à qual foi submetido, o endereço do editor responsável (*editor)*, o endereço do *Paper* e o *timestamp* indicando o momento em que o artigo foi submetido;
- *invite_editor*, informa qual editor foi escolhido para o artigo, através do endereço do editor (*editor*), do endereço do artigo (*paper*) e do *timestamp* do momento em que a ação foi executada;
- *improvement_submitted*, informa os revisores do artigo que uma nova versão foi submetida através do endereço do artigo (*paper*), o endereço do revisor (*reviewer*) e o *timestamp* do momento em que a ação foi executada;
- *commented*, indica que um comentário sobre o artigo foi enviado, informando o endereço do artigo (*paper*), o endereço do autor do comentário (*comment_author*), e o *timestamp* de quando o comentário foi enviado;

- *authors_data_submitted*, indica que os autores submeteram seus dados, informando o endereço do artigo (*paper*), o endereço do autor do comentário (*comment_author*), e o *timestamp* de quando o comentário foi enviado;
- *final_version_submitted*, indica que a versão final do artigo foi submetida, informando o endereço do artigo (*paper*), o endereço do autor do comentário (*comment_author*), e o *timestamp* de quando o comentário foi enviado.

```
83      function submit(string calldata _paper_ipfs, uint _venue_id, uint256 _amount) external
84      {
85          Venue v = Venue(venues[_venue_id]);
86
87          require(v.get_submission(), "Submission Closed");
88          require(_amount == v.get_fee(), "Wrong PubToken amount");
89
90          address editor = (v.get_editor(0));
91          Paper paper = new Paper(_paper_ipfs, msg.sender, venues[_venue_id], editor, address(token));
92          v.submit_paper(address(paper));
93
94          require(token.transferFrom(msg.sender, address(paper), _amount), "PubToken transfer failed");
95
96          emit invite_editor(editor, address(paper), block.timestamp);
97          emit submited(venues[_venue_id], editor, address(paper), block.timestamp);
98      }
```

Figura 39 – Trecho de código 2 *Submission*. Fonte: Elaborado pelo autor.

A função *submit* (figura 39) é responsável por realizar a submissão de um novo artigo à plataforma. Ela recebe por parâmetro o *CID* do arquivo no *IPFS* (*_paper_ipfs*), o identificador da *venue* (*_venue_id*) e o valor da taxa de revisão que está sendo enviado (*_amount*). Ela verifica se a submissão à *venue* está habilitada e se a quantidade de *PubTokens* enviados corresponde à taxa de revisão. Para submeter o artigo uma nova instância do *smart contract Paper* é criada e as informações necessárias são enviadas por parâmetro. Em seguida o endereço do *Paper* recém criado é adicionado à lista de artigos submetidos da *venue* em questão. A taxa de revisão é transferida do autor para o *Paper* e os eventos *invite_editor* e *submitted* são emitidos. O editor deve ser escolhido aleatoriamente, mas na implementação do protótipo, como a quantidade de endereços disponíveis para teste era limitada e para simplificar esta etapa utilizou-se somente um editor.

As funções *add_paper_information* e *add_authors_information* (figura 40) são responsáveis por adicionar as informações sobre o artigo e sobre os autores, respectivamente, ao *Paper*. Elas simplesmente recebem titulo (*_title*), palavras-chave (*_keywords*), nomes (*_names*) e orcid (*_orcid*), e enviam estas informações para as funções correspondentes no *Paper*.

Uma nova versão do artigo pode ser submetida somente pelo primeiro autor por meio da função *submit_new_version* (figura 41), ela recebe o *CID* do arquivo

```
100     function add_paper_information(address _paper, string calldata _title, string[] calldata _keywords) external
101     {
102         Paper p = Paper(_paper);
103 
104         p.add_paper_information(msg.sender, _title, _keywords);
105     }
106 
107     function add_authors_information(address _paper, string[] calldata _names, string[] calldata _orcid) external
108     {
109         Paper p = Paper(_paper);
110         Venue v = Venue(p.get_venue());
111 
112         p.add_authors_information(msg.sender, _names, _orcid);
113 
114         if(!v.get_open_identities())
115             emit authors_data_submited(_paper, p.get_editor(), block.timestamp);
116     }
```

Figura 40 – Trecho de código 3 *Submission*. Fonte: Elaborado pelo autor.

```
118     function submit_new_version(address _paper, string calldata _paper_ipfs) external
119     {
120         Paper p = Paper(_paper);
121         require(p.is_author(msg.sender), "You are not the Author of this Paper");
122 
123         address[] memory reviewers;
124         p.add_new_version(_paper_ipfs);
125 
126         if(p.get_paper_status() == 3 || p.get_paper_status() == 4)
127         {
128             reviewers = p.get_reviewers();
129             for(uint i = 0; i < reviewers.length; i++)
130             {
131                 emit improvement_submited(_paper, reviewers[i], block.timestamp);
132             }
133         }
134 
135         if(p.get_paper_status() == 5)
136             emit final_version_submited(_paper, p.get_editor(), block.timestamp);
137     }
138 
139     function comment_paper(address _paper, string calldata _comment, string calldata _name) external
140     {
141         Paper p = Paper(_paper);
142         p.comment(_comment, _name);
143 
144         emit commented(_paper, msg.sender, block.timestamp);
145     }
```

Figura 41 – Trecho de código 4 *Submission*. Fonte: Elaborado pelo autor.

no *IPFS* e o adiciona ao *Paper*. Caso o *status* do *Paper* seja 4 (Em revisão - 2ª rodada) ou a revisão seja *Open Participation*, então ele está em revisão, e o evento *improvement_submitted* é emitido para o revisor do artigo informando que uma nova versão foi submetida. Se o *status* for 5, então a revisão do artigo foi finalizada e a versão final foi requerida, assim o evento *final_version_submitted* é emitido. Enquanto isso, a função *comment_paper* recebe o *CID* do *IPFS* e o nome do autor do comentário, os registra no *Paper* e emite o evento *commented* informando que o comentário foi

enviado.

4.2.5 Editorial Board

```
8   contract EditorialBoard
9   {
10      address[] private venues;
11
12      event reviewer_invited(address reviewer, address paper, uint timestamp);
13      event published(address paper, address author, address venue, uint timestamp);
14      event project_accepted(address paper, address author, address venue, uint timestamp);
15      event rejected(address paper, address author, address venue, uint timestamp);
16      event improve(address paper, address author, address venue, uint timestamp);
17      event revoked(address paper, address author, address venue, uint timestamp);
18      event ready_to_publish(address paper, address author, uint timestamp);
19      event review_started(address paper, uint timestamp);
20
```

Figura 42 – Trecho de código 1 *EditorialBoard*. Fonte: Elaborado pelo autor.

O *smart contract EditorialBoard* também é implantado pelo administrador do sistema e interage com a interface de usuário, permitindo acesso às funcionalidades dos editores. Ele armazena uma lista com as *venues* cadastradas. Todas as suas funções são executadas somente pelo editor designado para o artigo e executam as funções necessárias em *Paper* e *Venue*.

Eventos são utilizados para comunicar a interface de usuário que determinadas ações foram executadas no *blockchain*, eles são apresentados a seguir:

- *reviewer_invited*, indica que um revisor foi convidado à revisar o artigo, informando o endereço do revisor (*reviewer*), o endereço do *Paper* e o *timestamp* indicando o momento em que o revisor foi convidado;
- *published*, indica que o artigo foi publicado, informando os endereços do *Paper*, do autor (*author*) e da *venue* à qual o artigo foi publicado, além do *timestamp* do momento em que a ação foi executada;
- *project_accepted*, indica que o projeto foi aceito pelo editor, informando os endereços do *Paper*, do autor (*author*) e da *venue* à qual o artigo foi publicado, além do *timestamp* do momento em que a ação foi executada;
- *rejected*, indica que o artigo foi rejeitado, informando os endereços do *Paper*, do autor (*author*) e da *venue* à qual o artigo foi publicado, além do *timestamp* do momento em que a ação foi executada;
- *improve*, indica que que melhorias no artigo, baseadas nos relatórios de revisão, foram solicitadas pelo editor, informando os endereços do *Paper*, do autor

(*author*) e da *venue* à qual o artigo foi publicado, além do *timestamp* do momento em que a ação foi executada;

- *revoked*, indica que um artigo foi revogado, informando os endereços do *Paper*, do autor (*author*) e da *venue* à qual o artigo foi publicado, além do *timestamp* do momento em que a ação foi executada;
- *ready_to_publish*, indica que um artigo está pronto para ser publicado, a versão final é aguardada e, se a revisão não for *Open Identities*, as informações de identificação dos autores também são solicitadas. O evento informa os endereços do *Paper* e do autor (*author*), além do *timestamp* do momento em que a ação foi executada;
- *review_started*, indica que a revisão foi iniciada, somente para revisões *Open Participation*, informando o endereço do *Paper* e o *timestamp* em que ela foi iniciada.

```
21    function invite_reviewers(uint[] calldata _reviewers_id, address _paper) external
22    {
23        Paper p = Paper(_paper);
24        Venue v = Venue(p.get_venue());
25
26        require(p.is_editor(msg.sender), "Err1");
27        require(!v.get_open_participation(), "Err5");
28
29        address rev;
30
31        p.update_paper_status(msg.sender, 3);
32        for(uint i = 0; i < _reviewers_id.length; i++)
33        {
34            rev = v.get_reviewer(_reviewers_id[i]);
35            p.add_reviewer_invited(msg.sender, rev);
36
37            emit reviewer_invited(rev, _paper, v.get_deadline_review() + block.timestamp);
38        }
39    }
40
41    function start_review(address _paper) external
42    {
43        Paper p = Paper(_paper);
44        Venue v = Venue(p.get_venue());
45
46        require(v.get_open_participation(), "Only for Open Participation Venues");
47        require(p.is_editor(msg.sender), "Only Editor");
48
49        p.update_paper_status(msg.sender, 3);
50        emit review_started(_paper, block.timestamp);
51    }
```

Figura 43 – Trecho de código 2 *EditorialBoard*. Fonte: Elaborado pelo autor.

A função *invite_reviewers* (figura 43) é responsável por convidar os revisores para o artigo na revisão tradicional. Ela recebe por parâmetro um vetor com os índices

que identificam os revisores escolhidos (*_reviewers_id*) e o endereço do *Paper*. O *status* do *Paper* é atualizado para 3 (Em revisão - 1ª Rodada) e cada revisor é adicionado aos revisores convidados do artigo. Em seguida o evento *reviewer_invited* é emitido para cada um. Enquanto isso, a função *start_review* é a responsável, na revisão *Open Participation*, por iniciar o processo de revisão. Ela atualiza o seu *status* para 3 também e emite o evento *review_started*, informando que a revisão está iniciada e qualquer usuário da plataforma pode atuar como revisor, exceto o editor designado e o autor do artigo.

```
92      function finish_review(address _paper) external
93      {
94          Paper p = Paper(_paper);
95
96          require(p.is_editor(msg.sender));
97
98          uint8 status = p.get_paper_status();
99          require(status == 3 || status == 4);
100
101         p.update_paper_status(msg.sender, 5);
102         emit ready_to_publish(_paper, p.get_author(msg.sender), block.timestamp);
103     }
104
105     function improve_paper(address _paper) external
106     {
107         Paper p = Paper(_paper);
108         Venue v = Venue(p.get_venue());
109
110         require(p.is_editor(msg.sender));
111         require(v.get_venue_type() == 1 || v.get_venue_type() == 2);
112         require(p.get_paper_status() == 3);
113
114         p.update_paper_status(msg.sender, 4);
115
116         emit improve(_paper, p.get_author(msg.sender), p.get_venue(), block.timestamp);
117     }
```

Figura 44 – Trecho de código 3 *EditorialBoard*. Fonte: Elaborado pelo autor.

A função *finish_review* (figura 44) é responsável por informar o autor do artigo que a revisão foi finalizada. Portanto, a versão final do artigo ou as informações dos autores, caso a revisão não seja *Open Identities*, devem ser submetidas. Sua execução é restrita ao editor do artigo. Ela atualiza o *status* do artigo para 5 (Aguardando versão final) e emite o evento *ready_to_publish*, para informar o autor que ele está pronto para ser publicado. Enquanto isso, *improve_paper* informa o autor que melhorias são necessárias no artigo e inicia a segunda rodada de revisões, emitindo o evento *improve*. Ela é executada somente para *venues* do tipo *Journal* e *Conference* quando o artigo está na primeira rodada de revisões.

O artigo é publicado por meio da função *publish_paper* (figura 45), utilizada somente para *Journal* e *Conference*. Ela atualiza o *status* do *Paper* para 6 (Publi-

```
63      function publish_paper(address _paper) external
64      {
65          Paper p = Paper(_paper);
66          require(p.is_editor(msg.sender));
67          Venue v = Venue(p.get_venue());
68          require((v.get_venue_type() == 1) || (v.get_venue_type() == 2), "Only for Journals or Conferences");
69          uint8 status = p.get_paper_status();
70          p.update_paper_status(msg.sender, 6);
71          v.publish_paper(_paper);
72          p.transfer_tokens(msg.sender);
73          if(v.get_open_identities())
74          {
75              require(status == 3 || status == 4 || status == 5);
76              emit published(_paper, p.get_author(msg.sender), p.get_venue(), block.timestamp);
77          }
78          else
79          {
80              require(status == 5);
81              emit published(_paper, p.get_author(msg.sender), p.get_venue(), block.timestamp);
82          }
83      }
84
85      function accept_project(address payable _paper) external
86      {
87          Paper p = Paper(_paper);
88          Venue v = Venue(p.get_venue());
89          require(p.get_editor() == msg.sender, "Err1");
90          p.update_paper_status(msg.sender, 6);
91          v.publish_paper(_paper);
92          p.transfer_tokens(msg.sender);
93          emit project_accepted(_paper, p.get_author(msg.sender), p.get_venue(), block.timestamp);
94      }
```

Figura 45 – Trecho de código 4 *EditorialBoard*. Fonte: Elaborado pelo autor.

cado/Aceito), inclui o seu endereço na lista de artigos publicados em *Venue* e emite o evento *published*. Ela também gera *PubTokens* e os transfere aos revisores e ao editor, além de transferir a taxa de publicação do *Paper* para a *Venue*. De maneira similar, a função *accept_project* aceita projetos de pesquisa, porém para *venues* do tipo *Research Project Call*, emitindo o evento *project_accepted* ao final da execução.

A função *reject_paper* é responsável por rejeitar o artigo em qualquer etapa da revisão. Se ele for rejeitado antes de iniciar a revisão, acontece o chamado *Desk Reject* e o seu *status* é atualizado para 2. Quando a rejeição acontece em qualquer outro ponto do processo de revisão, o *status* é atualizado para 7 (Rejeitado) e os *PubTokens* são gerados e enviados para o editor e os revisores. Ao final os *PubTokens* armazenados no *Paper* são transferidos de volta para o autor e o evento *rejected* é emitido. Enquanto isso, a função *revoke_paper* é responsável por revogar o artigo. Isso acontece quando, após a publicação, ela deve ser cancelada por qualquer motivo. Ela atualiza o *status* para 8 (Revogado) e emite o evento *revoked*.

As funções *add_reviewer* e *add_reviewer_blind* (figura 47) são responsáveis por adicionar novos revisores às *venues*. A primeira é aplicada à *venues* que utilizam revisão *Open Identities*, pois adiciona informações referentes às identidades, nome (*_name*) e orcid (*_orcid*). Enquanto a segunda aplica-se para revisão não *Open Identities*, pois não é adicionada nenhuma identificação dos revisores, somente o seu

```
125     function reject_paper(address _paper) external
126     {
127         Paper p = Paper(_paper);
128         require(p.is_editor(msg.sender));
129         Venue v = Venue(p.get_venue());
130 
131         if(p.get_paper_status() == 1)
132             p.update_paper_status(msg.sender, 2);
133         else
134         {
135             p.update_paper_status(msg.sender, 7);
136             p.transfer_tokens(msg.sender);
137         }
138 
139         p.return_tokens();
140         v.reject_paper(_paper);
141 
142         emit rejected(_paper, p.get_author(msg.sender), address(v), block.timestamp);
143     }
144 
145     function revoke_paper(address _paper) external
146     {
147         Paper p = Paper(_paper);
148         require(p.get_editor() == msg.sender);
149         require(p.get_paper_status() == 6);
150 
151         Venue v = Venue(p.get_venue());
152         p.update_paper_status(msg.sender, 8);
153         v.revoke_paper(_paper);
154 
155         emit revoked(_paper, p.get_author(msg.sender), p.get_venue(), block.timestamp);
156     }
```

Figura 46 – Trecho de código 5 *EditorialBoard*. Fonte: Elaborado pelo autor.

```
175     function add_reviewer(uint _id_venue, address _rev, string calldata _name, string calldata _orcid) external
176     {
177         Venue v = Venue(venues[_id_venue]);
178 
179         v.add_reviewer(_rev, msg.sender, _name, _orcid);
180     }
181 
182     function add_reviewer_blind(int _id_venue, address _rev) external
183     {
184         Venue v = Venue(venues[uint(_id_venue)]);
185 
186         v.add_reviewer_blind(_rev, msg.sender);
187     }
```

Figura 47 – Trecho de código 6 *EditorialBoard*. Fonte: Elaborado pelo autor.

endereço.

A função *set_reviewer_open_participation* (figura 48) permite que o editor determine, na revisão *Open Participation*, quais revisores contribuíram durante o processo. Esta abordagem busca evitar comportamento mau intencionado, como alguém simplesmente enviar um texto sem relação com o artigo, somente para receber *PubTokens* ao final do processo.

```
164     function set_reviewer_open_participation(address _paper, address _rev) external
165     {
166         Paper p = Paper(_paper);
167         Venue v = Venue(p.get_venue());
168 
169         require(v.get_open_participation());
170 
171         p.add_reviewer_open_participation(msg.sender, _rev);
172     }
```

Figura 48 – Trecho de código 7 *EditorialBoard*. Fonte: Elaborado pelo autor.

As funções do tipo *get* fornecem acesso às informações contidas em *Paper*, disponibilizando-as para a interface de usuário.

4.2.6 Review

```
7   contract Review
8   {
9       event reviewer_accepted(address editor, address reviewer, address paper, uint timestamp);
10      event reviewer_declined(address editor, address reviewer, address paper, uint timestamp);
11      event submited_review(address editor, address reviewer, address paper, uint timestamp);
```

Figura 49 – Trecho de código 1 *Review*. Fonte: Elaborado pelo autor

O *smart contract Review* também é implantado pelo administrador do sistema e interage com a interface de usuário, permitindo acesso às funcionalidades dos revisores.

Eventos são utilizados (figura 49) para comunicar a interface de usuário que determinadas ações foram executadas no *blockchain*, eles são apresentados a seguir:

- *reviewer_accepted*, indica que um revisor aceitou revisar o artigo, informando os endereços do revisor (*reviewer*), do editor e do *Paper*, assim como o *timestamp* do momento em que o revisor aceitou;
- *reviewer_declined*, indica que o revisor recusou revisar o artigo, informando os endereços do editor, do revisor (*reviewer*) e do *Paper*, além do *timestamp* do momento em que a ação foi executada;
- *submitted_review*, indica que o revisor submeteu seu relatório de revisão, informando os endereços do editor, do revisor (*reviewer*) e do *Paper*, além do *timestamp* do momento em que a ação foi executada.

A função *submit_review* (figura 50) recebe por parâmetro o *CID* do relatório de revisão no *IPFS* e o adiciona ao *Paper*. Ela não permite que o autor e o edi-

```
20      function submit_review(address _paper, string calldata _review) external
21      {
22          Paper p = Paper(_paper);
23
24          require(!p.is_author(msg.sender));
25          require(!p.is_editor(msg.sender));
26
27          p.add_review(msg.sender, _review);
28          emit submited_review(p.get_editor(), msg.sender, _paper, block.timestamp);
29      }
30
31      function open_participation_name(address _paper, string calldata _name) external
32      {
33          Paper p = Paper(_paper);
34
35          p.add_name_open_participation(msg.sender, _name);
36      }
```

Figura 50 – Trecho de código 2 *Review*. Fonte: Elaborado pelo autor

tor enviem revisões. Ao final ela emite o evento *submitted_review*. Enquanto isso, *open_participation_name* é responsável por adicionar o nome do revisor, quando a revisão é do tipo *Open Participation* e *Open Identities* ao mesmo tempo.

```
38      function reviewer_accept(address _paper) external
39      {
40          Paper p = Paper(_paper);
41
42          p.reviewer_accept(msg.sender);
43          emit reviewer_accepted(p.get_editor(), msg.sender, _paper, block.timestamp);
44      }
45
46      function reviewer_decline(address _paper) external
47      {
48          Paper p = Paper(_paper);
49
50          p.reviewer_decline(msg.sender);
51          emit reviewer_declined(p.get_editor(), msg.sender, _paper, block.timestamp);
52      }
```

Figura 51 – Trecho de código 3 *Review*. Fonte: Elaborado pelo autor

As funções *reviewer_accept* e *reviewer_decline* (figura 51) permitem que um revisor aceite ou recuse participar da revisão do artigo, elas emitem os eventos *reviewer_accepted* e *reviewer_declined*, respectivamente. A função *get_reviews* permite o acesso aos relatórios de revisão presentes no *Paper*.

```
10  contract Publisher is Ownable
11  {
12      address[] public publishers;
13      address[] private venues;
14
15      mapping(address => bool) private publishers_map;
16      mapping(address => string) private publisher_name;
17
18      event venue_created(uint id, address venue, string name, uint fee);
19
```

Figura 52 – Trecho de código 1 *Publisher*. Fonte: Elaborado pelo autor.

4.2.7 Publisher

O *smart contract Publisher* (figura 52) também é implantado pelo administrador do sistema e interage com a interface de usuário, permitindo acesso às funcionalidades das editoras. Ele armazena uma lista com os endereços das editoras e uma lista com os endereços das *venues*. Duas estruturas do tipo *mapping* são utilizadas: *publishers_map* cria o mapeamento entre um endereço e um valor *bool*, que indica se o endereço é de uma editora cadastrada e; *publisher_name* que mapeia o endereço da editora para uma *string* que armazena o seu nome. O evento *venue_created* é utilizado para informar a plataforma quando uma nova *venue* é adicionada, os dados informados são: o índice da *venue* (*id*), o endereço da *venue*, o nome (*name*) e a taxa de publicação (*fee*).

A função *add_publisher* (figura 53) é responsável por adicionar uma nova editora, executada pelo administrador da plataforma. Ela recebe o endereço (_address) e o nome da editora (_name) e os armazena nas variáveis do *Publisher*. Enquanto isso, *add_venue* adiciona uma nova *venue* à plataforma. Uma nova instância do *smart contract Venue* é criada e as seguintes informações são adicionadas: nome (*_name*), tipo (*_type*), o prazo que os revisores têm para enviar os relatórios de revisão (*_deadline_review*), a quantidade de revisores para cada artigo (*_number_rev*), o endereço do *PubToken* (*_token*) e o valor da taxa de publicação (*_fee*). Ao final da execução o evento *venue_created* é emitido. De maneira similar, *add_editor* adiciona um novo editor a uma *venue*, acrescentando o nome do editor (*_name*), orcid e o seu endereço (*_editor*). Enquanto *get_venues* fornece acesso à lista de *venues* cadastradas.

As funções *open_submission* e *close_submission* (figuras 54 e 55) permitem habilitar e desabilitar a submissão de artigos a qualquer *venue*. De maneira similar, *set_open_identities_true*, *set_open_identities_false*, *set_open_participation_true*, *set_open_participation_false*, *set_open_fv_commenting_true* e *set_open_fv_commenting_false* permitem habilitar ou desabilitar os tipos diferentes de revisão aberta supor-

```
20    function add_publisher(string calldata _name, address _address) external onlyOwner
21    {
22        publishers.push(_address);
23        publisher_name[_address] = _name;
24        publishers_map[_address] = true;
25    }
26
27    function add_venue(string calldata _name, uint8 _type, int _deadline_review,
28                       uint8 _number_rev, address _token, uint256 _fee) external
29    {
30        require(publishers_map[msg.sender], "That Publisher is not registered");
31
32        Venue v = new Venue(msg.sender, _name, _type, _deadline_review, _number_rev, _token, _fee);
33
34        venues.push(address(v));
35
36        uint fee = _fee/1 ether;
37        uint id = venues.length-1;
38        emit venue_created(id, address(v), _name, fee);
39    }
40
41    function add_editor(uint _id_venue, string calldata _name, string calldata _orcid, address _editor) external
42    {
43        Venue v = Venue(venues[_id_venue]);
44
45        v.add_editor(_editor, _name, _orcid, msg.sender);
46    }
```

Figura 53 – Trecho de código 2 *Publisher*. Fonte: Elaborado pelo autor.

```
55    function open_submission(uint _id_venue) external
56    {
57        Venue v = Venue(venues[_id_venue]);
58
59        v.open_submission(msg.sender);
60    }
61
62    function close_submission(uint _id_venue) external
63    {
64        Venue v = Venue(venues[_id_venue]);
65
66        v.close_submission(msg.sender);
67    }
68
69    function set_open_identities_true(int _id_venue) external
70    {
71        Venue v = Venue(venues[uint(_id_venue)]);
72        v.set_open_identities_true(msg.sender);
73    }
74
75    function set_open_identities_false(int _id_venue) external
76    {
77        Venue v = Venue(venues[uint(_id_venue)]);
78        v.set_open_identities_false(msg.sender);
79    }
```

Figura 54 – Trecho de código 3 *Publisher*. Fonte: Elaborado pelo autor.

tados pela plataforma. Enquanto isso *set_deadline_review* permite alterar o prazo de entrega dos relatórios de revisão. Todas estas funções fornecem para a interface de

```
81      function set_open_participation_true(int _id_venue) external
82      {
83          Venue v = Venue(venues[uint(_id_venue)]);
84          v.set_open_participation_true(msg.sender);
85      }
86
87      function set_open_participation_false(int _id_venue) external
88      {
89          Venue v = Venue(venues[uint(_id_venue)]);
90          v.set_open_participation_false(msg.sender);
91      }
92
93      function set_open_fv_commenting_true(int _id_venue) external
94      {
95          Venue v = Venue(venues[uint(_id_venue)]);
96          v.set_open_fv_commenting_true(msg.sender);
97      }
98
99      function set_open_fv_commenting_false(int _id_venue) external
100     {
101         Venue v = Venue(venues[uint(_id_venue)]);
102         v.set_open_fv_commenting_false(msg.sender);
103     }
```

Figura 55 – Trecho de código 4 *Publisher*. Fonte: Elaborado pelo autor.

usuário acesso às funções correspondentes no *smart contract Venue*.

4.2.8 Tokens

A plataforma possui dois tipos de *tokens*, o *PubToken*, desenvolvido baseado no padrão ERC-20, e *PaperPublished*, baseado no padrão ERC-721. Ambos são implementados utilizando a biblioteca OpenZeppelin[2], que oferece uma implementação segura e eficiente. Contudo, algumas alterações foram feitas para adaptar o seu funcionamento às necessidades da plataforma.

O *PubToken* (figura 56) utiliza um *mapping* aninhado para garantir que possam ser cunhados novos *tokens* somente uma vez para cada revisor e editor do artigo. Ele faz o mapeamento do endereço do *Paper* para outro *mapping* que associa o endereço dos participantes a um valor do tipo *bool*, este valor indica se o participante já recebeu *tokens*. Trechos de código foram adicionados à função *mint* para garantir que somente o editor do artigo possa solicitar sua execução e que novos *tokens* sejam gerados unicamente para artigos publicados e já rejeitados, além da verificação do *mapping papers*.

De maneira similar, o *PaperPublished* (figura 57) utiliza um *mapping* (*papers*) para garantir que possa ser cunhado um novo *token* somente uma vez para cada

[2] Disponível em: https://www.openzeppelin.com/

```
15  contract PubToken is ERC20, ERC20Burnable, Ownable
16  {
17      // Paper => (Reviewer/Editor => bool)
18      mapping(address => mapping(address => bool)) private papers;
19
20      constructor() ERC20("PublicationToken", "PubToken")
21      {
22          _mint(msg.sender, 100 ether);
23      }
24
25      function mint(address to, address editor, uint256 amount) external
26      {
27          Paper p = Paper(msg.sender);
28          require(p.is_editor(editor), "Only editors can mint");
29          require(p.get_paper_status() == 6 || p.get_paper_status() == 7,
30                  "Only for finished reviews");
31          require(!papers[msg.sender][to], "Can be minted only once per participant");
32          require(p.is_editor(to) || p.is_reviewer(to), "Only for Reviewers and Editor");
33
34          papers[msg.sender][to] = true;
35          _mint(to, amount);
36      }
37
38      function burnFrom(address account, uint256 amount) public override onlyOwner
39      {
40          _burn(account, amount);
41      }
42  }
```

Figura 56 – Código *PubToken*. Fonte: Elaborado pelo autor.

artigo publicado. Ele faz o mapeamento do endereço do *Paper* para um valor do tipo *bool*, que indica se já foi gerado um novo *token* para o artigo. Trechos de código foram adicionados à função *mint* para garantir que somente o editor do artigo possa executá-la e que novos *tokens* sejam gerados unicamente para artigos publicados, além da verificação do *mapping papers*. Assim como a função *burn* (figura 58), onde foi adicionada uma verificação para permitir que seja executada somente pelo editor do artigo, caso ele seja revogado.

4.3 EXPERIMENTOS

Com o objetivo de avaliar o modelo proposto na seção 4.1, experimentos foram conduzidos para obter-se os resultados referentes aos custos de transação para cada ator ao desempenhar o seu papel ao longo do processo de revisão. Os experimentos consistem em cinco cenários de teste, chamados de *Open Identities*, Revisão Tradicional, *Open Participation*, *Open Participation* com *Open Identities* e *Research Project Call*. As funções são executadas na ordem em que são apresentadas nas tabelas do capítulo 5.

Os experimentos foram conduzidos através da execução do protótipo no *Gana-*

```
14  contract PaperPublished is ERC721, ERC721URIStorage, ERC721Burnable, Ownable
15  {
16      using Counters for Counters.Counter;
17      Counters.Counter private _tokenIds;
18
19      mapping(address => bool) private papers;
20
21      constructor() ERC721("PaperPublished", "PP")  {
22      }
23
24      function mint_paper(address paper, string memory tokenURI) external
25      {
26          Paper p = Paper(paper);
27          require(!papers[paper], "Only once for Paper");
28          require(!p.is_author(owner()), "Recipient cannot be the owner of the contract");
29          require(p.is_editor(msg.sender), "Only Editor can mint");
30          require(p.get_paper_status() == 6, "Only for Published Papers");
31
32          uint256 newItemId = _tokenIds.current();
33          _tokenIds.increment();
34
35          _safeMint(p.get_author(msg.sender), newItemId);
36          _setTokenURI(newItemId, tokenURI);
37          papers[paper] = true;
38      }
```

Figura 57 – Trecho de código 1 *PaperPublished*. Fonte: Elaborado pelo autor.

```
40      function burn(address paper, uint256 tokenId) external
41      {
42          Paper p = Paper(paper);
43          require(p.is_editor(msg.sender), "Only Editor can Burn");
44
45          _burn(tokenId);
46      }
```

Figura 58 – Trecho de código 2 *PaperPublished*. Fonte: Elaborado pelo autor.

che. Ele oferece, por padrão, dez endereços de contas para testes e trabalha com o valor do *gas* fixo em 20 *Gwei*, portanto este foi valor adotado para o cálculo dos resultados. Para a execução das funções utilizou-se a interface de usuário implementada em *Javascript* em conjunto com o *Metamask* para interagir com os *smart contracts*. As contas fornecidas pelo *Ganache* foram adicionadas ao *Metamask* e cada uma foi utilizada para representar um ator participante do processo de revisão.

A princípio foram implantados os *smart contracts* necessários para a execução da plataforma, nesta etapa foram cunhados 100 *PubTokens* para serem utilizados durante os experimentos. Para medir o custo de adição de um novo *Publisher* utilizou-se a *string* “Publisher teste” como parâmetro *_name*. Os parâmetros utilizados para adicionar uma nova *Venue* foram: “Venue teste” (*_name*), 1 (*_type*), 5656944988 (*_deadline_review*) e 2 (*_number_rev*), 10000000000000000000 (*_fee*). De maneira similar os parâmetros ao adicionar um novo editor foram: 0 (*_id_venue*), “João da Silva”

(*_name*) e “https:orcid.org0000-0000-0000-0000” (*_orcid*). Assim como para adicionar um revisor: 0 (*_id_venue*) e “Isabella Atilano Gomes” (*_name*), “https:orcid.org0000-0000-0000-0000” (*_orcid*). Todos os parâmetros do tipo *address* têm o mesmo tamanho em qualquer caso.

Durante os cenários *Open Identities*, Revisão Tradicional, *Open Participation* e *Open Participation* com *Open Identities* os valores passados por parâmetros para as funções *submit*, *add_authors_information* e *add_paper_information* são iguais, com exceção do endereço do *smart contract* do artigo. O parâmetro *_names* é um vetor contendo duas *strings*, onde a primeira possui 19 caracteres e a segunda 17. Enquanto *orcid* é um vetor contendo as *strings* “https://orcid.org/0000-0000-0000-0000” e "https://orcid.org/0000-0000-0000-0001". O parâmetro *_title* é a *string* “Título Utilizado como Teste” e *keywords* é um vetor contendo as três *strings* “keyword1”, “keyword2”, “keyword3”. Os outros parâmetros têm tamanho fixo para qualquer caso.

No cenário *Research Project Call*, os parâmetros *_names* e *orcid* são uma *string* de 19 caracteres e “https://orcid.org/0000-0000-0000-0000”, respectivamente. O parâmetro *keywords* contém as *strings* “keyword1”, “keyword2”, “keyword3”. Os outros parâmetros são iguais aos casos anteriores. Neste caso não é executada a segunda rodada de revisões. Assim como, nos cenários que utilizam *Open Identities* não foi solicitada a versão final do artigo após a segunda rodada de revisões.

Em todos os cenários a plataforma está configurada para cunhar e transferir um *PubToken* para cada revisor e editor participante do processo. As taxas de publicação nos quatro primeiros cenários são de 10 *PubTokens*, enquanto o cenário *Research Project Call* não possui taxa de publicação.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos durante a execução dos experimentos descritos na seção 4.3, apresentando os custos de execução em *gas* e *ether* para diferentes cenários, assim como o custo de implantação dos *smart contracts*. Para os cálculos de custo em *ether* considerou-se o valor de 1 *gas* como 20 *Gwei*, como disponibilizado pelo *Ganache*. Os valores totais em Reais também são apresentados, considerando o valor do *ether* de aproximadamente R$7.100 em 26 de Setembro de 2022. As limitações são discutidas na seção 5.8. Os valores aqui apresentados referem-se somente aos custos de execução, não são consideradas as taxas de publicação eventualmente cobradas pelas *venues*.

5.1 CUSTOS DE IMPLANTAÇÃO

A tabela 3 apresenta os custos de implantação dos *smart contracts* utilizados no sistema, mostrados no capítulo 4. O custo total para disponibilizar o sistema on-line é de 12142830 *gas* ou 0,242857 *ether*. O valor total para implantação do sistema é de aproximadamente R$1724.

Tabela 3 – Custos de Implantação. Fonte: Elaborado pelo autor.

Smart Contract	**Custo de transação (gas)**	**Ether**
Submission	4037363	0,08074726
EditorialBoard	2391261	0,04782522
Review	498945	0,0099789
Publisher	2397359	0,04794718
PubToken	1043620	0,0208724
PaperPublished	1774282	0,03548564
Total	**12142830**	**0,242857**

5.2 CUSTOS OPEN IDENTITIES

A tabela 4 apresenta os resultados para o cenário *Open Identities*, que possui um custo total para a publicação do artigo, ou seja, somando todos os participantes, de 4.358.038 *gas* equivalente à 0,08716076 *ether* com valor aproximado de R$614. Neste cenário o autor gasta 2.983.275 *gas*, ou 0,0596655 *ether*, cerca de R$420, para executar todas as funções que lhe cabem, considerando as informações utilizadas como parâmetros das transações. Ao mesmo tempo, o editor teve um custo total de 703.419 *gas* ou 0,01406838 *ether*, aproximadamente R$100. Enquanto o revisor 1 gastou 358.172 *gas* (0,00716344 *ether*), e o revisor 2 313.172 *gas* (0,00626344 *ether*), aproximadamente R$50 e R$44, respectivamente.

Tabela 4 – Custos do cenário *Open Identities*. Fonte: Elaborado pelo autor.

Função	**Custo de transação (gas)**	**Ether**
Submissão (autor)		
approve	24973	0,000499
submit	2483533	0,049671
add_authors_information	249901	0,004998
add_paper_information	137243	0,002745
	2870677	**0,057414**
invite_reviewers (editor)	164186	0,003284
revisor_accept (revisor 1)	114358	0,002287
revisor_accept (revisor 2)	99358	0,001987
submit_review (revisor 1)	121084	0,002422
submit_review (revisor 2)	106084	0,002122
improve_paper (editor)	46497	0,00093
submit_new_version (autor)	112598	0,002252
submit_review (revisor 1) - 2ª rodada	122730	0,002455
submit_review (revisor 2) - 2ª rodada	107730	0,002155
publish_paper (editor)	307063	0,006141
mint_paper (editor)	185673	0,003713
Total	**4358038**	**0,087161**

O custo para executar a função *reject_paper* isoladamente é de 63272 *gas*, ou 0,00126544 *ether*, este valor equivale a aproximadamente R$9. O custo total para rejeitar um artigo depende do momento da revisão em que ele foi rejeitado. Caso seja antes da revisão (*Desk Reject*), então o custo total seria o custo de submissão somado ao de rejeição, portanto: 0,057414 *ether* + 0,00126544 *ether* o que resulta em 0,05867944 *ether*, ou aproximadamente R$417. Quanto mais o artigo avança no processo de revisão, maior o custo de rejeição, por exemplo, caso ele seja rejeitado após a primeira rodada de revisões o custo é de 0,07018218 *ether*, ou aproximadamente R$500.

5.3 CUSTOS REVISÃO TRADICIONAL

De maneira similar, a tabela 5 apresenta os resultados para o cenário Revisão Tradicional, que possui um custo total para publicação do artigo de 4.386.063 *gas* equivalente à 0,087721 *ether*, com valor de aproximadamente R$622. Neste cenário o autor gasta 2.982.371 *gas*, ou 0,05964742 *ether*, cerca de R$423, para executar todas as funções que lhe cabem, considerando as informações utilizadas como parâmetros das transações. Ao mesmo tempo, o editor teve um custo total de 747.448 *gas* ou 0,01494896 *ether*, aproximadamente R$106. Enquanto o revisor 1 gastou 335.622 *gas* (0,00671244 *ether*) e o revisor 2 gastou 320.622 *gas* (0,00641244 *ether*), aproximadamente R$48 e R$46, respectivamente.

O custo total para rejeitar um artigo depende do momento da revisão em que ele foi rejeitado, similarmente ao que acontece no cenário anterior, apresentado na

Tabela 5 – Custos cenário Revisão Tradicional. Fonte: Elaborado pelo autor.

Função	**Custo de transação (gas)**	**Ether**
Submissão (autor)		
approve	24973	0,000499
submit	2339774	0,046795
add_paper_information	137221	0,002744
	2501968	**0,050039**
invite_reviewers (editor)	164231	0,003285
reviewer_accept (revisor 1)	111808	0,002236
reviewer_accept (revisor 2)	96808	0,001936
submit_review (revisor 1)	103584	0,002072
submit_review (revisor 2)	118584	0,002372
improve_paper (editor)	46431	0,000929
submit_new_version (autor)	112466	0,002249
submit_review (revisor 1) - 2ª rodada	120230	0,002405
submit_review (revisor 2) - 2ª rodada	105230	0,002105
finish_review (editor)	44044	0,000881
Submeter Versão Final (autor)		
submit_new_version	113411	0,002268
add_authors_information	254526	0,005091
	367937	**0,007359**
publish_paper (editor)	306997	0,00614
mint_paper (editor)	185745	0,003715
Total	**4386063**	**0,087721**

seção 5.2. Caso ele seja rejeitado antes da revisão (*Desk Reject*), então o custo total seria o custo de submissão somado ao custo de rejeição, portanto: 0,050039 *ether* + 0,00126544 *ether* o que resulta em 0,0513048 *ether*, ou R$364. Quanto mais o artigo avança no processo de revisão, maior o custo de rejeição, por exemplo, caso ele seja rejeitado após a primeira rodada de revisões o custo é de 0,0632051 *ether*, ou aproximadamente R$450. Como o fluxo de execução do processo de revisão neste caso é semelhante ao anterior os custos são muito similares, havendo algumas diferenças referentes aos trechos de código diferente executados nos *smart contracts* para cada caso.

5.4 CUSTOS OPEN PARTICIPATION

A tabela 6 apresenta os resultados para o Cenário *Open Participation*, que possui um custo total para a publicação do artigo de 4.729.158 *gas* equivalente à 0,094583 *ether*, ou aproximadamente R$672. Neste cenário o autor gasta 3.342.131 *gas*, ou 0,06560022 *ether*, cerca de R$466, para executar todas as suas funções, considerando as informações utilizadas como parâmetros das transações. Ao mesmo tempo, o editor teve um custo total de 745.213 *gas* ou 0,01490426 *ether*, aproximadamente R$106. Enquanto o revisor 1 gastou 488.672 *gas* (0,00651564 *ether*), o revisor

Tabela 6 – Custos cenário *Open Participation*. Fonte: Elaborado pelo autor.

Função	**Custo de Transação (gas)**	**Ether**
Submissão (autor)		
approve (autor)	24973	0,000499
add_paper_information (autor)	137235	0,002745
submit (autor)	2468533	0,049371
	2630741	**0,052615**
start_review (editor)	37204	0,000744
submit_review (revisor 1)	187485	0,00375
submit_review (revisor 2)	157485	0,00315
submit_review (revisor 3)	157485	0,00315
submit_new_version (autor)	106236	0,002125
submit_review (revisor 1)	138297	0,002766
submit_review (revisor 2)	138297	0,002766
submit_new_version (autor)	106236	0,002125
finish_review (editor)	44116	0,000882
Submeter Versão Final (autor)		
submit_new_version	107129	0,002143
add_authors_information	254554	0,005091
	498918	**0,009978**
set_reviewer_open_participation (editor)	92969	0,001859
set_reviewer_open_participation (editor)	77969	0,001559
publish_paper (editor)	307216	0,006144
mint_paper (editor)	185739	0,003715
Total	**4729158**	**0,094583**

2 gastou 295.782 *gas* (0,00591564 *ether*) e o revisor 3 gastou 157.485 *gas* (0,0031497 *ether*), cerca de R$46, R$42 e R$22, respectivamente. Vale ressaltar, neste caso, que o revisor 3 enviou somente um relatório de revisão e o editor não o considerou relevante para a versão final do artigo.

O custo de rejeição de um artigo deve ser tratado como nos cenários anteriores, com um custo de rejeição antes da revisão (*Desk Reject*) de 0,05388026 *ether*, equivalente a aproximadamente R$383. Contudo, como neste caso a revisão é *Open Participation*, não existe controle sobre a quantidade de revisores que participam do processo, portanto quanto mais participantes, maior o custo de rejeição depois que a revisão é iniciada.

5.5 CUSTOS OPEN IDENTITIES E OPEN PARTICIPATION

De maneira semelhante, a tabela 7 apresenta os resultados para o cenário *Open Participation* e *Open Identities*, que possui um custo total para a publicação do artigo de 4.745.308 *gas* equivalente à 0,09490616 *ether*, com valor de aproximadamente R$674. Neste cenário o autor gasta 3.111.536 *gas*, ou 0,06223072 *ether*, cerca de R$442, para executar todas as funções que lhe cabem, considerando as informações

Tabela 7 – Custos cenário *Open Participation* e *Open Identities*. Fonte: Elaborado pelo autor.

Função	**Custo de transação (gas)**	**Ether**
Submissão (autor)		
approve	24973	0,000499
submit	2486949	0,049739
add_authors_information	249907	0,004998
add_paper_information	137235	0,002745
	2899064	**0,057981**
start_review (editor)	38614	0,000772
submit_review (revisor 1)	187497	0,00375
open_participation_name (revisor 1)	47144	0,000943
submit_review (revisor 2)	157497	0,00315
open_participation_name (revisor 2)	47084	0,000942
submit_review (revisor 3)	157497	0,00315
open_participation_name (revisor 3)	47096	0,000942
submit_new_version (autor)	106236	0,002125
submit_review (revisor 1)	138297	0,002766
open_participation_name (revisor 1)	27944	0,000559
submit_review (revisor 2)	138297	0,002766
open_participation_name (revisor 2)	27884	0,000558
submit_new_version (autor)	106236	0,002125
set_revisor_open_participation (editor)	92969	0,001859
set_revisor_open_participation (editor)	77969	0,001559
publish_paper (editor)	262244	0,005245
mint_paper (editor)	185739	0,003715
Total	**4745308**	**0,094906**

utilizadas como parâmetros das transações. Ao mesmo tempo, o editor teve um custo total de 657.535 *gas* ou 0,0131507 *ether*, aproximadamente R$93. Enquanto o revisor 1 gastou 400.882 *gas* (0,00801764 *ether*), o revisor 2 gastou 370.762 *gas* (0,00741524 *ether*) e o revisor 3 gastou 204.593 *gas* (0,00409186 *ether*), cerca de R$57, R$53, R$29, respectivamente. Vale ressaltar que, como no cenário apresentado na seção 5.4 o revisor 3 enviou somente um relatório de revisão que não foi considerado relevante para a versão final do artigo.

O custo de rejeição antes da revisão (*Desk Reject*) é de 0,05924644 *ether*, ou aproximadamente R$420. Como neste caso a revisão também é *Open Participation*, não existe controle sobre a quantidade de revisores, portanto quanto mais participantes no processo maior o custo de rejeição depois que a revisão é iniciada.

5.6 CUSTOS RESEARCH PROJECT CALL

Os resultados do cenário *Research Project Call* são apresentados na tabela 8, com um custo total para que o projeto seja aceito de 3.640.059 *gas* equivalente

Tabela 8 – Custos cenário *Researh Project Call*. Fonte: Elaborado pelo autor.

Função	**Custo de Transação (gas)**	**Ether**
Submissão (autor)		
submit	2489277	0,049786
add_authors_information	161064	0,003221
add_paper_information	137235	0,002745
	2787576	**0,055752**
invite_reviewers (editor)	164274	0,003285
revisor_accept (revisor 1)	114291	0,002286
revisor_accept (revisor 2)	99291	0,001986
submit_review (revisor 1)	121084	0,002422
submit_review (revisor 2)	106084	0,002122
accept_project (editor)	247459	0,004949
Total	**3640059**	**0,072801**

à 0,07280118 *ether*, com valor aproximado de R$517. Neste cenário o autor gasta 2.787.576 *gas* ou 0,05575152 *ether*, cerca de R$396, para executar todas as funções que lhe cabem, considerando as informações utilizadas como parâmetros das transações. Ao mesmo tempo, o editor teve um custo total de 411.733 *gas* ou 0,00823466 *ether*, aproximadamente R$58. Enquanto o revisor 1 gastou 235.375 *gas* (0,0044075 *ether*) e o revisor 2 gastou 205.375 *gas* (0,0041075 *ether*), cerca de R$31 e R$29, respectivamente. Neste caso, o custo de rejeição antes da revisão é de 0,05701744 *ether*, enquanto o custo ao final do processo de revisão é de 0,06911744 *ether*, ou aproximadamente R$405 e R$490, respectivamente.

5.7 CUSTOS PARA ADICIONAR PARTICIPANTES

Tabela 9 – Custos para adicionar participantes à plataforma. Fonte: Elaborado pelo autor.

Função	**Custo de transação (gas)**	**Ether**
add_publisher (admin)	107560	0,002151
add_venue (publisher)	1295433	0,025909
add_editor (publisher)	176171	0,003523
add_reviewer (editor)	162019	0,00324

A tabela 9 apresenta os custos para adicionar novos participantes à plataforma com base nas condições apresentadas na seção 4.3, indicando quem tem autorização para tal. O administrador do sistema adiciona uma nova editora (*publisher*) com custo de 0,002151 *ether*, o que equivale à aproximadamente R$15. Cada *publisher* pode adicionar uma nova *venue*, com custo observado no teste de 0,025909 *ether* ou aproximadamente R$184. Também cabe ao *publisher* adicionar um novo editor à uma *venue*, onde o custo obtido de 0,003523 *ether*, cerca de R$25. Finalmente, cada editor

pode adicionar um novo revisor à sua *venue* com custo aproximado de 0,00324 *ether* ou R$23.

5.8 DISCUSSÃO

Os resultados mostram que a complexidade do fluxo de trabalho do sistema tem relação direta com o custo total de execução, pois, quanto maior a quantidade de etapas ou de participantes, maior a quantidade de transações que devem ser executadas pelo *blockchain*. O custo de revogação de um artigo ou projeto de pesquisa, ou seja, retirar o seu *status* de publicado ou aceito, é de 106799 *gas* (0,00213598 *ether*), ou aproximadamente R$15. Desse modo, o custo total para revogar o artigo deve somar este valor ao custo de publicação ou aceitar o projeto. Esta ação pode ser necessária, por exemplo, caso após a publicação do artigo verifiquem-se problemas com os dados utilizados pelos autores, problemas éticos sobre a pesquisa etc. Em casos de projetos de pesquisa, eles podem ser revogados por baixo desempenho dos pesquisadores ou por questões éticas por exemplo. É importante ressaltar que os valores aqui obtidos referem-se aos parâmetros apresentados na seção 4.3, diferenças no tamanho destas variáveis afetarão o resultado final. Principalmente no caso dos nomes, pois são *strings* e podem variar bastante em tamanho no mundo real, portanto para nomes maiores o custo tende a ser maior e para nomes menores o custo tende a ser menor.

Com base nos resultados percebe-se um custo de execução de aproximadamente R$620 para a publicação de um artigo na revisão tradicional e *Open Identities*. Nos dois casos de revisão *Open Participation* o custo é superior a R$670, enquanto para *Research Project Call* é de quase R$517. Os custos de rejeição de um artigo antes da revisão (*Desk Reject*) são de aproximadamente R$360 no cenário Revisão Tradicional, R$405 para *Research Project Call* e cerca de R$383 para *Open Participation*. Enquanto nos dois casos que utilizam *Open Identities* os custos são próximos de R$420. Os custos mostraram-se relativamente altos, principalmente em caso de rejeição, portanto mecanismos para redução de custos e otimização da execução devem ser estudados.

Vale destacar que o valor do *gas* na rede principal do *Ethereum* oscila de acordo com as condições da rede, no dia 24 de Setembro de 2022 ele chegou a custar aproximadamente 9 *Gwei*[1]. Este valor reduziria os custos de execução para menos da metade dos obtidos no *Ganache*, considerando que os custos de transação seriam os mesmos aqui apresentados. Contudo no dia 27 de Setembro de 2022 o valor do *gas* foi de aproximadamente 22 *Gwei*, o que manteria os custos semelhantes, com um leve aumento.

[1] Disponível em: https://bityli.com/KniRqHIv

O *Ethereum* demora um certo tempo para confirmar as transações no *blockchain*, Giraldo, Milton C. e Gamboa (2020) mencionam cerca de 16 segundos. Para a aplicação aqui apresentada este atraso não prejudica o seu funcionamento, pois o tempo não é crítico para a execução.

O anonimato dos participantes, nos casos em que se aplica, é mantido através do pseudo-anonimato, onde as identidades dos autores são solicitadas somente após a publicação do artigo. Esta abordagem utiliza o endereço das contas de propriedade externa do *Ethereum* para identificar os participantes, não incluindo sua identificação no *blockchain*. Uma abordagem mais avançada deve ser adotada futuramente.

A plataforma adota por padrão as características de revisão aberta *Open Pre-review Manuscripts* e *Open Reports*, elas disponibilizam publicamente os manuscritos e relatórios de revisão, respectivamente. Esta abordagem aumenta a transparência do processo de revisão, mas limita o poder de escolha das editoras e instituições sobre a revisão. Para solucionar esta questão algum módulo *off-chain* deve ser desenvolvido, pois toda informação armazenada *on-chain* no *Ethereum* é pública.

Vale destacar também que o gerenciamento do *PubToken* na plataforma está em fase inicial de desenvolvimento, suficiente para a realização dos testes e validação de resultados. Maneiras mais eficientes, capazes de incentivar um comportamento positivo dos participantes devem ser estudadas. Possíveis soluções para os problemas aqui citados, assim como propostas de estudos e pesquisas posteriores para resolver estas limitações são abordadas na seção 6.1.

6 CONCLUSÃO

O objetivo desta pesquisa foi propor um modelo para um sistema de revisão de artigos científicos e propostas de projeto de pesquisa baseado em *blockchain*. O modelo proposto busca resolver ou mitigar problemas relatados sobre o referido processo, oferecendo um meio transparente para execução e incentivos para revisores e editores. O anonimato dos autores e revisores é mantido através do pseudo anonimato, mas uma solução mais eficiente deve ser implementada futuramente.

Os registros permanentes confiáveis dos *blockchains* públicos criam um ambiente propício à revisão aberta, disponibilizada pelo sistema em suas diferentes formas. Assim como podem ser utilizados para reduzir o risco de fraude durante o processo de revisão, criando uma prova de existência para os artigos ou projetos de pesquisa submetidos à plataforma. Apesar de a interface de usuário desenvolvida neste trabalho possuir as mínimas funcionalidades necessárias para a realização dos testes, foi possível observar a integração entre o *front end* e o *back end* do sistema funcionando de maneira adequada. Assim como a interação dos usuários, através dos seus endereços no *Metamask*, com os *smart contracts* funcionando conforme esperado.

Um mecanismo de incentivos foi aplicado em conjunto com a criação do *token* chamado de *PubToken*, cunhando-o e transferindo-o aos participantes do processo de revisão, como forma de incentivo. Este mecanismo busca atrair mais participantes interessados em atuar como revisores ou editores, e assim reduzir o tempo de revisão e agilizar a publicação de artigos ou a aprovação de projetos de pesquisa. De maneira similar demonstrou-se a aplicação de um *NFT* para associar a propriedade intelectual do artigo aos autores de maneira permanente.

6.1 TRABALHOS FUTUROS

Para trabalhos futuros, visando a redução dos custos de execução, propõe-se a avaliação do funcionamento do modelo em um *blockchain* privado, como o *Hyperledger*, mesmo que neste caso a transparência seja reduzida. Esta pode ser uma boa alternativa, principalmente para *Research Project Calls*. Assim como uma análise dos custos de execução com base na nova versão da rede principal do *Ethereum*, após a atualização *The Merge* que abandonou o algoritmo de consenso *proof-of-work* e adotou o *proof-of-stake*.

Buscando manter o anonimato dos participantes de maneira mais eficiente, uma solução semelhante à proposta por Mackey *et al.* (2019) pode ser adicionada ao sistema atual, utilizando um *blockchain* privado, como o *Hyperledger*, para executar a revisão fechada. Desta maneira as identidades dos autores e revisores podem ser armazenadas *on-chain*, pois as informações não serão públicas e ao final do processo o artigo é disponibilizado em um *blockchain* público. Assim como, para permitir que as

opções *Open Reports* e *Open Pre-review Manuscripts* sejam desabilitadas, pode-se criptografar os artigos, propostas de projeto e relatórios de revisão antes de enviá-los para o *IPFS*. Contudo esta solução requer um banco de dados para o armazenamento das chaves *off-chain*, permitindo controlar o acesso à elas somente por usuários autorizados. De maneira similar pode-se armazenar os próprios arquivos em um banco de dados *off-chain*, permitindo controlar o acesso a eles, porém exigiria que as editoras implementassem o banco de dados.

Uma análise sobre o melhor valor inicial para o *PubToken* deve ser conduzida, com o objetivo de definir a quantidade de *tokens* que devem ser criados durante a implantação da plataforma. Isto se faz necessário porque a relação entre a quantidade de *tokens* e *ether* enviada ao *liquidity pool* define o seu valor inicial, a partir deste momento ele oscila de acordo com as condições de oferta e demanda do mercado. Com base nesta análise, deve-se avaliar a quantidade de *PubTokens* que deve ser distribuída para cada editor e revisor após a finalização da revisão. O desenvolvimento da interface de usuário voltada para oferecer a melhor experiência aos usuários da plataforma também se faz necessário.

O *NFT* aplicado à plataforma será modificado para a criação de um *SBT*, permitindo o desenvolvimento de um sistema de reputação. A implementação do *SBT* será expandida para os relatórios de revisão, contribuindo para a reputação dos revisores. Esta abordagem será de grande importância para permitir o desenvolvimento de um ecossistema integrado para revisão de artigos e projetos de pesquisa, incorporando diferentes revistas científicas, conferências e agências de fomento à pesquisa.

REFERÊNCIAS

ALI, Parveen Azam; WATSON, Roger. Peer review and the publication process, p. 193–202, 2016. DOI: 10.1002/nop2.51.

AVITAL, Michel. Peer review: Toward a blockchain-enabled market-based ecosystem. **Communications of the Association for Information Systems**, v. 42, n. 1, p. 646–653, 2018. ISSN 15293181. DOI: 10.17705/1CAIS.04228.

BAILEY JR., Charles W. **Open Access Bibliography**. [*S.l.*: *s.n.*], 2005. P. 107. ISBN 9781594076701. Disponível em: http://digital-scholarship.org/oab/oab.pdf.

BASHIR, Imran. **Mastering Blockchain, Second Edition**. [*S.l.*]: Packt Publishing, 2018. P. 656. ISBN 9781788839044.

BORNMANN, Lutz. Scientific Peer Review. *In*: ANNUAL Review of Information Science and Technology. 45. ed. [*S.l.*: *s.n.*], 2011. cap. 5, p. 197–245.

BRAVO, Giangiacomo *et al.* The effect of publishing peer review reports on referee behavior in five scholarly journals. **Nature Communications**, Springer US, v. 10, n. 1, p. 1–8, 2019. ISSN 20411723. DOI: 10.1038/s41467-018-08250-2. Disponível em: http://dx.doi.org/10.1038/s41467-018-08250-2.

BUTERIN, Vitalik. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. January, p. 1–36, 2014.

BUTERIN, Vitalik. **Soulbound**. [*S.l.*: *s.n.*], 2022. Disponível em: https://vitalik.ca/general/2022/01/26/soulbound.html. Acesso em:

CHOI, Dong Hoon; SEO, Tae Sul. Development of an open peer review system using blockchain and reviewer recommendation technologies. **Science Editing**, v. 8, n. 1, p. 104–111, 2021. ISSN 22887474. DOI: 10.6087/kcse.237.

COELHO, Flávio Codeço; BRANDÃO, Adeilton. Decentralizing scientific publishing: Can the blockchain improve science communication? **Memorias do Instituto Oswaldo Cruz**, v. 114, n. 7, p. 1–4, 2019. ISSN 16788060. DOI: 10.1590/0074-02760190257.

CROSBY, M. *et al.* Applied Innovation Review. **Applied Innovation Review**, n. 2, 2016.

DAS, A. K. 'Peer review' for scientific manuscripts: Emerging issues, potential threats, and possible remedies. **Medical Journal Armed Forces India**, Director General, Armed Forces Medical Services, v. 72, n. 2, p. 172–174, 2016. ISSN 22134743. DOI: 10.1016/j.mjafi.2016.02.014. Disponível em: http://dx.doi.org/10.1016/j.mjafi.2016.02.014.

DUH, Emilija Stojmenova *et al.* Publish-and-flourish: Using blockchain platform to enable cooperative scholarly communication. **Publications**, v. 7, n. 2, p. 1–15, 2019. ISSN 23046775. DOI: 10.3390/publications7020033.

FERREIRA, Agata. Emerging Regulatory Approaches to Blockchain-based Token Economy. v. 3, n. 1, p. 1–9, 2020.

GIPP, Bela *et al.* CryptSubmit: Introducing Securely Timestamped Manuscript Submission and Peer Review Feedback Using the Blockchain. **Proceedings of the ACM/IEEE Joint Conference on Digital Libraries**, IEEE, 2017. ISSN 15525996. DOI: 10.1109/JCDL.2017.7991588.

GIRALDO, Faber D.; MILTON C., Barbosa; GAMBOA, Carlos E. Electronic Voting Using Blockchain and Smart Contracts: Proof of Concept. **IEEE Latin America Transactions**, v. 18, n. 10, p. 1743–1751, 2020. ISSN 15480992. DOI: 10.1109/TLA.2020.9387645.

HE, Ying; TIAN, Kun; FU, Jiangyang. An incentive mechanism-based framework to assure the quality of self-organizing peer review in preprint. **Data Technologies and Applications**, 2021. ISSN 25149288. DOI: 10.1108/DTA-08-2020-0181.

HILDENBRANDT, Everett *et al.* KEVM: A complete formal semantics of the ethereum virtual machine. **Proceedings - IEEE Computer Security Foundations Symposium**, 2018-July, p. 204–217, 2018. ISSN 19401434. DOI: 10.1109/CSF.2018.00022.

JAIN, Arpit; JAT, Dharm Singh. A Review on Consensus Protocol of Blockchain Technology. **Lecture Notes in Networks and Systems**, v. 334, p. 813–829, 2022. ISSN 23673389. DOI: 10.1007/978-981-16-6369-7_72.

KHAN, Imtiaz; SHAHAAB, Ali. A Peer-To-Peer Publication Model on Blockchain. **Frontiers in Blockchain**, v. 4, February, p. 1–8, 2021. DOI: 10.3389/fbloc.2021.615726.

KITCHENHAM, Barbara. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

KOSMARSKI, Artyom; GORDIYCHUK, Nikolay. Token-curated registry in a scholarly journal: Can blockchain support journal communities? **Learned Publishing**, v. 33, n. 3, p. 333–339, 2020. ISSN 17414857. DOI: 10.1002/leap.1302.

KRAVITZ, Richard L. *et al.* Editorial peer reviewers' recommendations at a general medical journal: Are they reliable and do editors care? **PLoS ONE**, v. 5, n. 4, p. 2–6, 2010. ISSN 19326203. DOI: 10.1371/journal.pone.0010072.

LEONARDO BARBOZA, Hugo; SCHERREIER FERNEDA, Ariê; BEATRIZ SASS, Liz. A garantia de autenticidade e autoria por meio de Non-Fungible Tokens (NFT's) e sua

(in)validade para a proteção de obras intelectuais. **International Journal of Digital Law**, v. 2, n. 2, p. 99–118, 2021. DOI: 10.47975/ijdl.barboza.v.2.n.2.

MACKEY, Tim K *et al.* A Framework Proposal for Blockchain-Based Scientific Publishing Using Shared Governance. v. 2, November, p. 1–12, 2019. DOI: 10.3389/fbloc.2019.00019.

NAKAMOTO, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System, p. 9, 2008. ISSN 15309185. DOI: 10.1162/ARTL_a_00247.

OLIVEIRA, Marcela T. *et al.* Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload. **Proceedings of the 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN 2019**, Icin, p. 180–187, 2019. DOI: 10.1109/ICIN.2019.8685888.

PALMA, Lucas M. *et al.* Blockchain and smart contracts for higher education registry in Brazil. **International Journal of Network Management**, v. 29, n. 3, p. 1–21, 2019. ISSN 10991190. DOI: 10.1002/nem.2061.

PALMA, Lucas Machado da. **Blockchain-Based Academic Record System**. [*S.l.*: *s.n.*], 2020. P. 75.

PANDA, Sandeep Kumar; SATAPATHY, Suresh Chandra. An Investigation into Smart Contract Deployment on Ethereum Platform Using Web3.js and Solidity Using Blockchain, p. 549–561, 2021. DOI: 10.1007/978-981-16-0171-2_52.

PPGTIC. **Programa de Pós-Graduação em Tecnologias da Informação e Comunicação**. [*S.l.*: *s.n.*], 2020. Disponível em: https://ppgtic.ufsc.br/linhas-de-pesquisa/. Acesso em: 3 dez. 2020.

ROSA, Reginaldo José da. **Modelo de logística reversa baseado em contabilidade distribuída - Blockchain**. 2019. F. 136. Tese (Doutorado) – Universidade Federal de Santa Catarina. ISBN 0012387592.

ROSS-HELLAUER, Tony. What is open peer review? A systematic review. **F1000Research**, v. 6, 2017. ISSN 1759796X. DOI: 10.12688/f1000research.11369.2.

ROSSUM, JORIS VAN. **Blockchain for Research**. [*S.l.*], 2017. ISBN 9780995624573. DOI: 10.1524/phil.1912.71.14.566.

ROWLAND, Fytton. The peer-review process, p. 247–258, 2002.

SCHUEFFEL, Patrick. Alternative Distributed Ledger Technologies Blockchain vs. Tangle vs. Hashgraph - A High-Level Overview and Comparison -. **SSRN Electronic Journal**, p. 1–8, 2018. ISSN 1556-5068. DOI: 10.2139/ssrn.3144241.

SILBIGER, Nyssa J.; STUBLER, Amber D. Unprofessional peer reviews disproportionately harm underrepresented groups in STEM. **PeerJ**, v. 2019, n. 12, p. 1–14, 2019. ISSN 21678359. DOI: 10.7717/peerj.8247.

SZABO, Nick. Formalizing and Securing Relationships on Public Networks. **First Monday**, v. 2, n. 9, 1997. DOI: 10.5210/fm.v2i9.548. Disponível em: https://firstmonday.org/ojs/index.php/fm/article/view/548.

TROVÒ, Bianca; MASSARI, Nazzareno. Ants-Review: A Privacy-Oriented Protocol for Incentivized Open Peer Reviews on Ethereum. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, 12480 LNCS, p. 18–29, 2021. ISSN 16113349. DOI: 10.1007/978-3-030-71593-9_2.

VALEONTI, Foteini *et al.* Crypto collectibles, museum funding and openGLAM: Challenges, opportunities and the potential of non-fungible tokens (NFTs). **Applied Sciences (Switzerland)**, v. 11, n. 21, 2021. ISSN 20763417. DOI: 10.3390/app11219931.

WANG, Taotao; CHANG LIEW, Soung; ZHANG, Shengli. PubChain: A decentralized open-access publication platform with participants incentivized by blockchain technology. **2020 International Symposium on Networks, Computers and Communications, ISNCC 2020**, 2020. DOI: 10.1109/ISNCC49221.2020.9297213. arXiv: 1910.00580.

WOOD, Gavin. Ethereum: a secure decentralised generalised transaction ledger. **Ethereum Project Yellow Paper**, p. 1–32, 2014. ISSN 1098-6596. arXiv: arXiv:1011.1669v3.

XU, Xiwei *et al.* A Taxonomy of Blockchain-Based Systems for Architecture Design. **Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017**, IEEE, p. 243–252, 2017. DOI: 10.1109/ICSA.2017.33.

YOO, Min Jae; WON, Yoojae. Smart Contract Based Academic Paper Review System. **Lecture Notes in Electrical Engineering**, Springer, Singapore, 536 LNEE, p. 259–264, dez. 2018. DOI: 10.1007/978-981-13-9341-9_44. Disponível em: https://link.springer.com/chapter/10.1007/978-981-13-9341-9%7B%5C_%7D44.

YOO, Soonduck. How to design the token reinforcement based on token economy for blockchain model. v. 8, n. 1, p. 157–164, 2020.

ZHENG, Xiaoying; ZHU, Yongxin; SI, Xueming. A Survey on Challenges and Progresses in Blockchain Technologies : A Performance and Security Perspective, p. 1–24, 2019. DOI: 10.3390/app9224731.