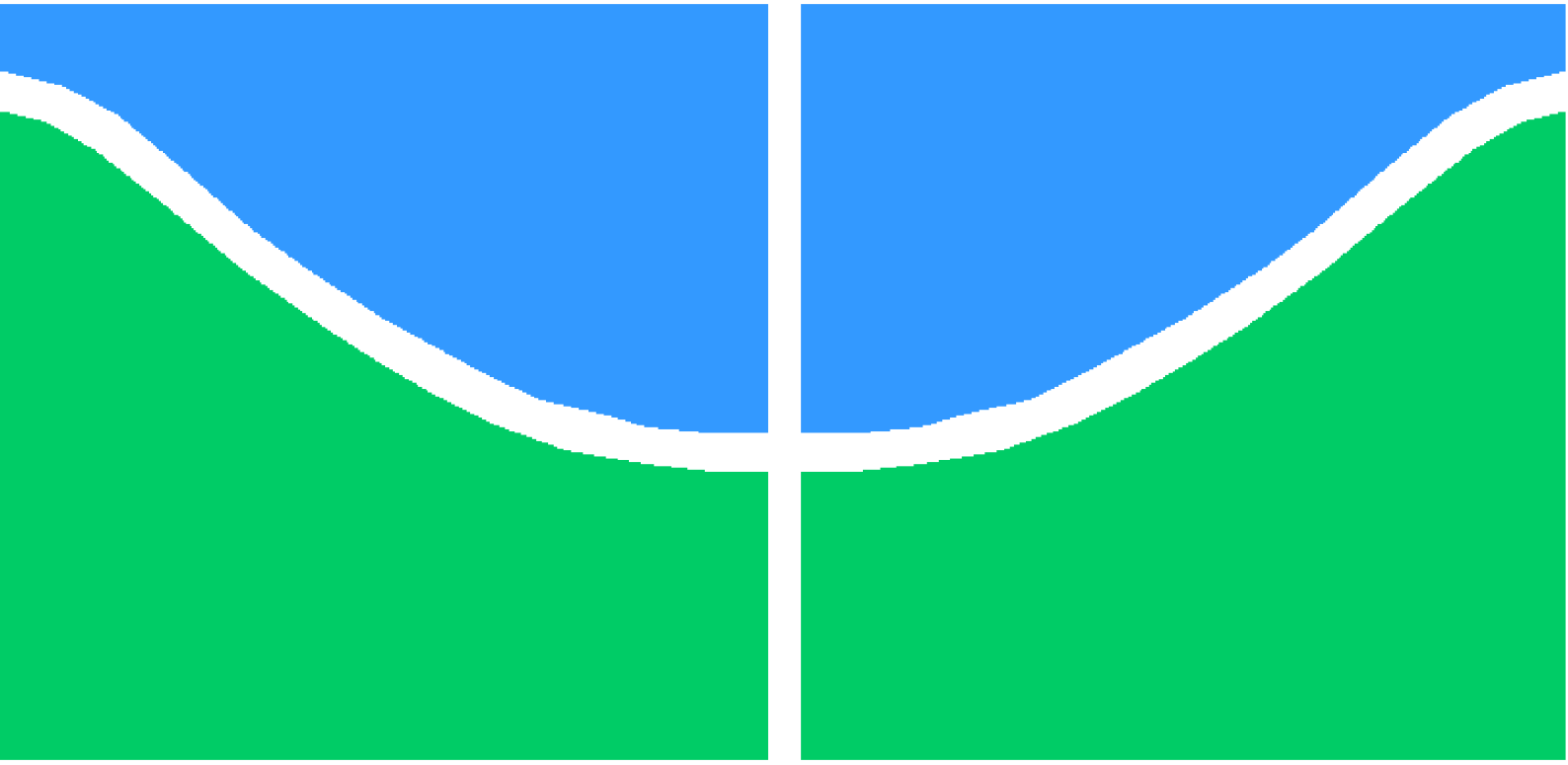


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Aplicação descentralizada de votação aberta usando blockchain Ethereum

Autores: Lucas Melo dos Santos, Eduardo Nunes Picolo

Orientador: Prof. Dr. Ricardo Matos Chaim

Brasília, DF

2023

Lucas Melo dos Santos, Eduardo Nunes Picolo

Aplicação descentralizada de votação aberta usando blockchain Ethereum

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 17 de Julho de 2023:

Prof. Dr. Ricardo Matos Chaim
Orientador

Prof. Dr. Vandor Roberto Vilardi Rissoli
Convidado 1

Prof. Dr. Giovanni Almeida Santos
Convidado 2

Brasília, DF
2023

Resumo

A tecnologia *blockchain*, originada como solução para transações financeiras com o Bitcoin, tem se destacado em diversas aplicações de software. Ao utilizar contratos inteligentes na *blockchain* do Ethereum, é possível estabelecer regras de negócio e aplicá-las em diferentes setores, incluindo sistemas de votação. Neste trabalho, foi desenvolvida uma aplicação descentralizada (DApp) que utiliza a tecnologia Ethereum para garantir segurança, transparência e integridade no processo de uma votação aberta. A metodologia adotada incluiu pesquisa bibliográfica para embasamento teórico, coleta de dados e a utilização do *framework* Scrum e da metodologia ágil Kanban para apoiar o desenvolvimento tecnológico. É realizada uma análise das vantagens e desafios do uso da tecnologia *blockchain* do Ethereum nesse contexto específico, identificando suas contribuições positivas e limitações. A aplicação desenvolvida proporciona aos usuários a capacidade de auditar seus votos, assegurando que sejam armazenados de forma imutável.

Palavras-chave: blockchain, contratos inteligentes, Ethereum, votação, aplicação descentralizada, DApp.

Abstract

Blockchain technology, which originated as a solution for financial transactions with Bitcoin, has been highlighted in several software applications. By using smart contracts on the Ethereum blockchain, it is possible to establish business rules and apply them in different sectors, including voting systems. In this work, a decentralized application (DApp) was developed that uses Ethereum technology to guarantee security, transparency, and integrity in the process of an open vote. The methodology adopted included bibliographical research for theoretical basis, data collection and the use of the Scrum framework and the agile Kanban methodology for technological development. An analysis of the advantages and challenges of using Ethereum blockchain technology in this specific context is carried out, identifying its positive contributions and limitations. The developed application provides users with the ability to audit their votes, ensuring that they are stored immutably.

Key-words: blockchain, smart contracts, Ethereum, voting, decentralized application, DApp.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
DApp	Decentralized application
DPoS	Delegated Proof of Stake
ETH	Ether
E2E	End-to-End
EVM	Ethereum Virtual Machine
IDE	Integrated Development Environment
MACI	Minimal Anti-Collision Infrastructure
PoS	Proof of Stake
PoW	Proof of Work
UnB	Universidade de Brasília

Sumário

1 Introdução

1.1 Contexto

Eleição é o processo no qual cada participante pode votar em seus representantes de uma dada lista de candidatos (BOSRI et al., 2019). A forma mais tradicional de votação é a com uso de papel. Esse sistema de votação enfrenta riscos com falha humana, lenta contagem dos votos e confiabilidade no resultado. Nesse contexto, e-*voting* foi proposto como solução para muitos dos desafios de sistemas de votação em papel para garantir eleições livre de erros e vieses (TAŞ; TANRıöVER, 2020).

Eleições tradicionais são caras, especialmente quando há centenas de centros de votação distribuídos e milhões de eleitores. *E-voting* possui várias vantagens sobre sistemas de votação tradicionais, especialmente custo operacional, logística, falha humana e resultados mais rápidos. O voto eletrônico contribui para a contagem mais rápida dos votos e entrega dos resultados das eleições, além disso, é esperado que o voto eletrônico leve a resultados mais confiáveis, pois o erro humano é excluído(ACE, c2023).

Apesar dos benefícios, a proposta de votação eletrônica traz alguns riscos, como segurança e integridade dos dados. Vários dos sistemas de votação eletrônica atuais são baseados em projetos privados e controlados por uma única entidade, diminuindo a confiança e credibilidade dos eleitores no processo de votação (MOURA; GOMES, 2017). Essa realidade estimula a busca por pesquisas e projetos inovadores que possam trazer melhorias para as condições atuais.

1.2 Justificativa

A facilidade de realizar o processo de votação eletronicamente possui o potencial de promover uma maior participação pública (ANANE; FREELAND; THEODOROPOULOS, 2007). No entanto, em um sistema de votação do mundo real, é crucial garantir a precisão, privacidade, auditabilidade e elegibilidade durante o processo. Embora a votação eletrônica ofereça diversos benefícios para as eleições, também apresenta desafios, como a integridade dos dados, confiabilidade, transparência e sigilo do voto (TAŞ; TANRıöVER, 2020).

Nesse contexto, a tecnologia *blockchain* surge como uma solução recente para impulsionar os sistemas utilizados em diversos domínios. Especificamente nos sistemas de votação, a *blockchain* mostra-se uma proposta promissora, pois pode contribuir significativamente para a garantia da confidencialidade e integridade dos dados (TAŞ; TANRıöVER,

2020). A Ethereum, como uma plataforma de *blockchain* de código aberto, destaca-se como uma das mais populares. Por meio de contratos inteligentes, que são trechos de código que definem regras de negócio (ETHEREUM, 2022a), é possível desenvolver uma ampla variedade de soluções.

Sendo assim, a finalidade deste trabalho é utilizar a blockchain Ethereum para criar um sistema de votação aberta, onde os votos de cada eleitor são tornados públicos e registrados de forma transparente.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral desenvolver uma aplicação descentralizada de votação aberta utilizando a *blockchain* Ethereum e avaliar a solução segundo as propriedades que um sistema de votação eletrônica deve possuir.

1.3.2 Objetivos Específicos

- Implementar um sistema que permita realizar votações abertas de diferentes tipos, utilizando contratos inteligentes na *blockchain* Ethereum;
- Garantir a integridade, a autenticidade e a imutabilidade dos resultados das votações;
- Prover mecanismos de auditoria e verificação do código-fonte e dos registros das transações realizadas na *blockchain*;
- Enumerar as propriedades de avaliação com base nos padrões estabelecidos na literatura especializada;
- Avaliar solução desenvolvida baseando-se nos requisitos gerais de uma votação eletrônica.

1.4 Metodologia

Este trabalho é um desenvolvimento tecnológico apoiado por uma pesquisa bibliográfica. A pesquisa bibliográfica incluiu a escolha e delimitação do tema, coleta de dados, localização de informações e anotações. Foram realizadas buscas nas bases de conhecimento IEEE e Scopus por artigos sobre votação eletrônica, blockchain e Ethereum. A metodologia de desenvolvimento foi realizada através do *framework* Scrum e da metodologia ágil Kanban. A metodologia será mais detalhada do capítulo 3

1.5 Organização do Trabalho

Este trabalho de conclusão de curso está organizado nos seguintes capítulos:

- **Capítulo 1 - Introdução:** Neste capítulo foram apresentados o contexto do trabalho, a justificativa e os objetivos deste trabalho;
- **Capítulo 2 - Referencial teórico:** Descreve os conceitos que fundamentam este trabalho. O capítulo discorre sobre os conceitos de votação e blockchain, assim como da tecnologia Ethereum.
- **Capítulo 3 - Metodologia:** Apresenta a proposta deste trabalho. O capítulo contém as seções que descrevem a metodologia de desenvolvimento, assim como a aplicação a ser desenvolvida.
- **Capítulo 4 - Resultados e avaliação:** Este capítulo apresenta os resultados obtidos na implementação da aplicação de votação eletrônica. Além disso, é feita uma avaliação dos resultados em relação aos objetivos propostos, destacando os pontos fortes e as áreas que necessitam de melhorias.
- **Capítulo 5 - Conclusões:** Neste capítulo são apresentadas as conclusões do trabalho, destacando os principais resultados alcançados. São discutidos os desafios enfrentados, as limitações do estudo e as sugestões para trabalhos futuros.

2 Referencial Teórico

2.1 Considerações Iniciais

Para melhor entendimento acerca do conceito de *blockchain* e de sistemas de votação eletrônica, ou *e-voting*, como será referido adiante, este capítulo introduz o processo de votação e as propriedades principais que este deve possuir, além de apresentar a arquitetura da tecnologia *Ethereum* e o funcionamento básico de uma rede blockchain, bem como o uso de contratos inteligentes. Também é apresentada uma seção voltada a descrever diferentes implementações de *e-voting*.

2.2 Votação eletrônica

E-voting é um método de votação que utiliza dispositivos eletrônicos para registrar ou contar os votos (TAş; TANRıöVER, 2020). Esse método traz grandes vantagens para uma sociedade cada vez mais globalizada e digital. O voto eletrônico é mais barato, aumenta a participação e a confiança dos eleitores e transforma um processo que existe há milhares de anos em um que é excepcionalmente eficiente e compatível com os avanços culturais e tecnológicos (TSAHKNA, 2013). No entanto, ainda há muitas ressalvas sobre este processo, principalmente em relação à segurança. Os sistemas de voto eletrônico, em particular o usado na Estônia, foram amplamente criticados, sendo a centralização inerente a maior das preocupações (TAş; TANRıöVER, 2020).

A votação eletrônica pode ser realizada tanto em ambientes controlados quanto em ambientes não controlados. Em ambientes controlados a votação é realizada em locais supervisionados por funcionários designados pelo órgão responsável pela administração eleitoral (WOLF; NACKERDIEN; TUCCINARDI, 2011). Dessa forma, a administração eleitoral pode controlar em grande parte a tecnologia de votação, bem como os procedimentos e condições em que os eleitores estão votando. A votação eletrônica em ambientes controlados pode ser vista como o equivalente da votação tradicional em papel realizada em centros de votação.

A votação eletrônica em ambientes não controlados ocorre sem supervisão e a partir de dispositivos de votação que não podem ser controlados pela administração eleitoral. Isso pode acontecer em casa, no computador pessoal do eleitor ou potencialmente em qualquer lugar em dispositivos móveis, ou públicos. Preocupações com o sigilo do voto, votação em família, intimidação, compra de votos, perda do dia da eleição, impacto da divisão digital e a separação técnica da identidade do eleitor e o voto, bem como a integridade

técnica do dispositivo a partir do qual os votos são lançados, precisam de considerações específicas (WOLF; NACKERDIEN; TUCCINARDI, 2011). As formas atuais de votação pela internet ainda não conseguiram fornecer uma solução definitiva para tais preocupações (TAş; TANRıöVER, 2020). A votação eletrônica em ambientes não controlados pode ser vista como o equivalente da votação por correio ou voto ausente.

2.2.1 Tipos de *e-voting*

Ao discutir as vantagens e desvantagens dos diversos sistemas de voto eletrônico, é útil distinguir várias tipologias de sistemas. Todos os tipos têm diversos pontos fortes e fracos, tanto quando comparados entre si como quando comparadas ao voto em papel tradicional. Não existe um sistema de voto eletrônico perfeito e os sistemas disponíveis continuam a evoluir com os avanços tecnológicos em andamento (WOLF; NACKERDIEN; TUCCINARDI, 2011).

2.2.1.1 Sistemas de votação com cartão perfurado

Com sistemas de votação com cartão perfurado, a cédula é um cartão no qual os eleitores perfuram buracos com um dispositivo de perfuração fornecido ao lado de seu candidato ou escolha. Após perfurar os buracos, o eleitor pode colocar a cédula em uma urna eleitoral ou em um dispositivo eletrônico de apuração de votos no local de votação.

A recontagem de votos na Flórida durante a eleição presidencial de 2000 criou um debate sobre a confiabilidade dos sistemas de votação com cartão perfurado (ACE, c2023).

2.2.1.2 Sistemas de votação por escaneamento ótico

Esses sistemas usam um scanner ótico para ler e contar cédulas marcadas. Os sistemas de votação por escaneamento ótico combinam papel com dispositivos eletrônicos. Todos os sistemas mantêm uma cédula de papel que serve como registro tangível da intenção do eleitor, permitindo a recontagem manual de cédulas caso o sistema falhe. A grande vantagem é que o processo de contagem pode ser feito em um local central e a contagem é muito mais rápida, além de ser facilmente compreensível pelo eleitor (ACE, c2023).

2.2.1.3 Máquinas de votação eletrônica com registro direto

Com uma máquina, a votação pode ser feita no dia da eleição ou pode ser usada como um dispositivo de votação antecipada em postos de votação. O eleitor apenas pressiona um botão ou tela sensível ao toque para votar em seu candidato ou escolha. Após a eleição, a máquina produz uma tabulação dos votos armazenados em um componente

de memória removível e/ou em cópia impressa. O sistema também pode permitir a transmissão de cédulas individuais ou totais de votos para um local central onde o resultado pode então ser consolidado (ACE, c2023).

2.2.1.4 Trilha de auditoria em papel verificada pelo eleitor

Uma trilha de auditoria em papel, verificada pelo eleitor, ou um registro em papel verificado, não constitui um sistema de votação eletrônica em si, mas refere-se a um componente que pode ser combinado com diferentes tipos de sistemas de votação. Esse registro implica que um voto em papel é impresso por um dispositivo eletrônico utilizado para realizar o voto, sendo destinado a um sistema de verificação independente projetado para permitir que os eleitores verifiquem se o voto foi registrado corretamente, detectem possíveis fraudes eleitorais ou falhas, e forneçam um meio de auditar os resultados eletrônicos armazenados (ACE, c2023).

2.2.1.5 Sistemas de votação pela internet

Os sistemas de votação pela internet, também conhecidos como *I-voting*, são uma forma específica de *e-voting* em que o voto é realizado e/ou transmitido através da internet (ACE, c2023). Essa modalidade de votação pode assumir diferentes formas, dependendo se é utilizada em ambientes controlados ou não controlados. No caso da votação remota em um ambiente não controlado, os eleitores têm a possibilidade de votar de qualquer lugar e utilizando qualquer dispositivo. Nesse cenário, as máquinas e o ambiente físico não estão sob o controle direto das autoridades eleitorais, e o voto é transmitido pela internet. Embora esse método ofereça diversas vantagens aos eleitores, é também o mais suscetível a preocupações de segurança, como questões relacionadas à confiabilidade da internet como meio de transmissão de informações confidenciais, ataques de hackers e a possibilidade de coerção durante o processo de votação (ACE, c2023).

Em um ambiente controlado, a autenticação dos eleitores pode ser feita por meio de métodos tradicionais, e a realização do voto ocorre em máquinas localizadas fisicamente em postos de votação oficiais ou em locais públicos supervisionados por autoridades eleitorais (ACE, c2023).

2.2.2 Benefícios e riscos

A votação eletrônica é frequentemente considerada uma ferramenta para avançar a democracia, construir confiança na gestão eleitoral, adicionar credibilidade aos resultados e aumentar a eficiência geral do processo eleitoral. Quando implementadas corretamente, as soluções de votação eletrônica têm o potencial de eliminar fontes de fraude, agilizar o processamento de resultados, aumentar a acessibilidade e tornar a votação mais conveni-

ente para os cidadãos. Além disso, em alguns casos, pode reduzir os custos das eleições a longo prazo (ACE, c2023).

No entanto, nem todos os projetos de votação eletrônica conseguem cumprir essas promessas. Há conflitos legislativos e técnicos, bem como ceticismo e oposição à introdução de novas tecnologias de votação. Os desafios inerentes à votação eletrônica são consideráveis e estão ligados à complexidade dos sistemas e procedimentos eletrônicos (ACE, c2023). Muitas soluções de votação eletrônica carecem de transparência tanto para os eleitores quanto para os administradores eleitorais. A maioria dessas soluções é compreendida apenas por um pequeno número de especialistas, e a integridade do processo eleitoral frequentemente depende de um grupo limitado de operadores de sistemas, em vez de milhares de funcionários das seções eleitorais.

Conforme a análise realizada por Bokslag (BOKSLAG; VRIES, 2016) e Wolf, Nackerdien e Tuccinardi (2011), os benefícios do *e-voting* incluem:

- Contagem rápida: A tabulação e contagem dos votos podem ser feitas em poucos minutos;
- Resultados mais confiáveis;
- Menos intensivo em mão de obra: Embora as soluções de votação eletrônica ainda exijam funcionários para supervisão, não é necessária mão de obra adicional para a contagem;
- Redução de custos: Possíveis economias a longo prazo com a redução do tempo dos trabalhadores das seções eleitorais e dos custos de produção e distribuição de cédulas de papel;
- Melhoria de acessibilidade. A votação eletrônica aumenta a conveniência dos eleitores e é mais acessível do que a votação em papel para pessoas com deficiência. Isso inclui recursos como áudio ou leitores de tela para eleitores cegos, bem como a votação pela internet para eleitores que não podem sair de casa ou para eleitores no exterior. O *i-voting* oferece ainda mais acessibilidade, eliminando a necessidade de deslocamento até uma seção de votação, o que pode ser especialmente útil para pessoas com deficiência física ou cidadãos no exterior. Além disso, a votação pela internet pode reduzir a necessidade de votação por procuração, como ocorre em alguns países;
- Mais sintonizado com as necessidades de uma sociedade cada vez mais móvel;
- Prevenção de fraudes por meio da redução de intervenção humana;
- Aumento da participação eleitoral.

Alguns dos problemas associados ao voto eletrônico incluem:

- Falta de transparência e dificuldade de compreensão do sistema: Os sistemas de votação eletrônica podem ser muito complexos e, em muitos casos, os cidadãos não têm acesso suficiente a informações sobre como eles funcionam e como os votos são contados. Isso pode dificultar a compreensão do funcionamento do sistema e diminuir a confiança do eleitor no uso do mesmo (BOKSLAG; VRIES, 2016; WOLF; NACKERDIEN; TUCCINARDI, 2011).
- Falta de padrões acordados: Os sistemas de votação eletrônica não possuem um conjunto de padrões amplamente aceitos que garantam a segurança e a integridade dos votos ou a certificação do sistema (ACE, c2023; WOLF; NACKERDIEN; TUCCINARDI, 2011).
- Risco de violação do sigilo do voto: Em sistemas que realizam tanto a autenticação do eleitor quanto a votação, há um risco maior de violação do sigilo do voto (WOLF; NACKERDIEN; TUCCINARDI, 2011).
- Risco de manipulação por funcionários internos: Os sistemas de votação eletrônica podem ser vulneráveis a manipulação por funcionários com acesso privilegiado (WOLF; NACKERDIEN; TUCCINARDI, 2011).
- Redução do controle da administração eleitoral: A administração eleitoral pode ter menos controle sobre o processo eleitoral em decorrência da dependência em relação aos fornecedores e/ou tecnologias (WOLF; NACKERDIEN; TUCCINARDI, 2011).
- Necessidade de campanhas de educação adicionais: Os eleitores podem precisar de campanhas de educação adicionais para compreender como funcionam os sistemas de votação eletrônica (ACE, c2023; WOLF; NACKERDIEN; TUCCINARDI, 2011).

A votação pela internet apresenta algumas desvantagens adicionais. Primeiramente, assume-se que os sistemas dos clientes são confiáveis. No entanto, é importante ressaltar que muitos computadores domésticos são vulneráveis e podem ser comprometidos por *botnets*, representando um risco para a integridade dos resultados eleitorais. Além disso, garantir a resistência à coerção na votação pela internet é uma tarefa desafiadora. Enquanto na votação presencial o eleitor pode votar de forma independente e sem deixar rastros tangíveis do seu voto, na votação pela internet, o eleitor pode ser observado ou até mesmo coagido a fornecer uma confirmação do seu voto, prejudicando a garantia da liberdade de escolha do eleitor (BOKSLAG; VRIES, 2016).

2.2.3 Processo de votação

Um sistema eleitoral é composto por um conjunto de regras e protocolos que regulam todo o processo de votação. Em um ambiente eleitoral tradicional, o processo de votação segue um procedimento muito específico:

1. É coletada uma lista de eleitores elegíveis;
2. A elegibilidade dos eleitores é verificada, garantindo que apenas os eleitores legítimos possam votar;
3. As cédulas são coletadas;
4. O total é calculado contando os votos.

Esse padrão deve ser cautelosamente seguido na implementação do sistema eletrônico (ANANE; FREELAND; THEODOROPOULOS, 2007).

2.2.4 Requisitos gerais

De acordo com Anane, Freeland e Theodoropoulos (2007) e Bokslag e Vries (2016), as seguintes propriedades principais devem ser garantidas:

- Transparência/Integridade: Para garantir que o público, bem como outros principais *stakeholders* no processo, tenham confiança na solução de votação eletrônica;
- Sigilo/privacidade do voto. Para proteger o sigilo do voto em todas as etapas do processo de votação;
- Unicidade: Para garantir que cada voto seja contado apenas uma vez;
- Elegibilidade do eleitor. Para garantir que apenas as pessoas com o direito de voto possam votar;
- Verificabilidade/auditoria: Idealmente, o eleitor pode verificar se seu voto foi contado no resultado. Se não, auditores independentes devem conseguir verificar a integridade do resultado da eleição;
- Acessibilidade: Para garantir acessibilidade ao maior número possível de eleitores, especialmente em relação às pessoas com deficiência;
- Liberdade de voto/resistência à coerção. Para manter o direito dos eleitores de formar e expressar sua opinião de maneira livre, sem coerção ou influência indevida;
- Disponibilidade: Para garantir que a solução de votação esteja disponível para o eleitor durante a eleição;

- Equidade: Para garantir que resultados preliminares não afetem as decisões de outros eleitores.

2.3 Projetos de votação eletrônica

A revisão dos trabalhos relacionados é fundamental para entender o estado da arte e os desafios existentes na área de estudo deste projeto. Nesta seção são apresentados os principais estudos relevantes para o tema.

2.3.1 *Secure Digital Voting System based on Blockchain Technology*

O sistema proposto por Khan, Arshad e Khan (2018) em sua publicação *Secure Digital Voting System based on Blockchain Technology* (Sistema de Votação Segura baseado em Tecnologia Blockchain) é projetado para suportar votações reais, considerando privacidade, elegibilidade, conveniência, sem necessidade de recibo e verificabilidade, e projetado usando uma interface baseada na web, com medidas como digitalização de impressões digitais para proteção contra votos duplos, além de uma interface de administrador, permitindo iguais direitos de participação e mantendo anonimidade dos eleitores.

Os seguintes requisitos de um sistema *e-voting* foram atendidos:

- Privacidade: O sistema usa *blockchain* para garantir privacidade do eleitor, gerando um identificador único protegido e dificultando rastrear o voto, assim protegendo o eleitor.
- Elegibilidade: O sistema permite somente eleitores registrados votarem, usando identificadores únicos e tecnologia de biometria para garantir autenticidade e evitar votos duplos.
- Sigilo/privacidade do voto: O sistema permite ao eleitor votar de acordo com sua escolha e cria um *hash* criptográfico para cada voto, o que é importante para verificar se o voto foi incluído na contagem, mas não permite extrair informações sobre como o eleitor votou.
- Verificabilidade/auditoria: Ao votar, o usuário recebe um ID único da transação em forma de *hash* criptográfico, que pode ser usado para rastrear se o voto foi contabilizado, mas não permite ver como foi votado, essa medida foi adotada para proteger o eleitor de ameaças.

2.3.2 *Blockchain-Based E-Voting System*

Hjálmarsson et al. (2018) em seu trabalho publicado *Blockchain-Based E-Voting System* (Sistema de Votação Eletrônica baseado em Blockchain) propõe um sistema de

votação eletrônica baseado em *blockchain* que utiliza contratos inteligentes para garantir segurança e eficiência em custos, preservando a privacidade dos eleitores. Foi mostrado que a tecnologia *blockchain* oferece uma nova possibilidade para superar as limitações e barreiras de adoção dos sistemas de votação eletrônica, garantindo a segurança e integridade das eleições e transparência, usando *blockchain* privado Ethereum. Porém, para países de maior tamanho, algumas medidas adicionais seriam necessárias para suportar maior capacidade de transações por segundo.

2.3.3 *Follow My Vote*

O sistema Follow My Vote oferece uma solução baseada em *Delegated Proof of Stake* (DPoS) para votação *online* (FOLLOWMYVOTE, Ano desconhecido). Isso permite que os eleitores deleguem sua influência na rede para um delegado em quem confiam. O protocolo do Follow My Vote inclui uma etapa inicial de registro, na qual as informações do usuário (como foto e cartão de identidade governamental) são verificadas por uma entidade centralizada. Embora o Follow My Vote afirme usar assinaturas cegas para garantir anonimato após o voto ser enviado para a *blockchain*, essa etapa de registro apresenta uma vulnerabilidade preocupante, já que os eleitores são anônimos para a entidade responsável pela verificação.

2.3.4 Estonia *i-voting*

Desde 2005, a Estônia tem sido defensora do voto eletrônico, quando realizou pela primeira vez eleições municipais usando-o. Pouco tempo depois, em 2007, tornou-se o primeiro país do mundo a usar voto eletrônico nas eleições parlamentares. Hoje em dia, mais de 30% de seus votos são computados *online* (VALIMISED, 2019). No entanto, seu sistema de *i-voting* (veja a Seção 2.2 para uma melhor descrição desse conceito) foi criticado devido às suas graves limitações arquiteturais e lacunas procedurais (SPRINGALL et al., 2014). Além disso, a falta de verificabilidade de ponta a ponta (E2E, sigla em inglês) é preocupante, por requerer confiança na integridade do computador do eleitor, dos componentes do servidor e dos funcionários eleitorais. Para fornecer transparência ao sistema, foram fornecidas gravações ao vivo para o público. No entanto, em algumas ocasiões, os funcionários digitavam acidentalmente senhas raiz para os servidores eleitorais à vista das câmeras e usavam *drives* de armazenamento pessoais para mover os resultados oficiais das eleições fora do servidor de contagem, como enfatizado em Springall et al. (2014). Além disso, o pessoal eleitoral foi gravado usando computadores pessoais para preparar o *software* cliente para a eleição, o que seria posteriormente distribuído para o público, criando a possibilidade de espalhar código malicioso para o sistema do eleitor.

2.4 *Blockchain*

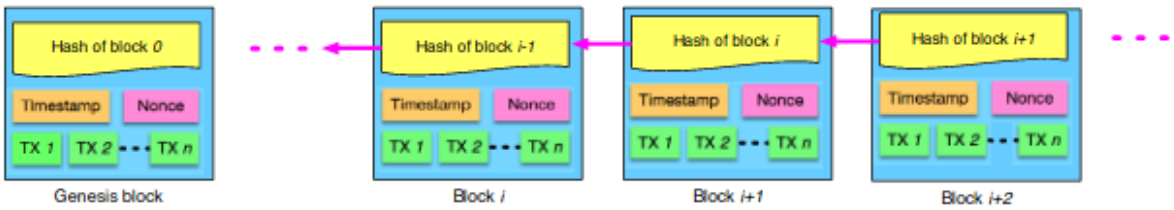
As primeiras pessoas a descreverem e trabalharem com uma cadeia de blocos conectados criptograficamente foram Haber e Stornetta (1991). No trabalho publicado, descrevem a projeção de um registro de tempo de transação (*timestamp*) para documentos digitais que evitasse adulterações. O primeiro grande sistema baseado em *blockchain* foi a criptomoeda Bitcoin, que descreveu como a tecnologia poderia ser usada para resolver o problema de manter a ordem das transações e evitar o problema de gasto duplo (NAKAMOTO, 2008).

A tecnologia *blockchain* pode ser entendida como um livro-razão distribuído. Um livro-razão é um registro contábil que documenta todas as transações financeiras de uma organização. O documento contém informações importantes, como o valor, data, hora e outras informações relevantes para cada transação. O livro-razão é crucial para garantir que todas as transações sejam registradas com precisão e transparência, permitindo que as partes envolvidas tenham confiança nas informações registradas.

No contexto da *blockchain*, a construção desse livro-razão se da através de uma estrutura de cadeia de blocos definidos por um *timestamp* e *hashes* criptográficos, além de um número arbitrário encontrado pelo minerador, chamado de *nonce*, para provar o esforço computacional realizado em uma rede que use o *Proof of Work* como algoritmo de consenso. Essa cadeia cresce ao adicionar mais bloco, sendo que cada novo bloco possui também a informação da *hash* do bloco anterior, com exceção do bloco gênesis, conforme exemplificado na Figura 1. Antes de um bloco ser acrescentado à *blockchain*, as transações são verificadas pelos nós ou validadores. Dentro de cada bloco, as transações são organizadas a partir de uma estrutura chamada de árvore Merkle (NAKAMOTO, 2008).

Os dados armazenados em uma rede *blockchain* não podem ser modificados ou adulterados, por serem transparentes e constantemente verificados por uma rede de computadores, tornando-a descentralizada e resistente a ataques maliciosos, proporcionando assim um nível de segurança que não é possível com bancos de dados convencionais (TAŞ; TANRıöVER, 2020). Dessa forma, a *blockchain* permite a criação de confiança sem a necessidade de uma autoridade central (NAKAMOTO, 2008).

Figura 1 – Exemplo de sequência de blocos que constituem a *blockchain*



Fonte: (WANG et al., 2018).

2.4.1 Criptografia

A criptografia é o processo pelo qual as informações são codificadas para transmissão ou armazenamento seguro, de modo que apenas indivíduos autorizados possam decodificá-las. Cifras são algoritmos utilizados para criptografar ou descriptografar dados, tornando-os ininteligíveis para terceiros não autorizados sem a chave secreta necessária para descriptografá-los. A proteção de um ecossistema de blockchain requer o uso de vários primitivos criptográficos, incluindo funções de *hash*, criptografia de chave simétrica, assinaturas digitais e criptografia de chave pública (BASHIR, 2018).

2.4.1.1 Criptografia simétrica

A criptografia simétrica, também conhecida como criptografia de chave compartilhada, refere-se a um tipo de criptografia na qual a chave usada para criptografar os dados é a mesma usada para descriptografá-los. A chave deve ser estabelecida ou acordada antes que a troca de dados ocorra entre as partes que se comunicam (BASHIR, 2018).

2.4.1.2 Criptografia assimétrica

Os sistemas baseados em blockchain são fortemente dependentes da técnica de criptografia de chaves públicas e privadas (BASHIR, 2018).

A criptografia assimétrica é um tipo de criptografia na qual a chave usada para criptografar os dados difere da chave usada para descriptografar os dados. Também é conhecida como criptografia de chave pública. Utiliza tanto chaves públicas quanto privadas para criptografar e descriptografar dados, respectivamente (BASHIR, 2018). A chave pública é compartilhada entre vários usuários, enquanto a decodificação dos dados é realizada com uma chave privada que não é compartilhada. Embora seja mais lenta, a criptografia assimétrica é mais segura. A chave pública pode ser acessada por qualquer pessoa, mas a chave privada é mantida em sigilo. Nesta técnica de criptografia, uma mensagem codificada com a chave pública pode ser decodificada somente com a chave privada

correspondente. Já uma mensagem codificada com a chave privada pode ser decodificada com a chave pública (STALLINGS, 2006).

2.4.2 Funções *hash*

Funções *hash* são usadas para criar resumos de comprimento fixo de *strings* de entrada de tamanho arbitrário (BASHIR, 2018). O propósito central da utilização de uma função *hash* é assegurar a integridade dos dados.

Uma função *hash* F aceita um bloco de tamanho variável de dados X como entrada e produz um valor de *hash* de tamanho fixo H = F(X), sendo que uma mudança em *bits* de X causa alteração no valor do *hash* obtido (STALLINGS, 2006).

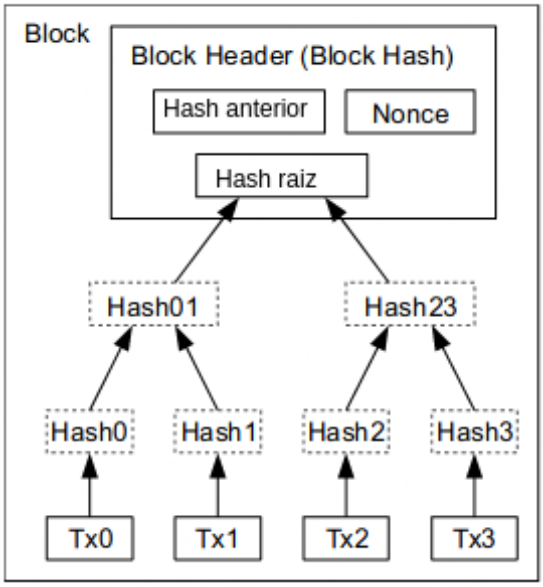
As funções *hash* usadas em aplicações de segurança são chamadas de funções *hash* criptográficas e devem ser inviáveis de serem decifradas. Isso significa que não deve ser possível encontrar dados que produzam um resultado de *hash* pré-especificado ou dois conjuntos de dados que produzam o mesmo resultado de *hash*. Devido a essas características, as funções *hash* são comumente usadas para verificar se os dados foram alterados ou não (STALLINGS, 2006).

2.4.2.1 Árvore de Merkle

A Árvore de Merkle é uma estrutura de dados usada para verificar a integridade de dados em uma *blockchain*. A estrutura de uma Árvore de Merkle é uma árvore binária em que cada nó tem dois filhos, exceto os nós folha que representam os *hashes* das transações de dados, conforme a Figura 2. Os nós intermediários são gerados a partir dos *hashes* dos filhos (MERKLE, 1988). Este processo é repetido sucessivamente até chegar ao nó raiz, o qual é o *hash* resumido de todas as transações naquele bloco.

A tecnologia blockchain se beneficia dessa estrutura, pois a mesma permite uma verificação eficiente da integridade dos dados, já que qualquer alteração em uma transação resultaria em mudanças em todos os *hashes* ao longo da árvore, o que facilitaria detectar qualquer tentativa de alteração maliciosa (NAKAMOTO, 2008).

Figura 2 – Árvore de Merkle



Fonte: (NAKAMOTO, 2008).

2.4.3 *Zero-knowledge proofs* (Provas de conhecimento zero)

A origem das provas de conhecimento zero remonta a um artigo de Goldwasser, Micali e Rackoff (1985) chamado "The knowledge complexity of interactive proof systems" (A complexidade do conhecimento dos sistemas de prova interativos), onde é apresentada uma definição amplamente aceita das provas de conhecimento zero: "Um protocolo de conhecimento zero é um método pelo qual uma das partes (o '*prover*') pode demonstrar à outra parte (o '*verifier*') a veracidade de algo, sem revelar nenhuma informação além do fato de que a afirmação específica em questão é verdadeira." (GOLDWASSER; MICALI; RACKOFF, 1985)

As provas de conhecimento zero tiveram um impacto significativo no campo da criptografia aplicada, oferecendo uma solução promissora para aprimorar a segurança das informações pessoais (ETHEREUM, 2022b). Considerando a demonstração da elegibilidade para participar do processo de votação em uma jurisdição específica sem revelar dados pessoais a terceiros, normalmente seria necessário apresentar documentos de identificação, como um título de eleitor ou identidade, o que poderia comprometer a privacidade.

No entanto, por meio do uso de provas de conhecimento zero, é possível comprovar a elegibilidade para participar do processo de votação sem expor informações confidenciais. O protocolo de conhecimento zero permite a geração de uma prova concisa de elegibilidade com base em critérios ou atributos predefinidos, como idade, residência e cidadania. Essa prova garante a validade da elegibilidade sem revelar detalhes pessoais ou comprometer a privacidade.

Ao aplicar esse conceito no contexto do processo de votação é possível apresentar uma prova de conhecimento zero às autoridades eleitorais, as quais podem verificar as propriedades da prova sem ter acesso a informações confidenciais. Isso assegura que o processo de votação mantenha a privacidade dos eleitores ao mesmo tempo que cumpra os requisitos de verificação e equilíbrio necessários para prevenir fraudes e garantir a integridade do processo de votação.

Para atingir esse objetivo, os protocolos de conhecimento zero se baseiam em algoritmos que recebem dados específicos como entrada e produzem uma resposta de "verdadeiro" ou "falso" (ETHEREUM, 2022b). O provador busca demonstrar conhecimento de informações ocultas. Essas informações confidenciais são a "testemunha" da prova, estabelecendo um conjunto de perguntas que só podem ser respondidas por alguém que possua esse conhecimento. O provador seleciona aleatoriamente uma pergunta, calcula a resposta correspondente e a envia ao verificador. O verificador escolhe outra pergunta aleatoriamente do conjunto e a apresenta ao provador para que ele a responda. Por fim, o provador aceita o desafio, calcula a resposta e a retorna ao verificador. A resposta do provador permite que o verificador verifique se o provador realmente tem acesso à testemunha. Para evitar que o provador esteja apenas adivinhando ou obtendo respostas corretas por acaso, o verificador pode fazer mais perguntas. Repetindo essa interação várias vezes, a probabilidade de o provador falsificar o conhecimento da testemunha diminui consideravelmente até que o verificador esteja satisfeito.

2.4.3.1 ZK-SNARKs

Zero-Knowledge Succinct Non-Interative Argument of Knowledge (Argumento de conhecimento sucinto não interativo de conhecimento zero) é uma forma de prova de conhecimento zero que possibilita que um provador convença concisamente qualquer verificador sobre a veracidade de uma afirmação específica. Além disso, garante o zero conhecimento computacional, sem a necessidade de interação entre o provador e o verificador.

2.4.4 Tipos de redes

De acordo com a natureza da acessibilidade de dados, a *blockchain* pode ser categorizada da seguinte maneira (LIN; LIAO, 2017):

- *Blockchain* pública: Neste tipo de *blockchain* qualquer pessoa pode ler e enviar transações;
- *Blockchain* privada: Neste tipo de *blockchain,* apenas uma organização ou todas as organizações subsidiárias do mesmo grupo têm permissão para ler e enviar transações;

- *Blockchain* de comunidade/consórcio: Neste tipo de *blockchain* vários grupos de organizações formam um consórcio com permissão para enviar transações e ler dados de transações;
- *Blockchain* híbrida: Esta é uma nova categoria em que qualquer uma das três *blockchain*, pública, privada ou de comunidade/consórcio pode ser combinada para facilitar transações. Uma plataforma de *blockchain* pode ser configurada em modo múltiplo usando uma *blockchain* híbrida.

Em relação à funcionalidade principal e ao suporte a contratos inteligentes, *blockchain* pode ser categorizada da seguinte maneira (HILEMAN; RAUCHS, 2017):

- *Stateless blockchain*: O sistema de *blockchain Stateless* se concentra apenas na otimização de transações e na funcionalidade de cadeia, o qual é a verificação da transação através do cálculo de *hashes*. Por ser independente da camada de lógica de contrato inteligente, não é afetado por erros ou vulnerabilidades no código de contrato inteligente;
- *Stateful blockchain*: Este tipo de *blockchain* oferece capacidades de computação de contratos inteligentes e transações. Também suporta lógicas de negócios multifacetadas, sua otimização e preservação de estados lógicos.

Segundo Michael, Cohn e Butcher (2018), a *blockchain* pode ser classificado em três tipos, baseado na necessidade de autorização para participar dele:

- *Blockchain* não-permissionada: Neste tipo de *blockchain*, não é necessária nenhuma permissão prévia para participar. Todos são permitidos a participar do processo de verificação e podem se juntar à rede de *blockchain* com sua própria capacidade computacional;
- *Blockchain* permissionada: Para participar deste tipo de *blockchain* é necessária uma permissão prévia. Somente as partes autorizadas podem executar nós para verificar as transações na rede de *blockchain*;
- *Blockchain* híbrido: É possível que um nó participe de um *blockchain* com permissão e de outro sem permissão ao mesmo tempo, para facilitar a comunicação entre os blockchains. Esse tipo de *blockchain* pode ser chamado de *blockchain* híbrido. Uma plataforma de *blockchain* também pode ser configurada para suportar o modelo com permissão e/ou sem permissão.

A Tabela 1 resume as principais características de cada rede de *blockchain* em relação ao permissionamento, mecanismos de consenso e gerenciamento.

Tabela 1 – Características das redes

Propriedade	**Pública**	**Privada**	**Consórcio**
Gerenciamento	Decentralizado	Uma organização	Múltiplas organizações
Participantes	Não permissionado	Permissionado	Permissionado
Centralizada	Não	Sim	Parcialmente
Eficiência	Baixa	Alta	Alta
Determinação do consenso	Todos Validadores/Mineradores	Participantes da organização	Validadores/Mineradores selecionados
Duração da transação	Longa	Curta	Curta

Fonte: (TAş; TANRıöVER, 2020).

2.4.5 Mecanismos de consenso

Os nós de uma rede podem trocar mensagens entre eles. Esses podem ser honestos, com falhas ou maliciosos, e eles possuem memória e processador. Um nó que exibe comportamento irracional também é conhecido como um nó bizantino (BASHIR, 2018). O processo de acordo entre nós sem relação de confiança sobre o estado final dos dados é chamado de consenso (BASHIR, 2018).

Segundo Bashir (2018), há vários requisitos que devem ser atendidos para fornecer os resultados desejados em um mecanismo de consenso:

- Concordância: Todos os nós honestos decidem o mesmo valor;
- Término: Todos os nós honestos encerram a execução do processo de consenso e eventualmente chegam a uma decisão;
- Validade: O valor acordado por todos os nós honestos deve ser o mesmo que o valor proposto inicialmente por pelo menos um nó honesto;
- Tolerância a falhas: O algoritmo de consenso deve ser capaz de ser executado na presença de nós falhos ou maliciosos (nós bizantinos);
- Integridade: Este é um requisito de que nenhum nó tome a decisão mais de uma vez em um único ciclo de consenso.

No contexto de *blockchain*, consenso é um conceito utilizado para prover meios de alcançar uma única versão da verdade em todos os *peers* da rede (BASHIR, 2018). Diferentes algoritmos podem ser utilizados para atingir consenso em uma rede, sendo os principais:

- *Proof of Work (PoW)*: É o mecanismo adotado pela *blockchain* do Bitcoin. Consiste na mineração, ou seja, no ato de adicionar blocos válidos na *blockchain* utilizando poder computacional. O objetivo da prova de trabalho é estender a cadeia. A cadeia mais longa é mais crível como a válida por ter o maior trabalho computacional feito para gerá-la (ETHEREUM, 2022a);
- *Proof of Stake (PoS)*: Em uma rede que utiliza *proof-of-stake* os validadores usam como garantia suas moedas. Na *blockchain Ethereum*, por exemplo, os validadores explicitamente apostam capital na forma de ETH em um contrato inteligente. O ETH pode ser destruído caso o validador se comporte de maneira desonesta ou preguiçosa. Dessa forma, o validador será responsável por validar os novos blocos adicionados a rede, recebendo ETH como recompensa (ETHEREUM, 2022a);

- *Delegated Proof of Stake (DPoS)*: Trata-se de uma inovação em relação ao *Proof of Stake* padrão, em que cada nó pode delegar a validação de uma transação a outros nós por meio de votação (BASHIR, 2018).

O uso do algoritmo de consenso está relacionado ao tipo de *blockchain* utilizada (BASHIR, 2018). Por exemplo, em *blockchains* públicas não-permissionada faz mais sentido utilizar PoW ou PoS como mecanismo de consenso ao invés de um mecanismo de prova de autoridade.

2.4.6 Plataforma *blockchain*

Nos últimos anos, várias plataformas *blockchain* foram desenvolvidas similares à plataforma Bitcoin, mas com funcionalidades adicionais. A plataforma *blockchain* é o núcleo da rede blockchain e fornece serviços-chave para os nós participantes. Uma plataforma *blockchain* típica (COUNCIL, 2017) deve ter os seguintes módulos:

- Ambiente de execução *blockchain*: Para ser capaz de processar transações e contratos inteligentes, a *blockchain* precisa de um ambiente de hospedagem seguro. Geralmente, o sistema operacional seguro, a linguagem de programação, as bibliotecas de tempo de execução e as bibliotecas de suporte residem nesta camada;
- Serviços criptográficos: Essa camada fornece acesso a algoritmos criptográficos, como função de *hash* e assinatura digital;
- Módulo de contrato inteligente: O módulo de contrato inteligente é opcional e se aplica exclusivamente ao *blockchain Stateful*. Esse módulo engloba as lógicas de negócios que podem ser implementadas utilizando qualquer linguagem de programação, como Go, Solidity, Java, Rust, C++, etc;
- Armazenamento secundário da *blockchain*: A plataforma *blockchain* processa grandes quantidades de transações e precisa de uma solução de armazenamento altamente segura, confiável e escalável para armazenar dados de bloco no livro-razão distribuído da *blockchain*. Essa camada fornece capacidades de armazenamento permanente para a plataforma. Geralmente, o LevelDB, RocksDB, H2 Database, MongoDB são usados como armazenamento, com outras soluções de armazenamento de dados distribuídos para armazenar informações do livro-razão;
- Armazenamento em memória da *blockchain*: Essa camada armazena as transações mais recentes na memória para suportar a recuperação de dados mais rápida e acelerar a execução de transações. A árvore Merkle, *Trie*, o gráfico dirigido acíclico, a matriz associativa, etc. são algumas das estruturas de dados usadas como armazenamento em memória em várias plataformas *blockchain*;

- Módulo de protocolo de consenso: Este módulo contém mecanismos para alcançar acordo entre os nós sobre a validade e autenticidade das transações. Uma vez que a maioria dos nós concorda e o nível de consenso é alcançado, a transação dada é tratada como válida e registrada no livro-razão distribuído. *Proof of Service*, *Proof of Stake*, *Proof of Importance (PoI)*, Raft, *Byzantine Fault Tolerance (BFT)*, etc. são alguns dos famosos protocolos de consenso do *blockchain* em uso (SANKAR; SINDHU; SETHUMADHAVAN, 2017);

- Camada de serviços *blockchain*: Usando essa camada, as plataformas *blockchain* podem ser reforçadas com capacidades extras, como gerenciamento de membros, gerenciamento de autorização e acesso, serviços de distribuição e notificação de eventos, exposição de serviços da plataforma usando a Interface de Programação de Aplicativos (API), etc. No entanto, alguns dos serviços, como gerenciamento de membros, autorização e acesso, são somente significativos para *blockchain* permissionada e opcionais para *blockchain* não permissionada;

- Protocolo de comunicação: "O protocolo *blockchain* implementa regras padrão projetadas para facilitar a comunicação distribuída *peer-to-peer* entre os nós participantes na rede *blockchain*. O Bitcoin usa transmissão via conexão TCP, enquanto o Ethereum usa o protocolo *devp2p* (ETHEREUM. . . , 2023). Em geral, os usuários enviam transações para um nó na rede *blockchain*. O nó agrupa um conjunto de transações em um bloco e, em seguida, transmite esse bloco para todos os nós para processamento. Os nós calculam as funções de *hash* criptográficas, processam as transações da *blockchain* e transmitem o resultado de sucesso para todos os nós na rede *blockchain*. Com base no consenso e acordo, o bloco é adicionado ao livro-razão distribuído e a transação é concluída com sucesso.

2.4.7 Beneficios da *blockchain*

- Segurança aprimorada: A *blockchain* possibilita criar registros que não podem ser alterados que são criptografados do início ao fim. A *blockchain* também permite anonimidade de dados pessoais e impedimento de acesso, criando assim mecanismos de privacidade. As informações são armazenadas em uma rede de computadores em vez de em um único servidor, tornando difícil para os *hackers* visualizarem os dados (IBM, c2022a);

- Transparência: Como a *blockchain* usa um livro-razão distribuído, as transações e os dados são registrados de forma idêntica em vários locais. Todos os participantes da rede com acesso autorizado visualizam as mesmas informações ao mesmo tempo, proporcionando total transparência (IBM, c2022a);

- Eficiência e velocidade: Os processos tradicionais em papéis são demorados, propensos a erros humanos e geralmente exigem mediação de terceiros. A documentação pode ser armazenada na *blockchain*, juntamente com os detalhes da transação, eliminando a necessidade de troca de papel (IBM, c2022a);

- Rastreamento: A tecnologia *blockchain* cria uma trilha de auditoria que documenta a procedência de um ativo em cada etapa de sua jornada (IBM, c2022a). Como cada uma das transações na *blockchain* é validada e registrada com uma data e hora, os usuários podem facilmente verificar e rastrear os registros anteriores acessando qualquer nó da rede distribuída (WANG et al., 2018).

2.5 Contratos inteligentes

O termo contrato inteligente tem sido usado ao longo dos anos para descrever uma ampla variedade de coisas diferentes. O criptógrafo Szabo (1994) definiu o termo como "um conjunto de promessas, especificado em forma digital, incluindo protocolos dentro dos quais as partes cumprem as outras promessas". Desde então, o conceito de contratos inteligentes evoluiu, especialmente após a introdução de plataformas blockchain descentralizadas com a invenção do Bitcoin em 2009.

Contratos inteligentes são aplicativos que funcionam exatamente como planejado sem qualquer possibilidade de censura, fraude ou interrupção (BUTERIN, 2014). São usados como acordos legalmente vinculativos entre partes, assim como os contratos escritos convencionais, mas oferecem vantagens adicionais, como a automatização de transações e a possibilidade de chegar a acordos direta e automaticamente, sem a necessidade de um intermediário. Os contratos inteligentes são executados automaticamente quando uma condição é cumprida, sem necessidade de processamento de papelada ou reconciliação de erros que podem ocorrer com o preenchimento manual de documentos (IBM, c2022b). Com isso é possível aplicar automação de processos de um sistema de votação com registro de votos, garantia de unicidade, elegibilidade do eleitor e contagem dos votos. A Figura 3 mostra um exemplo de contrato inteligente para registro de votos escrito na linguagem de programação Solidity.

Figura 3 – Exemplo de contrato inteligente para registrar voto

```
function vote(uint8 toProposal) public {
    Voter storage sender = voters[msg.sender];
    if (sender.isVoted || toProposal
    >= proposals.length && !sender.hasRightToVote)
        return;
    sender.isVoted = true;
    sender.vote = toProposal;
    proposals[toProposal].voteCount += 1;
}
```

Fonte: (KOç et al., 2018)

2.6 Ethereum

Ethereum é uma tecnologia para criar aplicativos e organizações, manter ativos, realizar transações e se comunicar sem ser controlado por uma autoridade central, por meio de contratos inteligentes (ETHEREUM, 2022a). Embora também documente e garanta a segurança das transações, a Ethereum é muito mais flexível que a *blockchain* Bitcoin (COINBASE, c2022). A rede Ethereum oferece uma gama mais ampla de usos com contratos inteligentes, que podem desencadear transações com os critérios incluídos no contrato (TAŞ; TANRıöVER, 2020).

Ao contrário do Bitcoin, que possui uma linguagem de *script* muito limitada, o Ethereum é projetado para ser uma cadeia de blocos programável que executa uma máquina virtual capaz de executar códigos de complexidade arbitrária e ilimitada. Enquanto a linguagem de programação do Bitcoin é, intencionalmente, restrita a avaliações verdadeiras/falsas simples das condições de gastos, a linguagem do Ethereum pode funcionar de maneira direta como um computador de propósito geral (ANTONOPOULOS; Gavin Wood, 2018).

O Ethereum, por ser de código aberto e utilizar contratos inteligentes, permite a utilização por uma ampla gama de desenvolvedores (TAŞ; TANRıöVER, 2020). Os contratos inteligentes no Ethereum podem ser programados usando linguagens relativamente amigáveis para o desenvolvedor (ETHEREUM, 2022a). A linguagem mais popular e usada com mais frequência para contratos inteligentes Ethereum é a Solidity, uma linguagem de programação procedural (imperativa) com sintaxe semelhante a JavaScript, C++ e Java (ANTONOPOULOS; Gavin Wood, 2018). A *blockchain* Ethereum está em constante evolução. Uma das mudanças mais recentes foi do mecanismo de consenso que mudou para *proof-of-stake* (PoS) em 2022 por ser mais seguro, consumir menos energia e ser melhor

para implementar novas soluções de escalabilidade em comparação com a antiga arquitetura de *proof-of-work* (PoW) (ETHEREUM, 2022a).

2.6.1 Terminologia

- Nó: Um "nó" é um software cliente Ethereum que se conecta a outros computadores na rede, formando uma rede descentralizada. Os nós desempenham um papel essencial na transmissão de informações, processamento de transações e validação de dados. Os clientes Ethereum são implementações do protocolo que verificam e mantêm a segurança da rede, garantindo que as transações sigam as regras estabelecidas e evitando fraudes e ataques maliciosos. (ETHEREUM, 2022a);
- *Gas*: Unidade que mede a quantidade de esforço computacional necessária para executar operações específicas na rede Ethereum (ETHEREUM, 2022a);
- EVM: A Máquina Virtual Ethereum (EVM) é o computador virtual global cujo estado todos os participantes da rede Ethereum armazenam e concordam. A EVM define as regras para calcular um novo estado válido de bloco em bloco na Ethereum (ETHEREUM, 2022a);
- *Ether* (ETH): É a criptomoeda nativa do Ethereum. Os usuários pagam ETH a outros usuários para que suas solicitações de execução de código sejam atendidas (ETHEREUM, 2022a);
- Conta: Local onde ETH são armazenados. O ETH é armazenado em contas que podem ser inicializadas, depositadas e transferidas por usuários. As contas e os saldos de conta são armazenados na EVM como parte do estado geral da mesma (ETHEREUM, 2022a);
- Carteiras: São aplicativos que permitem que você interaja com sua conta Ethereum, lendo o saldo, enviando transações e se conectando a aplicativos;
- Transações: São instruções criptograficamente assinadas de contas usadas para atualizar o estado da rede Ethereum. Uma transação Ethereum é uma ação iniciada por uma conta externamente controlada e pode incluir a transferência de ETH de uma conta para outra (ETHEREUM, 2022a).

2.6.2 Tipos de Nós

Os clientes podem executar três tipos diferentes de nós: *full node*, *light node* e *archive node*. Além disso, existem diferentes estratégias de sincronização disponíveis, permitindo uma sincronização mais rápida. A sincronização é o processo de obter as informações mais recentes sobre o estado atual do Ethereum de forma ágil e eficiente. Dessa

forma, os clientes têm flexibilidade para selecionar a opção que melhor atende às suas necessidades, garantindo acesso rápido e atualizado às informações na rede Ethereum.

Um *full node* desempenha papel crucial na rede Ethereum, por ser responsável por armazenar todos os dados da *blockchain*. Embora periodicamente reduza o armazenamento de dados antigos, o nó mantém informações essenciais para validar os blocos e verificar todos os blocos e estados. Através do *full node*, é possível derivar todos os estados presentes na rede, embora estados muito antigos precisem ser reconstruídos mediante solicitações feitas aos *archive nodes*. Além de seu papel na validação, o *full node* também serve à rede, fornecendo dados quando solicitados, contribuindo para o funcionamento eficiente e seguro da rede Ethereum (ETHEREUM, 2022a).

Já os *light nodes* ao invés de fazer baixar completamente cada bloco, o *light node* apenas obtém os cabeçalhos dos blocos. Esses cabeçalhos contêm apenas informações resumidas sobre o conteúdo dos blocos. Qualquer outra informação necessária pelo *light node* é solicitada a um *full node*. O *light node*, então, verifica independentemente os dados recebidos em relação às raízes de estado nos cabeçalhos dos blocos. Os *light nodes* permitem que os usuários participem da rede Ethereum sem a necessidade de ter hardware poderoso ou uma conexão de internet de alta velocidade, como é exigido para executar *full nodes*. Os *light nodes* não participam do consenso (ou seja, não podem ser mineradores/validadores), mas têm acesso à *blockchain* Ethereum com a mesma funcionalidade e garantias de segurança de um *full node* (ETHEREUM, 2022a).

O *archive node* é responsável por armazenar todas as informações mantidas pelo *full node* e criar um registro de estados históricos. Esse tipo de nó é essencial caso seja necessário realizar consultas. Dessa forma é possível testar as próprias transações sem a necessidade de minerá-las através do rastreamento. Ao sincronizar os clientes em qualquer modo que não seja o de arquivo, os dados da *blockchain* são reduzidos. Isso significa não haver um arquivo contendo todos os estados históricos, mas o *full node* consegue construí-los sob demanda (ETHEREUM, 2022a).

2.6.3 Transações

As transações no Ethereum consistem em mensagens de dados criptograficamente assinadas que incluem um conjunto de instruções. Essas instruções podem ser entendidas como a transferência de Ether de uma conta Ethereum para outra ou a interação com um contrato inteligente implantado na *blockchain*. As transações, que modificam o estado do EVM, devem ser divulgadas para toda a rede. Qualquer nó pode transmitir uma solicitação para executar uma transação no EVM; uma vez feito isso, um validador executará a transação e disseminará a alteração resultante do estado para o restante da rede (ETHEREUM, 2023).

No contexto do Ethereum, podem ser identificados vários tipos de transações distintos, cada um com uma finalidade específica e características próprias. As transações regulares consistem na transferência de valor monetário de uma conta Ethereum para outra. Essas transações são utilizadas para enviar a criptomoeda Ether (ETH) entre contas (ETHEREUM, 2023).

Outro tipo relevante é a transação de implantação de contrato. Essas transações são empregadas para criar e disponibilizar contratos inteligentes na rede Ethereum. Nesse caso, o campo de dados da transação é usado para armazenar o código-fonte do contrato que será implantado. Assim, quando essa transação é executada, o contrato inteligente é registrado na *blockchain* do Ethereum, tornando-se acessível para futuras interações e execuções (ETHEREUM, 2023).

Por fim, destacam-se as transações de execução de contrato, com a finalidade de interagir com contratos inteligentes já implantados na rede Ethereum. Nesse tipo de transação, o endereço de destino da transação identifica o contrato inteligente alvo da interação. Através dessas transações, é possível executar as instruções e operações específicas contidas nos contratos inteligentes, permitindo uma ampla gama de funcionalidades e possibilidades de automação (ETHEREUM, 2023).

Essa diversidade de transações e a capacidade de interagir com contratos inteligentes conferem ao Ethereum uma grande flexibilidade e potencial para o desenvolvimento de aplicativos descentralizados, sistemas autônomos e soluções baseadas em blockchain.

2.6.3.1 Ciclo de vida da transação

Uma vez que a transação tenha sido submetida, uma série de processos são desencadeados para garantir a segurança e a integridade da transação. Primeiramente, um código *hash* criptograficamente gerado é atribuído à transação. Esse *hash* serve como uma identificação única e exclusiva da transação, fornecendo uma camada adicional de segurança (ETHEREUM, 2023).

Em seguida, a transação é enviada para a rede e adicionada a um *pool* de transações pendentes. A *pool* consiste em todas as outras transações que continuam aguardando validação. Uma vez no *pool*, a transação fica disponível para os validadores. A tarefa de um validador é selecionar transações do *pool* e incluí-las em um bloco. Esse processo envolve verificar a autenticidade e a validade de cada transação. Ao selecionar sua transação, o validador a inclui no bloco, dando-lhe a chance de ser processada e considerada como "bem-sucedida". Conforme o tempo passa, o bloco contendo sua transação passa por duas etapas de atualização: "justificado" e "finalizado". Essas atualizações são essenciais para fortalecer a confiabilidade da transação. Quando um bloco é "justificado", significa que o mesmo foi validado e considerado aceitável pela maioria dos validadores da rede. Já

quando um bloco é "finalizado", indica que foi completamente validado e que a transação nele contida é considerada permanentemente confirmada e imutável (ETHEREUM, 2023).

Esses mecanismos de atualização tornam altamente improvável qualquer modificação posterior na transação. De fato, a alteração de um bloco "finalizado" exigiria um ataque de nível de rede extremamente poderoso e custoso, envolvendo bilhões de dólares (ETHEREUM, 2023). Portanto, o processo de geração de *hash*, transmissão, validação, atualização e finalização do bloco assegura que sua transação seja protegida contra adulteração, proporcionando uma base sólida para a confiabilidade e a segurança no contexto da rede.

2.6.4 Redes

As redes são diferentes ambientes Ethereum que podem ser acessadas para casos de uso de desenvolvimento, teste ou produção. Uma conta Ethereum funcionará em diferentes redes, mas o saldo e histórico de transações não serão transferidos da rede principal do Ethereum (ETHEREUM, 2022b).

A Mainnet é a principal *blockchain* pública de produção da Ethereum, em que transações reais são registradas e as aplicações descentralizadas são executadas (ETHEREUM, 2022b).

Além da Mainnet, existem *testnets* públicos. Estas são redes usadas por desenvolvedores para testar atualizações de protocolos e potenciais contratos inteligentes em um ambiente semelhante ao de produção antes de serem implantados na Mainnet (ETHEREUM, 2022b).

A maioria das *testnets* começaram usando um mecanismo de consenso de prova com permissão. Isso significa que apenas um pequeno número de nós é escolhido para validar as transações e criar novos blocos, estabelecendo sua identidade no processo. Algumas *testnets* possuem um mecanismo de prova de participação aberto, em que qualquer pessoa pode testar a execução de um validador, assim como na Mainnet Ethereum (ETHEREUM, 2022b).

2.6.5 Aplicativos descentralizados

Além de contratos inteligentes, a Ethereum também é utilizada para criar DApps, ou aplicativos descentralizados. Esses aplicativos podem ser construídos na plataforma Ethereum e utilizados para diversos fins, incluindo finanças, saúde, mídia e jogos. Os aplicativos usam contratos inteligentes para se conectar a uma rede descentralizada e oferecer serviços sem um único ponto de falha (ETHEREUM, 2022a).

Segundo Johnston (2022) e Raval (2016), uma aplicação deve cumprir os seguintes requisitos para ser considerada uma DApp:

- *Open Source*: A aplicação deve ser totalmente de código aberto e operar de forma independente, sem que nenhuma entidade controle a maioria de seus *tokens*. A mesma pode ajustar seu protocolo em resposta as sugestões de melhoria e *feedback* do mercado, mas todas as mudanças devem ser decididas pelo consenso de seus usuários.

- Sem ponto de falha central: Os dados e registros de operação da aplicação devem ser armazenados em uma *blockchain* pública e descentralizada usando criptografia, para evitar pontos de falha centrais.

- Suporte à criptomoeda interna: A aplicação deve usar um *token* criptográfico (como o Bitcoin ou um *token* específico de seu sistema) para o acesso e os usuários que contribuem com valor devem ser recompensados com os *tokens* da aplicação.

- Consenso descentralizado: O consenso entre nós descentralizados é a base da transparência.

- A aplicação deve criar *tokens* usando um algoritmo criptográfico padrão como prova do valor contribuído pelos nós para a aplicação (por exemplo, o Bitcoin usa o algoritmo *Proof of Work*).

2.6.6 *Minimum Anti-Collusion Infrastructure* (MACI)

O MACI é um protocolo criptográfico projetado para garantir a privacidade e a integridade dos votos, ao mesmo tempo, em que permite a verificabilidade dos resultados (JIE, 2021). Utiliza técnicas de criptografia baseadas em provas de conhecimento zero e *accumulators* (acumuladores) para proteger a identidade dos eleitores e o conteúdo de seus votos. O protocolo permite que os votos sejam criptografados de forma anônima, garantindo que nem mesmo o sistema de votação ou terceiros tenham conhecimento da escolha individual de cada eleitor. O MACI foi originalmente proposto por Vbuterin et al. (2019) em um post na plataforma "Ethresear.ch". Suas características principais incluem:

- Resistência à colusão: O MACI assegura que apenas um coordenador confiável possua o conhecimento da validade de um voto. Isso reduz significativamente a eficácia da corrupção, uma vez que nenhum outro participante, exceto o coordenador, consegue obter tal certeza.

- Ausência de comprovantes: O sistema garante que nenhum eleitor consiga comprovar, a não ser para o coordenador, em qual direção votou. Isso preserva a confidencialidade do voto, evitando qualquer tentativa de coerção ou influência indevida.

- Privacidade: A descriptografia de um voto só pode ser realizada pelo coordenador confiável. Isso impede que terceiros tenham acesso indevido às informações dos votantes, garantindo assim a privacidade dos participantes.
- Impossibilidade de censura: Nem mesmo o coordenador confiável consegue censurar um voto. Isso assegura que todas as opiniões e preferências dos eleitores sejam devidamente consideradas, sem a interferência de qualquer entidade central.
- Impossibilidade de falsificação: Apenas o proprietário da chave privada de um usuário é capaz de emitir um voto vinculado à sua chave pública correspondente. Isso garante a autenticidade dos votos, evitando a possibilidade de falsificação por parte de terceiros.
- Não repúdio: Uma vez emitido, nenhum voto pode ser modificado ou excluído. No entanto, um usuário tem a opção de emitir outro voto para anular o anterior. Essa característica garante a integridade do processo de votação e evita a manipulação dos resultados.
- Execução correta: O MACI assegura que nem mesmo o coordenador confiável possa produzir uma contagem falsa de votos. Isso garante a precisão e a confiabilidade dos resultados, promovendo a transparência e a confiança no sistema.

O MACI é baseado em contratos inteligentes Ethereum e usa provas de conhecimento zero (zk-SNARK). Essa combinação de tecnologias herda a segurança e a impossibilidade de censura oferecidas pela *blockchain* Ethereum subjacente. Além disso, o uso de criptografia assimétrica garante a impossibilidade de falsificação. Dessa forma, o MACI alcança resistência à colusão, privacidade e execução correta, fornecendo um ambiente confiável para a realização de votações digitais (JIE, 2021).

3 Metodologia

3.1 Metodologia de pesquisa bibliográfica

Uma pesquisa bibliográfica geralmente contém as seguintes etapas: a escolha e delimitação do tema; a coleta de dados; a localização das informações; anotações e fichamentos (ANDRADE, 2022). Após a definição do tema, Aplicação descentralizada de votação aberta usando blockchain Ethereum, realizou-se a busca de materiais relacionados nas bases de conhecimento IEEE e Scopus, através da pesquisa foi possível obter alguns artigos, conforme demonstrado na Tabela 2.

Tabela 2 – Strings de busca

Base	**String de Busca**	**Nº de artigos obtidos**
IEEE	"ethereum"AND "voting"	86
IEEE	"blockchain"and "ethereum"AND "e-voting"	105
IEEE	"blockchain"AND "voting"	582
IEEE	"e-voting"AND "ethereum"	44
Scopus	"e-voting"	1.898
Scopus	"ethereum"AND "blockchain"	4.952
Scopus	"ethereum"AND "e-voting"	87

Fonte: Autores

Após a triagem de alguns artigos resultantes das pesquisas nas bases de dados, iniciou-se o processo de coleta de dados e a localização das informações. Dessa forma, foi realizada a leitura prévia a fim de determinar as obras que serão examinadas mais detidamente. A próxima etapa foi a leitura seletiva para identificar as informações úteis para elaboração deste trabalho. Nessa etapa, os artigos selecionados foram aqueles que falavam sobre: votação eletrônica e destacavam a diferença entre votações tradicionais e eletrônicas; *Ethereum* e suas aplicações em um sistema de votação, *blockchain* e suas aplicações; votação eletrônica usando a *blockchain Ethereum*. Por fim, foram realizadas a leitura crítica e interpretativa para um entendimento mais profundo dos artigos. Para compor o trabalho também foram utilizados *sites* da internet e as documentações oficiais das tecnologias como da *Ethereum*, que foram utilizadas para o desenvolvimento tecnológico.

3.2 Metodologia de desenvolvimento

Scrum é um *framework* para desenvolver e sustentar produtos, geralmente complexos (SCHWABER, 2004). O Scrum define papéis com responsabilidades definidas, como

o Scrum Master, o Dono do Produto e o time. Também define artefatos como o *backlog* do produto e o *sprint backlog*. No contexto de desenvolvimento de software, o *backlog* do produto é uma lista de tarefas ou funcionalidades que precisam ser realizadas para alcançar os objetivos. As *sprints* definem uma faixa de tempo para se realizar tarefas específicas do *backlog* do produto. A lista de tarefas definidas em uma *sprint* é chamada de *sprint backlog*.

O Kanban é uma abordagem ágil que visa a melhoria contínua, flexibilidade no gerenciamento de tarefas e fluxo de trabalho aprimorado (KISSFLOW, 2022). Ao contrário do Scrum, o Kanban não possui papéis definidos e não segue uma estrutura rígida de *sprints*. Isso proporciona uma maior flexibilidade no uso do método, permitindo que a equipe se adapte de forma mais fluida às mudanças no projeto. O Kanban utiliza um quadro que exibe as diferentes fases do processo de desenvolvimento.

Para o desenvolvimento tecnológico deste trabalho foi utilizado uma adaptação do Scrum e Kanban. O artefato usado do Kanban foi o quadro Kanban, demonstrado na Figura 4 que auxiliou a visualização do processo de desenvolvimento. Já o recurso do Scrum utilizado foi o *backlog* do produto, pela sua natureza dinâmica que progride conforme o desenvolvimento do produto (SCHWABER, 2004).

O quadro Kanban possui as tarefas a serem feitas, que por sua vez envolvem as histórias de usuário definidas no *backlog* do produto. Para a visualização do fluxo de trabalho foram definidas quatro etapas do progresso da execução das tarefas:

- A fazer: Tarefas que ainda não foram iniciadas. Essas tarefas estão prontas para serem selecionadas e movidas para a próxima etapa por um membro da equipe.
- Em progresso: Tarefas que estão sendo atualmente trabalhadas, mas ainda não estão concluídas.
- Em revisão: Tarefas concluídas, mas que devem ser verificadas para ver se estão conforme os critérios de aceitação. Qualquer feedback ou alterações necessárias podem ser identificados nesta etapa.
- Feito: Indica as tarefas que foram revisadas e aprovadas.

Figura 4 – Quadro Kanban utilizado para guiar o desenvolvimento da aplicação

Fonte: Autores.

3.3 *Backlog* do produto

O *backlog* do produto desempenhou um papel fundamental no desenvolvimento da aplicação, foi construído através da definição de épicos e histórias de usuários. Para organizá-lo, foram definidos três épicos principais. O primeiro épico, "Autenticação de Usuário", focou-se em garantir acesso à aplicação para eleitores e chefes de eleição. O segundo épico, "Apuração", abordou as necessidades relacionadas à contagem de votos, geração de relatórios e divulgação dos resultados eleitorais. Já o terceiro épico, "Gerenciamento de Eleições", concentrou-se na implementação de funcionalidades que permitissem a criação, organização e monitoramento de eleições. Os requisitos foram escritos em forma de histórias de usuário e registradas como issues no GitHub, conforme ilustrado na Figura 5. Cada história de usuário foi acompanhada por seus respectivos critérios de aceitação, que serviram como um guia claro para determinar quando a história poderia ser considerada concluída. O Apêndice A.1 mostra as histórias de usuário definidas no *backlog*.

Figura 5 – *Backlog* do produto

	Title	Assignees	Status	Labels	Milestone
1	US04 - Eu, como eleitor gostaria de visualizar o resultado da votação, para saber a opç... #4	EduardoPicolo	Feito	US	Épico 2
2	US05 - Eu, como usuário desejo ser capaz de visualizar as transações da votação na bl... #7	EduardoPicolo	Feito	US	Épico 2
3	US02 - Eu, como um eleitor, quero exercer meu direito de voto #1	luucas-melo	Feito	US	Épico 2
4	US03 - Eu, como oficial de eleição quero iniciar um processo de votação #3	luucas-melo	Feito	US	Épico 3
5	US01 - Eu, como usuário gostaria de me autenticar no sistema, para participar de uma... #2	luucas-melo	Feito	US	Épico 1

Fonte: Autores.

3.4 Cronograma

As Tabelas 3 e 4 exibem os cronogramas das atividades executadas nos TCC's 1 e 2.

3.4.1 TCC1

Tabela 3 – Cronograma do TCC1

Atividades	**Nov/2022**	**Dez/2022**	**Jan/2023**	**Fev/2023**
Definir tema	X			
Introdução(contexto)		X		
Introdução(problema e objetivos)			X	
Referencial Teórico	X	X	X	
Proposta			X	X
Apresenta à banca				X

Fonte: Autores.

3.4.2 TCC2

Tabela 4 – Cronograma do TCC2

Atividades	**Abr/2023**	**Maio/2023**	**Jun/2023**	**Jul/2023**
Aplicar melhorias	X			
Definir backlog	X			
Implementar requisito definido na história de usuário US01	X	X		
Implementar requisito definido na história de usuário US02		X	X	
Implementar requisito definido na história de usuário US03		X	X	
Implementar requisito definido na história de usuário US04			X	X
Implementar requisito definido na história de usuário US05		X	X	X
Implantar os contratos inteligentes na rede Sepolia				X
Fazer *deploy* do *frontend* da aplicação				X
Documentar resultados e conclusão				X
Apresentar à banca				X

Fonte: Autores.

4 Resultados e avaliação

4.1 Considerações Iniciais

Este capítulo elabora como os resultados são avaliados e o nível de sucesso da solução proposta. Utilizando os métodos propostos por Bokslag e Vries (2016) em sua publicação "*Evaluating e-voting: theory and practice*", serão abordados critérios como transparência, sigilo do voto, unicidade, elegibilidade do eleitor, verificabilidade, acessibilidade, resistência à coerção e disponibilidade. A utilização desses critérios de avaliação visa fornecer uma análise consistente e comparável da aplicação em questão, identificando seus pontos fortes e áreas de melhoria. Os resultados dessa avaliação serão fundamentais para a tomada de decisões informadas sobre a viabilidade e aprimoramento da aplicação, visando um processo de votação mais transparente, seguro e confiável. As fontes completas do projeto estão disponíveis no repositório do GitHub.

4.2 Solução desenvolvida

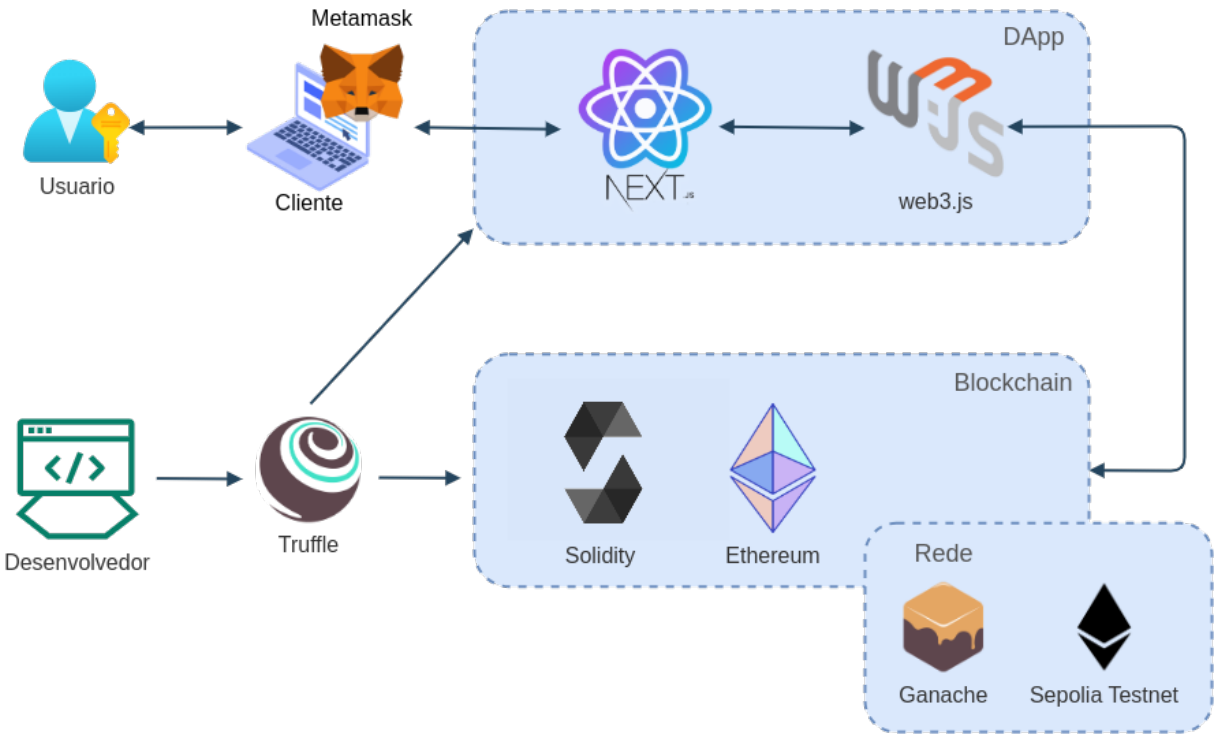
A aplicação desenvolvida demonstra o funcionamento de um sistema de votação aberta, utilizando a tecnologia blockchain Ethereum, considerando fatores como privacidade, elegibilidade, conveniência, comprovação de voto e verificabilidade. Para isso, o sistema é projetado como uma aplicação web. A aplicação possui uma interface que permite ao chefe da eleição gerenciar o processo, eleitores e candidatos. O sistema garante direitos iguais de votação para todos os eleitores e um processo justo. O eleitor pode verificar se o voto foi registrado por meio de uma API web. Tanto o chefe da eleição quanto o eleitor são identificados pelo endereço público da conta que estão usando na Metamask.

4.3 Visão arquitetural de alto nível

O DApp proposto consiste em duas partes principais, como mostra a Figura 6: o contrato, que roda na rede Ethereum, e o cliente. O usuário acessa o DApp por meio de um navegador web que possui a extensão Metamask instalada, conectando sua conta na rede *blockchain*. A rede possui contratos inteligentes escritos em Solidity, que são acessíveis através da comunicação com a biblioteca Web3.js, equipada com artefatos compilados usando o Truffle. Foi utilizado a ferramenta Ganache para implementar e testar o DApp em um ambiente local determinístico. Já a *testnet* Sepolia é utilizada para disponibilizar o sistema em um ambiente público que simula a Ethereum *mainnet*, tendo suas transações

acionadas e assinadas através da Metamask. O cliente Next.js utiliza a biblioteca Web3.js para interagir com os contratos inteligentes instalados na *blockchain* Ethereum.

Figura 6 – Arquitetura da aplicação desenvolvida



Fonte: Autores.

4.3.1 Tecnologias e ferramentas utilizadas

- Truffle: É um *framework* que cuida do gerenciamento de artefatos de um contrato inteligente. Inclui suporte para implantações personalizadas, vinculação de bibliotecas e aplicativos Ethereum complexos (TRUFFLE, c2022);
- Ganache: É uma ferramenta para configurar localmente sua própria blockchain da Ethereum. Ganache pode ser usado em todo o ciclo de desenvolvimento, permitindo que você desenvolva, implante e teste seus DApps em um ambiente seguro e determinístico (GANACHE, c2022);
- Remix IDE: É uma IDE para desenvolver contratos inteligentes (REMIXIDE, c2022);
- Web3.js: A biblioteca Web3.js é uma coleção de módulos que contém funcionalidade para o ecossistema Ethereum (WEB3.JS, c2016)
- Metamask: É uma carteira criptográfica e um gateway para aplicativos *blockchain*. Funciona como uma extensão de navegador que permite aos usuários interagir com a *blockchain* Ethereum e outras redes compatíveis. Com o Metamask, os usuários podem gerenciar suas chaves privadas, armazenar e enviar criptomoedas, e acessar

aplicativos descentralizados (DApps) diretamente de seu navegador. A carteira Metamask é auto-gerenciada, o que significa que os usuários têm controle total sobre suas chaves privadas e fundos. (METAMASK, c2023).

- Sepolia Testnet: A *testnet* Sepolia foi utilizada como a rede de teste da aplicação. Essa rede utiliza um conjunto de validadores autorizados, fornecendo um ambiente seguro para testes e experimentação antes da implantação em redes principais (ETHEREUM, 2022b). A utilização da Sepolia permitiu testar e validar a funcionalidade da aplicação em um ambiente controlado, sem a necessidade de ser implantada em uma rede principal Ethereum.

- Next.js: É um *framework* web de código aberto que permite a criação de aplicativos web baseados em *React* com renderização do lado do servidor e geração de *sites* estáticos (VERCEL, c2023).

- Vercel: A plataforma Vercel foi utilizada para realizar o *deploy* do *frontend*. A mesma oferece uma infraestrutura de hospedagem escalável e confiável para aplicativos web, permitindo que o *frontend* seja implantado e acessível de forma rápida e segura.

- Git: O Git é um sistema de código aberto para versionamento de código.

4.3.2 Contratos inteligentes

A solução implementa dois contratos inteligentes: “ *Voting*.sol” e “ *VotingFactory*.sol”. O contrato “ *Voting*.sol” possibilita gerenciar uma votação mediante um sistema seguro e transparente, onde apenas os endereços autorizados podem votar. Este registra as propostas de votação, permite a alteração de informações relevantes, como título e prazo, e fornece mecanismos para controlar o processo de votação, garantindo a integridade e verificabilidade dos votos. O contrato fornece funcionalidades para adicionar e remover endereços da lista de permissões, obter a lista de propostas de votação, obter a lista de endereços permitidos, editar propostas, editar o título e o prazo da votação, realizar votos, iniciar e encerrar a votação, cancelar a votação e obter os resultados. O contrato “ *Voting*” foi testado por meio de testes unitários que abrangem diferentes aspectos de sua funcionalidade (ver Apêndice A.2.1).

Já o "*VotingFactory*"é responsável por criar e gerenciar contratos de votação. O contrato permite que novos contratos de votação sejam criados e mantém um registro dos contratos criados. Além disso, fornece uma função para recuperar os contratos de votação nos quais um determinado endereço está autorizado a votar.

4.3.2.1 Dados

O armazenamento em blockchain tem um custo significativo de *gas*, portanto, o design dos contratos requer considerações cuidadosas sobre o que será armazenado e quais tipos de dados serão utilizados.

No contrato *Voting*, a variável pública *title* define o título ou tema da votação. A estrutura *Proposal* representa uma opção de voto individual, contendo um nome e o número de votos acumulados para essa opção. As opções de voto disponíveis são armazenadas em um vetor de *Proposals*. Essa escolha é feita devido à possibilidade de iterar sobre as opções e calcular o resultado.

A variável pública *electionCommission* armazena o endereço público da carteira que implantou o contrato e é responsável por gerenciar a votação. A variável *whitelist* representa a lista de eleitores elegíveis para a votação. Utiliza-se o tipo *mapping*, em que a chave é o endereço público da carteira do eleitor e o valor é um booleano que indica se o eleitor já votou. Essa escolha de tipo de dados permite acesso eficiente em O(1) aos dados, garantindo o melhor desempenho das funções que utilizam essa variável. A declaração pública dessa variável não representa um problema, pois não há necessidade de ocultar quais carteiras já votaram, uma vez que essa informação está disponível no histórico de transações. Além disso, também são definidas variáveis para controle de duração da votação.

4.3.2.2 Funções

O contrato inteligente de votação possui diversas funcionalidades importantes para o seu funcionamento adequado. Uma dessas funcionalidades é o registro de eleitores. O contrato mantém uma relação de endereços de eleitores autorizados a participar da votação. Essa lista é gerenciada pela Comissão Eleitoral, que adiciona os endereços dos eleitores a mesma. Isso garante que apenas os eleitores registrados possam votar e impede que endereços não autorizados participem da votação.

Outra funcionalidade relevante é a criação de propostas. O contrato permite que a Comissão Eleitoral crie diferentes propostas que serão votadas pelos eleitores. Cada proposta é representada por uma estrutura que contém o nome da proposta e o número de votos que recebeu. Isso permite que os eleitores tenham opções claras para escolher durante a votação.

O contrato também possui funcionalidades relacionadas à configuração da votação. Ele possui um título que identifica a votação em questão. Além disso, a duração da votação é definida em segundos, permitindo que se estabeleça um período específico para a votação ocorrer. O contrato também registra o endereço da Comissão Eleitoral responsável pela

administração da votação, garantindo que apenas essa entidade autorizada possa realizar ações importantes relacionadas à votação.

Para garantir o controle de acesso adequado, o contrato utiliza modificadores. Esses modificadores estabelecem condições que devem ser atendidas para certas funções poderem ser executadas. Por exemplo, apenas a Comissão Eleitoral pode executar determinadas ações, como adicionar ou remover eleitores da *whitelist*. Além disso, certas ações só podem ocorrer antes ou durante o período de votação, como a criação ou edição de propostas. Isso ajuda a manter a integridade do processo de votação e evitar a ocorrência de ações indesejadas em momentos inadequados.

A realização da votação é um dos pontos centrais do contrato inteligente. A votação pode ser iniciada pela Comissão Eleitoral, após a qual os eleitores registrados na *whitelist* podem votar. O contrato garante que cada eleitor possa votar apenas uma vez, evitando assim votos duplicados. Os votos são contabilizados para cada proposta, permitindo que sejam analisados posteriormente.

O encerramento da votação é outra funcionalidade importante do contrato. A Comissão Eleitoral pode encerrar a votação quando considerar apropriado. Após o encerramento, nenhuma alteração adicional é permitida no processo de votação. Isso garante que os resultados sejam preservados e que nenhuma manipulação ocorra após o término da votação.

Por fim, o contrato oferece a possibilidade de consultar os resultados da votação. Após o encerramento da votação, é possível verificar o número de votos recebidos por cada proposta. Essa funcionalidade permite uma análise transparente dos resultados e fornece informações sobre a preferência dos eleitores.

No geral, o contrato inteligente de votação apresentado oferece uma série de funcionalidades essenciais para realizar uma votação segura, transparente e controlada na *blockchain*. O contrato garante o controle de acesso adequado, a integridade dos votos e a disponibilidade dos resultados, proporcionando um ambiente confiável para o processo.

4.3.3 A aplicação cliente

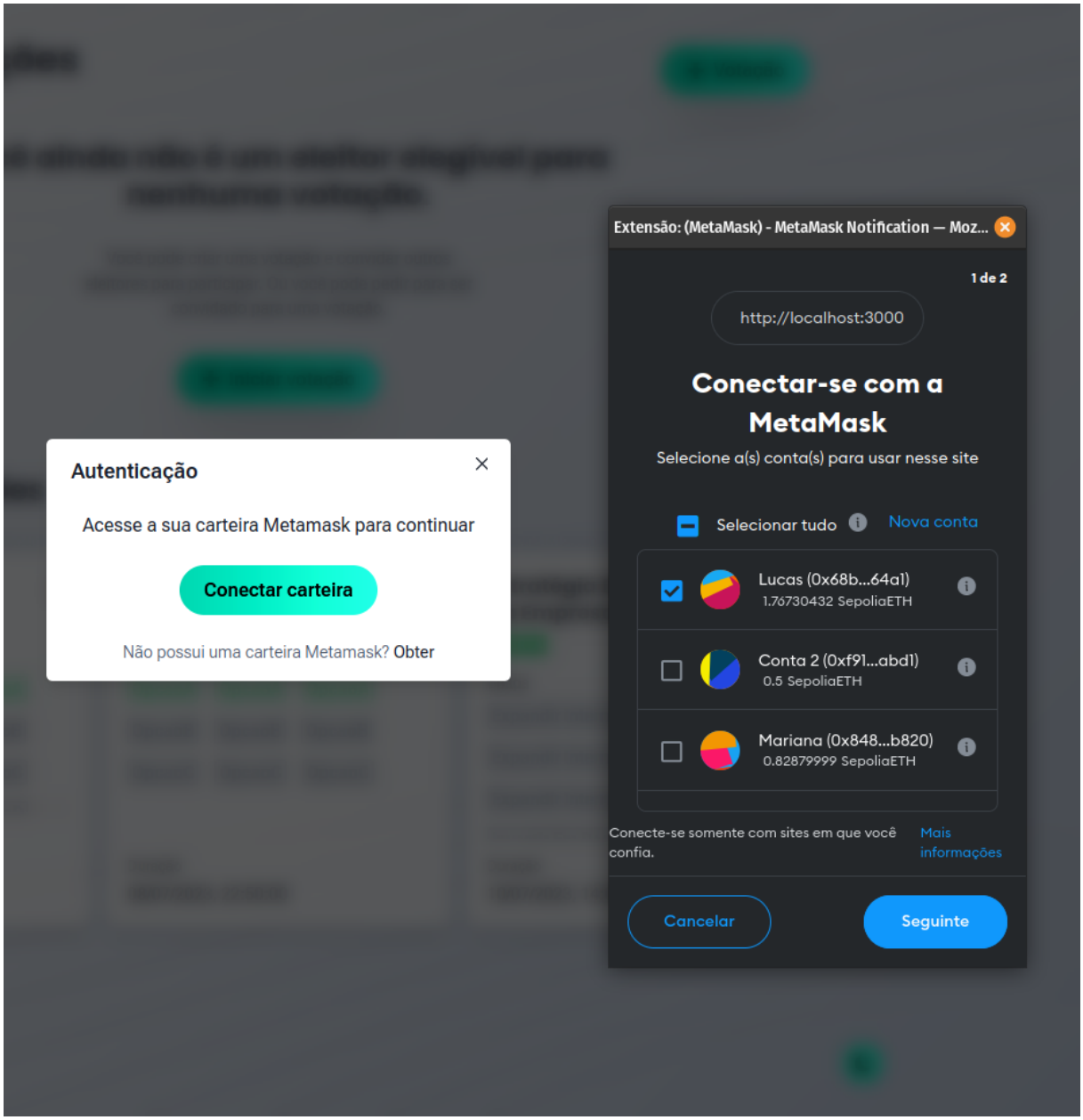
A aplicação cliente é uma aplicação web. Para acessar a API da rede Ethereum é utilizada a biblioteca Web3.js e uma conexão com um nó da rede Ethereum. A Metamask irá servir como uma ponte para conectar à rede Ethereum e injetar uma instância Web3 na aplicação. A extensão permite selecionar diferentes redes e contas Ethereum. A aplicação cliente é composta por recursos estáticos que podem ser executados localmente ou disponibilizados em um servidor web. Neste projeto, o código JavaScript foi implantado na Vercel e as fontes originais foram escritas usando TypeScript e React no *framework* Next.js.

4.3.4 Processo

O sistema pode ser dividido em 5 fases:

1. Autenticação: A autenticação é realizada através da extensão Metamask. Esta abordagem foi escolhida, por fornecer uma camada extra de segurança ao permitir o gerenciamento das chaves privadas e assinar transações sem expor informações pessoais diretamente ao aplicativo. Ao acessar o sistema é necessário autenticar o usuário através da extensão Metamask, conforme demonstrado na Figura 7;

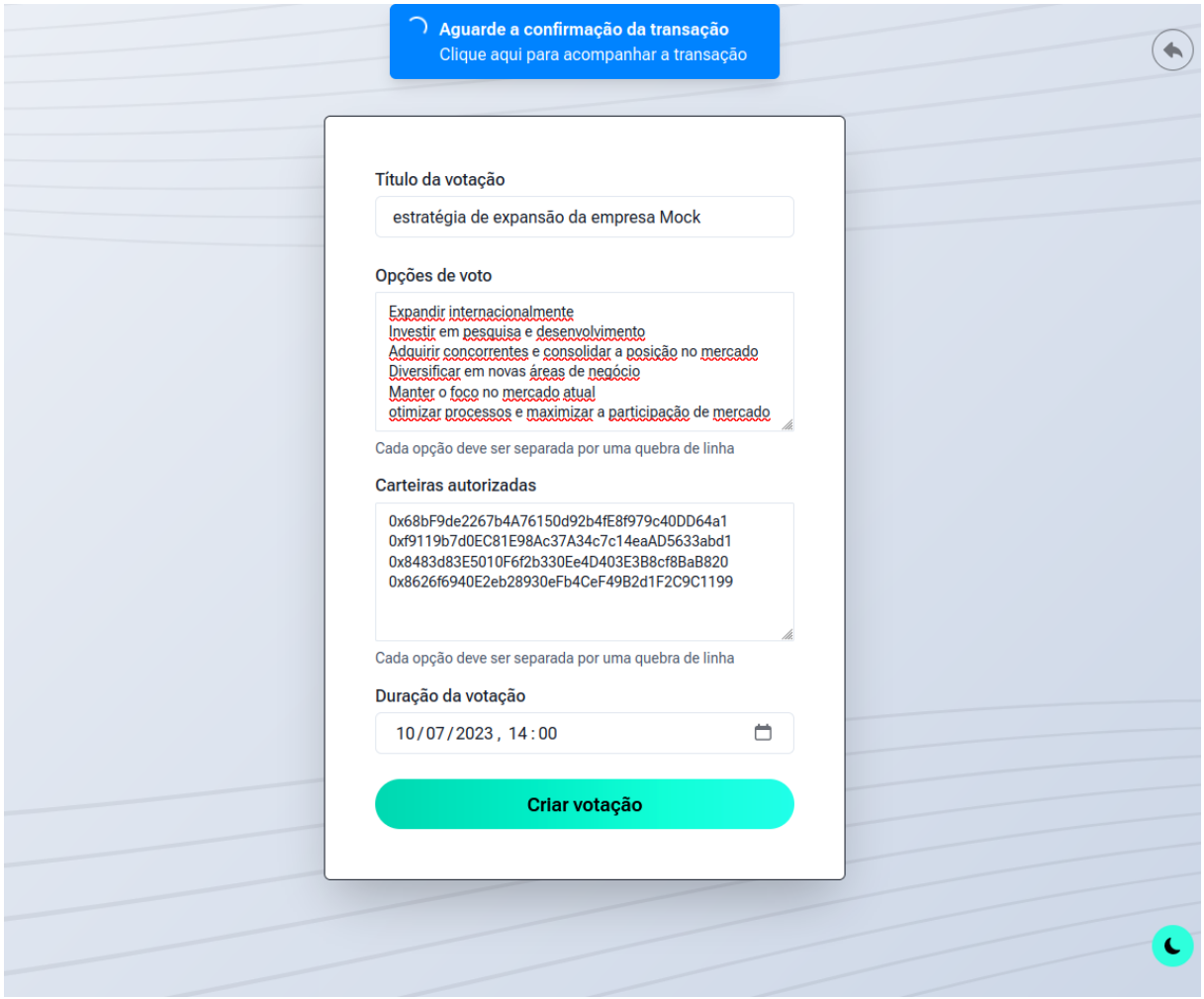
Figura 7 – Autenticação através da extensão Metamask



Fonte: Autores.

2. Preparação da votação: A autoridade eleitoral deve cadastrar o endereço das carteiras de eleitores elegíveis, fornecer um título significativo e a lista de opções. Dessa forma, a votação poderá ser iniciada pelo criador da votação através do menu de opções exibido na Figura 9. Mediante um formulário do *frontend*, representado na Figura 8, é possível criar uma votação;

Figura 8 – Formulário de criação de uma votação



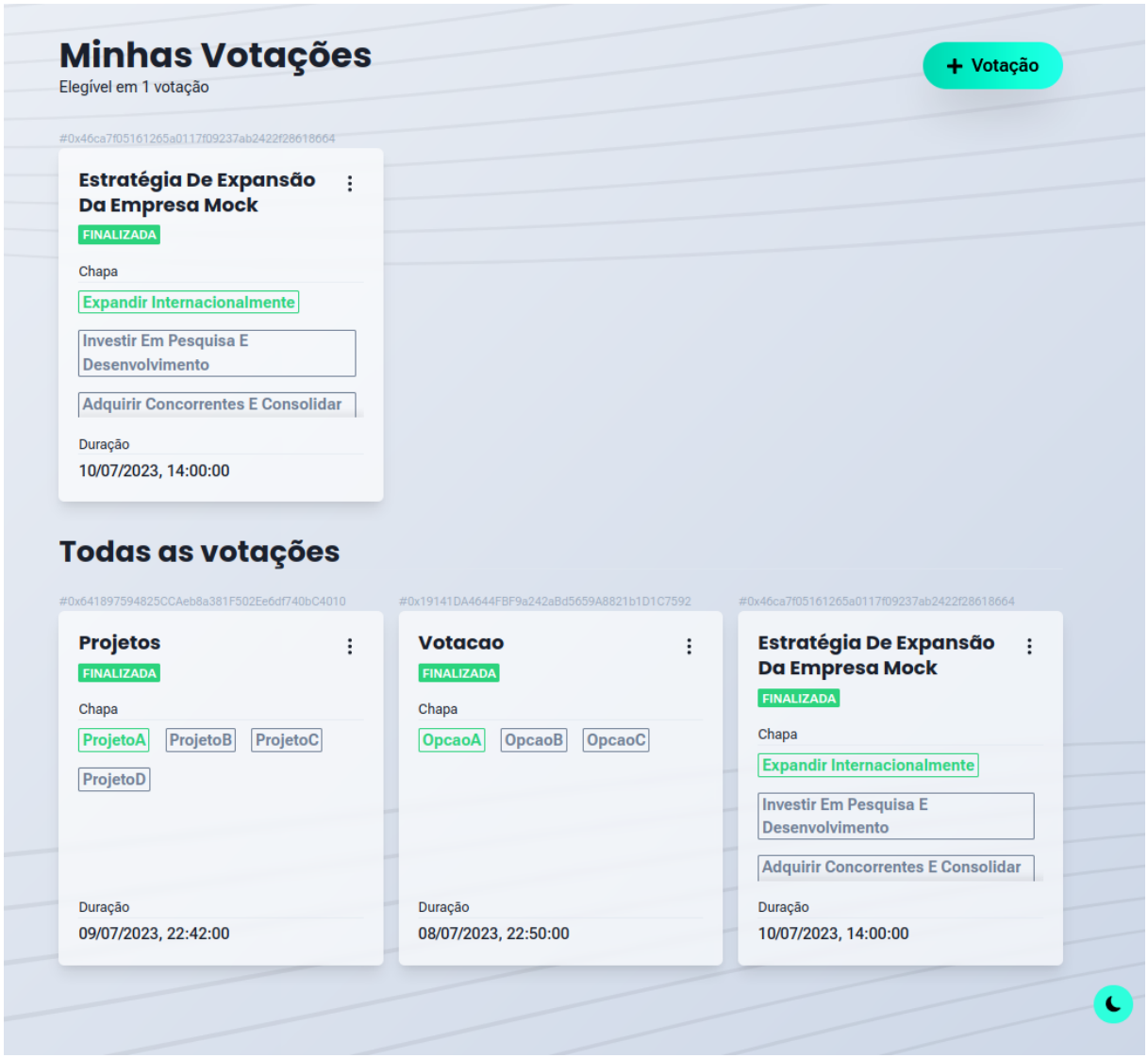
Fonte: Autores.

Figura 9 – Menu de opções de uma votação

Fonte: Autores.

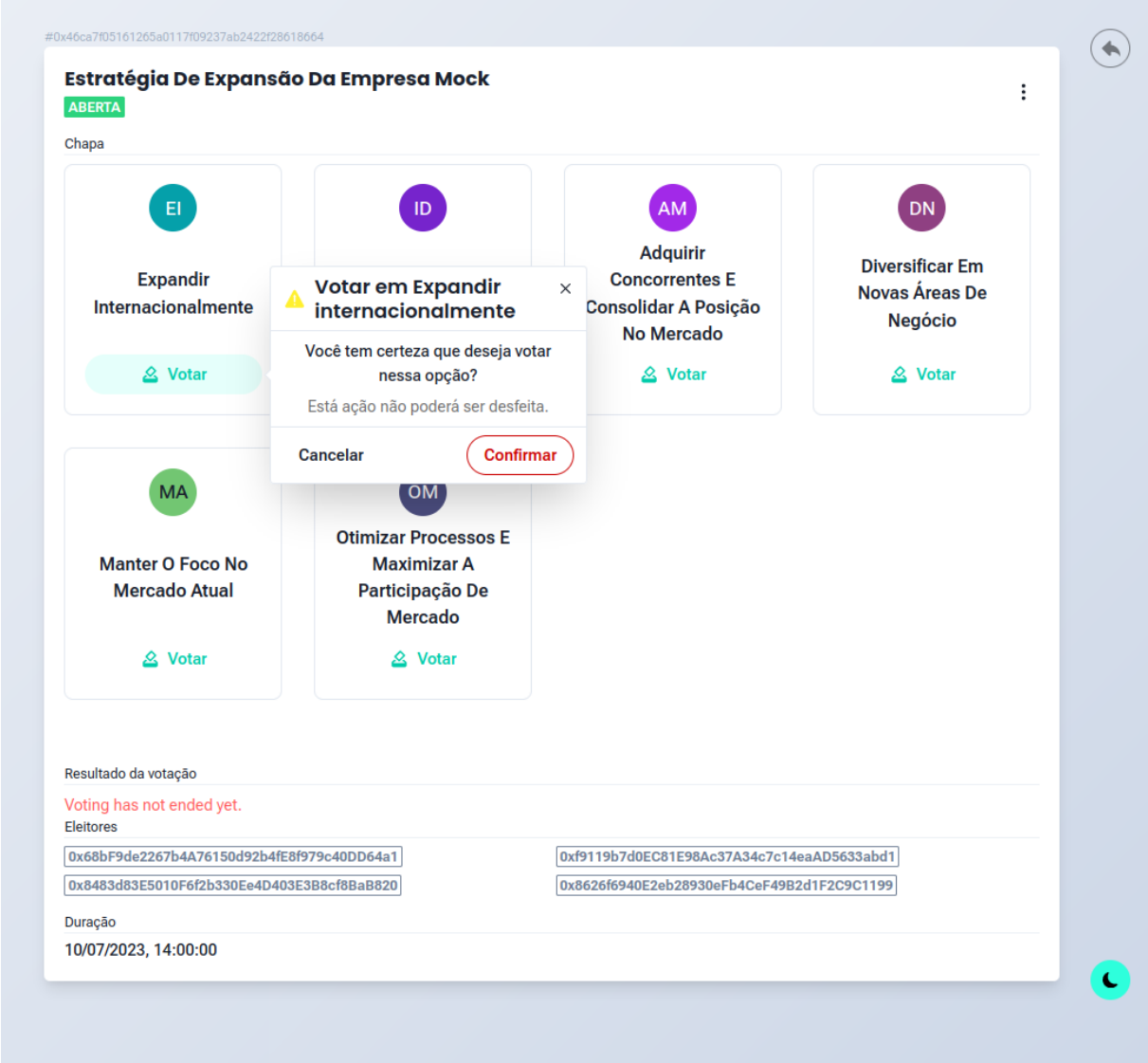
3. Votação: Após de realizar o *login* no portal da web utilizando um endereço Ethereum válido presente no Metamask, os eleitores podem acessar uma votação específica através da tela demonstrada na Figura 10 e submeterem seu voto nas votações que forem elegíveis através do botão “Votar”, conforme a Figura 11

Figura 10 – Lista todas as votações e votações que o usuário é elegível



Fonte: Autores.

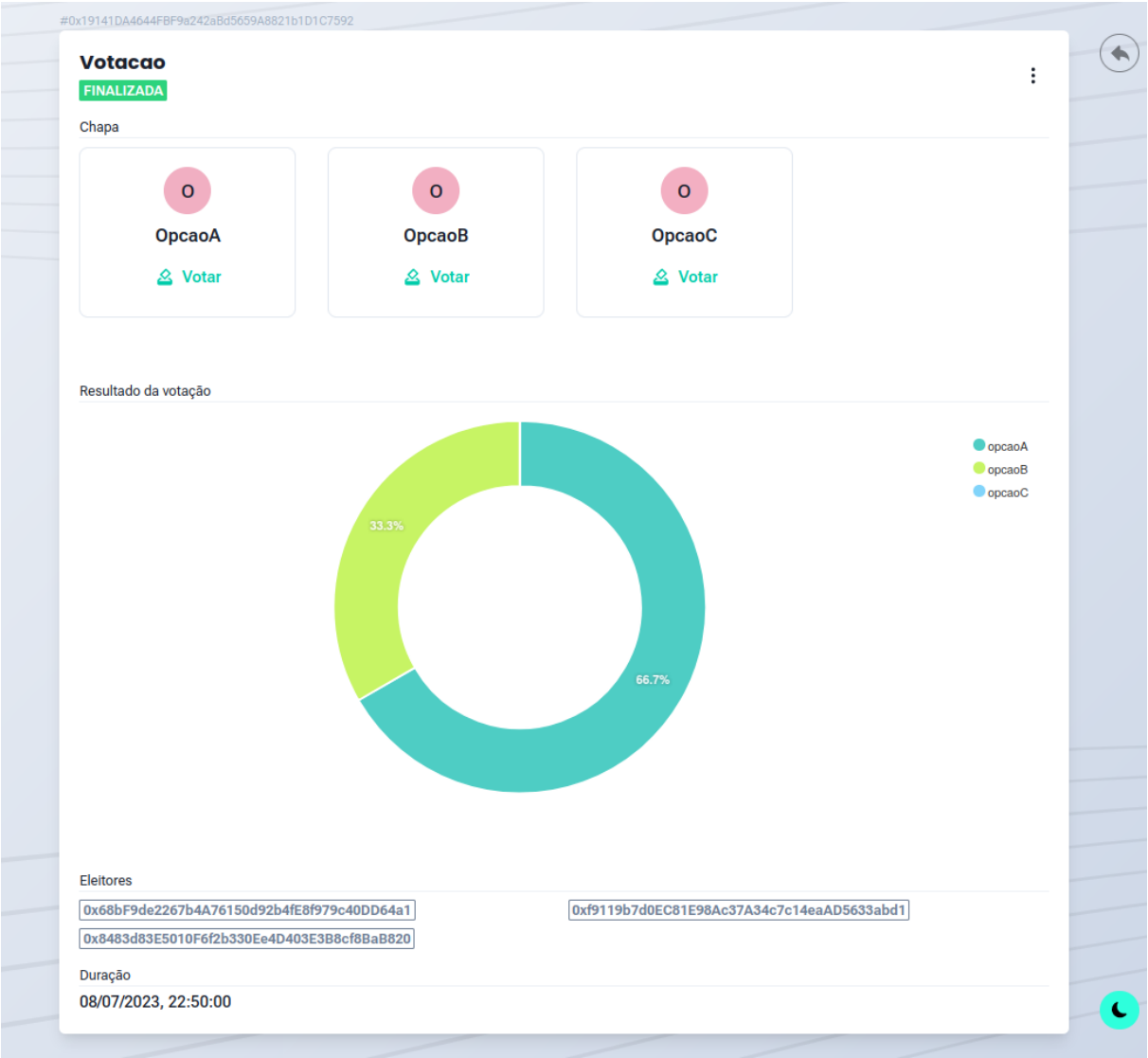
Figura 11 – Interface para realizar voto em uma das opções



Fonte: Autores.

4. Apuração: A apuração não é realizada manualmente, mas sim automaticamente por meio do contrato inteligente. Durante a votação, os resultados não são revelados aos eleitores até o encerramento da votação. Ao término do processo, os votos calculados para cada opção serão disponibilizados para visualização, conforme demonstrado na Figura 12.

Figura 12 – Resultado da votação

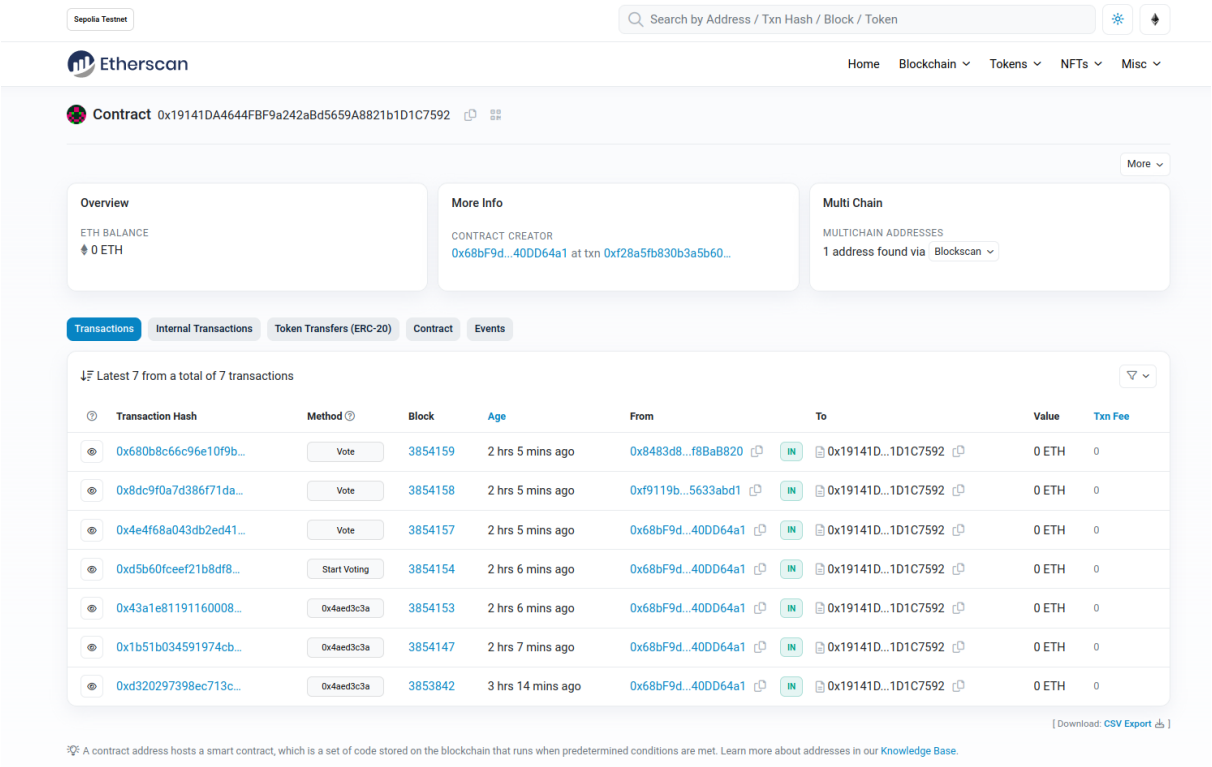


Fonte: Autores.

5. Verificação: Após submeterem seus votos, qualquer pessoa pode verificar se os votos foram registrados conforme a intenção. Para realizar essa verificação é necessário possuir o *hash* da transação recebido durante a fase de votação. Durante o processo de verificação, é confirmado se a transação foi incluída em um bloco ou não. As transações podem ser vistas pelo Etherscan [1], conforme ilustrado na Figura 13.

[1] Etherscan é uma ferramenta *online* para explorar, rastrear e verificar transações, endereços, contratos inteligentes e outras atividades na *blockchain* do Ethereum.

Figura 13 – Etherscan



Fonte: (ETHERSCAN.IO, c2023).

4.4 Propriedades de avaliação

- Transparência: A transparência refere-se à capacidade do sistema em fornecer informações claras e compreensíveis sobre todo o processo de votação. Isso inclui a divulgação de informações sobre o funcionamento do sistema, as etapas do processo e contagem, bem como a divulgação dos resultados de forma acessível e compreensível para os eleitores e demais interessados. A transparência é essencial para garantir a confiança no sistema e a verificação da validade dos resultados;

- Sigilo do voto: O sigilo do voto é uma propriedade crucial que protege a privacidade e a confidencialidade do eleitor. O sistema deve garantir que o voto seja registrado de forma anônima, impedindo qualquer tentativa de identificar o eleitor e sua escolha. É essencial que os eleitores se sintam seguros em exercer seu direito de voto sem medo de retaliação ou exposição;

- Unicidade: A propriedade de unicidade assegura que cada eleitor possa exercer seu voto apenas uma vez. O sistema de *e-voting* deve ser capaz de detectar e prevenir a duplicação de votos, evitando que um eleitor vote mais de uma vez em uma eleição específica. Isso requer a adoção de mecanismos robustos para garantir a identificação única de cada eleitor e a contagem precisa dos votos;

- Elegibilidade do eleitor: A elegibilidade do eleitor refere-se à verificação de que apenas os eleitores elegíveis possam participar do processo de votação. O sistema deve ter mecanismos eficientes para autenticar e verificar a elegibilidade de cada eleitor, impedindo o acesso de indivíduos não autorizados ou impedidos de votar;

- Verificabilidade: A verificabilidade é a capacidade dos eleitores e demais interessados verificarem a precisão e integridade dos resultados da votação. O sistema deve permitir que os eleitores verifiquem se seu voto foi registrado corretamente e que a contagem dos votos foi feita precisamente. Isso pode ser alcançado por meio de mecanismos de auditoria, registros digitais, criptografia e outras técnicas que possibilitem a verificação independente dos resultados;

- Acessibilidade: A acessibilidade diz respeito à garantia de que o sistema seja acessível a todos os eleitores, independentemente de suas habilidades físicas, visuais, auditivas ou cognitivas. O sistema deve ser projetado considerando a inclusão e a usabilidade, garantindo que todos os eleitores possam exercer seu direito de voto de maneira independente e sem obstáculos;

- Resistência à coerção: A resistência à coerção refere-se à capacidade do sistema em proteger os eleitores contra qualquer forma de pressão, intimidação ou coerção durante o processo de votação. O sistema deve garantir que os eleitores possam expressar livremente suas escolhas, sem serem influenciados ou coagidos por terceiros. Isso requer medidas de segurança que impeçam o rastreamento de votos e garantam a confidencialidade e integridade do processo;

- Disponibilidade: A propriedade de disponibilidade diz respeito à garantia de que o sistema esteja acessível e operacional durante o período de votação. Isso envolve a implementação de medidas para evitar interrupções, falhas de sistema ou ataques cibernéticos que comprometam a disponibilidade do sistema. A garantia da disponibilidade é fundamental para permitir que os eleitores votem sem obstáculos e para manter a confiança no processo eleitoral.

4.5 Avaliação

- Transparência/Integridade: O sistema de votação proporciona transparência e integridade aos eleitores e outros *stakeholders* envolvidos no processo. Por meio de uma arquitetura baseada em *blockchain* é possível garantir a imutabilidade e a distribuição dos registros de votação. Isso permite que qualquer pessoa verifique a autenticidade dos votos e garante a confiança na solução de votação eletrônica proposta;

- Sigilo/Privacidade do Voto: A aplicação fornecer uma camada de privacidade, pois o usuário é identificado por um endereço público, porém caso alguém consiga associar o endereço público a uma pessoa específica, será possível visualizar os detalhes de votação, como o candidato escolhido, por meio de uma API web, como o "Etherscan";
- Unicidade: A tecnologia blockchain do Ethereum garante que cada voto seja contado apenas uma vez. Como os registros de votação são imutáveis e rastreáveis, é impossível alterar os votos registrados. A utilização de contratos inteligentes e a lógica de programação implementada no sistema garantem a contagem precisa e a unicidade dos votos;
- Elegibilidade: A elegibilidade do eleitor é garantida por meio da utilização de uma *whitelist* de endereços de carteira autorizados. Somente os eleitores com endereços de carteira registrados na *whitelist* têm permissão para votar. Isso impede que pessoas não autorizadas participem do processo de votação e garante a elegibilidade dos eleitores;
- Verificabilidade/Auditoria: O sistema baseado em Ethereum permite que os eleitores verifiquem seus próprios votos e também permite a auditoria independente dos resultados da votação. Como todas as transações são registradas na *blockchain*, qualquer pessoa pode verificar a integridade dos votos e o resultado da votação. A transparência da *blockchain* e a capacidade de acesso público aos registros garantem a verificabilidade e a auditabilidade do sistema;
- Acessibilidade: O sistema visa garantir a acessibilidade aos eleitores. A natureza digital do sistema permite que eleitores participem da votação remotamente, usando uma carteira Ethereum e uma conexão à internet. Além disso, interfaces amigáveis e adaptáveis são projetadas para facilitar a interação e a participação de eleitores com diferentes necessidades e habilidades. Porém, é necessário possuir conhecimentos básicos em informática para realizar esse procedimento;
- Liberdade de Voto/Resistência à Coerção: Considerando que o sistema se destina a votações abertas, a resistência à coerção não é um requisito. No entanto, em casos de votações que necessitem de garantia de sigilo do voto, a utilização de endereços de carteira não garante resistência à coerção, embora forneça uma camada adicional de segurança. No sistema implementado, se houver a possibilidade de associar o eleitor ao endereço público de sua carteira, a liberdade de expressar sua opinião de forma independente, sem influência ou pressão indevida, estará vulnerável;
- Disponibilidade: A disponibilidade contínua do sistema é garantida pela infraestrutura da *blockchain* Ethereum. A rede Ethereum é projetada para ser altamente disponível e resistente a falhas. Isso assegura que o sistema de votação esteja operacional e acessível durante o período de votação, evitando interrupções significativas;

- Equidade: A equidade é garantida por meio da neutralidade do sistema. Os votos são tratados de maneira igualitária, independentemente das preferências dos eleitores ou dos resultados preliminares. A contagem dos votos é feita imparcialmente, seguindo a lógica implementada nos contratos inteligentes, garantindo assim a equidade no processo de votação.

5 Conclusões

A partir da análise realizada, é possível concluir que a solução desenvolvida atende à maioria dos requisitos gerais de um sistema de votação eletrônico. Utilizando os critérios enumerados por Bokslag e Vries (2016), foi possível avaliar a aplicação em diversos aspectos, identificando tanto seus pontos fortes quanto áreas que necessitam de melhorias. Os contratos inteligentes "Voting.sol" e "VotingFactory.sol" garantem uma gestão segura e transparente do processo eleitoral. Esses tornam todas as interações visíveis para os participantes, garantindo transparência e verificabilidade. Além disso, garantem a unicidade, integridade dos votos e restringem a gestão da votação apenas à autoridade eleitoral.

No entanto, se for necessário garantir o sigilo do voto, é imprescindível adotar medidas de segurança adicionais para proteger a manipulação de dados sensíveis e garantir a autenticação adequada. Isso se torna particularmente importante, uma vez que todas as comunicações entre os clientes e o contrato são visíveis na rede *blockchain*. Além disso, a necessidade de os usuários instalarem a extensão Metamask pode limitar a acessibilidade e a facilidade de uso do sistema para alguns eleitores.

Embora o trabalho tenha alcançado resultados significativos na implementação de um sistema de votação aberta baseado em Ethereum, é importante destacar as dificuldades encontradas ao lidar com as soluções de criptografia mencionadas nas Seções 2.4.1 e 2.6.6. A implementação de provas de conhecimento zero e MACI exigem um conhecimento técnico avançado e uma compreensão profunda dos algoritmos e protocolos subjacentes. Uma das principais dificuldades enfrentadas foi a complexidade dos algoritmos criptográficos envolvidos. A implementação correta e segura desses algoritmos requer um domínio específico em criptografia, além de um cuidado minucioso na escolha dos parâmetros adequados e na gestão das chaves criptográficas. Além disso, a integração dessas soluções de criptografia com a infraestrutura existente do sistema de votação pode apresentar desafios técnicos adicionais. É necessário garantir a interoperabilidade entre as diferentes camadas do sistema, como os contratos inteligentes Ethereum, a interface do usuário e os protocolos de comunicação.

Outra dificuldade seria a necessidade de manter um equilíbrio entre segurança e desempenho. A aplicação de técnicas criptográficas avançadas pode introduzir uma sobrecarga significativa no processamento e armazenamento dos dados, o que pode afetar a escalabilidade e a eficiência do sistema. É necessário considerar cuidadosamente o custo de *Gas* das transações na rede Ethereum e a capacidade do sistema de lidar com o aumento da carga de processamento devido à criptografia.

Em suma, a solução desenvolvida usando Ethereum possibilita a realização segura, transparente e verificável de votações abertas por meio de uma aplicação web. Através dessa abordagem, os eleitores têm a conveniência de acessar o sistema de votação remotamente. O código-fonte [1] do projeto está disponível no Github, promovendo mais transparência e colaboração, permitindo que outros pesquisadores e desenvolvedores possam contribuir, revisar e aprimorar o sistema.

5.1 Trabalhos futuros

Nesta seção é apresentada uma visão dos trabalhos futuros que podem ser realizados para aprimorar o sistema desenvolvido. Foram identificadas diversas áreas de melhoria, destacando aspectos como criptografia das transações de voto, encerramento automático da votação, opções de conexão, acessibilidade web e simplificação do processo de assinatura das transações.

- Criptografia das transações de voto: Em casos de votações que necessitem de garantia de sigilo do voto, é essencial implementar criptografia robusta para garantir a segurança das transações de voto. A aplicação deve utilizar algoritmos criptográficos confiáveis para proteger os dados sensíveis dos eleitores, como a escolha de voto, durante o processo de transmissão e armazenamento. Uma das abordagens promissoras para a criptografia de votos é a utilização do MACI, um sistema que fornece garantias fundamentais para a realização de votações digitais, com ênfase na resistência à corrupção, segurança e privacidade.

- Utilização de endereços de carteira anônimos: Ao fornecer endereços de carteira aleatórios ou gerados especificamente para cada eleitor, é possível dificultar a vinculação direta entre um endereço e a identidade de um eleitor. Isso ajuda a preservar a privacidade do voto, mesmo que não seja uma solução criptográfica completa.

- Encerramento automático da votação: O sistema atual precisa ser aprimorado para permitir o encerramento automático da votação após a duração especificada. A implementação de uma funcionalidade que encerre a votação automaticamente evita a necessidade de intervenção manual e garante que o processo de votação seja concluído conforme os prazos estabelecidos.

- Mais opções de conexão além da Metamask: Para melhorar a acessibilidade e a usabilidade da aplicação é recomendável oferecer aos eleitores mais opções de conexão além da Metamask. Isso pode incluir a integração de outras carteiras digitais ou soluções de autenticação que sejam amplamente utilizadas e ofereçam uma experiência mais acessível e conveniente para os eleitores.

[1] <https://github.com/luucas-melo/TCC_blockchain_voting>

- Acessibilidade web: Para garantir a acessibilidade da aplicação a todos os usuários, é importante incorporar práticas de acessibilidade web. Isso inclui fornecer suporte para leitores de tela, utilizando atributos como "aria-labels" e outras técnicas que tornam a aplicação acessível para pessoas com deficiências visuais ou outras necessidades especiais.

- Simplificação do processo de assinatura das transações: Para que os eleitores assinem as transações de voto, é necessário simplificar o processo de conexão da carteira digital e assinatura. Isso pode ser alcançado por meio de orientações claras, instruções passo a passo e interfaces intuitivas que guiem os eleitores durante o processo de assinatura, tornando-o mais fácil e compreensível.

Referências

ACE. *Focus on E-Voting —*. c2023. Disponível em: <https://aceproject.org/ace-en/focus/e-voting/default>. Citado 5 vezes nas páginas 10, 14, 15, 16 e 17.

ANANE, R.; FREELAND, R.; THEODOROPOULOS, G. e-Voting Requirements and Implementation. In: *The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC-EEE 2007)*. Tóquio, Japão: IEEE, 2007. p. 382–392. ISBN 978-0-7695-2913-4. Disponível em: <http://ieeexplore.ieee.org/document/4285237/>. Citado 2 vezes nas páginas 10 e 18.

ANDRADE, M. M. de. Introdução à metodologia do trabalho científico: elaboração de trabalhos na graduação 9788522458561, 9788522478392. In: . 10ª. ed. [s.n.], 2022. ISBN 9788522478392. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788522478392/>. Citado na página 39.

ANTONOPOULOS, A. M.; Gavin Wood. Mastering Ethereum. In: *Mastering Ethereum*. California, Estados Unidos: O'Reilly Media, 2018. ISBN 978-1-4919-7194-9. Citado na página 32.

BASHIR, I. *Mastering blockchain: distributed ledger technology, decentralization, and smart contracts explained*. Second edition, fully revised and updated. Birmingham Mumbai: Packt, 2018. (Expert insight). ISBN 978-1-78883-904-4. Citado 4 vezes nas páginas 22, 23, 28 e 29.

BOKSLAG, W.; VRIES, M. Evaluating e-voting: theory and practice. fev. 2016. Citado 5 vezes nas páginas 16, 17, 18, 45 e 60.

BOSRI, R. et al. Towards A Privacy-Preserving Voting System Through Blockchain Technologies. In: . [S.l.: s.n.], 2019. Citado na página 10.

BUTERIN, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2014. Citado na página 31.

COINBASE. *O que é o Ethereum?* c2022. Disponível em: <https://www.coinbase.com/pt/learn/crypto-basics/what-is-ethereum>. Citado na página 32.

COUNCIL, C. S. C. Cloud Customer Architecture for Blockchain. 2017. Disponível em: <https://www.omg.org/cloud/deliverables/cloud-customer-architecture-for-blockchain.htm>. Citado na página 29.

ETHEREUM. *Intro to Ethereum*. 2022. Acesso em: 22 dez. 2022. Disponível em: <https://ethereum.org/>. Citado 6 vezes nas páginas 11, 28, 32, 33, 34 e 36.

ETHEREUM. *Networks*. 2022. Acesso em: 22 dez. 2022. Disponível em: <https://ethereum.org>. Citado 4 vezes nas páginas 24, 25, 36 e 47.

ETHEREUM. *Transactions*. 2023. Acesso em: 04 jul. 2023. Disponível em: <https://ethereum.org>. Citado 3 vezes nas páginas 34, 35 e 36.

ETHEREUM Community. Ethereum, 2023. Disponível em: <https://github.com/ethereum/devp2p>. Citado na página 30.

ETHERSCAN.IO. *TESTNET Sepolia (ETH) Blockchain Explorer*. c2023. Acesso em: 09 jul. 2023. Disponível em: <http://sepolia.etherscan.io/>. Citado na página 56.

FOLLOWMYVOTE. *Blockchain Voting: The End To End Process*. Ano desconhecido. Acesso em: 17 jan. 2023. Disponível em: <https://followmyvote.com/blockchain-voting-the-end-to-end-process/>. Citado na página 20.

GANACHE. *Ganache | Overview - Truffle Suite*. c2022. Acesso em: 26 jan. 2023. Disponível em: <https://trufflesuite.com/docs/ganache/>. Citado na página 46.

GOLDWASSER, S.; MICALI, S.; RACKOFF, C. The knowledge complexity of interactive proof-systems. In: *Symposium on the Theory of Computing*. [S.l.: s.n.], 1985. Citado na página 24.

HABER, S.; STORNETTA, W. S. How to time-stamp a digital document. *Journal of Cryptology*, v. 3, n. 2, p. 99–111, Jan 1991. ISSN 1432-1378. Disponível em: <https://doi.org/10.1007/BF00196791>. Citado na página 21.

HILEMAN, G.; RAUCHS, M. 2017 Global Blockchain Benchmarking Study. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. Disponível em: <https://www.ssrn.com/abstract=3040224>. Citado na página 26.

HJáLMARSSON, F. et al. Blockchain-Based E-Voting System. In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. [S.l.: s.n.], 2018. p. 983–986. ISSN: 2159-6190. Citado na página 19.

IBM. *Benefits of blockchain - IBM Blockchain | IBM*. c2022. Disponível em: <https://www.ibm.com/topics/benefits-of-blockchain>. Citado 2 vezes nas páginas 30 e 31.

IBM. *What are smart contracts on blockchain? | IBM*. c2022. Acesso em: 26 dez. 2022. Disponível em: <https://www.ibm.com/topics/smart-contracts>. Citado na página 31.

JIE, K. W. *Minimum anti-collusion infrastructure*. Privacy amp; Scaling Explorations, 2021. Disponível em: <https://privacy-scaling-explorations.github.io/maci/index.html>. Citado 2 vezes nas páginas 37 e 38.

JOHNSTON, D. *The General Theory of Decentralized Applications, Dapps*. 2022. Original-date: 2013-11-20T05:58:04Z. Disponível em: <https://github.com/DavidJohnstonCEO/DecentralizedApplications>. Citado na página 36.

KHAN, K. M.; ARSHAD, J.; KHAN, M. M. Secure digital voting system based on blockchain technology. *Int. J. Electron. Gov. Res.*, IGI Global, USA, v. 14, n. 1, p. 53–62, jan 2018. ISSN 1548-3886. Disponível em: <https://doi.org/10.4018/IJEGR.2018010103>. Citado na página 19.

KISSFLOW. *What is Kanban Methodology | Introduction to Kanban Framework*. 2022. Acesso em: 04 fev. 2023. Disponível em: <https://kissflow.com/project/agile/kanban-methodology/>. Citado na página 40.

KOç, A. et al. Towards Secure E-Voting Using Ethereum Blockchain. In: . [S.l.: s.n.], 2018. Citado na página 32.

LIN, I.-C.; LIAO, T.-C. A survey of blockchain security issues and challenges. *International Journal of Network Security*, v. 19, p. 653–659, 09 2017. Citado na página 25.

MERKLE, R. C. A Digital Signature Based on a Conventional Encryption Function. In: POMERANCE, C. (Ed.). *Advances in Cryptology — CRYPTO '87*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988. p. 369–378. ISBN 978-3-540-48184-3. Citado na página 23.

METAMASK. *The crypto wallet for Defi, Web3 Dapps and NFTs | MetaMask*. c2023. Acesso em: 26 jan. 2023. Disponível em: <https://metamask.io/>. Citado na página 47.

MICHAEL, J.; COHN, A.; BUTCHER, J. R. Blockchain technology. *The Journal*, v. 1, n. 7, 2018. Citado na página 26.

MOURA, T.; GOMES, A. *Benefits of blockchain - IBM Blockchain | IBM*. 2017. Disponível em: <https://www.ibm.com/topics/benefits-of-blockchain>. Citado na página 10.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. 2008. Citado 3 vezes nas páginas 21, 23 e 24.

RAVAL, S. *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. 1st. ed. [S.l.]: "O'Reilly Media, Inc.", 2016. Google-Books-ID: fvywDAAAQBAJ. ISBN 978-1-4919-2452-5. Citado na página 36.

REMIXIDE. *Remix - Ethereum IDE & community*. c2022. Acesso em: 26 jan. 2023. Disponível em: <https://remix-project.org/>. Citado na página 46.

SANKAR, L. S.; SINDHU, M.; SETHUMADHAVAN, M. Survey of consensus protocols on blockchain applications. In: *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Coimbatore, India: IEEE, 2017. p. 1–5. ISBN 978-1-5090-4559-4. Disponível em: <http://ieeexplore.ieee.org/document/8014672/>. Citado na página 30.

SCHWABER, K. *Agile Project Management with Scrum*. Redmond, WA: Microsoft Press, 2004. ISBN 978-0-7356-1993-7. Disponível em: <http://my.safaribooksonline.com/9780735619937>. Citado 2 vezes nas páginas 39 e 40.

SPRINGALL, D. et al. Security analysis of the estonian internet voting system. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2014. (CCS '14), p. 703–715. ISBN 9781450329576. Disponível em: <https://doi.org/10.1145/2660267.2660315>. Citado na página 20.

STALLINGS, W. *Cryptography and network security: principles and practice*. 4th ed. ed. Upper Saddle River, N.J: Pearson/Prentice Hall, 2006. OCLC: ocm63126393. ISBN 978-0-13-187316-2. Citado na página 23.

SZABO, N. *Smart Contracts*. 1994. Disponível em: <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html>. Citado na página 31.

TAş, R.; TANRıöVER, A Systematic Review of Challenges and Opportunities of Blockchain for E-Voting. *Symmetry*, v. 12, p. 1328, ago. 2020. Citado 7 vezes nas páginas 10, 11, 13, 14, 21, 27 e 32.

TRUFFLE. c2022. Acesso em: 26 jan. 2023. Disponível em: <https://trufflesuite.com/truffle/>. Citado na página 46.

TSAHKNA, A.-G. E-voting: Lessons from Estonia. *European View*, v. 12, n. 1, p. 59–66, jun. 2013. ISSN 1781-6858. Publisher: SAGE Publications Ltd. Disponível em: <https://doi.org/10.1007/s12290-013-0261-7>. Citado na página 13.

VALIMISED. *Statistics about Internet voting in Estonia | Elections in Estonia*. 2019. Acesso em: 17 jan. 2023. Disponível em: <https://www.valimised.ee/en/archive/statistics-about-internet-voting-estonia>. Citado na página 20.

VBUTERIN et al. *Minimal anti-collusion infrastructure*. 2019. Disponível em: <https://ethresear.ch/t/minimal-anti-collusion-infrastructure/5413>. Citado na página 37.

VERCEL. *Next.js by Vercel - The React Framework*. c2023. Acesso em: 26 jan. 2023. Disponível em: <https://nextjs.org>. Citado na página 47.

WANG, H. et al. Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, v. 14, p. 352–375, out. 2018. Citado 2 vezes nas páginas 22 e 31.

WEB3.JS. *Web3.js 1.0.0 documentation*. c2016. Acesso em: 26 jan. 2023. Disponível em: <https://web3js.readthedocs.io/en/v1.5.2/getting-started.html>. Citado na página 46.

WOLF, P.; NACKERDIEN, R.; TUCCINARDI, D. *Introducing electronic voting: essential considerations*. Stockholm: International Institute for Democracy and Electoral Assistance, 2011. (Policy paper). ISBN 978-91-86565-21-3. Citado 4 vezes nas páginas 13, 14, 16 e 17.

Apêndices

APÊNDICE A – Documentação de software

A.1 Histórias de Usuário

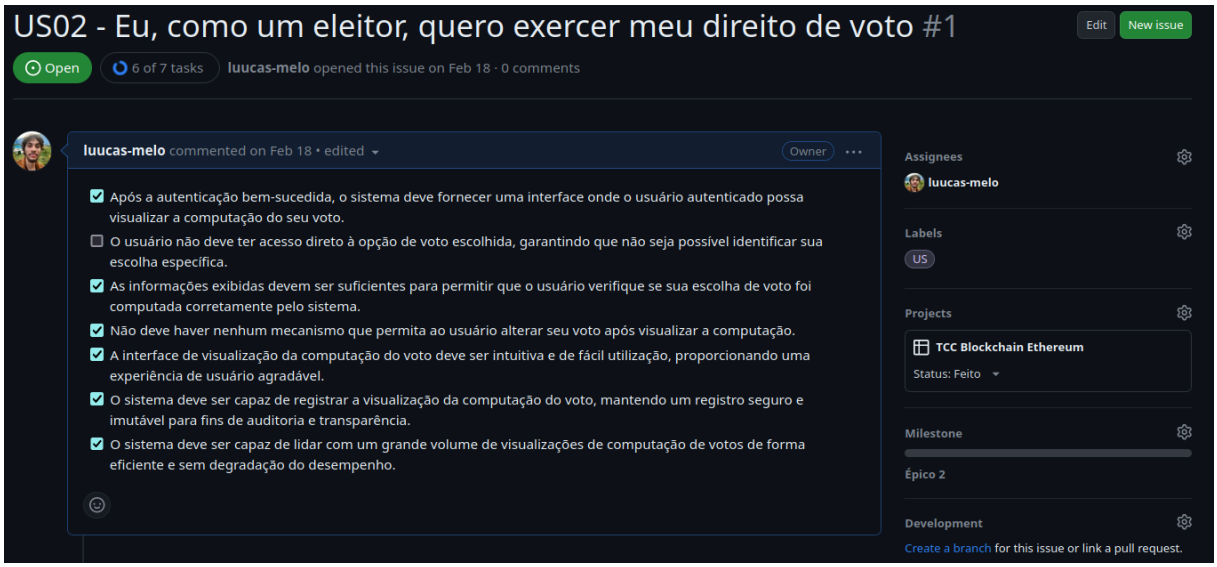
A.1.1 US01

Figura 14 – US01 - História de Usuário que contempla a autenticação dos usuários e seus critérios de aceitação

Fonte: Autores.

A.1.2 US02

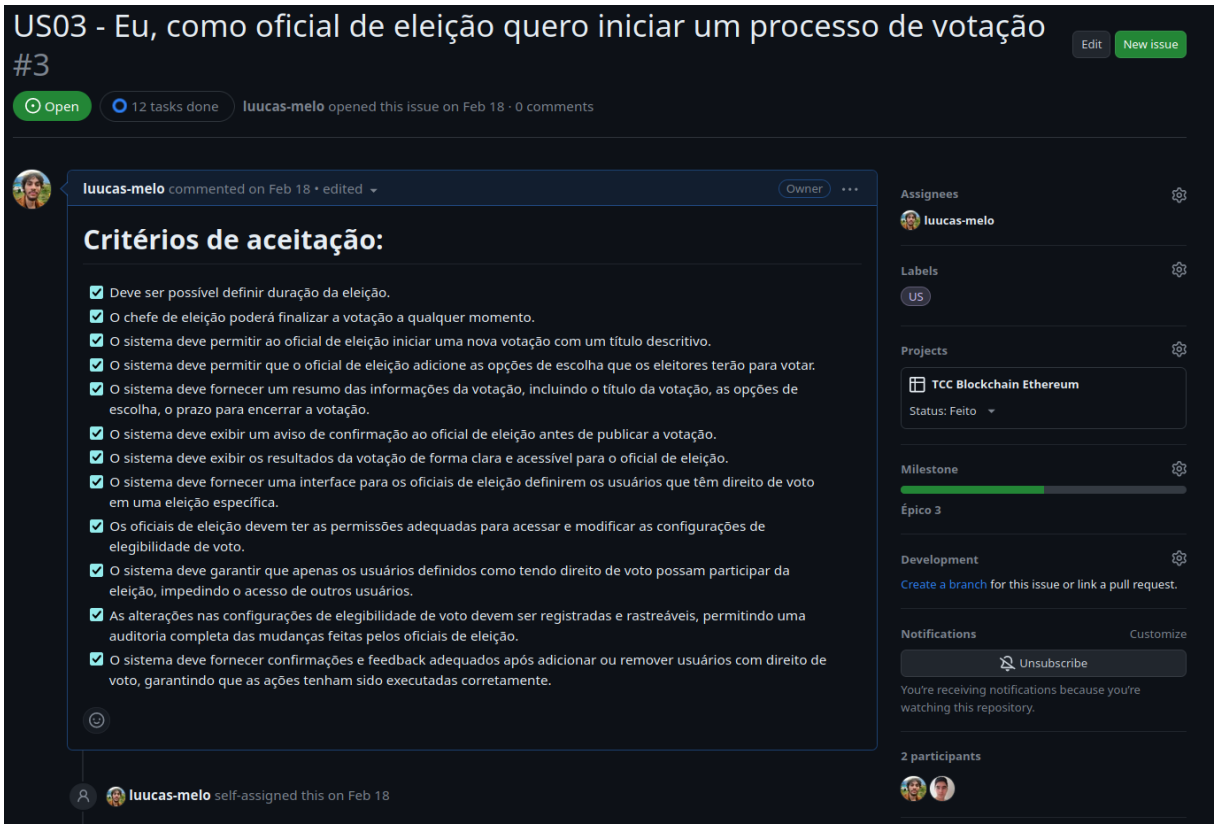
Figura 15 – US02 - História de Usuário que contempla a realização do voto e seus critérios de aceitação



Fonte: Autores.

A.1.3 US03

Figura 16 – US03 - História de Usuário que contempla a inicialização da votação e seus critérios de aceitação



Fonte: Autores.

A.1.4 US04

Figura 17 – US04 - História de Usuário que contempla a visualização dos resultados da votação e seus critérios de aceitação

Fonte: Autores.

A.1.5 US05

Figura 18 – US05 - História de Usuário que contempla a visualização das transações na blockchain e seus critérios de aceitação

- Fonte: Autores.

A.2 Testes unitários

A.2.1 Teste unitário do contrato *Voting*

```
pragma solidity ^0.8.0;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/Voting.sol";

contract TestVoting {
    Voting voting;

    function beforeEach() public {
        // Deploy the Voting contract before each test
        string[] memory initialProposals = new string[](2);
        initialProposals[0] = "Proposal 1";
        initialProposals[1] = "Proposal 2";
        address[] memory initialWhiteList = new address[](3);
        initialWhiteList[0] = address(0x1);
        initialWhiteList[1] = address(0x2);
        initialWhiteList[2] = address(this);
        voting = new Voting(
            address(this),
            "Test Voting",
            initialProposals,
            initialWhiteList,
            block.timestamp + 1720636849
        );
    }

    // Verify the initial state of the contract
    function testInitialState() public {
        Assert.equal(voting.title(), "Test Voting", "Incorrect title"
        Assert.equal(
            voting.votingDuration(),
            block.timestamp + 1720636849,
            "Incorrect voting duration"
        );
        Assert.equal(
```

```
            voting.votingEnded(),
            false,
            "Voting should not have ended"
        );
        Assert.equal(
            voting.votingStarted(),
            false,
            "Voting should not have started"
        );
        Assert.equal(
            voting.votingCancelled(),
            false,
            "Voting should not be cancelled"
        );
        Assert.equal(
            voting.electionCommission(),
            address(this),
            "Incorrect election commission"
        );
    }

    // Verify the initial proposals
    function testInitialProposals() public {
        string[] memory initialProposals = voting.getProposals();
        Assert.equal(
            initialProposals.length,
            2,
            "Incorrect number of initial proposals"
        );
        Assert.equal(
            initialProposals[0],
            "Proposal 1",
            "Incorrect initial proposal 1"
        );
        Assert.equal(
            initialProposals[1],
            "Proposal 2",
            "Incorrect initial proposal 2"
        );
```

```
}

// Verify the initial whitelist
function testInitialWhiteList() public {
    // Verify the initial whitelist
    address[] memory initialWhiteList =
        voting.getWhiteListedAddresses();
    Assert.equal(
        initialWhiteList.length,
        3,
        "Incorrect number of initial whitelist addresses"
    );
    Assert.equal(
        initialWhiteList[0],
        address(0x1),
        "Incorrect initial whitelist address 1"
    );
    Assert.equal(
        initialWhiteList[1],
        address(0x2),
        "Incorrect initial whitelist address 2"
    );
}

function testWhiteList() public {
    // Add an address to the whitelist
    voting.addToWhiteList(address(this));

    // Call the vote function with an address
    //not in the whitelist
    (bool success, bytes memory data) = address(voting).call(
        abi.encodeWithSignature("vote(uint256)", 0)
    );

    // Verify the result
    Assert.isFalse(
        success,
        "Only whitelisted addresses should be able to vote"
    );
```

```
    }

    function testStartVoting() public {
        voting.startVoting();
        Assert.equal(
            voting.votingStarted(),
            true,
            "Voting should have started"
        );
    }

    function testGetResultVotingNotEnded() public {
        // Call the getResult function before the voting has ended
        (bool success, ) = address(voting).call(
            abi.encodeWithSignature("getResult()")
        );

        // Verify the result
        Assert.isFalse(
            success,
        "Should not be able to get the result before the voting ends"
        );
    }
}
```