UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANDRÉ DEFRÉMONT

DESENVOLVIMENTO DE UMA EDITORA CIENTÍFICA SOBRE UMA REDE BLOCKCHAIN PERMISSIONADA

BELÉM-PA

Julho / 2019

ANDRÉ DEFRÉMONT

DESENVOLVIMENTO DE UMA EDITORA CIENTÍFICA SOBRE UMA REDE BLOCKCHAIN PERMISSIONADA

Trabalho de Conclusão de Curso apresentado no curso de Sistemas de Informação da Universidade Federal do Pará, como requisito parcial para obtenção do titulo de Bacharel em Sistemas de Informação

Orientador: Antônio Jorge Gomes Abelém

Co-Orientador: Vagner de Brito Nascimento

BELÉM-PA

Julho / 2019

ANDRÉ DEFRÉMONT

DESENVOLVIMENTO DE UMA EDITORA CIENTÍFICA SOBRE UMA REDE BLOCKCHAIN PERMISSIONADA

Trabalho de Conclusão de Curso apresentado no curso de Sistemas de Informação da Universidade Federal do Pará, como requisito parcial para obtenção do titulo de Bacharel em Sistemas de Informação

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Antônio Jorge Gomes Abelém
Universidade Federal do Pará
Orientador

Prof. Me. Vagner de Brito Nascimento
Universidade da Amazônia
Co-orientador

Prof. Dr. Denis Lima do Rosário
Universidade Federal do Pará

Prof. Dr. Marcos Cesar da Rocha Seruffo
Universidade Federal do Pará

CAPÍTULO 1

AGRADECIMENTOS

Aos meus pais Fernanda Dias Machado Defrémont e Alain Jacques André Defrémont; minhas irmãs Isabelle Machado Defrémont, Fanny Machado Defrémont e à minha amiga e companheira, Bruna Paola de Souza Sena.

A todos os familiares que sempre apoiaram minha paixão pela tecnologia e estiveram ao meu lado ao longo desse caminho rumo a formatura.

Aos alunos e professores da UFPA, especialmente os que convivi no GERCOM, Antônio Abelém, Billy Pinheiro, Vagner Nascimento e Bruno Evaristo, os quais ajudaram a me aprofundar nas tecnologias distribuídas e descentralizadas as quais abriram portas para que um trabalho como esse fosse desenvolvido por mim.

E finalmente, a todos da turma de Sistemas de Informação de 2013, sem eles o período acadêmico seria extremamente mais difícil.

Resumo

Resumo do Trabalho de Conclusão de Curso apresentado à UFPA como parte dos requisitos necessários para obtenção do título de Bacharel em Sistemas de Informação.

Desenvolvimento de uma Editora Científica sobre uma rede Blockchain Permissionada

Orientador: Prof. Dr. Antônio Jorge Gomes Abelém
Co-orientador: Prof. Me. Vagner de Brito Nascimento
Palavras-chave: Blockchain; Hyperledger; Editora Científica; Descentralizada; IPFS

Este Trabalho trata do desenvolvimento (especificação, implementação e avaliação) de uma aplicação web que deve funcionar como uma editora científica descentralizada, utilizando a tecnologia *Blockchain* para contratos inteligentes e armazenamento de hipermídia. A segurança dos dados armazenados é uma preocupação em diversos *softwares*, o que contribuiu para um melhor entendimento sobre processos e *softwares* distribuídos. Sistemas de armazenamento de dados que utilizam a tecnologia *Blockchain* surgem como alternativa para garantir a integridade das informações, através da descentralização do armazenamento, auditabilidade e transparência. Aplicações descentralizadas provaram-se mais eficientes, seguras e baratas. Essa discussão estende-se ao meio científico no cenário de publicação e revisão de artigos científicos. Pesquisadores utilizam ferramentas de publicação para divulgar seus trabalhos e desejam obter um certo controle quanto ao lucro, direitos autorais e revisão do conteúdo. Com a *Blockchain*, soluções como o *Hyperledger* e o *IPFS (Interplanetary File System)*, possibilitaram a gestão desse conteúdo mediante as lógicas de uma editora tradicional de forma autônoma, descentralizada e segura. Este trabalho é a prova de conceito de uma pesquisa mais ampla realizada no laboratório GERCOM na UFPA, o *DASP (Decentralized Autonomous Software Publisher)*, que tem como objetivo ampliar o gerenciamento do processo editorial, retirando intermediários, afim de otimizar o processo entre os entes envolvidos (comunidade). O sistema desenvolvido neste trabalho, possibilita o cadastro de autores e permite que eles enviem artigos à rede *IPFS* para ter seu *hash* armazenado no *Hyperledger Fabric* onde estará disponível para ser avaliado por Revisores cadastrados na rede, através das regras do contrato inteligente. A implementação do código foi feita na linguagem *Javascript (NodeJS),* com o auxílio das Dependências: *angular4, react, go-ipfs, hyperledger-composer*, a base de dados é o *IPFS* para hipermídia e o *Hyperledger Fabric* para transações, participantes e ativos. O sistema desenvolvido foi validado através da avaliação técnica dos diagramas elaborados e por questionários avaliativos com usuários reais, em um cenário que contempla o envio, revisão e publicação de um artigo científico, em uma intranet.

Abstract

Course Conclusion Paper presented at the Information Systems course of the Federal University of Pará, as a partial requirement to obtain the Bachelor's degree in Information Systems.

Development of a Scientific Publishing House on a Blockchain Permissioned Network

Advisor: Prof. Dr. Antônio Jorge Gomes Abelém
Co-advisor: Prof. Me. Vagner de Brito Nascimento
Keywords: Blockchain; Hyperledger; Scientific Publishing House; Decentralized

This paper deals with the development (specification, implementation and evaluation) of a web application that should act as a decentralized scientific publisher, using Blockchain technology for intelligent contracts and hypermedia storage. The security of stored data is a concern in many software, which has contributed to a better understanding of distributed processes and softwares. Data storage systems that use Blockchain technology emerge as an alternative to ensure information integrity through decentralized storage, auditing, and transparency. Decentralized applications have proven to be more efficient, safer and cheaper. This discussion extends to the scientific community in the scenario of publication and revision of scientific articles. Researchers use publishing tools to publicize their work and want to gain some control over profit, copyright, and content review. With Blockchain, solutions such as Hyperledger and IPFS (Interplanetary File System) enabled the management of this content through the logics of a traditional publisher in an autonomous, decentralized and secure way. This work is the proof of concept of a broader research carried out in the GERCOM laboratory at UFPA, the DASP (Decentralized Autonomous Software Publisher), which aims to broaden the management of the editorial process, removing intermediates, in order to optimize the process between the entities involved (community). The system developed in this work allows authors to register and allows them to submit articles to the IPFS network to have their hash stored in Hyperledger Fabric where it will be available to be evaluated by registered reviewers on the network through the rules of the smart contract. The implementation of the code was done in the Javascript language (NodeJS), with the help of the Dependencies: angular4, react, go-ipfs, hyperledger-composer, the database is IPFS for hypermedia and Hyperledger Fabric for transactions, participants and assets . The implemented system was validated through the technical evaluation of the elaborated diagrams and by evaluation questionnaires with real users, in a scenario that contemplates the sending, revision and publication of a scientific article, in an intranet.

Lista de Abreviaturas

UFPA	Universidade Federal do Pará
SBC	Sociedade Brasileira de Computação
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
ACM	Association for Computing Machinery
DApps	Decentralized Applications
P2P	Peer To Peer
DHTs	Distributed Hash Tables
UML	Unified Modeling Language
BFT	Bizantine fault tolerance
TCP	Transmission Control Protocol
BND	Business Network Definition
npm	Node Package Manager
MVP	Minimum Viable Product
API	Application Programming Interface

Lista de Figuras

Figura 3.1 Diagrama da arquitetura de rede de Paul Baran. [Baran 1964] 4

Figura 3.2 Como a rede Blockchain funciona [Adaptado de http://www.agiboo.com/] .. 6

Figura 3.3 Conjunto de setores Hyperledger [Hyperledger.org] 8

Figura 3.4 O Processo Editorial [Ferreira and Silva 2013] 9

Figura 3.5 Scrum: A Metodologia Ágil [http://www.mindmaster.com.br/scrum/] 13

Figura 3.6 Exemplo de Casos de uso 16

Figura 3.7 Exemplo de Diagrama de Atividades 17

Figura 3.8 Exemplo diagrama de Classes 18

Figura 3.9 Exemplo de diagrama de Sequência 18

Figura 4.1 Hyperledger Composer Playground 21

Figura 4.2 Node-RED - Flow para envio de e-mail 22

Figura 5.1 O editor on-line da plataforma EUREKA 24

Figura 5.2 Beta DEIP: Decentralized Research Platform 25

Figura 5.3 Website Open Science Network - OSN 25

Figura 6.1 Personas Autor e Revisor 29

Figura 6.2 Diagrama de Casos de Uso de Contexto 31

Figura 6.3 Diagrama de Casos de Uso do subsistema do Autor 32

Figura 6.4 Diagrama de Casos de Uso do subsistema do Revisor 37

Figura 6.5 Diagrama de Atividades do sistema 41

Figura 6.6 Diagrama de Classes do sistema 42

Figura 6.7 Diagrama de sequência do Caso de uso 4 do subsistema de Submissão 43

Figura 6.8 Diagrama de sequência do Caso de uso 4 do subsistema de Revisão 44

Figura 6.9 Definição do Participante Author no arquivo modelo (.cto) 45

Figura 6.10 Definição dos Ativos (Artigo, Revisão e Detalhes no arquivo modelo (.cto) . 46

Figura 6.11 Definição das Transações no arquivo modelo (.cto) 47

Figura 6.12 Exemplo de sequência lógica (logic.js) 48

Figura 6.13 Exemplos de regra de acesso (permissions.acl) 48

Figura 6.14 Painel principal da aplicação web 51

Figura 6.15 Menu "*To Review*"da aplicação web 51

Figura 7.1 Respostas da Primeira pergunta objetiva da Avaliação da aplicação 53

Figura 7.2 Respostas da Segunda pergunta objetiva da Avaliação da aplicação 54

Figura 7.3 Respostas da Terceira pergunta objetiva da Avaliação da aplicação web 55

Figura 7.4 Respostas da Quarta pergunta objetiva da Avaliação da aplicação web 55

Figura 7.5 Respostas da Quinta pergunta objetiva da Avaliação da aplicação web 56

Figura 7.6 Respostas da Sexta pergunta objetiva da Avaliação da aplicação web 57

Figura 7.7 Respostas da Sétima pergunta objetiva da Avaliação da aplicação web 58

Figura 7.8 Respostas da Oitava pergunta objetiva da Avaliação da aplicação web 58

Figura 7.9 Respostas da Pergunta subjetiva da Avaliação da aplicação web 59

Lista de Tabelas

Tabela 5.1 Ferramentas relacionadas encontradas 23

Tabela 5.2 Comparação das propostas 26

Sumário

2 Introdução ..... p. 1
2.1 Motivações e Justificativas ..... p. 1
2.2 Objetivos ..... p. 2
2.2.1 Objetivo Geral ..... p. 2
2.2.2 Objetivo Específico ..... p. 3
2.3 Organização do Trabalho ..... p. 3

3 Referencial Teórico ..... p. 4
3.1 Aplicações Descentralizadas ..... p. 4
3.2 Blockchain ..... p. 5
3.3 Hyperledger ..... p. 7
3.4 Processo de publicação de artigo ..... p. 8
3.5 Engenharia de Software ..... p. 10
3.6 Processo de Software ..... p. 11
3.7 Modelo de Processo de Software ..... p. 11
3.7.1 Método Ágil ..... p. 12
3.7.2 Scrum ..... p. 12
3.8 Engenharia de Requisitos ..... p. 13
3.8.1 Documentação dos Requisitos ..... p. 14
3.8.2 UML ..... p. 14
3.8.3 Diagrama de Casos de uso ..... p. 16
3.8.4 Diagrama de Atividades ..... p. 16
3.8.5 Diagrama de Classes ..... p. 17
3.8.6 Diagrama de Sequência ..... p. 17

4 Ferramentas e Tecnologias Utilizadas ..... p. 19
4.1 Hyperledger Fabric ..... p. 19
4.2 Hyperledger Composer ..... p. 20
4.3 Hyperledger Composer Playground ..... p. 20
4.4 IPFS - Interplanetary File System ..... p. 21
4.5 Node-RED ..... p. 22

5 Trabalhos Relacionados ..... p. 23
5.1 Sistemas Editoriais que utilizam Blockchain ..... p. 23
5.1.1 EUREKA ..... p. 24
5.1.2 DEIP ..... p. 24
5.1.3 Open Science Network - OSN ..... p. 25
5.2 Avaliação das Ferramentas que utilizam Blockchain para Revisão e Publicação de Artigos Científicos ..... p. 26

6 Editora científica sobre uma rede Blockchain Permissionada ..... p. 28
6.1 Documento de especificação de requisitos ..... p. 28
6.1.1 Ambiente do sistema ..... p. 28
6.1.2 Objetivos do produto ..... p. 29
6.1.3 Benefícios ..... p. 30
6.2 Diagramas de Casos de Uso ..... p. 30
6.2.1 Diagrama de Casos de Uso de Contexto ..... p. 31
6.2.2 Subsistema de Submissão de Artigos Científicos ..... p. 31
6.2.3 Subsistema de Revisão de Artigos Científicos ..... p. 37
6.2.4 Requisitos não funcionais ..... p. 40
6.3 Documento de especificação e análise ..... p. 40
6.3.1 Diagrama de Atividades ..... p. 41
6.3.2 Diagrama de Classes ..... p. 42
6.3.3 Diagrama de Sequência ..... p. 42
6.4 Implementação do software editorial sobre *Blockchain* ..... p. 44
6.4.1 Banco de Dados ..... p. 45

6.4.2 API . . . . . p. 49
6.4.3 Client . . . . . p. 49
6.4.4 Finalização do desenvolvimento . . . . . p. 50
6.5 Interação do usuário . . . . . p. 50
6.5.1 Interface Web Gráfica da Ferramenta . . . . . p. 50

7 Avaliação da Proposta . . . . . p. 52
7.1 Usabilidade . . . . . p. 52
7.2 Funcionalidade . . . . . p. 53
7.3 Confiabilidade . . . . . p. 55
7.4 Eficiência . . . . . p. 57
7.5 Críticas e Sugestões . . . . . p. 59

8 Conclusões . . . . . p. 60
8.1 Conclusões Gerais . . . . . p. 60
8.2 Trabalhos Futuros . . . . . p. 60

Referências Bibliográficas . . . . . p. 63

Apêndice A – Configuração e instalação do sistema em produção . . . . . p. 65
A.1 Instalação . . . . . p. 65

CAPÍTULO 2

INTRODUÇÃO

2.1 Motivações e Justificativas

Sistemas distribuídos ou descentralizados possuem diversas vantagens para a segurança e auditabilidade dos dados. Com isso, a indústria e a comunidade acadêmica voltaram-se para aplicações baseadas em *Blockchain*, tecnologia que permite que partes mutuamente desconfiadas, formem uma rede distribuída mantendo um registro de transações da rede em comum [Swan 2015]. A tecnologia *Blockchain* evoluiu muito desde o surgimento do *Bitcoin*, moeda digital criptografada e descentralizada [Nakamoto et al. 2008], e hoje permite que outros grupos criem sua própria cadeia de blocos que, pode ter regras, modelos e lógicas diferentes, adequando-se a diversos processos empresariais e científicos, ao exigirem que seus usuários sejam verificados ou que determinados dados sejam restritos. Para garantir a integridade de determinados dados e evitar possíveis corrupções, precisamos de um modelo de *Blockchain* como o *Hyperledger Fabric* da *Linux Foundation*, que por ser permissionada, pode conter regras de acesso específicas para cada dado, transação e participante.

A validação do conhecimento científico através da publicação de artigos sempre foi alvo de adaptações e melhorias através do tempo [Van Wyk 1998]. Atualmente esse processo é gerenciado pelo o que conhecemos por editoras científicas. Estas são responsáveis pela coleta desses artigos, qualificação através de revisões e publicações dos mesmos, com base em suas revisões. Com a expansão do meio digital e o visível impacto do papel no meio ambiente, sistemas editoriais como *SBC*[1]*, IEEE*[2]*, Springer*[3]*, ELSEVIER*[4] *e ACM*[5]; utilizam sistemas terceirizados para gerenciar seu processo de submissão e revisão de artigos. Essa terceirização se deve ao fato da complexidade e confiabilidade relacionada a escolha dos que vão revisar os trabalhos submetidos.

Os sistemas das editoras científicas citadas, possuem seus dados centralizados e/ou pouco auditável, isso significa que há possibilidade de que uma parte maliciosa do serviço se beneficie ou dificulte nos processos de publicação dos artigos científicos da plataforma. Além do problema com a centralização, precisamos garantir que haja um acesso permissionado aos processos de revisão para que não haja, nem mesmo de forma indireta, a manipulação de re-

[1]https://www.acm.org/
[2]https://www.ieee.org/
[3]https://www.springer.com/gp
[4]https://www.loja.elsevier.com.br
[5]https://www.acm.org/

visões. Algumas editoras, apesar de descentralizadas, são não-permissionadas, isto é, todo seu registro é público, o que torna autores e revisores facilmente rastreáveis, um atributo não desejável em um sistema de revisão de artigos.

Muitas editoras também geram relatórios, ou seja, de certa forma auditam os processos de envio, revisão e publicação dos trabalhos; porém não utilizam tecnologias de redes descentralizadas em suas transações para que possam ser acompanhadas pelos interessados e garantir a imutabilidade desses registros. Esse aprimoramento garante ao usuário final, maior rastreabilidade sobre o que acontece com seus artigos, ao acompanhar os processos de revisão e publicação de forma automatizada.

Com o objetivo de promover uma maior descentralização sobre os processos de revisão de artigos científicos, foi desenvolvido um sistema web editorial descentralizado, que é alvo deste trabalho. Este sistema funcionará como uma Editora de Artigos Científicos e foi concebida como uma alternativa descentralizada para publicação de artigos científicos por quaisquer autores previamente registrados em uma rede permissionada e descentralizada, para garantir todos os requisitos de auditabilidade, segurança e controle de permissão não encontrados nas ferramentas atuais.

A melhoria desse processo através desta ferramenta trás para a comunidade científica um ganho muito grande, não apenas na velocidade do processo de publicação de artigos científicos, mas também por possuir um atrativo para os produtores e revisores de artigos científicos, bem como para as editoras tradicionais já que a ferramenta pode, e deve, ser instanciada, isto é, replicada em várias entidades, as quais compartilhariam das mesmas transações, regras de acesso e banco de dados; promovendo assim, a descentralização. Além disso, a transparência e a descentralização, pode criar um novo parâmetro de qualidade no processo de publicação de artigos científicos, podendo se tornar um vetor de confiabilidade entre as partes envolvidas. Nós somos diretamente afetados pela produção científica que, com muita dificuldade, contribui para o desenvolvimento e crescimento econômico de uma nação. Dificuldade pelo fato de que como muitas coisas no mundo atual, corrompe-se facilmente sempre que envolve lucro e muitos intermediários.

2.2 Objetivos

2.2.1 Objetivo Geral

O objetivo principal deste trabalho é o desenvolvimento de um sistema editorial descentralizado sob uma rede *Blockchain* permissionada que promova a interação entre autores de artigos científicos e revisores de forma autônoma, que garanta a qualidade do material revisado, além de disponibilizar publicamente os artigos que passaram pelo processo de revisão com sucesso, para a comunidade.

2.2.2 Objetivo Específico

Para alcançar o objetivo central neste trabalho, faz-se necessário alguns objetivos específicos como:

- Pontuar técnicas de ferramentas utilizadas para avaliação de artigos científicos atualmente.
- Implementar uma rede *Blockchain* baseada nas funcionalidades pontuadas.
- Implementar interface que faça integração das tecnologias para o usuário final.
- Fazer avaliações da usabilidade da interface de usuário.
- Discutir resultados e soluções para possíveis problemas futuros encontrados.

2.3 Organização do Trabalho

O Capítulo 2 apresenta uma breve descrição sobre o referencial teórico relacionado a pesquisas envolvendo aplicações descentralizadas, modelo de revisão de artigos científicos, métodos de engenharia de software do sistema.

O Capítulo 3 descreve as ferramentas e tecnologias utilizadas para alcançar os objetivos previstos.

O Capítulo 4 apresenta os trabalhos relacionados à criação de ferramentas ou sistemas de revisão e publicação de artigos científicos e que possuem algum tipo de tecnologia *Blockchain* implementada.

O Capítulo 5 descreve a proposta e seu desenvolvimento, informações das ferramentas, tecnologias e diagramas relacionados à ferramenta.

O Capítulo 6 expõe o cenários da avaliações utilizada, o formulário utilizado, os resultados obtidos e a análise desses resultados.

O Capítulo 7 contêm conclusões obtidas e os trabalhos futuros.

CAPÍTULO 3

REFERENCIAL TEÓRICO

Este Capítulo aborda as principais literaturas e estudos realizados para implementar o projeto do sistema proposto, além de obter embasamento nas metodologias e parâmetros utilizados.

3.1 Aplicações Descentralizadas

Um novo modelo para a construção de aplicativos escaláveis e lucrativos está surgindo. A *Bitcoin* abriu o caminho com a tecnologia de livro de registros criptografados e tecnologia *peer-to-peer*. Esses recursos fornecem um ponto de partida para a criação de um novo tipo de software chamado aplicativos descentralizados, ou *DApps*. Raval acredita que essas aplicações: "um dia se tornarão mais amplamente usadas do que os aplicativos da web mais populares do mundo. Eles são mais flexíveis, transparentes, distribuídos, resilientes e têm uma estrutura de incentivo melhor do que os modelos de software atuais."[Raval 2016]

Existem vários aplicativos atualmente em uso e a grande maioria dos aplicativos Web seguem um modelo de cliente-servidor centralizado. Alguns são distribuídos e uma pequena parcela são descentralizados. A Figura 3.1 mostra uma representação visual desses três modelos de software, centralizado, descentralizado e distribuído.

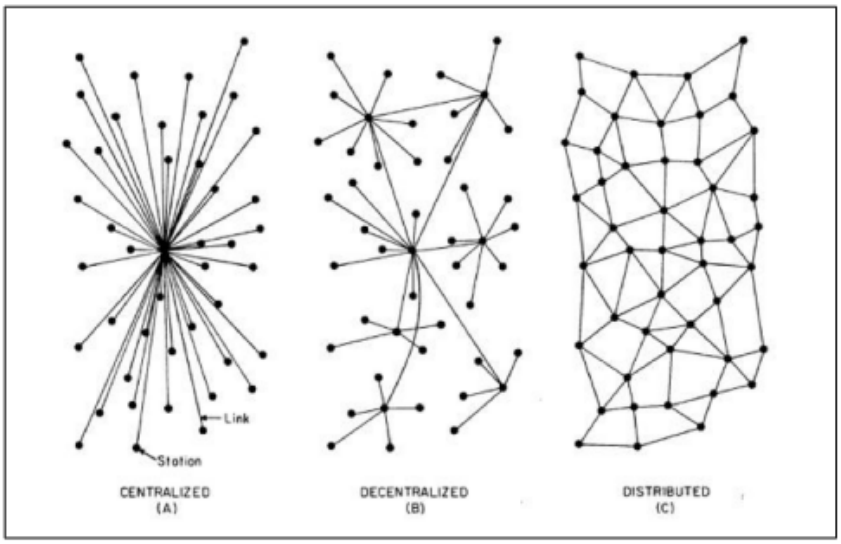

Figura 3.1: Diagrama da arquitetura de rede de Paul Baran. [Baran 1964]

Sistemas centralizados controlam diretamente a operação e o fluxo de informações dos usuários em um único ponto. Todos os indivíduos dependem diretamente do poder central para enviar e receber informações ou controla-lo. A maioria dos serviços tradicionais que usamos na internet são centralizados e fornecem um ótimo serviço, mas possuem falhas.

Distribuição em software significa que os dados do sistema estão distribuídos em vários nós ao invés de apenas um, como no *Bitcoin*, onde um registro datado e público é armazenado em vários computadores. Já a descentralização garante que nenhum nó da rede dependa de orientações de outro nó, isto é, os nós são independentes, novamente como no *Bitcoin* onde se um nó falha, a rede continua operando normalmente.

O cenário de aplicações descentralizadas, é atualmente um campo emergente, os desenvolvedores possuem opiniões diferentes sobre o que exatamente é um *dapp*. Alguns desenvolvedores acham que não ter um ponto central de falha é o suficiente e outros acham que existem mais requisitos. Os principais pontos discutidos são:

- Código aberto

  Os aplicativos de código aberto garantem aos usuários tudo o que foi prometido pelos desenvolvedores, provendo transparência ao projeto. Aplicativos de código fechado por outro lado, atuam como uma barreira à adoção para os usuários especialmente quando o aplicativo é projetado para receber, reter ou transferir ativos do usuário.

- Consenso Descentralizado

  Antes do *Bitcoin*, o consenso sobre a validade das transações sempre exigia algum grau de centralização. Se você quisesse efetuar um pagamento, sua transação teria que passar por um processo que monitora todas as transações. Atualmente, temos transações *peer to peer (P2P)*, onde os nós são capazes de falar diretamente uns com os outros, independendo de um intermediário. Redes P2P não são novidade; Tabelas de *hash* distribuídas (DHTs), como *BitTorrent*, nasceram antes da *Blockchain*. Os *DHTs* são ótimos para armazenar e transmitir dados descentralizados, mas se você quiser uma aplicação de alta complexidade que exija que todos concordem de maneira descentralizada, será necessário implementar algo com *Blockchain*.

- Sem ponto de falha central

  O fato de não haver um servidor central, uma aplicação descentralizada não pode ser desativada facilmente, isso pelo fato de que cada nó é independente, se um falhar, os outros ainda poderão manter a rede.

3.2 Blockchain

Um *Blockchain* é essencialmente um banco de dados distribuído de registros, ou razão pública de todas as transações ou eventos digitais que foram executados e compartilhados entre as partes participantes [Crosby et al. 2016].

A tecnologia *Blockchain* é uma tecnologia emergente que funciona como um livro de registros de transações entre participantes que não necessariamente têm confiança entre si e possui propriedades que antes não eram possíveis em sistemas tradicionais, como a imutabilidade, descentralização e criptografia desses registros. Com essa possibilidade, a troca de bens digitais se tornou mais eficiente e independente de confiança em uma parte específica, isso permitiu a criação de moedas digitais criptografadas mais sólidas como o *Bitcoin* [Nakamoto et al. 2008] e outras várias soluções confiáveis como sistemas de votação, registro de bens e cadeias de fornecimento substituindo entidades certificadoras e centralizadoras (Cartórios, Bancos, Governos e etc).

Na tecnologia *Blockchain*, todos os nós possuem e mantêm uma réplica do registro de transações, na forma de um livro-razão (*ledger*) distribuído, que é imutável, pode ser verificado e auditado, e está sempre disponível. Através de um protocolo de consenso *Byzantino*, a tecnologia se torna tolerante a ações de nós maliciosos, que podem subverter o sistema e determina a ordem em que as transações são executadas.

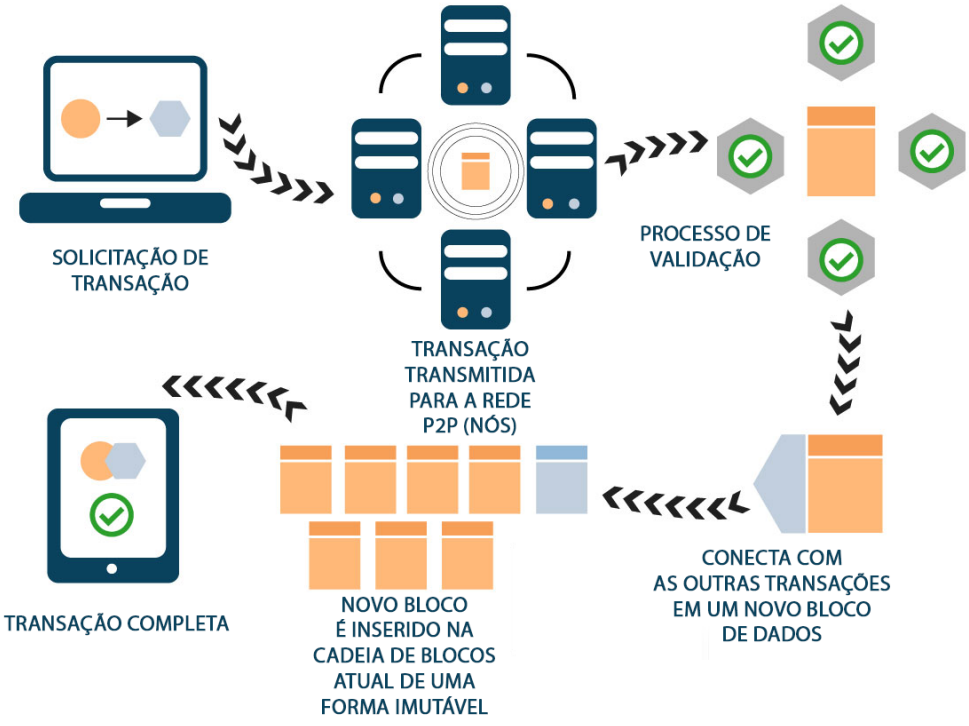

Figura 3.2: Como a rede Blockchain funciona [Adaptado de http://www.agiboo.com/]

As principais propriedades da tecnologia *Blockchain* para o desenvolvimento de aplicações e sistemas:

- Descentralização: Aplicações e sistemas executados de maneira distribuída e sem a necessidade de uma entidade centralizador.
- Disponibilidade e Integridade: Todos os dados e transações são replicados em diferentes nós de maneira segura.

- Transparência e Auditabilidade: Todas as transações registradas são públicas, podendo ser verificadas e auditadas.
- Imutabilidade e Irrefutabilidade: As transações registradas são imutáveis.
- Privacidade e Anonimidade: É possível oferecer privacidade aos usuários sem que terceiros envolvidos tenham acesso aos seus dados.
- Desintermediação: A *Blockchain* possibilita a integração entre diversos sistemas, isto é, elimina intermediários, simplificando o processo.

Swan diz que: "Como cada categoria de atividade humana organizada passou para a Internet e, atualmente, tem a possibilidade de ser reinventada e tornar-se mais eficiente, com o atributo *blockchain*, assim também a publicação acadêmica pode ser colocada no *blockchain*. Houve inovações em direção à abertura no campo da publicação acadêmica, como periódicos de acesso aberto, que embora forneçam acesso aberto ao conteúdo do artigo em vez de mantê-lo atrás de um portal de pagamento, obrigam os autores a apoiar taxas de publicação possivelmente proibitivas."[Swan 2015]

Diante dessas características, essa tecnologia deve ser vista como uma oportunidade para as instituições nacionais e internacionais defenderem os direitos daqueles que representam e para acelerar o nosso progresso coletivo no sentido de alcançar os Objetivos de Desenvolvimento Sustentável das Nações Unidas.

3.3 Hyperledger

O *Hyperledger*[1] é um conjunto de módulos colaborativos e de código aberto criado para promover as tecnologias de *Blockchain* em vários setores [Dhillon et al. 2017]. Criada pela *Linux Foundation*, este projeto inclui líderes em finanças, serviços bancários, Internet das Coisas, cadeias de suprimentos, manufatura e tecnologia.

A tecnologia *Blockchain* nada mais é do que um livro de ordens distribuído *peer-to-peer* mantido por consenso, combinado com um sistema para “contratos inteligentes” e outras tecnologias assistivas. Juntos, eles podem ser usados para construir uma nova geração de aplicativos transacionais que estabeleçam confiança, responsabilidade e transparência em sua essência, enquanto simplificam os processos de negócios e as restrições legais.

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

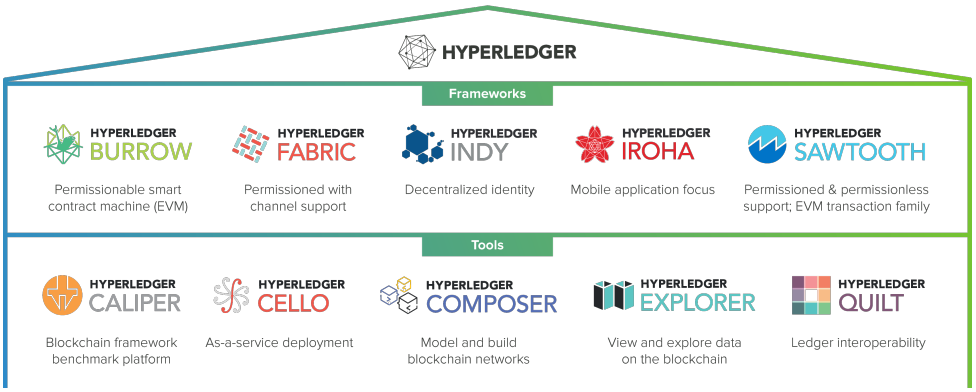

Figura 3.3: Conjunto de setores Hyperledger [Hyperledger.org]

Algo como um sistema operacional para mercados, redes de compartilhamento de dados, micro-moedas e comunidades digitais descentralizadas. O projeto *Hyperledger* tem o potencial de reduzir enormemente o custo e a complexidade de fazer as coisas no mundo real.

Com o desenvolvimento colaborativo de *software* de código aberto, podemos garantir a transparência, a longevidade, a interoperabilidade e o suporte necessários para levar as tecnologias *Blockchain* à adoção comercial convencional. É disso que se trata o *Hyperledger*, comunidades de desenvolvedores de software construindo estruturas e plataformas *Blockchain*.

3.4 Processo de publicação de artigo

O progresso da ciência está intimamente relacionada a pesquisa e publicação científica coordenada por especialistas. A publicação de artigos em periódicos confiáveis, valoriza o pesquisador ao garantir maiores prestígios e também as instituições com seus históricos de publicações. Essa busca por reputação também pode ser prejudicial quando nos referimos aos editores e revisores que detém o conhecimento de forma centralizada.

A Figura 3.4 é uma representação criada por Manuel Aníbal [Ferreira and Silva 2013] que fez uma adaptação da representação de *Van Wyk* [Van Wyk 1998] que por sua vez, analisou e designou o que compreendeu sobre o processo de publicação como um sistema. Esse processo tem como objetivo gerar conhecimento, aprendizagem e recompensas sociais para o pesquisador e para a instituição, e possui os componentes chaves descritos abaixo:

- **Autor**: Escreve o artigo e é responsável pela submissão do mesmo. Todo o trabalho científico e até mesmo a diagramação do conteúdo a um formato pré-estabelecido é de sua responsabilidade;

- **Revisor**: Avalia os artigos e atribui notas que determinam a aceitação do artigo. Não é raro que autores de um determinado periódico sejam convidados a compor o corpo de revisores do mesmo. Tal função é normalmente não remunerada, mas de reconhecido prestígio dentro do campo de atuação;

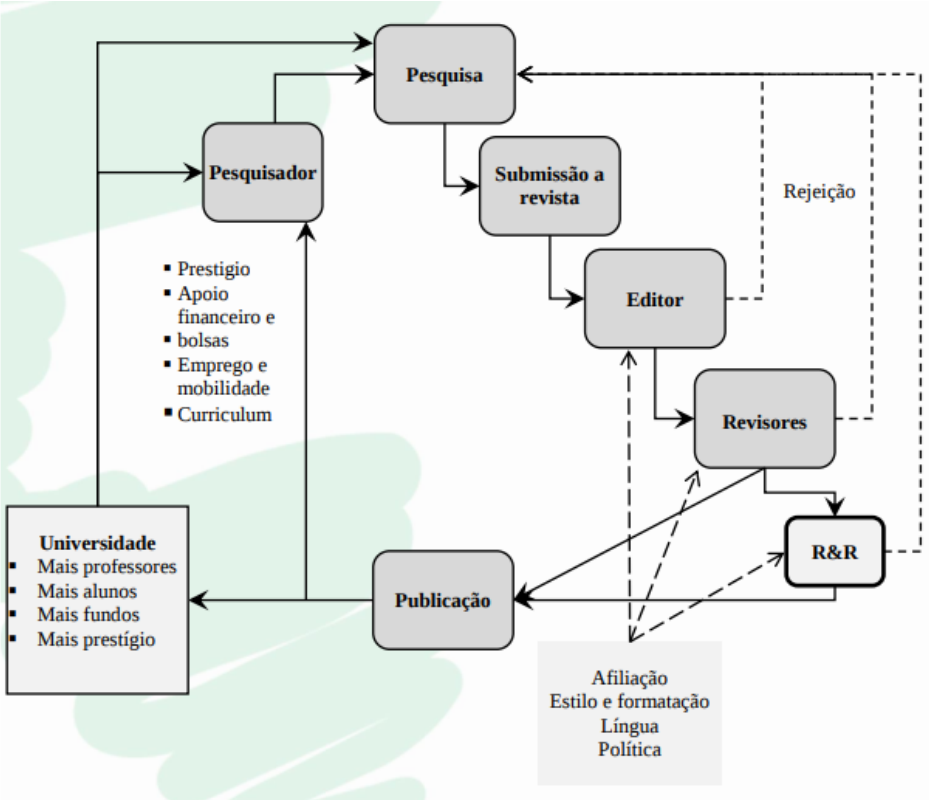

Figura 3.4: O Processo Editorial [Ferreira and Silva 2013]

- **Editor**: Responsável por receber as revisões, organizar os artigos aceitos e gerar a publicação final;
- **Editora**: Organizam os editores e revisores, fazendo as chamadas para publicações, nas quais os autores submetem seus artigos. Dependendo da organização do evento, os autores e leitores são taxados de alguma forma para realizarem a publicação ou leitura do trabalho;
- **Leitor**: Acessa os artigos que foram publicados, podendo eventualmente pagar por isso.

Com o avanço das tecnologias de processadores de texto e diagramação, como (LaTeX[2]), os trabalhos dos editores e das próprias editoras são minimizados, retirando todo o esforço manual do processo de diagramação por parte dos editores, restando a estes a escolha dos revisores e unificação das notas; e às editoras a escolha dos editores.

O processo de publicação é iniciado com a submissão do artigo seguida da distribuição destes para os revisores previamente cadastrados, alguns sistemas já fazem esse trabalho automaticamente, como o *JEMS*[3], substituindo o fator humano que pode influenciar no processo. Na sequência, cada revisor recebe uma notificação de que uma revisão foi solicitada podendo aceitar ou negar. Esta dinâmica entre editoras, revisores e autores pode variar já que existem

[2]https://www.latex-project.org/
[3]https://jems.sbc.org.br/

inúmeros modelos de revisões propostos por pesquisadores. Entre os principais modelos podemos citar:

- **Halfblind**: Somente os revisores conhecem os nomes dos autores do artigo.
- **Fullblind**: Nem o autor, nem o revisor sabem a identidade um do outro.
- **Open peer review**: A identidade dos autores e dos revisores são reveladas, tornando conhecido todos os envolvidos no processo;
- **Triple-blind**: A identidade dos autores, revisores e editores é oculta;
- **Crowdsourced peer review**: Permite que a comunidade em geral contribua para o processo de revisão;
- **Open pre-review manuscripts**: Os trabalhos ficam acessíveis (via Internet) antecipadamente, ou em sincronia com qualquer procedimento formal de revisão por pares. Normalmente, esse tipo de revisão é usado em repositórios institucionais.

Um levantamento abordando a avaliação por pares lista a saturação dos revisores em relação aos métodos existentes e alguns vícios de avaliações, já que no meio acadêmico, um grupo de estudo normalmente reconhece o trabalho ou a área de atuação que está envolvida um outro grupo, o que pode acabar gerando vantagens avaliativas comparado com grupos acadêmicos de estudos não tão conhecidos. Está claro que há disfunções no processo clássico [Starbuck 2005], mas é possível melhorar.

3.5 Engenharia de Software

Quase todos os países possuem sistemas complexos de computadores, bem como nossos produtos e serviços que tendem a automatização. Portanto, produzir e manter um software controlando os custos, é essencial para a economia nacional e internacional.

Para Pressman Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software [Pressman and Maxim 2016].

Sommerville define a engenharia de software como uma disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação [Sommerville 2011].

A importância da engenharia de software não se limita apenas ao projeto em desenvolvimento, mas também á comunidade científica e a sociedade, pois padrões de desenvolvimento e organização tornaram-se cada vez mais necessários para a obtenção de um software competitivo. [FURTADO and DA COSTA 2010]

Jalote refere-se a base da engenharia de software como os conjuntos de atividades para o processo de desenvolvimento de software. A existência de vários tipos de processo de

desenvolvimento de software e podemos dizer para resolver o problema do software usam estas atividades tais como: analise de requisito, design do software, código e teste [Jalote 2005].

3.6 Processo de Software

Um bom software é desenvolvido, a partir, da união de boas práticas da engenharia de software, com um processo de desenvolvimento consistente. Requisitos diferentes em determinados projetos exigem diferentes processos de desenvolvimento, isto é, há sistemas que podem ser desenvolvidos em paralelo com a especificação, outros softwares, por razões de segurança, por exemplo, devem ser completamente especificados antes de começar o desenvolvimento.

O uso de um processo de software inadequado pode reduzir a qualidade ou a utilidade do produto de software a ser desenvolvido e/ou aumentar os custos de desenvolvimento. Este fato leva as organizações que produzem software a usar processos de desenvolvimento que sejam eficientes e que atendam plenamente suas necessidades [Sommerville 2011].

3.7 Modelo de Processo de Software

Modelos de processo podem incluir atividades que fazem parte do processo de software, produtos de software, ex. descrições arquiteturais, código-fonte, documentação do usuário e as funções das pessoas envolvidas na engenharia de software.

Modelos de processo de software podem ser considerados estruturas de processo que podem ser estendidas e adaptadas para criar processos de engenharia de software mais específicos. Os modelos de processos amplamente utilizados na atual prática de engenharia de software são:

- **O modelo em cascata:** Neste modelo de processo de software, as atividades fundamentais de processo de especificação, desenvolvimento, validação e evolução são fases sequenciais do processo, tais como especificação de requisitos, projeto de software, implementação, teste.
- **Desenvolvimento evolutivo:** Esta abordagem intercala as atividades de especificação, desenvolvimento e validação. Um sistema inicial é desenvolvido rapidamente a partir de especificações abstratas. Em seguida, o sistema inicial é refinado pelas entradas do cliente para produzir um sistema que atenda às necessidades do cliente.
- **Engenharia de software baseada em componentes:** Os modelos de processo que usam essa abordagem são baseados na existência de um número significativo de componentes reutilizáveis. O processo de desenvolvimento do sistema se concentra na integração desses componentes em um sistema, em vez de desenvolvê-los.

Estes modelos não são mutuamente exclusivos e são frequentemente usados juntos, especialmente para o desenvolvimento de sistemas grandes. Sub-sistemas podem ser desenvolvidos usando diferentes abordagens. Projetos grandes podem usar diferente (ou vários) modelos de processo de software para desenvolver diferentes partes do software.

3.7.1 Método Ágil

Segundo Sommerville, os métodos prescritivos dominavam a área de engenharia de software no final de 1980 [Sommerville 2011], isto é, o desenvolvimento de um software levava em consideração rigorosas prescrições de segurança e planejamento e outros aspectos que tornavam o processo todo controlado e severo.

A insatisfação com essas abordagens levou vários desenvolvedores de software nos anos 90 a propor novos métodos. Isso permitiu que a equipe de desenvolvimento se concentrasse no software em si, e não em seu design e documentação. Os métodos ágeis dependem universalmente de uma abordagem iterativa para especificação, desenvolvimento e entrega de software e foram projetados principalmente para suportar o desenvolvimento de aplicativos de negócios em que os requisitos do sistema geralmente mudam rapidamente durante o processo de desenvolvimento.

Em fevereiro de 2001, desenvolvedores de software se reuniram em Utah para discutir métodos de desenvolvimento leves. Eles publicaram o Manifesto para o Desenvolvimento Ágil de Software para definir a abordagem agora conhecida como desenvolvimento de software ágil. O Manifesto Ágil diz o seguinte:

> "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais:
>
> - Indivíduos e interações do que processos e ferramentas;
> - Software em funcionamento do que documentação abrangente;
> - Colaboração do cliente do que negociação de contrato;
> - Respostas a mudanças do que seguir um plano;
>
> Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda."

Outras abordagens ágeis incluem *Extreme Programing*, *Crystal*, *Scrum* [Cohn 2000], Desenvolvimento de Software Adaptativo, DSDM e Desenvolvimento Orientado a Recursos. Embora esses métodos ágeis sejam todos baseados na noção de desenvolvimento incremental e entrega, eles propõem diferentes processos para conseguir isso. No entanto, eles compartilham um conjunto de princípios e, portanto, têm muito em comum.

3.7.2 Scrum

O *Scrum* é uma estrutura de desenvolvimento de software ágil e incremental para gerenciar projetos de software e desenvolvimento de produtos ou aplicativos. Esta metodologia

permite que as equipes se auto-organizem, incentivando a co-localização física de todos os membros da equipe e a comunicação diária face a face entre todos os membros da equipe e as disciplinas do projeto. Um princípio fundamental do *Scrum* é o reconhecimento de que, durante um projeto, os clientes podem mudar de ideia sobre o que querem e precisam, e que desafios imprevisíveis não podem ser facilmente abordados de maneira tradicional ou planejada. O Scrum se concentra em maximizar a capacidade da equipe de entregar rapidamente e responder aos requisitos emergentes. Esse método é formado por artefatos e eventos:

- *Backlog* do Produto: é uma lista de prioridades de tudo que é necessário para se ter o produto funcionando, baseada nos requisitos;
- *Backlog* da *Sprint*: são as tarefas que serão realizadas em uma determinada *Sprint*;
- Incremento: é a soma de todas as tarefas realizadas em determinada *Sprint* com as tarefas de *Sprints* anteriores;
- *Sprint*: são ciclos com duração de um mês ou menos. Este assunto é melhor tratado mais à frente nesta seção.
- Revisão da *Sprint*: realizado ao final de cada *Sprint*, para avaliar se os objetivos foram alcançados. O refinamento do *Backlog* também é feito nesta fase.
- Retrospectiva da *Sprint*: é um evento de reavaliação por parte do Time *Scrum*, onde este pode ver em quais pontos podem melhorar. Este evento ocorre após a revisão da *Sprint*.

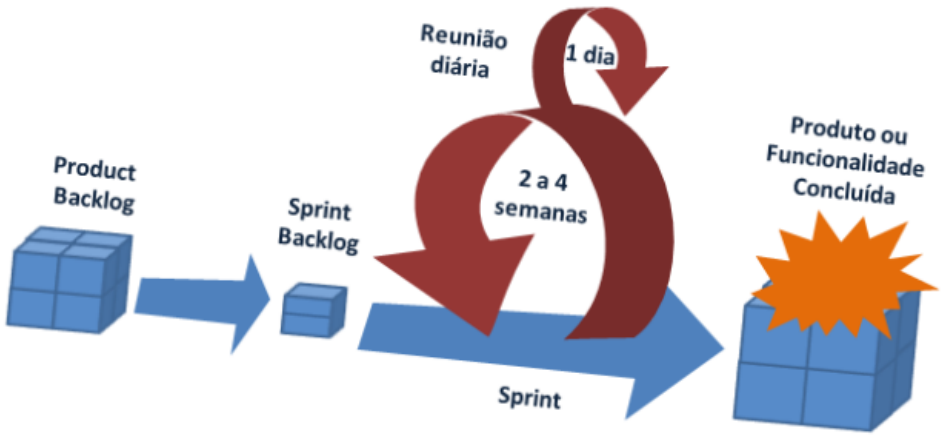

Figura 3.5: Scrum: A Metodologia Ágil [http://www.mindmaster.com.br/scrum/]

3.8 Engenharia de Requisitos

Os requisitos para um sistema são as descrições do que o sistema deve fazer, os serviços que ele fornece e as restrições em sua operação. Esses requisitos refletem as necessidades dos clientes de um sistema que atende a um determinado propósito, como controlar

um dispositivo, fazer um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado de engenharia de requisitos [FURTADO and DA COSTA 2010]. O termo "requisito"não é usado consistentemente na indústria de software. Em alguns casos, um requisito é simplesmente uma declaração abstrata de alto nível de um serviço que um sistema deve fornecer ou uma restrição em um sistema. No outro extremo, é uma definição detalhada e formal de uma função do sistema. Davis explica porque estas diferenças existem:

> "Se uma empresa deseja fechar um contrato para um grande projeto de desenvolvimento de software, deve definir suas necessidades de maneira suficientemente abstrata, para que uma solução não seja predefinida. Os requisitos devem ser escritos para que vários contratantes possam opinar sobre o contrato, oferecendo, talvez, maneiras diferentes de atender às necessidades da organização do cliente. Uma vez que um contrato tenha sido fechado, o contratado deve escrever uma definição de sistema para o cliente com mais detalhes para que o cliente compreenda e possa validar o que o software fará. Ambos os documentos podem ser chamados de documento de requisitos para o sistema. [Davis 1995] "

3.8.1 Documentação dos Requisitos

Para apresentação do software a futuros profissionais, a documentação dos requisitos deve possuir um modelo padrão, conhecido e aceito pela comunidade acadêmica, a *UML*, abreviação de *Unified Modeling Language*, uma linguagem de modelagem padronizada que consiste em um conjunto integrado de diagramas, desenvolvido para ajudar desenvolvedores de sistemas e software a especificar, visualizar, construir e documentar os artefatos de sistemas de software.

3.8.2 UML

A *UML* representa uma coleção das melhores práticas de engenharia que se mostraram bem-sucedidas na modelagem de sistemas grandes e complexos. A *UML* é uma parte muito importante do desenvolvimento de software orientado a objetos e do processo de desenvolvimento de software. A *UML* usa principalmente notações gráficas para expressar o design de projetos de software. Ou, como Booch et al explica em seu livro *The Unified Modeling Language User Guide*:

> "*A Unified Modeling Language* (*UML*) é uma linguagem padrão para escrever esquemas de software. A *UML* pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema intensivo em software [FURTADO and DA COSTA 2010]. A *UML* é apropriada para sistemas de modelagem que variam de sistemas de informações corporativas a aplicativos distribuídos baseados na *Web* e até mesmo a sistemas incorporados em tempo real. É uma linguagem muito expressiva, abordando todas as visões

> necessárias para desenvolver e depois implantar tais sistemas. Mesmo sendo expressivo, a *UML* não é difícil de entender e usar. Aprender a aplicar a *UML* efetivamente começa com a formação de um modelo conceitual da linguagem, que requer o aprendizado de três elementos principais: os blocos de construção básicos da *UML*, as regras que ditam como esses blocos podem ser montados e alguns mecanismos comuns que se aplicam ao longo língua [Booch et al. 2006]."

Os atuais padrões *UML* exigem 13 tipos diferentes de diagramas: classe, atividade, objeto, caso de uso, sequência, pacote, estado, componente, comunicação, estrutura composta, visão geral de interação, tempo e implantação. Esses diagramas são organizados em dois grupos distintos: diagramas estruturais e diagramas comportamentais ou de interação.

Diagramas *UML* Estruturais:

- Diagrama de classes
- Diagrama de pacotes
- Diagrama de objeto
- Diagrama de componentes
- Diagrama de estrutura composta
- Diagrama de implantação

Diagramas *UML* Comportamentais:

- Diagrama de atividades
- Diagrama de sequência
- Diagrama de casos de uso
- Diagrama de estado
- Diagrama de comunicação
- Diagrama de visão geral de interação
- Diagrama de tempo

Este trabalho apresentará os diagramas de classes, atividades, sequência e casos de uso. Os diagramas citados são os mais utilizados na especificação de *software*.

3.8.3 Diagrama de Casos de uso

Na *Unified Modeling Language* (UML), um diagrama de casos de uso resume os detalhes dos atores do sistema e suas interações com o sistema. Neste digrama utiliza-se um conjunto de símbolos e conectores especializados. Um diagrama de casos de uso efetivo pode ajudar a discutir e representar cenários nos quais seu sistema ou aplicativo interage com pessoas, organizações ou sistemas externos; objetivos do sistema ajuda essas atores a alcançar; e o escopo do sistema. A Figura 3.6 é o diagrama de caso de uso do sistema editorial descentralizado sobre *Blockchain* que contextualiza os atores (Author, Revisor e o software) interagindo com as funcionalidades básicas do sistema.

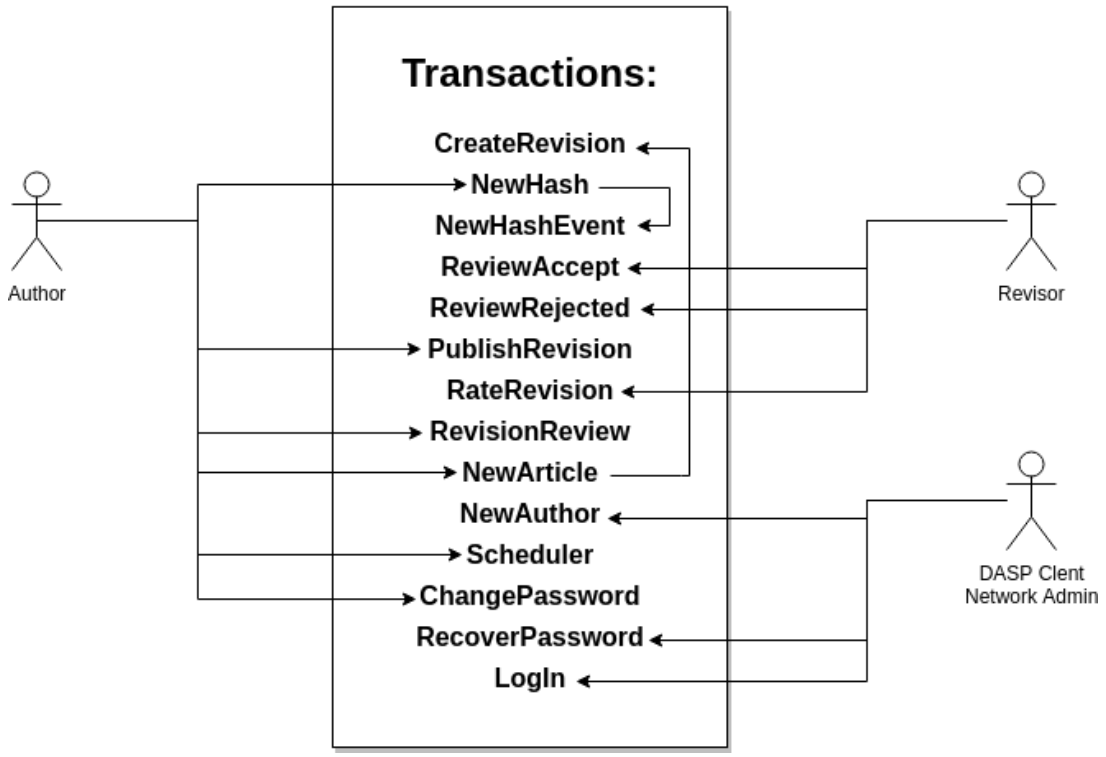

Figura 3.6: Exemplo de Casos de uso

3.8.4 Diagrama de Atividades

O diagrama de atividades, segundo Booch, é importante para descrever aspectos dinâmicos do sistema [Booch et al. 2006]. O diagrama de atividades é essencialmente uma versão avançada do fluxograma que modela o fluxo de uma atividade para outra. Abaixo, a Figura 3.7 mostra um exemplo de diagrama de atividades da aplicação, que mostra as atividades principais que o sistema deve suportar.

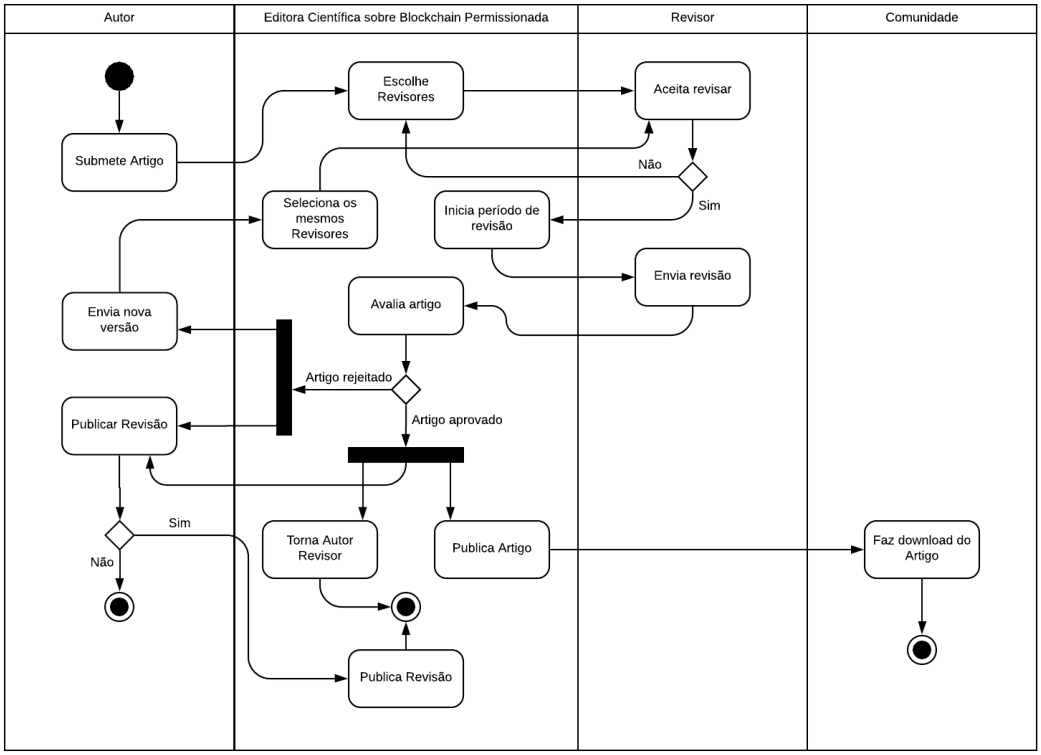

Figura 3.7: Exemplo de Diagrama de Atividades

3.8.5 Diagrama de Classes

Os diagramas de classes (Figura 3.8) são um dos tipos mais úteis de diagramas na *UML*, pois mapeiam claramente a estrutura de um determinado sistema, modelando suas classes, atributos, operações e relacionamentos entre objetos. Booch o define: "[...] um conjunto de classes, interfaces, colaborações e seus relacionamentos. Graficamente, um diagrama de classes é uma coleção de vértices e arcos [...]". [Booch et al. 2006]

3.8.6 Diagrama de Sequência

Os diagramas de sequência *UML* (Figura 3.9) modelam o fluxo de lógica dentro de seu sistema de maneira visual, permitindo documentar e validar a lógica. Este diagrama é o artefato *UML* mais popular para modelagem dinâmica, que se concentra na identificação do comportamento dentro de seu sistema.

Geralmente o diagrama de sequência tem como objetivo modelar interações de alto nível entre objetos ativos em um sistema e as interações entre instâncias de objetos que realizam um caso de uso ou uma operação. Também abrange interações genéricas de modelo (mostrando todos os caminhos possíveis por meio da interação) ou instâncias específicas de uma interação (mostrando apenas um caminho através da interação) [FURTADO and DA COSTA 2010].

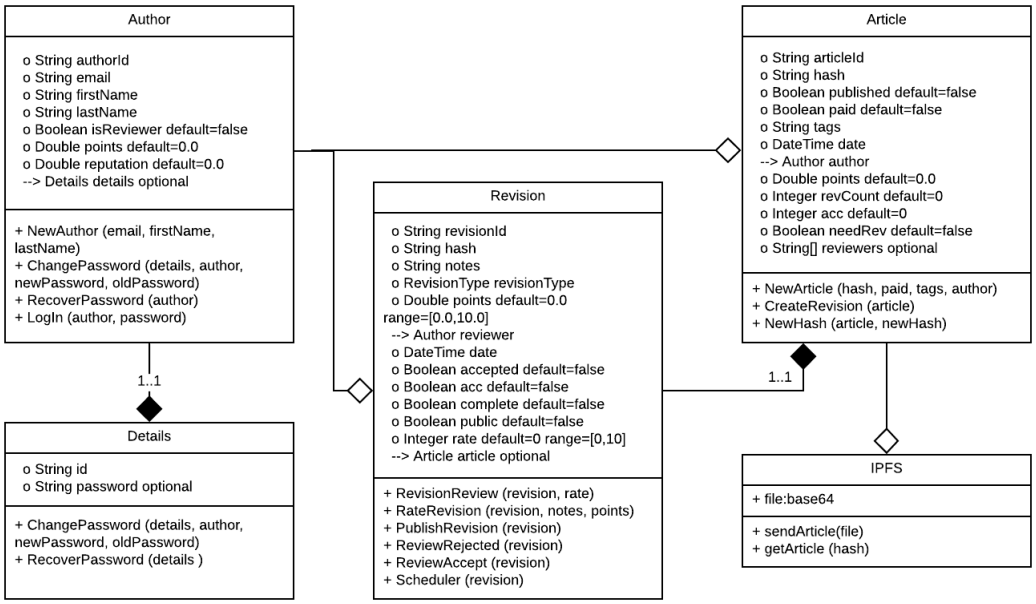

Figura 3.8: Exemplo diagrama de Classes

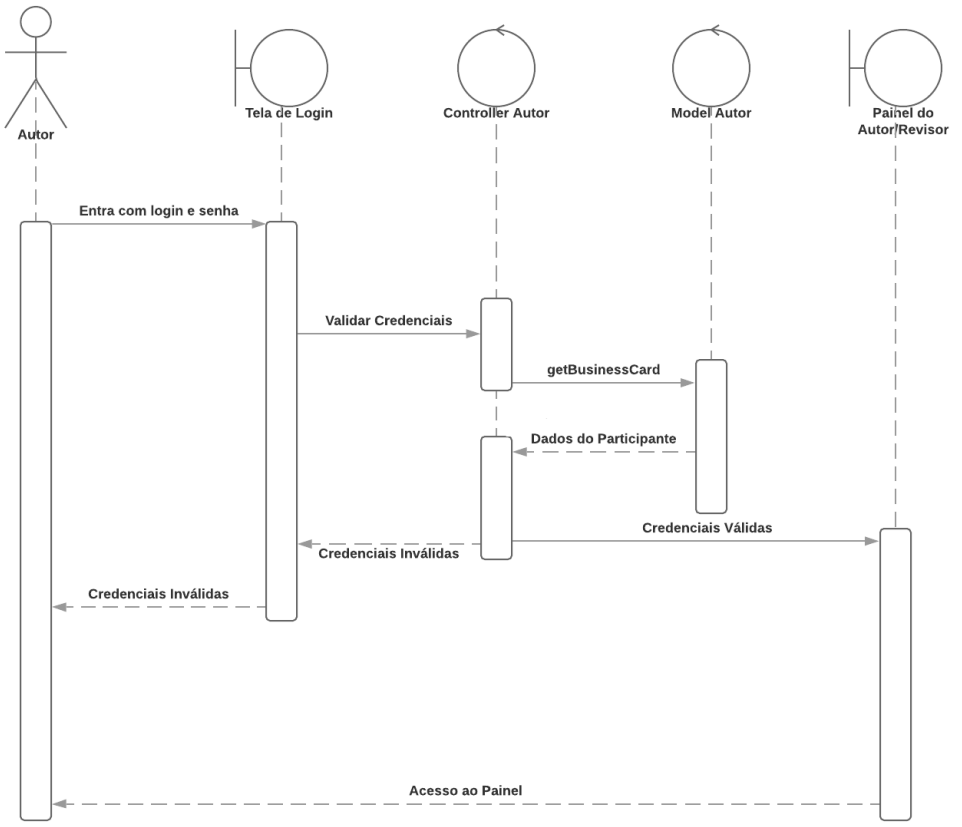

Figura 3.9: Exemplo de diagrama de Sequência

CAPÍTULO 4

FERRAMENTAS E TECNOLOGIAS UTILIZADAS

Este Capítulo trata das principais tecnologias e ferramentas utilizadas para desenvolver o software. Para trabalhar com a rede *Fabric*, a aplicação foi desenvolvida majoritariamente com a linguagem de programação *Javascript*, porém possui linguagens próprias das ferramentas como uma linguagem orientada a objeto para definição de modelos (.cto) em uma rede de negócios e a linguagem que faz o controle de acesso da rede (.acl).

A ferramenta *Composer do Hyperledger* é utilizado para facilitar os testes e a criação da rede *Blockchain* em si. O *Composer* trabalha diretamente com o *Docker* instanciando os recipientes e as imagens de cada parte da rede necessária para seu funcionamento. O *Trello* foi utilizado para gerenciar tarefas referentes a conclusão do projeto com as técnicas do método ágil descrito.

4.1 Hyperledger Fabric

O *Hyperledger Fabric* é um dos *frameworks* do projeto *Hyperledger*, é uma estrutura *Blockchain*. Esta estrutura funciona como uma base para o desenvolvimento de aplicativos ou soluções com uma arquitetura modular, permitindo que componentes, como consenso e serviços de associação, sejam *plug-and-play*. O *Hyperledger Fabric* aproveita a tecnologia de contêiner para hospedar contratos inteligentes chamados *"chaincode"*, que compõem a lógica de aplicação do sistema. *O Hyperledger Fabric* foi inicialmente contribuído pela Digital Asset e IBM, como resultado do primeiro *hackathon*.

O *Fabric* é uma implementação de *Blockchain* privada voltada para aplicações empresariais com objetivo de atender aos múltiplos e variados requisitos dos aplicativos de negócio, alavancar os estudos da tecnologia *Blockchain* e permitir a escalabilidade de serviços nele baseados [Androulaki et al. 2018]. Como já discutido, a arquitetura modular baseada em *Blockchain*, oferece altos graus de confiabilidade, resiliência, flexibilidade e escalabilidade [Nguyen et al. 2017].

A arquitetura do *Hyperledger Fabric* elimina o consenso em seu próprio componente - o serviço de ordenação. Esse é um recurso incomum porque significa que o *Fabric* pode suportar diferentes algoritmos de consenso simplesmente desativando a implementação do ordenador. Nem toda plataforma *blockchain* suporta isso. Essa arquitetura permite que os usuários

escolham um serviço de ordenação que implemente um algoritmo de consenso adequado a sua aplicação. Uma propriedade desejável é a tolerância a falhas bizantinas (*BFT, Bizantine fault tolerance*), que diz que o solicitante pode fazer seu trabalho mesmo na presença de agentes maliciosos.

É importante notar que o BFT implementado, no entanto, só se aplica à ordenação de transações no *Hyperledger Fabric*. Seu trabalho é garantir que cada par tenha a mesma lista de transações em seu livro. Para entrar no livro, em primeiro lugar, as transações precisam primeiro passar pelas políticas de endosso e endossar os pares que executaram o *chaincode* das transações. Depois que eles estão no *ledger*, as transações são verificadas uma última vez pela etapa de validação de cada *peer*. A cada passo do caminho, a *Fabric* possui um sistema para impedir que coisas ruins aconteçam.

4.2 Hyperledger Composer

O *Hyperledger Composer* é um extenso conjunto de ferramentas de desenvolvimento aberto para facilitar o desenvolvimento de aplicativos *Blockchain* com objetivo principal de acelerar o tempo de desenvolvimento e facilitar a integração de aplicativos *Blockchain* aos sistemas de negócios existentes. O *Composer* permite modelar sua rede de negócios e integrar sistemas e dados existentes com seus aplicativos *Blockchain*.

Esta ferramenta pode ser utilizada para modelar rapidamente uma rede comercial, contendo ativos, transações e participantes, os quais podem ser associados a uma identidade única, em várias redes de negócios. Ativos são bens tangíveis ou intangíveis; ou serviços. Além da possibilidade de definir transações que podem interagir com ativos e participantes.

4.3 Hyperledger Composer Playground

O *Hyperledger Composer Playground* é uma interface de usuário para configuração, implementação e teste de uma rede comercial, contida no *Hyperledger Composer*. Construído em *NodeJS*, os recursos do Playground permitem que os usuários gerenciem a segurança da rede de negócios, convidem os participantes para redes de negócios e conectem-se a várias redes de negócios *Blockchain*.

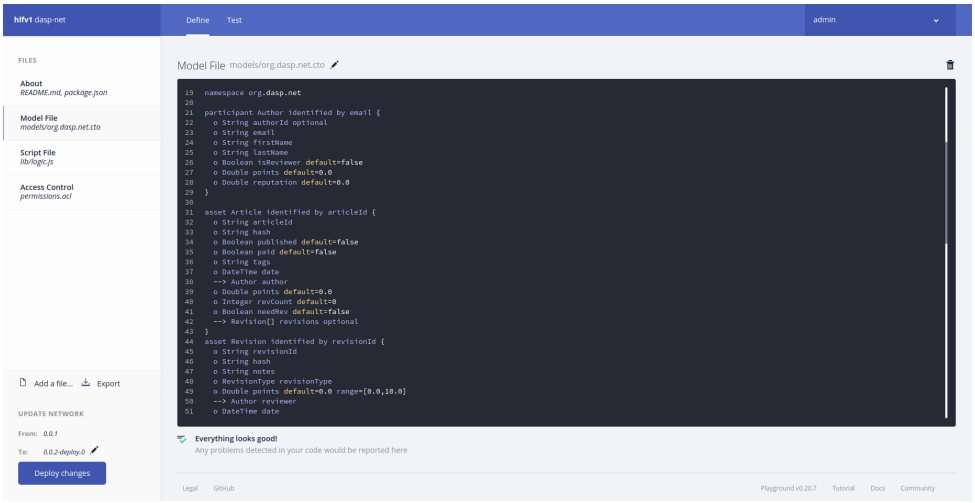

Figura 4.1: Hyperledger Composer Playground

A natureza *open source* desta interface, permitiu que fosse estudada, planejada e implementada uma solução de teste para a prova de conceito do *DASP* em tempo ágil.

4.4 IPFS - Interplanetary File System

O *IPFS* é um sistema distribuído para armazenar e acessar arquivos, sites, aplicativos e dados. Distribuído pois dispõe de tecnologias que não precisam buscar os dados em fontes distantes, mas sim nas fontes mais próximas e você, isto é, distribui bem os dados. Isso é possível para páginas da Web, mas também para qualquer tipo de arquivo que um computador possa armazenar, seja um documento do PDF, um e-mail, um arquivo MP3 ou um registro do banco de dados.

O conceito de distribuição da ferramenta torna possível baixar um arquivo de vários locais que não são gerenciados por uma única organização e isso possui vantagens:

- Torna difícil para um site ficar offline. Se alguém atacar os servidores da Web do sistema você ainda poderá obter a mesma página em outro lugar.
- Torna mais difícil para as autoridades censurarem o conteúdo. Como os arquivos no *IPFS* podem vir de muitos lugares e porque alguns desses locais podem estar próximos, é muito difícil para as autoridades (sejam estados, corporações ou outra pessoa) bloquear as coisas.
- Pode acelerar a Web quando você está longe ou desconectado. Você pude recuperar um arquivo de alguém próximo em vez de centenas ou milhares de quilômetros, isto é especialmente valioso se sua comunidade estiver conectada em rede, mas não tiver uma boa conexão com a Internet mais ampla.

Por trabalhar bem com arquivos hipermídia, o *IPFS* foi utilizado para armazenar os artigos PDF no sistema desenvolvido neste trabalho. Dessa forma os artigos estarão sempre disponíveis em uma rede distribuída e conta com as vantagens citadas acima.

4.5 Node-RED

O *Node-RED* é uma ferramenta visual de programação baseada em fluxo, utilizado para conectar dispositivos de hardware, *APIs* e serviços online como parte da Internet das Coisas. O *Node-RED* não pode ser usado apenas para aplicativos da Internet das coisas, mas é um mecanismo genérico de processamento de eventos. Pode ser utilizado para interagir com eventos de *http*, *websockets*, *TCP*, *Twitter* e muito mais, além armazenar esses dados em bancos de dados sem ter que programar muito.

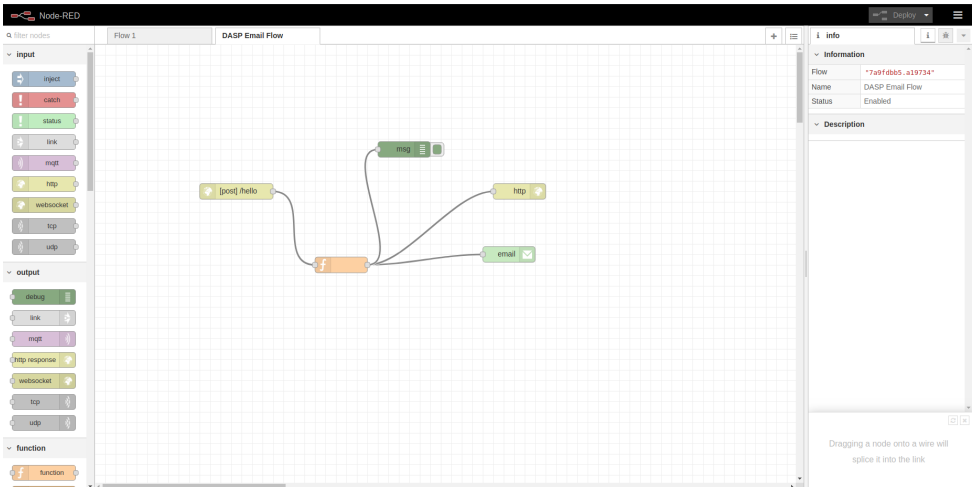

Figura 4.2: Node-RED - Flow para envio de e-mail

No sistema proposto por este trabalho, o *Node-RED* foi utilizado para gerir um fluxo no qual ao ser efetuado um *POST* em uma rota *HTTP* especificada, o sistema executa uma função que finaliza enviando um e-mail com os valores determinados pelo *POST*. Isto serve para que os revisores e autores sejam notificados sobre o estado da aplicação. Exemplo: Ao se cadastrar, ao receber uma revisão para revisar, quando o artigo do autor é publicado.

CAPÍTULO 5

TRABALHOS RELACIONADOS

Antes do desenvolvimento da ferramenta propriamente descrita, realizou-se pesquisas com o objetivo de encontrar ferramentas semelhantes ao apresentado neste trabalho, isto é, propostas de editoras científicas descentralizadas. A Tabela 5.1 apresenta as expressões utilizadas para obter esses resultados que serão detalhados mais a frente.

5.1 Sistemas Editoriais que utilizam Blockchain

Os sistemas editoriais mais utilizados atualmente, possuem características que podemos classificar seu funcionamento partindo de sistemas centralizados desiguais, até sistemas descentralizados igualitários. Essa classificação deve-se principalmente a cinco recursos e funcionalidades: Gestão de processos, Banco de Dados, Auditoria, Direitos Autorais e Inovação. Este capítulo trata de sistemas que possuem a mesma proposta: Ferramenta para publicação científica, utilizando *Blockchain*.

Foram realizadas pesquisas objetivando encontrar trabalhos relacionados ou ferramentas com algum nível de implementação, semelhantes ao apresentado neste trabalho. A tabela acima apresenta as expressões de buscas utilizadas, a plataforma de busca e os resultados obtidos.

Expressão pesquisada	**Buscador**	**Resultado**
scientific publisher blockchain	Google	EUREKA por ScienceMatters **https://eurekatoken.io/** **"A plataforma de ciência aberta alimentada por blockchain."**
scientific research decentralized	Google	DEIP **https://deip.world/** **"Uma plataforma de pesquisa descentralizada."**
blockchain open science network	Google	Open Science Network - OSN **https://open-science-network.github.io/** **"A Open Science Network (OSN) é uma rede de pesquisa distribuída"**

Tabela 5.1: Ferramentas relacionadas encontradas

5.1.1 EUREKA

EUREKA é uma plataforma de revisão e classificação de artigos científica alimentada pelo *token ERC-20* (um padrão técnico usado para contratos inteligentes no *blockchain* da *Ethereum* para implementação de tokens.) EUREKA da EKA *Blockchain* Solutions GmbH. Utiliza a *Blockchain* do *Ethereum* para tornar os resultados da pesquisa imutáveis, transparentes e descentralizados.

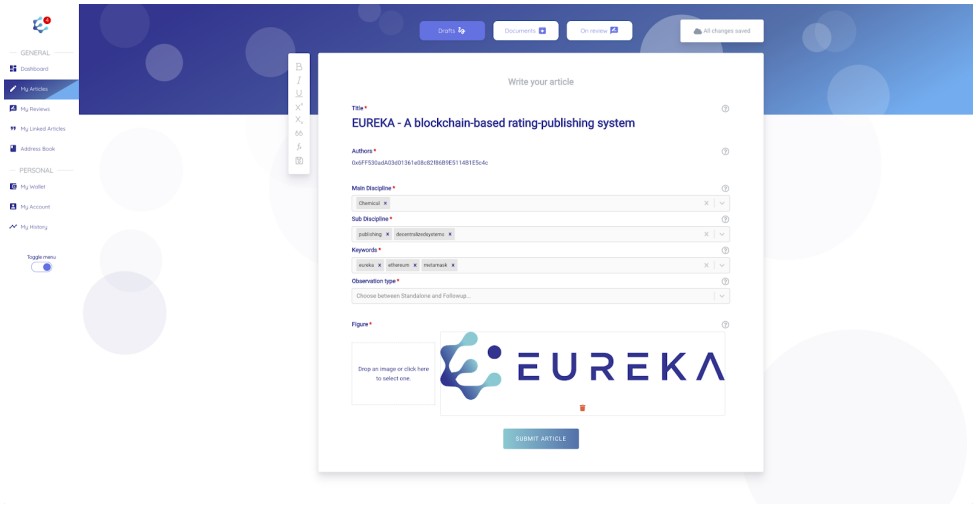

Figura 5.1: O editor on-line da plataforma EUREKA

Apesar de ser baseado em *tokens* na rede *Ethereum*, o que possibilita a descentralização de alguns dados, a rede *Ethereum*, diferente da rede *IPFS* utilizada na aplicação desenvolvida neste trabalho, não permite o armazenamento de hipermídia em sua cadeia de blocos. Sendo assim, a equipe da plataforma científica de publicação online *ScienceMatters*, que está por trás da plataforma *EUREKA*, apenas garante ao autor o registro da ideia ou invenção, a partir da marca de tempo da transação na *Blockchain*. Para o armazenamento dos artigos, a plataforma possui um banco de dados centralizado em *MongoDB* o que implica também que os direitos autorais de fato, são da *ScienceMatters*.

Também diferente do *Hyperledger Fabric* utilizado no sistema proposto este trabalho, cada transação na rede *Ethereum* possui, obrigatoriamente, um custo por cada transação e deve ser paga pelo emissor. Isso dificulta o acesso, a publicação e a revisão desses artigos além de contrariar alguns princípios da ciência aberta.

5.1.2 DEIP

O *DEIP* é uma plataforma de pesquisa descentralizada governada pela comunidade científica. Fornece aos seus membros ferramentas para publicar suas pesquisas, acesso aberto a todo o material publicado, um sistema de revisão descentralizado e mecanismos progressivos de financiamento. Sua infra-estrutura é totalmente descentralizada. Os participantes podem usar a plataforma sem quaisquer comissões ou intermediários.

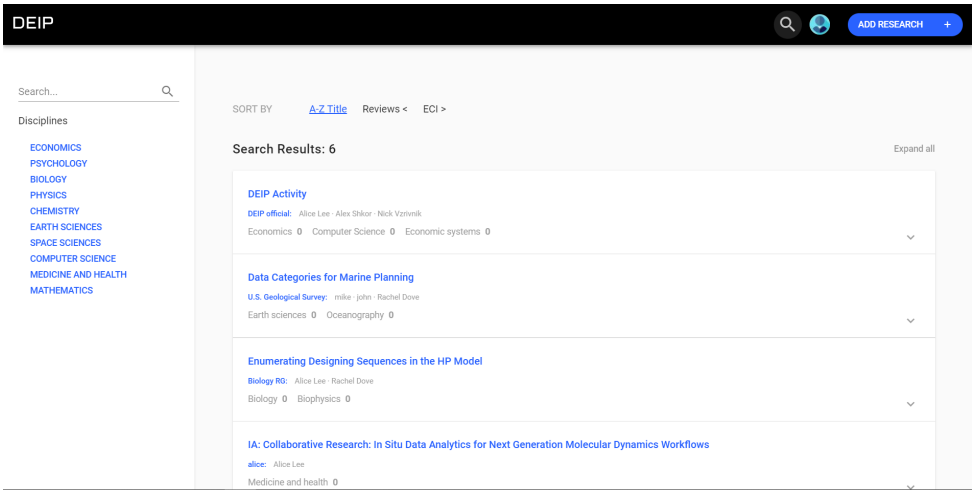

Figura 5.2: Beta DEIP: Decentralized Research Platform

Diferente da ferramenta *EUREKA*, a descentralização desta plataforma se deve a construção de uma rede *blockchain* a partir do código base da rede *STEEM* (uma plataforma *blockchain* que possibilita fluxos de receita imediatos para os usuários, recompensando-os pelo compartilhamento de conteúdo.). As implicações desse tipo de abordagem é que a *Steem* possui regras de funcionamento pré estabelecidas em sua rede de negócios. Como exemplo podemos citar a recompensa por publicação, o usuário não pode especificar o valor do acesso ao seu artigo ou lucrar por tempo indeterminado com sua publicação.

5.1.3 Open Science Network - OSN

A *Open Science Network* é um protocolo aberto compartilhado em *blockchain*, no qual pesquisadores, universidades e instituições governamentais podem interagir efetivamente com menores barreiras à entrada e reduzir o atrito em cada etapa do processo.

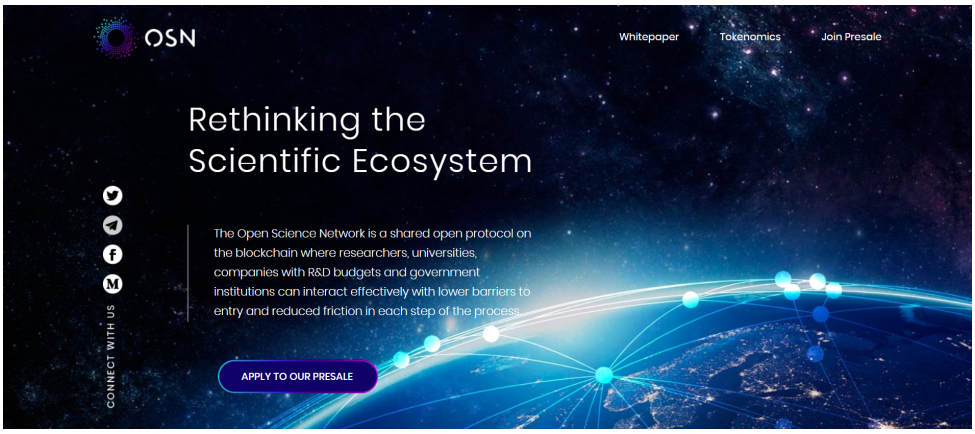

Figura 5.3: Website Open Science Network - OSN

Ainda em fase inicial, este protocolo também utilizará *tokens ERC-20* na rede *Ethereum*, além das problematizações já citadas, o custo por transação, limita o acesso do pesquisador que não possui esses *tokens*, o fato de que das regras da rede *Ethereum* que não podem ser

contornadas, sua natureza não-permissionada permite a identificação das carteiras dos participantes das redes em suas transações, permitindo a corrupção humana do sistema.

5.2 Avaliação das Ferramentas que utilizam Blockchain para Revisão e Publicação de Artigos Científicos

Este capítulo relaciona a proposta de ciência aberta no cenário inovador para revisão e publicação de artigos científicos o qual determina o fácil acesso aos trabalhos e pesquisas científicas, com o intuito de gerar um quadro comparativo entre as ferramentas já existentes que utilizam *Blockchain* para alcançar a proposta citada. Considerando que o uso de *Blockchain* implica da Gestão de processo descentralizado e na auditoria, o estudo das ferramentas teve como base alguns requisitos básicos: Banco de Dados, Direitos autorais, Acesso, Múltipls Organizações e Lucratividade. A Tabela 5.2 mostra o comparativo das ferramentas inovadoras existentes baseado no contexto de ciência aberta e tecnologia.

Características	Trabalhos relacionados			
	EUREKA	DEIP	OSN	Proposta
Banco de Dados	Centralizado	Descentralizado	Descentralizado	Descentralizado
Direitos Autorais	Desigual	Igualitário	Igualitário	Igualitário
Lucratividade	Média/Ilimitada	Baixa/Limitada	Média/Ilimitada	Alta/Ilimitada
Acesso	Pago	Livre	Pago	Livre
Múltiplas Orgs.	Não permite	Não permite	Não permite	Permite
Rede Blockchain	Pública	Pública	Pública	Permissionada

Tabela 5.2: Comparação das propostas

A tabela expõe a falta de equilíbrio de atributos essenciais para o impulsionamento da ciência aberta por meio das atuais ferramentas de avaliação e publicação de artigos científicos que utilizam *Blockchain*.

Observamos que a ferramenta *EUREKA* possui um banco de dados tradicional para armazenar seus artigos e outras informações, o que implica na centralização de determinadas informações, além de não assegurar corretamente os direito autorais, já que os artigos estão armazenados nessa estrutura centralizada.

Tanto a *EUREKA* como a *OSN*, utilizam *tokens* na rede *Ethereum* para transacionar em *Blockchain*, isso provoca a necessidade de cobrar fundos para enviar essas transações. Ou seja, o usuário, em algum momento, deve dispor de *Tokens* para participar da rede e enviar transações, por isso consideram-se ferramentas com acesso pago.

A base do *DEIP* é a *Blockchain* de código aberto *STEEM*, nela o método de distribuição de fundos por publicação é limitada por sete dias, isto significa que durante esse período a valorização dependerá das interações da comunidade com o artigo, bem como a "reputação"(aqueles

que possuem mais fundos) dos que interagem. Além disso, a natureza pública da rede escolhida não está a salvo de corrupções humanas como as que advém da identificação de usuários pelo rastro de transações.

A capacidade de delegar funções para entidades, permite com que a ferramenta descrita neste trabalho seja implementada e instanciada em diversas instituições, promovendo a descentralização dos dados e aumentar a confiança da rede, além de permitir regras de negócios específicas para a entidade que instancia. Em todas, não temos a possibilidade de entidades diferentes possuírem regras de negócios próprios ou participarem da descentralização sem a necessidade de trocar ou prover recursos como nas redes do *Ethereum* e da *STEEM*.

CAPÍTULO 6

EDITORA CIENTÍFICA SOBRE UMA REDE BLOCKCHAIN PERMISSIONADA

Este capítulo tem como objetivo tratar da fase de especificação da ferramenta (incluindo, requisitos, análise e projeto) e sua implementação. Para alcançar os objetivos da proposta deste trabalho, foram gerados documentos que especificam os requisitos para o sistema funcionar utilizando personas para gerar casos de uso que serão utilizados para a diagramação do sistema, além de definir a arquitetura do sistema, o projeto de interação humano-computador (IHC), o projeto de banco de dados e o projeto de componentes quais são necessários para a implementação da ferramenta. Por fim, mostra o desenvolvimento prático da ferramenta e sua subsequente avaliação por usuários reais.

6.1 Documento de especificação de requisitos

Para se ter uma ideia inicial do que se deseja construir, o processo de desenvolvimento de software também inclui a criação de personas (personagens que representam usuários reais da aplicação) para identificar casos de uso e requisitos que o sistema deve abranger. Como este trabalho não visa um cliente específico, definiu-se duas personas que representam os possíveis usuários do sistema, o autor, que submete um artigo pela ferramenta e o revisor que acessa a plataforma para efetuar a revisão dos artigos submetidos.

O personagem Matheus representa todos os autores de artigos científicos que poderiam se beneficiar das funcionalidades da ferramenta do sistema proposto. Sua principal necessidade é a melhoria do processo de publicação, referente aos ganhos e ao tempo necessário para ter uma publicação garantida.

A personagem Bruna representa todos os cientistas que, através de uma plataforma editorial, revisam artigos e propostas afim de obter experiência, prestígio e reconhecimento com esse processo. Sua principal necessidade é a lucratividade, transparência nos processos e a segurança dos registros de suas revisões dentro da plataforma.

6.1.1 Ambiente do sistema

Este sistema é uma ferramenta para submissão, revisão e eventual publicação de artigos científicos com o objetivo de promover a descentralização do processo, removendo intermediá-

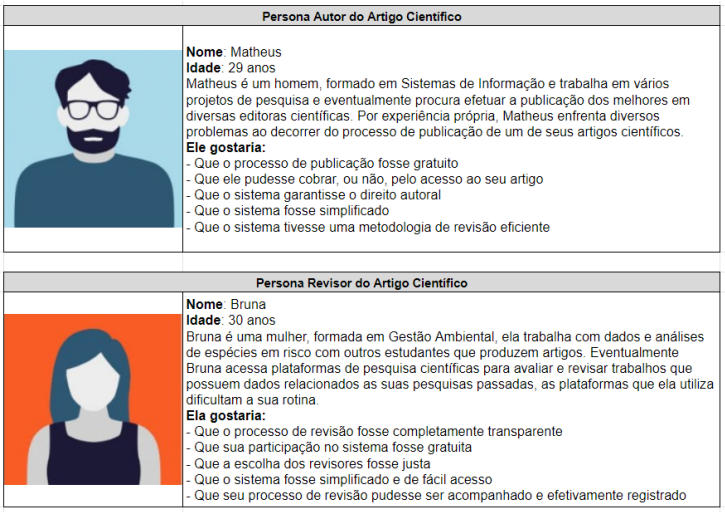

Figura 6.1: Personas Autor e Revisor

rios que antes selecionavam os revisores, calculavam as notas e controlavam as publicações, isto é, os editores e a própria editora, que agora é autônoma.

O Autor é uma pessoa interessada em adquirir prestígio no meio científico através de publicações em plataformas reconhecidas e pretende ter controle e direitos sobre isso. Ele é responsável pelo fluxo de envio de um arquivo para que o sistema distribua a revisores aleatoriamente, iniciando o ciclo de revisão.

O Revisor é uma pessoa que possui elevado grau de maestria e da mesma forma que o autor, tem interesse em adquirir mais publicações, seja para sua organização ou não. É responsável por avaliar os artigos e orientar o autor no sentido de melhoria do artigo científico.

Neste ambiente editoral descentralizado, que conecta Autores e Revisores diretamente, o autor cadastra o seu perfil, artigo e informações gerais. Para promover a qualidade do processo de revisão, foi determinado uma quantidade fixa de revisores que participarão da revisão, isto é, a plataforma deve convidar aleatoriamente exatamente 5 revisores para iniciar o processo de revisão, para garantir os 5 primeiros participantes do tipo revisor, uma das lógicas de contrato da rede é que os 6 primeiros cadastrados são obrigatoriamente revisores, um a mais para caso algum revisor enviar um artigo (o mesmo não pode participar do processo de revisão do seu próprio artigo). Nesse momento, os revisores escolhidos podem escolher se aceitam revisar ou não. Ao clicar em rejeitar o software convida outro revisor disponível. Caso ele clique em aceitar, o revisor inicia o processo de revisão baixando o artigo, revisando e preenchendo o formulário de avaliação na plataforma, com os campos para anotações e o campo do conceito. Caso o artigo seja publicado, para promover a autonomia do sistema sobre a inclusão de novos revisores, o autor do artigo se torna revisor.

6.1.2 Objetivos do produto

Esta ferramenta Web objetiva promover a descentralização no processo de publicação de artigos científicos. A plataforma *client* permite o cadastro de autores para que possam enviar

arquivos a um banco de dados descentralizado e executam transações de revisão e publicação que também são registradas em uma rede *blockchain* própria.

6.1.3 Benefícios

A ferramenta desenvolvida, possui as seguintes funcionalidades:

Painel de submissão como AUTOR

- Cadastro e manutenção do perfil;
- Recuperação de senha e dados;
- Envio de artigos;
- Lista de seus artigos
- Lista de revisões dos seus artigos
- Lista de revisões públicas
- Lista de artigos públicos

Painel de submissão como REVISOR

- Cadastro e manutenção do perfil;
- Recuperação de senha e dados;
- Envio de artigos;
- Lista de seus artigos
- Lista de revisões dos seus artigos
- Lista de revisões públicas
- Lista de artigos públicos
- Lista de revisões a serem feitas
- Lista de revisões completas
- Manutenção das revisões;

6.2 Diagramas de Casos de Uso

Nesta seção iremos pontuar cada caso de uso do sistema, considerando as personas envolvidas e os fluxos necessários para alcançar os principais objetivos dentro da plataforma.

6.2.1 Diagrama de Casos de Uso de Contexto

Nesta aplicação web, há dois atores interagindo: o Autor e o Revisor. No contexto do sistema temos um sistema de revisão e um subsistema de Autores conforme Figura a seguir.

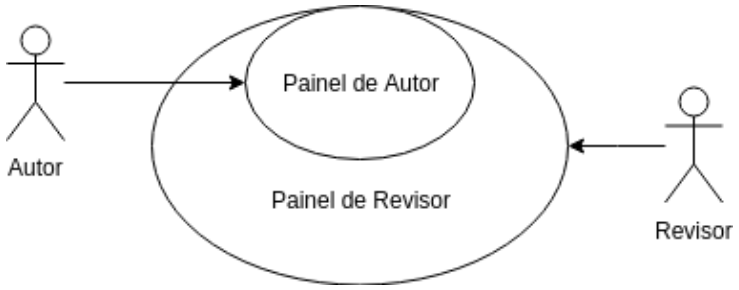

Figura 6.2: Diagrama de Casos de Uso de Contexto

A Figura acima apresenta o caso de uso de contexto do sistema. O Autor tem acesso subsistema de envio de artigos e quando se torna Revisor, tem acesso ao sistema completo, incluindo o processo de revisão.

6.2.2 Subsistema de Submissão de Artigos Científicos

CASO DE USO 1: CRIAR PERFIL Descrição: O Autor deverá fornecer informações como nome, sobrenome, e-mail e senha para que o sistema efetue seu registro no e conceda algum nível de acesso.

Ator primário: Autor

Precondições: Nenhuma

Fluxo Principal:

1. O autor acessa a área de cadastro do sistema
2. O autor preenche as informações necessárias
3. O autor envia o formulário

Pós-Condições: O Autor deverá receber um e-mail de cadastro efetuado com sucesso e ser redirecionado ao painel principal da ferramenta, onde ele pode iniciar o processo de submissão de artigos.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diário.

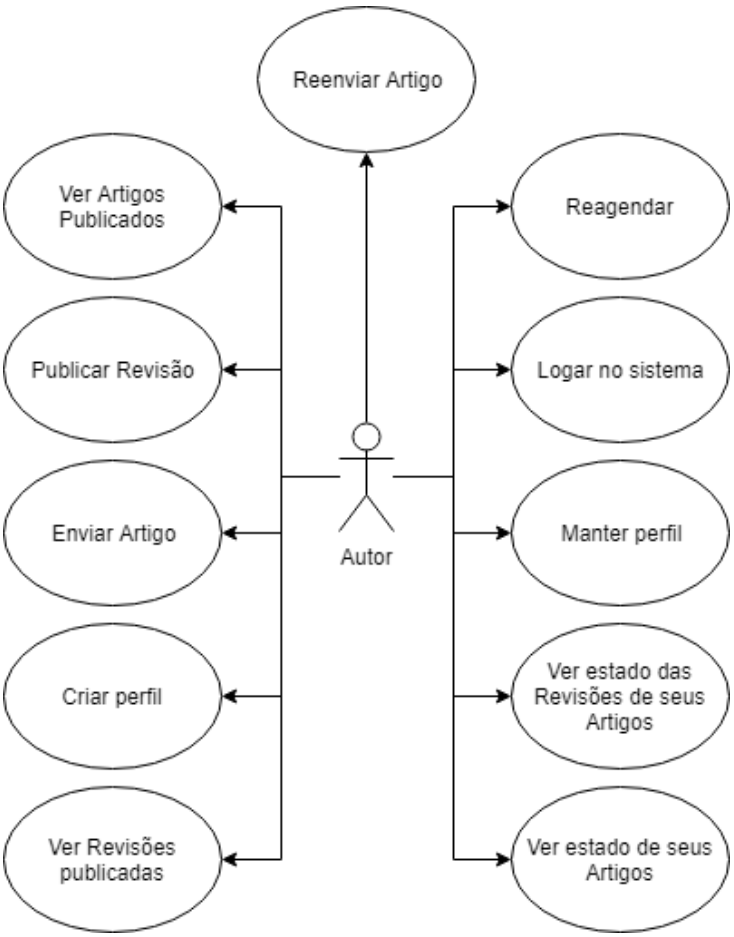

Figura 6.3: Diagrama de Casos de Uso do subsistema do Autor

CASO DE USO 2: LOGAR NO SISTEMA Descrição: O autor deverá entrar com login e senha e o sistema deverá validar suas credenciais permitindo acesso ou não ao sistema.

Ator primário: Autor

Precondições: O Autor já deve estar cadastrado no sistema.

Fluxo Principal:

1. O Autor entra com as credenciais de login e senha pré cadastradas.
2. O sistema processa as informações e valida o login.
3. O autor envia o formulário
4. Caso o login seja válido o usuário entra no painel principal de submissão.
5. Caso o login seja inválido o usuário não entra no painel principal e uma mensagem de erro é exibida.

Pós-Condições: Redirecionar ao painel principal caso o login seja válido.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diário.

CASO DE USO 3: MANTER PERFIL Descrição: O Autor é responsável pelas informações em seu perfil. Isso significa, por exemplo, a troca e recuperação de senha.

Ator primário: Autor

Precondições: O usuário deve estar logado ao sistema.

Fluxo Principal:

1. O Autor decide alterar sua senha.
2. O Autor acessa as informações de perfil.
3. O Autor clica em alterar senha.
4. O Autor preenche o formulário e envia.

Pós-Condições: Dados alterados com sucesso.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Eventual.

CASO DE USO 4: ENVIAR ARTIGO Descrição: O Autor submete um artigo científico de sua autoria através da ferramenta para que seja avaliado pela comunidade e eventualmente publicado.

Ator primário: Autor

Precondições: O usuário deve estar logado ao sistema, ter um artigo no formato *PDF* e o sistema possuir um mínimo de 6 pessoas cadastradas.

Fluxo Principal:

1. O Autor entra no painel principal.
2. O Autor clica em fazer upload.
3. O Autor preenche os campos do formulário e envia.

Pós-Condições: O Arquivo deve ser armazenado no *IPFS* e o *hash* do artigo deve ser armazenado em *blockchain* junto com as informações e vincular, todos esses dados, ao perfil do Autor. O artigo deve gerar 5 revisões cada uma com um Revisor convidado pelo sistema.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diário.

CASO DE USO 5: VER ESTADO DAS REVISÕES DE SEUS ARTIGOS Descrição: Dentro do sistema, o usuário pode ver a listagem das revisões que foram criadas a partir de seus artigos.

Ator primário: Autor

Precondições: O Autor estar logado no sistema.

Fluxo Principal:

1. O Autor entra no painel principal.
2. Clica em "My articles revisions".
3. Se houver artigos já submetidos, uma lista de revisões relacionadas aos seus artigos é exibida.

Pós-Condições: Exibir uma lista das revisões de cada artigo submetido.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Eventual.

CASO DE USO 6: VER ESTADOS DOS SEUS ARTIGOS Descrição: Dentro do sistema, o usuário pode ver a listagem dos artigos submetidos e os detalhes de cada artigo.

Ator primário: Autor

Precondições: O Autor estar logado no sistema.

Fluxo Principal:

1. O Autor entra no painel principal.
2. Clica em "My articles".
3. Se houver artigos já submetidos, uma lista desses artigos é exibida.

Pós-Condições: Exibir uma lista das revisões de cada artigo submetido.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diário.

CASO DE USO 7: REAGENDAR Descrição: O autor, se achar necessário, pode reagendar a revisão de um artigo autoral, se o revisor demorar mais que duas semanas para completar sua revisão ou aceitar revisar um artigo.

Ator primário: Autor

Precondições: Autor logado no sistema e uma proposta de revisão já ter expirado.

Fluxo Principal:

1. O Autor entra no painel principal.
2. Clica em "My articles revisions".
3. Se alguma revisão de seu artigo tiver expirado, um botão reagendar é exibido no cartão da revisão.
4. O Autor clica no botão e o sistema escolhe outro revisor disponível para revisar.

Pós-Condições: A revisão expirada deve receber um novo revisor, diferente dos que já estão envolvidos no mesmo processo de revisão.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Eventual.

CASO DE USO 8: VER ARTIGOS PUBLICADOS Descrição: Qualquer um que acessar o sistema, mesmo que não cadastrado, deve ver uma lista de artigos já publicados pela plataforma. Podendo baixar o artigo na íntegra.

Ator primário: Qualquer

Precondições: Nenhuma

Fluxo Principal:

1. O usuário acessa a página inicial da ferramenta.
2. Uma lista de artigos já publicados é exibida abaixo do formulário de login/cadastro.
3. Na lista, há informações sobre o artigo e um botão para download do PDF.
4. O usuário clica em download e o artigo é baixado.

Pós-Condições: Qualquer usuário pode ver e baixar os artigos já publicados pela plataforma.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

CASO DE USO 9: VER REVISÕES PUBLICADAS Descrição: Qualquer um que faça parte da comunidade deve ver uma lista de revisões marcadas como públicas.

Ator primário: Autor ou Revisor

Precondições: Nenhuma

Fluxo Principal:

1. O usuário acessa a página inicial da ferramenta.
2. Uma lista de revisões publicadas é exibida abaixo do formulário de login/cadastro.
3. Na lista, há informações sobre o artigo e um botão para download do PDF.
4. O usuário clica em download e o artigo é baixado.

Pós-Condições: Qualquer usuário pode ver e baixar os artigos já publicados pela plataforma.

Prioridade de Desenvolvimento: 2.

Frequência de uso: Eventual.

CASO DE USO 10: PUBLICAR REVISÃO Descrição: O Autor poderá publicar uma revisão já finalizada, de um artigo já publicado, para que a comunidade tenha acesso.

Ator primário: Autor

Precondições: Autor logado no sistema e pelo menos um artigo publicado.

Fluxo Principal:

1. O Autor entra no painel principal.
2. Clica em "My articles revisions".
3. Se algum artigo já estiver sido publicado com sucesso, um botão "publish"é liberado no cartão da revisão.
4. O Autor clica no botão e o sistema torna a revisão pública.

Pós-Condições: A revisão escolhida se torna pública para todos cadastrados.

Prioridade de Desenvolvimento: 2.

Frequência de uso: Eventual.

CASO DE USO 11: REENVIAR ARTIGO Descrição: O Autor reenvia um artigo científico que já foi revisado e tido como rejeitado, para que seja reavaliado pelos mesmo Revisores.

Ator primário: Autor

Precondições: O usuário deve estar logado ao sistema, ter um artigo rejeitado.

Fluxo Principal:

1. O Autor entra no painel principal.
2. O Autor clica em "My Articles".
3. No artigo com status rejeitado, o Autor clica no botão "Resend".
4. O Autor faz upload de um novo artigo e envia.

Pós-Condições: O Arquivo deve ser armazenado no *IPFS* e o *hash* do artigo deve ser substituido pelo novo e registrado em *blockchain*.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diário.

6.2.3 Subsistema de Revisão de Artigos Científicos

O ator Revisor nada mais é que um Autor que já obteve pelo menos um artigo publicado na plataforma, por tanto, ele herda todos os casos de uso anteriores e possui alguns extras que serão apresentados.

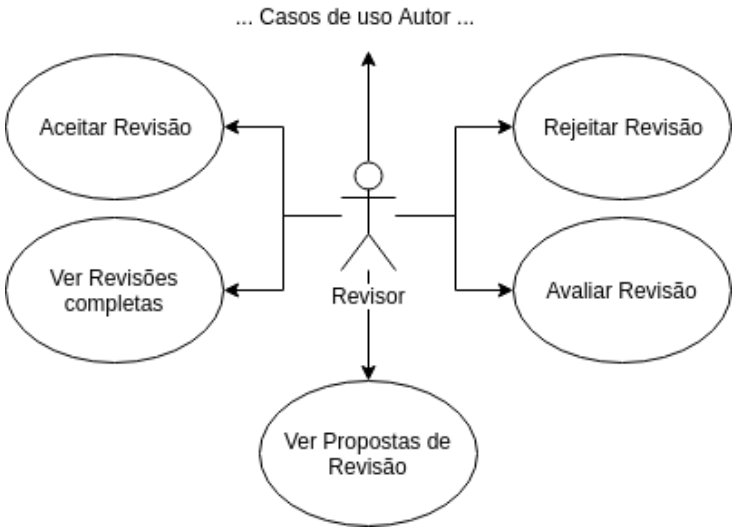

Figura 6.4: Diagrama de Casos de Uso do subsistema do Revisor

CASO DE USO 1: VER CONVITES PARA REVISÃO Descrição: O Revisor pode ver os convites para revisão recebidos.

Ator primário: Revisor

Precondições: Revisor estar logado no sistema.

Fluxo Principal:

1. O Revisor entra no painel principal.
2. Clica em "To Review".
3. Se o Revisor foi convidado para uma revisão, ele verá a lista de convites para revisão de artigos e poderá aceitar ou rejeitar trabalhar com elas.

Pós-Condições: Nenhuma

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

CASO DE USO 2: ACEITAR REVISÃO Descrição: Após analisar as informações iniciais do Artigo que ele foi convidado a revisar, o Revisor aceita revisar o Artigo proposto.

Ator primário: Revisor

Precondições: Revisor estar logado no sistema e ter sido escolhido, pelo sistema, para revisar um artigo.

Fluxo Principal:

1. O Revisor entra no painel principal.
2. Clica em "To Review".
3. Se o Revisor foi convidado para uma revisão, ele verá botões de ação no cartão da revisão.
4. O revisor clica em "Accept"

Pós-Condições: A revisão passa para o estado sob revisão com o parâmetro de aceita.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

CASO DE USO 3: REJEITAR REVISÃO Descrição: Após analisar as informações iniciais do artigo que ele foi convidado a revisar, o Revisor não aceita revisar, passando a responsabilidade a outro revisor.

Ator primário: Revisor

Precondições: Revisor estar logado no sistema e ter sido escolhido, pelo sistema, para revisar um artigo.

Fluxo Principal:

1. O Revisor entra no painel principal.
2. Clica em "To Review".
3. Se o Revisor foi convidado para uma revisão, ele verá botões de ação no cartão da revisão.
4. O revisor clica em "Reject"

Pós-Condições: O sistema escolhe outro revisor, diferente dos que já estão revisando o artigo.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

CASO DE USO 4: AVALIAR REVISÃO Descrição: Após aceitar uma revisão, baixar o PDF e analisar o artigo, o Revisor atribui seu conceito e seus comentários na revisão.

Ator primário: Revisor

Precondições: Revisor estar logado no sistema e ter aceito o convite para revisar um artigo.

Fluxo Principal:

1. O Revisor entra no painel principal.
2. Clica em "To Review".
3. Se o Revisor já aceitou trabalhar com uma revisão, aparecerá um botão "Rate", ao clicar, um formulário é exibido.
4. O revisor preenche o formulário dando seu conceito e observações e envia.

Pós-Condições: O conceito e anotações são salvas na revisão e somadas a pontuação do artigo que está sendo revisado.

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

CASO DE USO 5: VER REVISÕES COMPLETAS Descrição: Dentro do sistema, o revisor pode ver uma lista de revisões que ele já completou com sucesso.

Ator primário: Revisor

Precondições: Revisor estar logado no sistema.

Fluxo Principal:

1. O Autor entra no painel principal.
2. Clica em "Reviewed".
3. Uma lista de revisões feitas pelo Revisor é exibida.

Pós-Condições: Nenhuma

Prioridade de Desenvolvimento: 1.

Frequência de uso: Diária.

6.2.4 Requisitos não funcionais

- Segurança: O sistema deve prover facilidade para autenticação de todos os usuários, com a digitação de usuário e senha. As operações realizadas deverão ser registradas em *Blockchain*.
- Desempenho: Todas as operações realizadas deverão ser executadas em no máximo 1 segundo.
- Interface: Deve apresentar boa usabilidade, já que os usuários farão uso diariamente.
- Portabilidade: O sistema deve permitir ampla portabilidade, de modo a executar em ambientes operacionais diversos.

Com o Documento de Especificação de Requisitos pronto, foi possível ter clareza em que consistiria a ferramenta a ser implementada, com a execução das etapas posteriores do processo de desenvolvimento. Então, o próximo passo foi a produção do Documento de Especificação de Análise.

6.3 Documento de especificação e análise

O principal objetivo do Documento de Especificação de Análise é a identificação e a criação dos diagramas de classes da aplicação, com base nos casos de usos produzidos pelo Documento de Especificação de Requisitos. O diagrama de classe é o principal artefato em um documento de análise. Para complementar esse diagrama foram construídos outros artefatos: o diagrama de atividades, o diagrama de classes e o diagrama de sequência.

6.3.1 Diagrama de Atividades

O diagrama de atividades da ferramenta foi baseado nos requisitos apresentados e nas necessidades da comunidade científica a respeito da revisão de artigos científicos, isto é, contempla o processo de publicação de um artigo científico, desde o seu envio a plataforma passando pela escolha dos revisores e pelas revisões até obter a nota mínima onde finalmente, recebe o status de publicada e a comunidade científica pode acessar o artigo nesse processo, o sistema faz o uso de contratos inteligentes assinados por cada um dos envolvidos nas transações.

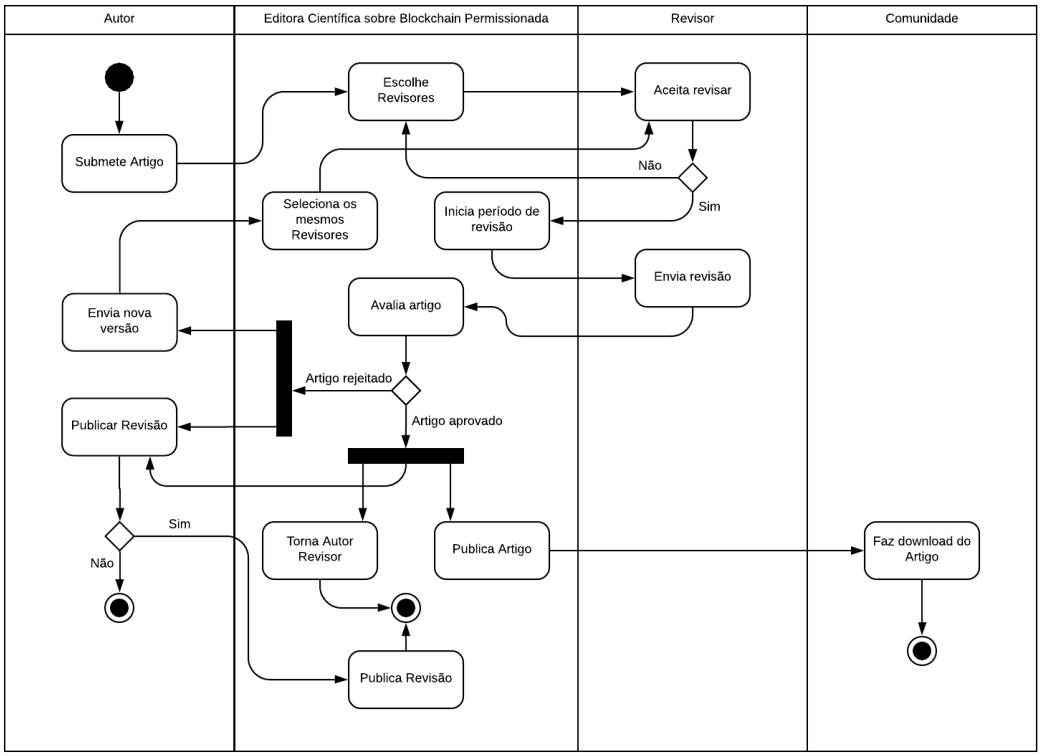

Figura 6.5: Diagrama de Atividades do sistema

O processo é iniciado quando o autor submete o artigo que é instanciado no IPFS, e seu *hash* é armazenado no *Hyperledger Fabric* junto com as informações das operações. Não existe uma instituição controlando a aplicação, sendo a gestão realizada pelos entes da comunidade.

A partir desse momento, através de contratos inteligentes, o sistema realiza a escolha dos 5 revisores que serão convidados a revisar o artigo. Com os revisores aceitando revisar, inicia-se o período de revisão, se não aceitarem revisar, o sistema convida outro Revisor, posteriormente, os revisores que aceitaram registram seus conceitos, que pode ser "Rejected", "Weak Rejected", "Border Line", "Weak Accepted"e "Accepted", além observações escritas, sobre o artigo. Assim que todos os revisores terminarem suas avaliações, um cálculo é feito para determinar se a média dos conceitos do artigo é o suficiente para publica-lo, aqui foi determinado que cada conceito possui os respectivos valores: 2, 4, 6, 8 e 10; dessa forma, a soma dos conceitos divida por 5, deve estar acima de 6 para que seja aprovado, caso não seja, o processo de revisão é reiniciado com os mesmos Revisores anteriores e o Autor necessita enviar uma nova versão

do Artigo.

6.3.2 Diagrama de Classes

Nesta seção será apresentado o principal diagrama de classe da ferramenta. As classes modelo abaixo, representam as entidades, ou seja, são todas as informações que a aplicação deve manter no banco de dados. Por meio do Documento de Especificação de Requisitos, foram identificadas as entidades: Autor, Detalhes, Revisão e o Artigo que possui relação com uma entidade do IPFS. Nessas classes temos todas as transações que são registradas em *blockchain* e contém regras de acesso específicas.

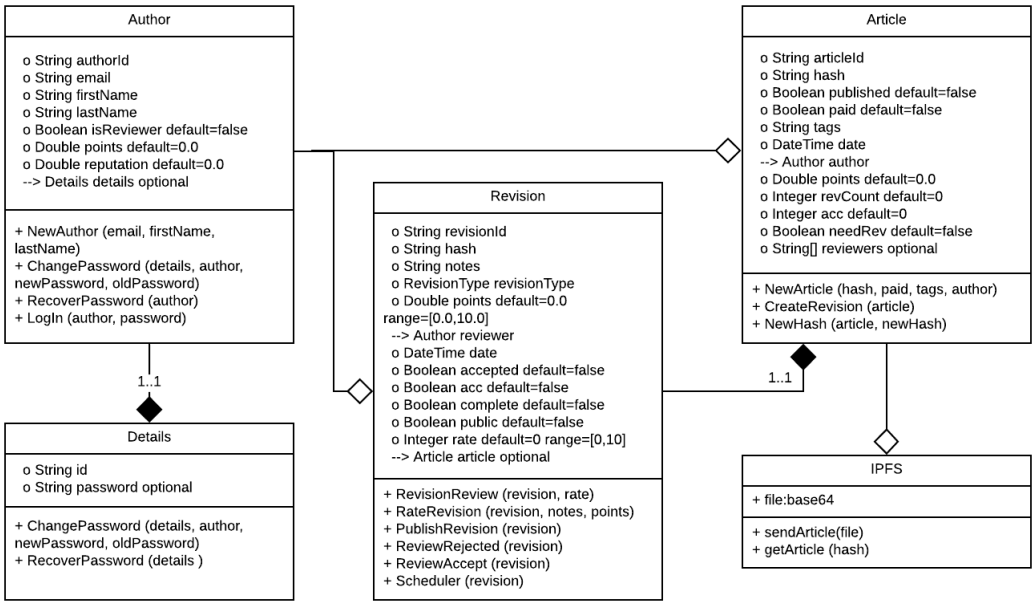

Figura 6.6: Diagrama de Classes do sistema

Para melhor entendimento por parte do desenvolvedor, foram criados diagramas de sequências. Estes diagramas - como explicado em capítulos anteriores, e afirmado por Sommerville (2011, p. 87): “tem por objetivo modelar as interações entre os atores e os objetos em um sistema e as interações entre os próprios objetos”.

6.3.3 Diagrama de Sequência

Abaixo encontra-se os principais diagramas de sequência que representam dois casos de usos da ferramenta, o de submissão de artigo (Caso de uso 04 do subsistema de Submissão) e o de revisão de artigo (Caso de uso 04 subsistema de Revisão).

O caso de uso 04 do subsistema de submissão de artigos científicos tem relação com o Autor, ou seja, este submete um arquivo, juntamente com as *Tags* (Marcações) e Título do Artigo. O diagrama deste caso de uso, detalha as interações que o Autor deve seguir para inserir

um artigo na plataforma, isso significa que além da hipermídia estar armazenada no IPFS, o *hash* deve estar registrado no *Fabric*.

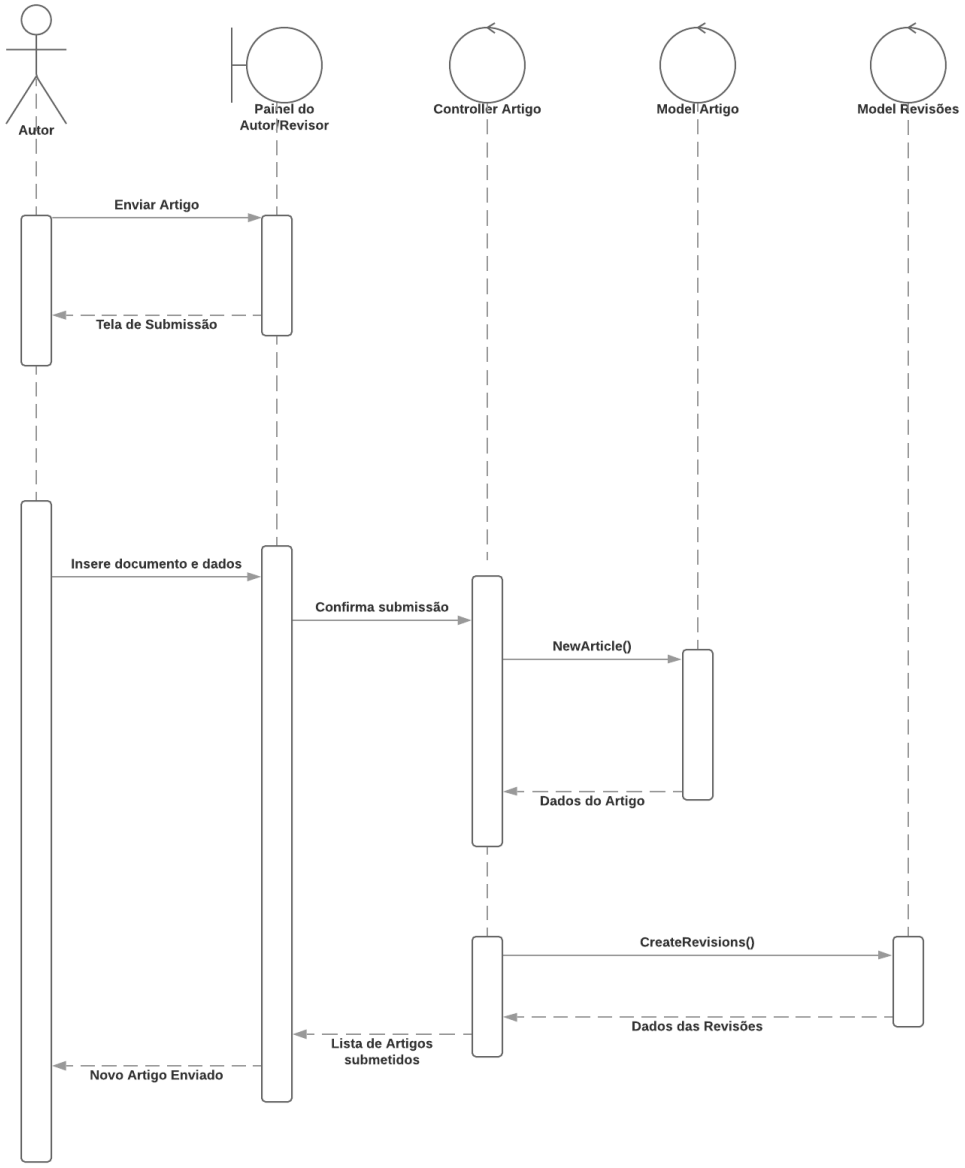

Figura 6.7: Diagrama de sequência do Caso de uso 4 do subsistema de Submissão

A sequência dos passos se inicia com o Autor clicando em enviar artigo e preenchendo o formulário de envio de Artigo no Painel principal, a submissão desses dados gera uma transação *NewArticle()* que possui um método na lógica da rede *Blockchain Fabric*; esse método cria um novo Artigo relacionado ao Autor e 5 revisões relacionadas ao Artigo criado, cada uma com um Revisor convidado. Ao fim da sequência, o *controller* deve retornar os dados novo Artigo enviado, com os detalhes e 5 novas revisões.

O caso de uso 04 do subsistema de revisão de artigos científicos tem relação com o Revisor, ou seja, este aceita participar de uma revisão, baixa o artigo, atribui suas anotações no campo Notes e seu conceito no *dropdown* Grade ao formulário e envia. O diagrama deste caso de uso, detalha as interações que o Revisor deve seguir para revisar um artigo na plataforma,

isso significa que a Revisão que ele aceitou trabalhar será preenchida e o artigo avaliado terá seus atributos atualizados.

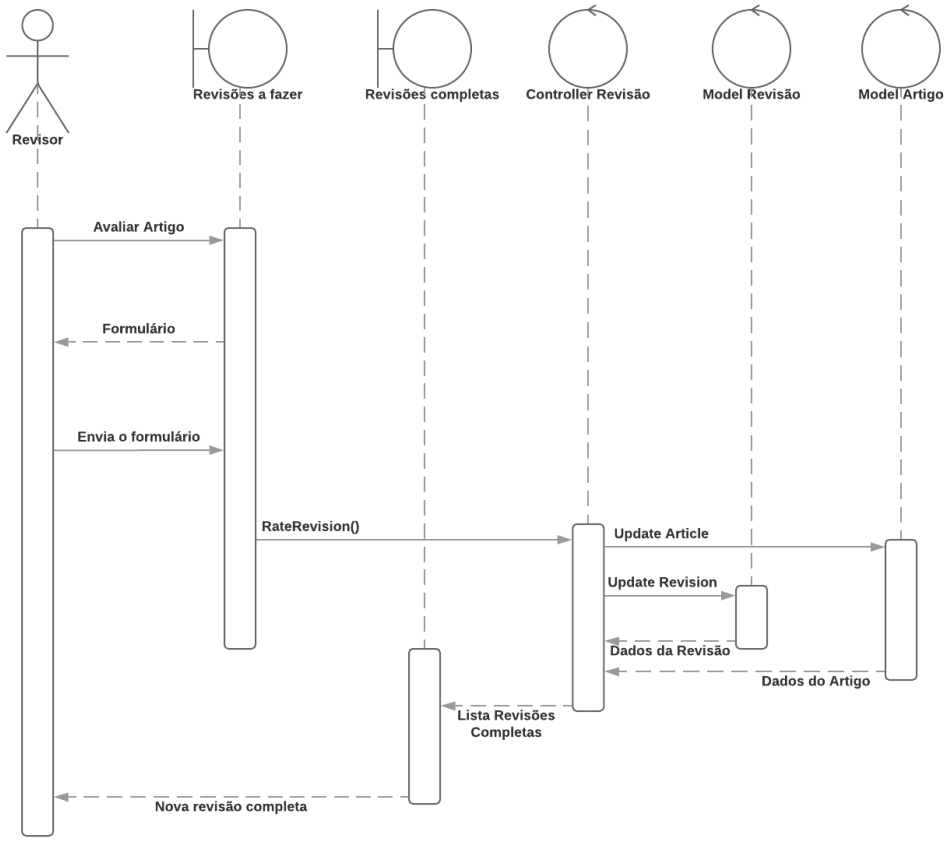

Figura 6.8: Diagrama de sequência do Caso de uso 4 do subsistema de Revisão

A sequência dos passos se inicia com o Revisor clicando em *"Rate"* e preenchendo o formulário de Revisão na área *"To Review"*, a submissão desses dados gera uma transação *RateRevision()* que possui um método na lógica da rede *Blockchain Fabric*; esse método atualiza os dados da Revisão na qual o Revisor está trabalhando e também atualiza os dados do Artigo relacionado a Revisão. Ao fim da sequência, o *controller* deve retornar os dados da nova Revisão completa e o Artigo avaliado deve ter seus dados atualizados.

6.4 Implementação do software editorial sobre *Blockchain*

Esta parte trata do desenvolvimento da aplicação, apresentando o passo a passo da construção do sistema após a coleta de requisitos: criação do banco de dados, da *API* e do *Client* e finalização. Nesta etapa da gerência do projeto de software, utilizou-se o Scrum como modelo de processo, entretanto, o sistema, que inclui 3 pacotes de software (*API, Client, Blockchain*), foi desenvolvido por uma pessoa (o autor), que representou todos os papéis na metodologia Scrum.

6.4.1 Banco de Dados

O processo de implementação da proposta do trabalho começa com a modelagem do "banco de dados", que neste caso, é uma Rede de Negócio implementada através do *Hyperledger Composer* que define o modelo de dados, lógica de transação e regras de controle de acesso da solução *blockchain*.

Para criar um *BND (Business Network Definition)*, precisamos criar uma estrutura de projeto com o gerador *Yeoman* com um parâmetro do *Hyperledger Composer* para criar um esqueleto de uma rede de negócios contendo todos os componentes de uma rede comercial.

Uma rede de negócios é composta de ativos, participantes, transações, regras de controle de acesso e, opcionalmente, eventos e consultas. Na esqueleto da rede comercial, há um arquivo modelo (*.cto*) que conterá as definições de classe para todos os ativos, participantes e transações na rede comercial. A estrutura da rede de negócios também contém um documento de controle de acesso (*permissions.acl*) com regras básicas de controle de acesso, um arquivo *script (logic.js)* contendo funções do processador de transações e um arquivo *package.json* contendo metadados da rede corporativa.

O principal documento a desenvolver é o arquivo modelo (*.cto*). Este arquivo deve ser escrito no modelo de linguagem do *Hyperledger Composer*. O arquivo de modelo contém as definições de cada classe de ativo, transação, participante e evento.

Definição do Participante:

```
namespace org.dasp.net

participant Author identified by email {
  o String authorId optional
  o String email
  o String firstName
  o String lastName
  o Boolean isReviewer default=false
  o Double points default=0.0
  o Double reputation default=0.0
  --> Details details optional
}
```

Figura 6.9: Definição do Participante Author no arquivo modelo (.cto)

Nas definições do Participante Autor indicamos que ele é identificado pelo e-mail e ele possui os atributos de perfil pessoal como primeiro nome, último nome e um identificador opcional todos do tipo *String*. Implementou-se também um campo Booleano para indicar se o

Autor é um Revisor ou não. O Autor possui uma pontuação e uma reputação, ambos do tipo *Double* para mensurar, respectivamente, a qualidade de seus artigos publicados e a qualidade desse usuário segundo a própria comunidade. Além disso, o Autor possui uma relação com um ativo do tipo Detalhes que possui informações complementares a esse Autor que vamos detalhar em seguida junto com outros ativos.

Definição dos Ativos:

```
asset Details identified by id {
  o String id
  o String password optional
}

asset Article identified by articleId {
  o String title
  o String tags
  o String articleId
  o String hash
  o Boolean published default=false
  o DateTime date
  --> Author author
  o Double points default=0.0
  o String concept default='none' optional
  o Integer revCount default=0
  o Boolean needRev default=false
  --> Revision[] revisions optional
}
}
asset Revision identified by revisionId {
  o String articleTitle
  o String articleTags
  o String revisionId
  o String concept default='none' optional
  o String hash
  o String notes
  o RevisionType revisionType
  o Double points default=0.0 range=[0.0,10.0]
  --> Author reviewer
  o DateTime date
  o Boolean acc default=false
  o Boolean complete default=false
  o Boolean public default=false
  o Integer rate default=0 range=[0,10]
  --> Article article optional
}
```

Figura 6.10: Definição dos Ativos (Artigo, Revisão e Detalhes no arquivo modelo (.cto)

Na definição dos Ativos, vemos 3 tipos de Ativos: Detalhes, Artigo e Revisão. Os Detalhes simplesmente guarda a senha do Autor/Revisor, assegurando que seja parte de um ativo no qual pertence a ele e só ele tem acesso. Aqui é importante lembrar que não há implementação de criptografia para o armazenamento da mesma, apesar de que todas as transações e informações são criptografadas em *Blockchain*, vejo como uma possível melhoria para por em produção já que melhora substancialmente a segurança da aplicação no seu nó.

O Ativo Artigo é identificado por um id, gerado automaticamente pela rede ao transacionar e possui campos como título, tags, hash do IPFS e conceito em formato *String*, na definição do próprio artigo armazenamos em formato Booleano, parâmetros que indicam se o Artigo precisa de revisão e se foi publicado e sua Data em formato *DateTime*. Por fim temos um atributo para fazer a contagem de revisões efetuadas, do tipo *Integer*, a pontuação do artigo do tipo *Double* e um array de relações com Ativos do tipo Revisão. Este último serve para armazenar junto com o Artigo, as Revisões que foram criadas a partir dele.

O Ativo Revisão também é identificado por um id, gerado automaticamente pela rede ao transacionar e possui campos como título do artigo, *tags* do artigo, *hash* artigo todos em formato *String*, na definição do próprio artigo armazenamos em formato Booleano, parâmetros que indicam se a Revisão, foi aceita pelo Revisor, foi completada pelo Revisor e se foi publicada e sua Data em formato *DateTime*. Por fim temos um atributo para fazer a contagem da avaliação

da Revisão pela comunidade do tipo *Integer*, os pontos do artigo revisado do tipo *Double* e duas relações com Ativos do tipo Artigo e Autor. Estes últimos servem para armazenar junto com a Revisão, o Artigo que está sendo revisado e o Revisor que é dono da Revisão.

Cada Participante, opera os Ativos na rede *Blockchain* através de transações, essas transações também são modeladas neste arquivo modelo (*.cto*).

```
enum RevisionType {
  o HALFBLIND
  o FULLBLIND
}
abstract transaction PublisherTransactions {
}
transaction CreateRevision extends PublisherTransactions {
  --> Article article
}
transaction NewHash extends PublisherTransactions {
  --> Article article
  o String newHash
}
event NewHashEvent {
  --> Article article
  o String oldHash
  o String newHash
}
transaction ReviewAccept extends PublisherTransactions {
  --> Revision revision
}
transaction ReviewRejected extends PublisherTransactions {
  --> Revision revision
}
transaction PublishRevision extends PublisherTransactions {
  --> Revision revision
}
transaction RateRevision extends PublisherTransactions {
  --> Revision revision
  o String notes
  o Double points
}
transaction RevisionReview extends PublisherTransactions {
  --> Revision revision
  o Double rate
}
transaction NewArticle extends PublisherTransactions {
  o String hash
  o String title
  o String tags
}
transaction NewAuthor extends PublisherTransactions {
  o String email
  o String firstName
  o String lastName
  o String password
}
transaction Scheduler extends PublisherTransactions {
  --> Revision revision
}
transaction ChangePassword extends PublisherTransactions {
  --> Details author
  --> Author user
  o String newPassword
  o String oldPassword
}
transaction RecoverPassword extends PublisherTransactions {
  --> Details author
}
transaction LogIn extends PublisherTransactions {
  --> Author author
  o String password
}
```

Figura 6.11: Definição das Transações no arquivo modelo (.cto)

Como podemos ver, há diversas transações para vários fins, porém, todas funcionam basicamente da mesma forma. Possuem atributos que funcionam como parâmetros que são utilizados posteriormente por essa transação, em sua sequência lógica. Algumas transações também podem conter um campo para relações de Ativos ou Participantes.

```
/**
 * Transaction for new Author, maybe not needed
 * @param {org.dasp.net.ChangePassword} changePassword
 * @transaction
 */

async function ChangePassword(changePassword) {
  changePassword.author.password = changePassword.newPassword;
  // Get the asset registry for the asset.
  let participantRegistry = await getAssetRegistry(DETAILS);
  // Update the asset in the asset registry.
  await participantRegistry.update(changePassword.author).then(
    await request.post({
      uri: "http://172.17.0.1:1880/hello",
      json: {
        to: changePassword.user.email,
        topic: "DASP - Password Changed",
        body: "You Password has been changed!"
      }
    })
  );
}
```

Figura 6.12: Exemplo de sequência lógica (logic.js)

A Figura 6.12 mostra a lógica da transação "*ChangePassord*", responsável por efetuar a alteração de senha do usuário, ao enviar sua nova senha, ele recebe um email de confirmação via *Node-RED*.

Cada transação possui uma lógica para executar e necessita ser definida no arquivo *script (logic.js)* além de também depender das permissões de acesso que são definidas no documento de controle de acesso (*permissions.acl*) com regras bem definidas de controle de acesso. Ambos são necessários para gerar o Arquivo da Rede de Negócios (*.bna*).

```
rule AuthorCanCreateRateRevision {
    description: "Allow all author access to resources from
    RateRevision"
    participant(a): "org.dasp.net.Author"
    operation: CREATE
    resource(r): "org.dasp.net.RateRevision"
    condition: (r.revision.reviewer.getIdentifier() ==
    a.getIdentifier() && !r.revision.complete)
    action: ALLOW
}
rule AuthorCanReadArticleFromRateRevision {
    description: "Allow all author access to resources from
    RateRevision"
    participant: "org.dasp.net.Author"
    operation: UPDATE, READ
    resource: "org.dasp.net.Article"
    transaction: "org.dasp.net.RateRevision"
    action: ALLOW
}
```

Figura 6.13: Exemplos de regra de acesso (permissions.acl)

A Figura 6.13 mostra duas regras de acesso. A primeira permite que o Revisor execute a transação "*RateRevision*", útil para avaliar uma revisão de um artigo. A regra diz que a

transação só pode ser executada pelo dono da revisão escolhida e se esta revisão ainda não está completa. A segunda regra refere-se permite que o Revisor leia e modifique artigos através da transação "*RateRevision*".

Agora que a rede de negócios foi definida através do arquivo modelo, da lógica e das permissões de acesso, ela deve ser empacotada em um Arquivo da Rede de Negócios (*.bna*) através de um comando do *Hyperledger Composer*.

Depois de criar o arquivo .bna, a rede comercial pode ser implantada na instância do *Hyperledger Fabric*. Depois que a rede comercial é instalada também através de comandos do *Hyperledger Composer*, a rede poderá ser iniciada.

6.4.2 API

O *Hyperledger Composer* possui um comando para gerar uma API REST sob medida baseada na rede comercial implementada. Por padrão, o servidor *REST* do *Hyperledger Composer* inclui uma funcionalidade que gera um conjunto de *APIs RESTful* para todos os ativos, participantes e transações em uma rede *Blockchain* implementada. Esse servidor *REST* do *Hyperledger Composer* também inclui as seguintes funcionalidades:

- Eventos usando *WebSockets*.
- A autenticação usando o middleware de autenticação do *Passport*.
- Modo de vários usuários, para que os usuários autenticados possam fornecer suas próprias credenciais na *Blockchain*.
- HTTPS e TLS para comunicações seguras entre cliente e servidor.

Esses recursos são todos projetados para serem de uso geral e fáceis de usar prontos para uso. O servidor *REST* do *Hyperledger Composer* é distribuído como um aplicativo chamado composer-rest-server, que pode ser instalado usando o *npm* ou o *Docker* e inclui todos esses recursos. Para desenvolver o sistema proposto neste trabalho, a *API REST* forneceu uma camada útil de abstração neutra de linguagem.

6.4.3 Client

O *Hyperledger Composer* também possui um módulo *Yeoman*, pacote do *NodeJS* que facilita a construção de projetos, usado para criar projetos para utilziar com o *Hyperledger Composer*. Esse gerador também permite criar aplicações *Angular*, porém suporta apenas definições simples e básicas de modelos de rede de negócios. O aplicativo gerado (incluindo os formulários da web que ele produz) não suportará tipos mais complexos de redes, por isso, o sistema proposto neste trabalho passou por severas modificações para se adequar a o que se espera da aplicação e seus casos de uso. Por ser uma aplicação *NodeJS*, alguns pacotes foram

adicionados para suportar as funcionalidades, o pacote *IPFS-http-client* por exemplo serve para se comunicar com a rede *IPFS* implementada.

6.4.4 Finalização do desenvolvimento

A etapa de finalização consistiu na realização de testes de validação e publicação da aplicação em ambiente de produção controlado, ou seja, a disponibilização da aplicação para uso em uma intranet. O fato do projeto ser parte de um estudo sobre *DAO* do laboratório GERCOM da UPFA, o próprio laboratório foi o ambiente de experimentação.

A realização do teste de validação tem como objetivo a validação dos requisitos, ou seja, a aplicação deve cumprir o que prometeu. Após a realização do teste e correções de falhas apresentadas, o sistema foi publicado na Intranet da rede virtual do laboratório GERCOM.

6.5 Interação do usuário

Esta seção apresenta a interface *web* do sistema, pela qual um usuário interage com a ferramenta para enviar, revisar e publicar um artigo científico.

6.5.1 Interface Web Gráfica da Ferramenta

Esta seção irá apresenta as principais telas da aplicação. As telas foram desenvolvidas considerando a usabilidade, experiência do usuário, acessibilidade e a comunicabilidade como características de interface e interação.

Na Figura 6.14, é apresentado o protótipo de tela após o usuário efetuar o login, o painel principal da aplicação. Nesta página o Autor pode iniciar o processo de submissão de um artigo ao fazer o *upload* de um documento em PDF e preencher os dados requeridos no formulário seguinte. O uso de um *card* central logo na entrada da plataforma, dá foco ao processo de submissão do artigo de forma simples e direta.

Na Figura 6.15, é apresentado o protótipo de tela após o Revisor efetuar o login e clicar em "*To Review*"no menu ao lado, o painel de Revisões pendentes. Nesta página o Revisor pode iniciar o processo de avaliação de um artigo ao visualizar o *card* com as informações de um Artigo, aceitar revisa-lo, baixar o arquivo e enviar o formulário de revisão com a nota e detalhes de sua análise. O fluxo do processo dentro do próprio *card* facilita a atenção e o processo.

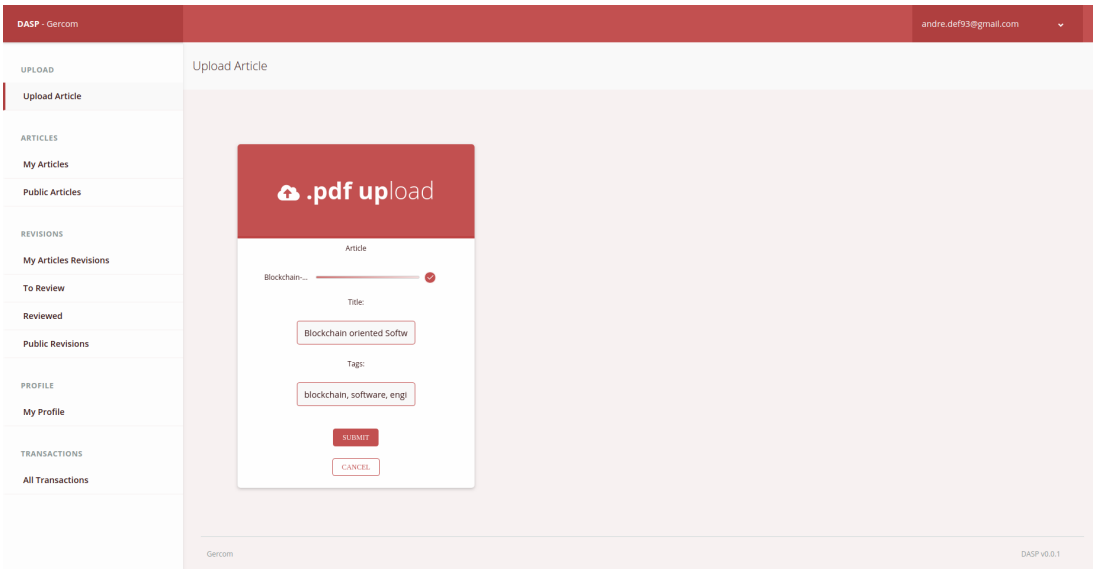

Figura 6.14: Painel principal da aplicação web

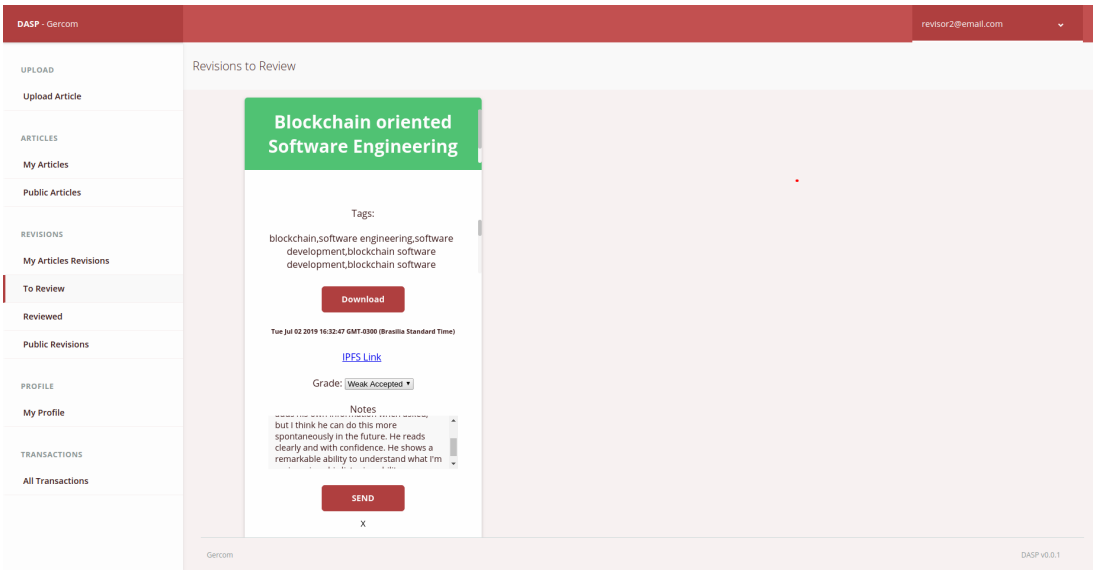

Figura 6.15: Menu "*To Review*"da aplicação web

CAPÍTULO 7

AVALIAÇÃO DA PROPOSTA

Este capítulo tem o objetivo de apresentar a avaliação do sistema web por usuários reais. Para que o sistema seja avaliado corretamente, um mínimo de seis participantes devem participar do cenário de submissão pois o sistema seleciona cinco revisores para iniciar o processo de revisão do artigo enviado pelo autor. Para estes foi enviado um formulário com um total de oito perguntas objetivas e uma discursiva.

Os seis usuários selecionados para fazer a avaliação, foram distribuídos em papeis dentro da ferramenta. Propositalmente, a ferramenta considera as seis primeiras pessoas cadastradas como revisores, como já explicado, isso permite com que todas os métodos dentro da ferramenta funcionem corretamente. Por convenção, os próximos cadastrados sempre serão Autores, a não ser que publiquem efetivamente um Artigo na plataforma. Já para a avaliação, necessitamos apenas de um Autor e cinco Revisores. Portanto, foi definido um cenário que contempla as principais funcionalidades, da submissão à eventual publicação de um artigo e um questionário com perguntas de usabilidade, funcionalidade, confiabilidade e eficiência da ferramenta.

7.1 Usabilidade

Define-se usabilidade como: “um atributo de qualidade relacionado à facilidade do uso de algo.” [Nielsen 1993] Ou seja, a interface deve ser autoexplicativa. O desenvolvimento da interface do sistema deve transmitir ao usuário a maneira correta de como a ferramenta deve ser utilizada. Se a ferramenta seguir bem esse critério, a próxima interação do usuário já será mais fluida.

Foram elaboradas duas perguntas relacionadas à usabilidade da aplicação web:

1. Ao acessar o *DASP*, quão claro ficou a proposta da ferramenta?
    - Completamente.
    - Algumas coisas não estão claras.
    - Nem um pouco.
2. A interface do *DASP* é atraente e intuitiva?

- Sim, o design da ferramenta facilita o entendimento das funções além de ser atraente.
- É esteticamente atraente porém nada intuitivo.
- É intuitiva, porém nada atraente.
- Não é atraente nem intuitiva.

A primeira pergunta tem como objetivo esclarecer se a aplicação é autoexplicativo, isto é, se o usuário entende a ferramenta através da interface, sem a necessidade de um manual ou tutorial. A Figura 7.1 mostra que a maior parte (83.3%) dos usuários avaliados envolvidos no processo de publicação, relataram que a ferramenta esclarece completamente a sua proposta através da interface. Nenhum usuário relatou o pior cenário desta pergunta ("nem um pouco"), mesmo assim, um usuário avaliado relatou que algumas coisas não estão claras, isso pode ter relação com a falta de elementos visuais na apresentação da ferramenta, para esclarecer sua proposta, antes do usuário se cadastrar.

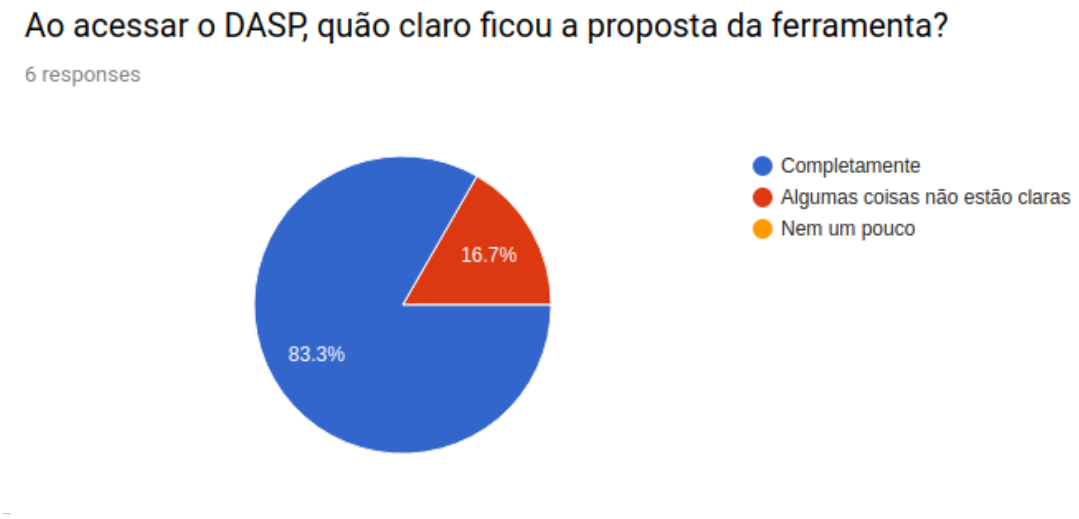

Figura 7.1: Respostas da Primeira pergunta objetiva da Avaliação da aplicação

A figura 7.2 mostra o resultado da segunda pergunta sobre o item de usabilidade. Esta pergunta tinha como objetivo saber se a ferramenta é agradável visualmente. Além da funcionalidade, um sistema deve possuir uma interface agradável esteticamente para o usuário. Para a maioria dos usuários, o design da ferramenta facilitou o entendimento das funções por meio da sua interface, além de ser atraente. Um usuário avaliado acredita que a proposta desenvolvida não é atraente, mas é intuitivo, isto é, como todo *MVP*, necessita de melhorias na interface e experiência do usuário, para conquistar o padrão estético.

7.2 Funcionalidade

Pressman define funcionalidade como: “O grau com que o software satisfaz às necessidades declaradas conforme indicado pelos seguintes subatributos: adequabilidade, exatidão,

Figura 7.2: Respostas da Segunda pergunta objetiva da Avaliação da aplicação

interoperabilidade, conformidade e segurança" [Pressman and Maxim 2016]. Em resumo, a aplicação deve entregar para o usuário o que foi prometido, e realizar isto de maneira adequada. Uma das características esperada pela aplicação web é a capacidade de realizar a submissão de um artigo científico e o mesmo passar pelo processo de submissão de maneira correta.

Foram elaboradas duas perguntas elaboradas para o item de funcionalidade:

1. O *DASP* cumpre sua função, isto é, fornece um meio para Revisão e eventual Publicação de Artigos Científicos?
   - Sim.
   - Sim, em partes.
   - Não.
   - Outro:
2. Em algum momento durante o uso, a aplicação exibiu alguma falha inesperada?
   - Não.
   - Uma vez.
   - Duas vezes.
   - Entre 3 e 5 vezes.
   - Mais que 5 vezes.

O intuito dessas perguntas é saber dos usuários, se a ferramenta funciona como prometido e se possui uma utilidade real. A aplicação deve cumprir seu objetivo. A Figura 7.3 mostra que para todos os usuários avaliados neste cenário, a ferramenta cumpre com sua função, fornecendo um meio para Revisão e eventual Publicação de Artigos Científicos.

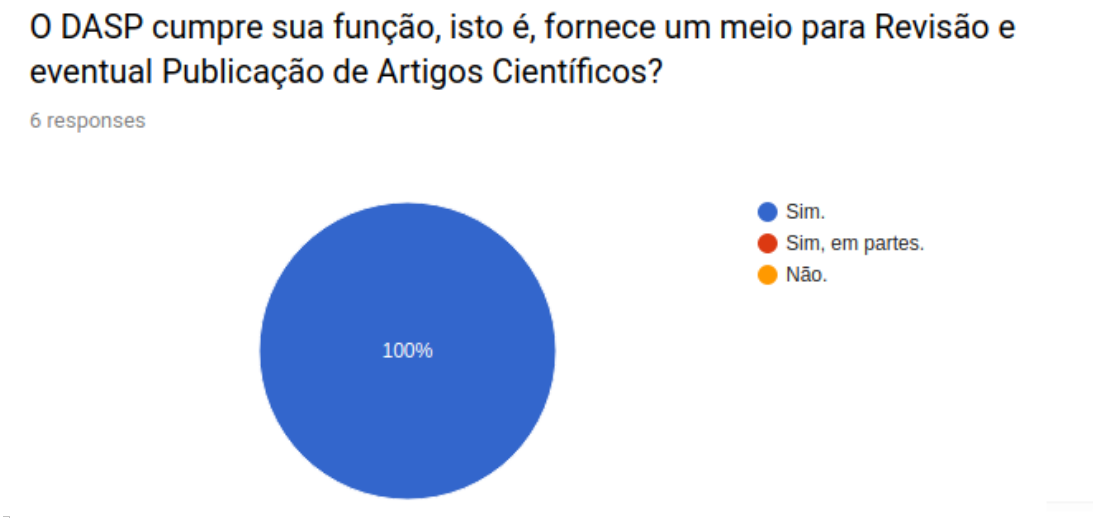

Figura 7.3: Respostas da Terceira pergunta objetiva da Avaliação da aplicação web

A Figura 7.4 demonstra experiência dos usuários em relação aos possíveis erros apresentados durante o uso da ferramenta, tenta extrair dos usuários quantas falhas ocorreram durante o processo. Para todos os usuários, a ferramenta não apresentou nenhuma falha inesperada. Falhas são comuns, é de se esperar que há melhorias a serem feitas em casos de uso específicos, mas até então, a ferramenta cumpre seu fluxo principal sem problemas.

Figura 7.4: Respostas da Quarta pergunta objetiva da Avaliação da aplicação web

7.3 Confiabilidade

Um *software* é confiável quando não apresenta falhas durante o seu uso, ou seja, funciona corretamente [Sommerville 2011]. A confiabilidade do software é a probabilidade de o sistema de software funcionar corretamente sem falhas durante um determinado período de tempo. É um dos atributos mais importantes na qualidade de software e através dela que con-

trolamos a frequência em que ocorrem as falhas, a severidade destas falhas e como o sistema se recupera dessas falhas. É difícil alcançar certo nível de confiabilidade em qualquer sistema com alto grau de complexidade [Pressman and Maxim 2016].

1. Os links da aplicação direcionam para as páginas corretas?
    - Sim.
    - Alguns.
    - Não.
2. O *DASP* possui *feedbacks* que orientam o usuário corretamente?
    - Sim.
    - Às vezes.
    - Não.

A confiança do usuário durante o uso da ferramenta, está intimamente relacionada com a prevenção de links quebrados ou páginas inexistentes, que podem levar até ao abandono da ferramenta. *Feedbacks* constantes são essenciais para informar ao usuário o estado do sistema além de manter o fluxo de transação que o usuário estava, recuperando do erro. A Figura 7.5 mostra que todos os usuários avaliados relataram que os links da aplicação direcionam para as páginas corretas. Devido a baixa complexidade de paginação do sistema, há poucas chances de falhas relacionadas ao redirecionamento de links.

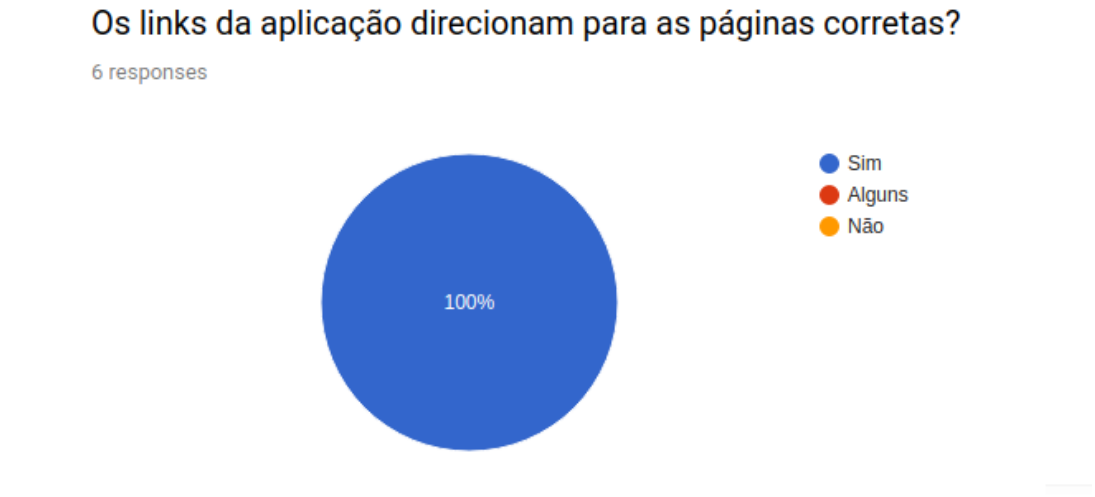

Figura 7.5: Respostas da Quinta pergunta objetiva da Avaliação da aplicação web

Sobre as respostas visuais da aplicação web, 80% dos usuários relataram positivamente para o fato de que há *feedbacks* que orientam o usuário durante o fluxo da aplicação. Ainda assim, uma pequena parcela indicou que somente algumas vezes a aplicação provém alguma resposta visual para o usuário. Realmente, apesar da aplicação web possuir muitos *feedbacks*, aplicações ideais em produção, possuem elementos visuais para cada ação dentro da plataforma, o que inclui um custo maior ao tempo de desenvolvimento e planejamento específico.

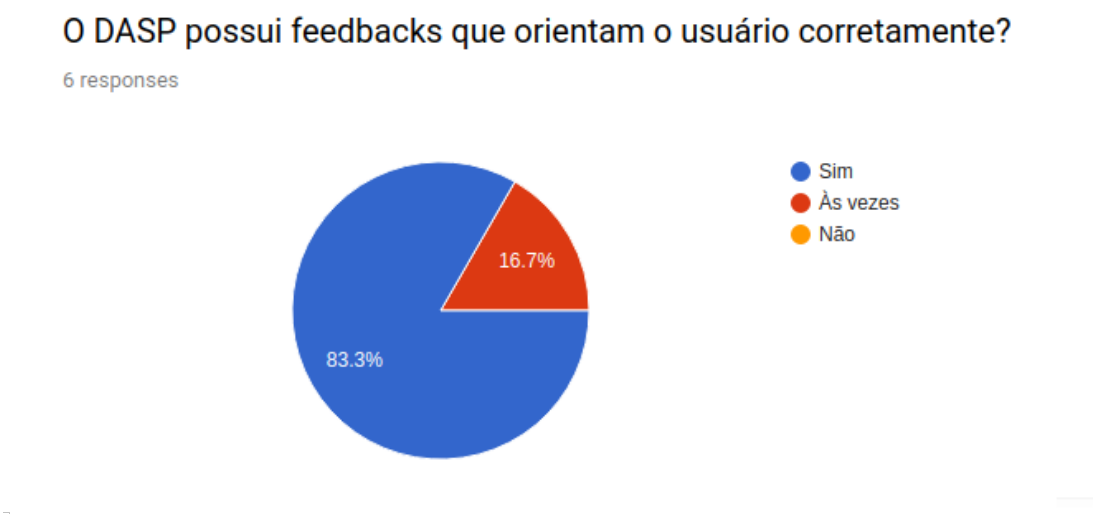

Figura 7.6: Respostas da Sexta pergunta objetiva da Avaliação da aplicação web

7.4 Eficiência

O ritmo acelerado do desenvolvimento de hardware fornece recursos de hardware aprimorados que aumentam a eficiência. Entre todos os tipos de softwares populares em computadores, existem apenas alguns, como os jogos, que podem fazer uso total dos recursos de hardware que são sensíveis à eficiência. Desenvolvedores de softwares menos complexos reduzem sua necessidade em utilizar recursos computacionais e isso pode aumentar sua competitividade no mercado. Desenvolver softwares mais eficientes requer melhor design e mais otimização, portanto, custos mais elevados.

1. O *DASP* processa em poucos segundos as requisições feitas?
    - Sim, o tempo de resposta é satisfatório.
    - Não, demora um pouco.
    - Não, demora muito.
2. O tempo de carregamento das páginas é satisfatório?
    - Sim.
    - Às vezes.
    - Não.

A figura 7.7 mostra o resultado da primeira pergunta sobre a eficiência do *software*, que questiona sobre o tempo de requisições feitas dentro do sistema. Para a maioria (66%), a aplicação web teve um tempo de resposta satisfatório para as requisições efetuadas. Entretanto, o restante dos usuários avaliados relataram que demora um pouco. Sabendo também que não houve usuários que relataram grande latência nas requisições, acredita-se que há algum processo um pouco demorado, talvez no cadastro, que deva ser melhorado. O cadastro de usuário nessa

versão da aplicação, efetua diversos processos por trás antes de efetivamente registrar e fornecer o *login* ao usuário: Cria participante na rede *Blockchain*, Cria uma identidade dentro da rede e relaciona ao participante anterior, além ativar a identidade deste que acabou de ser criado.

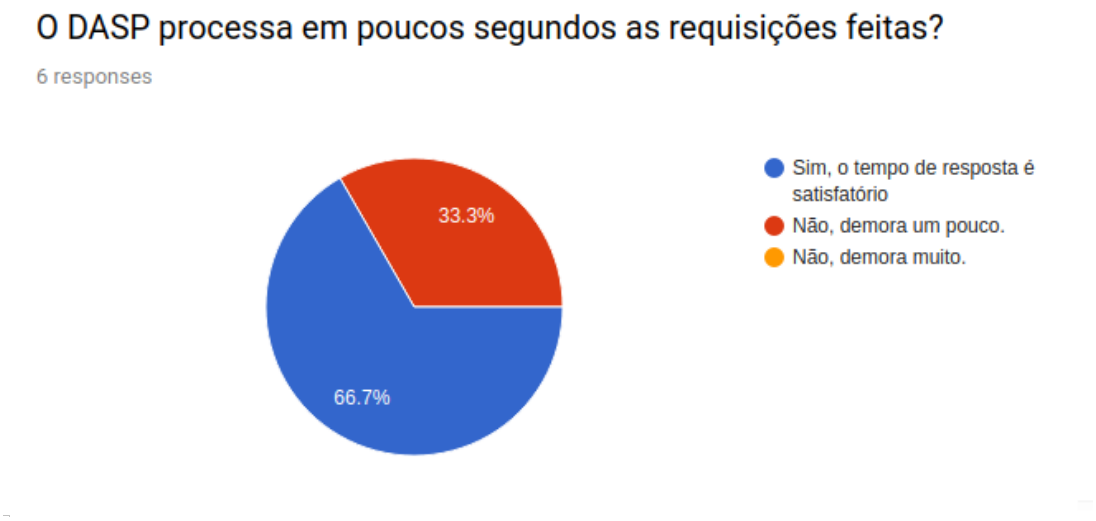

Figura 7.7: Respostas da Sétima pergunta objetiva da Avaliação da aplicação web

Sobre o tempo de carregamento das páginas, a figura 7.8 mostra que 80% dos usuários relataram que o tempo de carregamento das páginas é satisfatório. O restante pontuou que somente alguns momentos os carregamentos de páginas levaram um tempo satisfatório. Novamente, nenhum usuário relatou o pior caso, indicando que talvez haja casos de uso específicos que possuem uma certa lentidão devido ao número de transações em *Blockchain* efetuadas ao mesmo tempo.

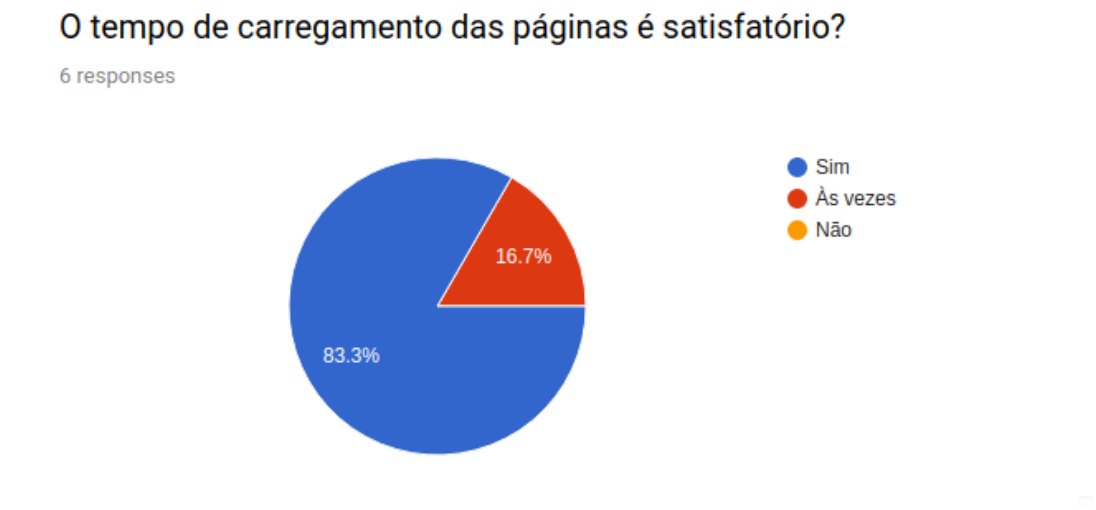

Figura 7.8: Respostas da Oitava pergunta objetiva da Avaliação da aplicação web

7.5 Críticas e Sugestões

A fim de obter informações adicionais dos usuários avaliados, foi adicionado ao questionário uma pergunta discursiva para o complemento de opiniões sobre a ferramenta:

1. Críticas e Sugestões. Se quiser, fale mais sobre a sua experiência com o *MVP* do *DASP*.

Os resultados obtidos a partir desta pergunta, podem ser vistos da figura 7.9 onde temos algumas sugestões, críticas e elogios. Pelo menos quatro das repostas fornecidas falam sobre experiencia do usuário e interface gráfica. Isto é, cobram melhorias visuais na plataforma que sinalizam eventos e atributos dentro da mesma, além de promover o esclarecimento sobre os objetivos de determinados menus. Dois usuários relataram lentidão no processo de *login* do sistema, concordando com o que já foi discutido sobre o número de transações no processo de criação de um usuário na rede *Blockchain* que deve ser melhorado.

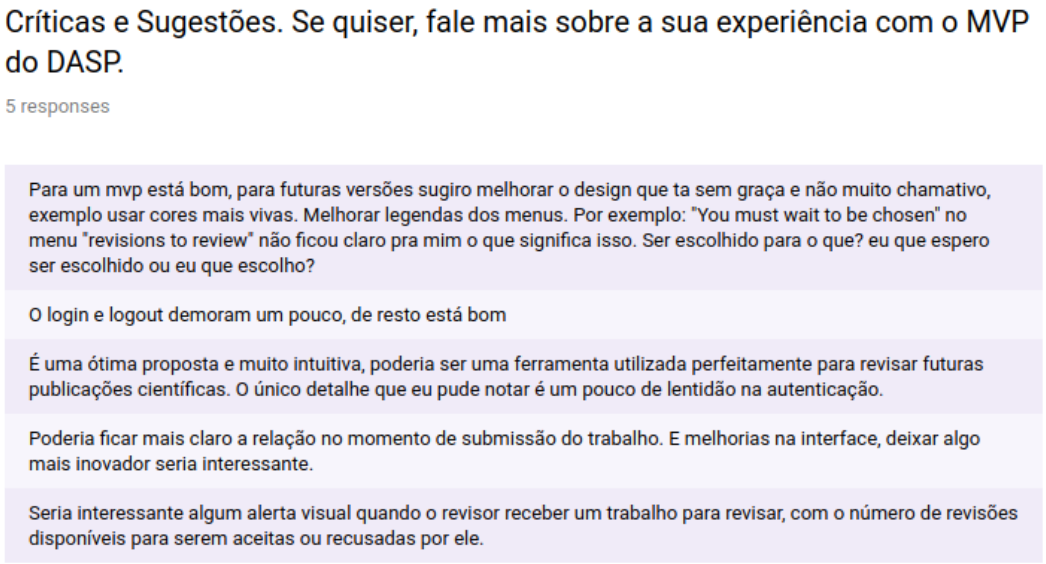

Figura 7.9: Respostas da Pergunta subjetiva da Avaliação da aplicação web

CAPÍTULO 8

CONCLUSÕES

8.1 Conclusões Gerais

Este trabalho é uma versão inicial de um objetivo que está sendo estudando no grupo de estudos GERCOM, relacionado a proposição futura de uma metodologia para disseminação e gerenciamento de trabalhos científicos de forma distribuída e sem intermediários (editoras).

Foram objetos de estudo deste trabalho: a especificação, a implementação e a avaliação do sistema proposto neste trabalho que tem como finalidade melhorar o processo de submissão, revisão e eventual publicação de artigos científicos através de tecnologias que descentralizam dados e processos. Foi realizado uma avaliação do cenário principal da ferramenta, para confirmar se o objetivo geral foi atingido. O resultado da avaliação validou o trabalho desenvolvido já que em maioria, tivemos respostas positivas.

Realizar esse trabalho proporcionou um aprendizado importante sobre como planejar e desenvolver aplicações sobre uma *Blockchain* permissionada para usuários finais. Mostrou que há ferramentas e tecnologias disponíveis que facilitam essa integração, otimizando o processo de construção e manutenção do banco de dados, por exemplo. Pode-se dizer que definir uma rede de negócios em uma rede *Blockchain* do *Hyperledger Fabric* com a ajuda do *Hyperledger Composer* é muito mais prático e intuitivo quando em comparação com a formatação de um banco de dados tradicional devido a abstração de modelos, regras de acesso, consultas e lógica; em simples códigos de programação em linguagens específicas.

O desenvolvimento do trabalho também permitiu entender e propor soluções para os problemas relacionados ao processo de publicação de artigos científicos e da construção de aplicações descentralizadas. O fato da ferramenta ter sido desenvolvida por apenas uma pessoa e em pouco tempo, algumas melhorias serão descritas afim de contemplar trabalhos futuros.

8.2 Trabalhos Futuros

O cadastro do sistema obedece um roteiro pré-definido para iniciar a rede com o mínimo de tipos de participantes registrados, neste roteiro, os seis primeiros cadastros do sistema são obrigatoriamente participantes do tipo revisor. Esta convenção deve ser melhor elaborada para alcançar o equilíbrio entre a escolha autônoma de revisores do sistema e manter a qualidade dos escolhidos. Além disso, a quantidade escolhida de revisores que participam do processo de

revisão, não parte de nenhuma metodologia ou regra pré-definida, foi arbitrariamente adotada. Recomenda-se a realização de um estudo para avaliar a quantidade ideal de revisores participantes do processo de revisão. A convenção de que o autor se torna revisor imediatamente ao publicar um artigo, também deve ser repensada afim de que haja qualidade entre os novos revisores, isto é, talvez se apenas autores com um número mínimo de artigos publicados se tornem revisores, a qualidade desses novos revisores seja maior.

Quando um artigo é publicado, ele é listado na página inicial da aplicação, possibilitando o download e visualização, para usuários ainda não registrados ou deslogados. Porém, uma funcionalidade essencial que não pôde ser encaixada no cronograma do desenvolvimento, é possibilitar a busca de artigos por termos digitados. No *Hyperledger Composer* há a possibilidade de implementação de um componente opcional chamado *Queries* que funciona através da própria *API* gerada e retorna dados específicos pré determinados no arquivo *queries.qry*.

A rede de negócios da aplicação já permite que o autor determine se o Artigo que ele quer publicar na aplicação possui seu acesso pago, porém, esta funcionalidade não foi devidamente implementada por depender de *APIs* externas para pagamento, o qual não era foco do trabalho. Portanto, sugere-se a implementação de um método que contemple pagamentos online. Essa funcionalidade garante ao Autor maior controle sobre seus ganhos já que ele pode decidir o valor pelo acesso de sua publicação.

Outro ponto importante a ser comentado é o fato de que, ao submeter um Artigo em *PDF*, o sistema não instrui o usuário sobre a formatação do mesmo, possibilitando a identificação do usuário pelo documento, por exemplo. Como a proposta do sistema também é evitar a identificação dos envolvidos, essa funcionalidade deve ser implementada para que, o usuário seja instruído a formatar corretamente o seu Artigo antes de envia-lo a plataforma, além de possuir alguma punição para os que não o fizerem (como diminuir sua pontuação ou instruir os Revisores a avaliar negativamente Autores que desobedecerem as instruções).

A aplicação permite que o nó (aquele que irá instalar e instanciar o *WebApp*) escolha o tipo de revisão que deseja trabalhar, no momento, apenas duas foram implementadas: *HALFBLIND* e *FULLBLIND*. Para que a descentralização e distribuição seja alcançada de fato, o sistema necessita ser instanciado em outras entidades ou servidores. Uma implementação recomendada é que cada instância possa determinar seu tipo de revisão, mesmo trabalhando com o mesmo banco de dados (*Blockchain*).

Os dados privados do usuário são armazenados explicitamente em um ativo dentro da *Blockchain*, onde somente o dono daquele ativo pode acessar, porém, é recomendado que os dados mais sensíveis sejam criptografados antes de seu armazenamento.

A tecnologia *Blockchain* não permite que dados sejam apagados de sua cadeia, isto é, removidos sem deixar rastros. Porém, pode-se fazer alterações que são registrados em cadeia. Por tanto, para que o usuário remova seu perfil da plataforma, deve ser implementado o que chamamos de *revoke ID*, onde o usuário poderá revogar sua identidade, tornando-o inativo para o sistema.

Por fim, o desenvolvimento do sistema descrito neste trabalho, mostrou-se muito pro-

veitoso em relação a quantidade de conhecimento adquirido, não só em desenvolvimento de sistemas e aplicações descentralizadas, mas também para melhor entendimento dos processos de submissão, revisão e publicação de artigos científicos e seus problemas. Há interesse na continuação desse projeto, para que, futuramente, ele seja disponibilizado em ambiente de produção na internet, contribuindo assim, para um cenário de produção científica, em geral, mais justo.

Referências Bibliográficas

[Androulaki et al. 2018] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM.

[Baran 1964] Baran, P. (1964). On distributed communications networks. *IEEE transactions on Communications Systems*, 12(1):1–9.

[Booch et al. 2006] Booch, G., Rumbaugh, J., and Jacobson, I. (2006). *UML: guia do usuário*. Elsevier Brasil.

[Cohn 2000] Cohn, M. (2000). *Desenvolvimento de software com Scrum: aplicando métodos ágeis com sucesso*. Bookman.

[Crosby et al. 2016] Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V., et al. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, 2(6-10):71.

[Davis 1995] Davis, A. M. (1995). *201 principles of software development*. McGraw-Hill, Inc.

[Dhillon et al. 2017] Dhillon, V., Metcalf, D., and Hooper, M. (2017). The hyperledger project. In *Blockchain enabled applications*, pages 139–149. Springer.

[Ferreira and Silva 2013] Ferreira, P. V. and Silva, M. A. (2013). Comentário editorial o processo editorial. da submissão à rejeição (ou aceite). *Revista Ibero Americana de Estratégia*, 12(3).

[FURTADO and DA COSTA 2010] FURTADO, A. and DA COSTA, J. (2010). *PRATICA DE ANALISE E PROJETO DE SISTEMAS*. JULIO VALENTE.

[Jalote 2005] Jalote, P. (2005). *Software project management in practice*. Tsinghua University Press Ltd.

[Nakamoto et al. 2008] Nakamoto, S. et al. (2008). *Bitcoin: A peer-to-peer electronic cash system*. Working Paper.

[Nielsen 1993] Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Pressman and Maxim 2016] Pressman, R. and Maxim, B. (2016). *Engenharia de Software-8^a Edição*. McGraw Hill Brasil.

[Raval 2016] Raval, S. (2016). *Decentralized applications: harnessing Bitcoin's blockchain technology*. "O'Reilly Media, Inc.".

[Sommerville 2011] Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, 137035152.

[Starbuck 2005] Starbuck, W. H. (2005). How much better are the most-prestigious journals? the statistics of academic publication. *Organization Science*, 16(2):180–200.

[Swan 2015] Swan, M. (2015). *Blockchain: Blueprint for a new economy*. "O'Reilly Media, Inc.".

[Van Wyk 1998] Van Wyk, G. (1998). Publish or perish. a system and a mess. *Systemic Practice and Action Research*, 11(3):245–257.

APÊNDICE A – Configuração e instalação do sistema em produção

A.1 Instalação

- **Sumário:**
  - **1. Pré-requisitos**
  - **2. Instalação**
  - **3. Inicializar ferramenta em produção**

- **1. Pré-requisitos**
  - A partir do *Ubuntu 16.04.5 LTS*
  - *Docker*: Versão 17.03 ou superior
  - *Docker-Compose*: Versão 1.8 ou superior
  - *Node*: 8.9 ou superior
  - *npm*: v5.x
  - *git*: 2.9.x ou superior
  - *Python*: 2.7.x

- **2. Instalação**

  Na raiz do projeto, há um *script em Bash* chamado *dasp-install.sh* que ao ser executado pelo terminal: "*./dasp-install.sh*"executa comandos para instalação e preparação do ambiente:

  - *Download* do *Hyperledger Fabric*.
  - Instalação dos pré-requisitos para o *Hyperledger Composer*.
  - Instalação do *GO-IPFS*.
  - Instalação dos módulos do *Hyperledger Composer*.
  - Inicia fluxos Node-RED.

  - Instala e Prepara os projetos Client e API para execução.

- **3. Inicializar ferramenta em produção**

  Ainda na raiz do projeto, há outro arquivo chamado *dasp-start.sh*, responsável pela inicialização das dependencias do sistema e das aplicações Client e API em produção. A execução do comando: "*./dasp-start.sh*"executa as seguintes funções:

  - Inicia o *Hyperledger Fabric*.
  - Instala o *BNA* no *Hyperledger Fabric*.
  - Inicia o *IPFS*.
  - Inicia o Client na porta 8080.

  Ao fim da execução, a página inicial da aplicação deve abrir no endereço: http://localhost:8080.