Sistema de eleições desenvolvido com a tecnologia de contratos inteligentes baseado em Blockchain

Lucas Lagrimante Martinho[1], **Filippe C. Jabour**[1]

[1]Núcleo de Informática – Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas Gerais (IF Sudeste MG)
Juiz de Fora - MG - Brasil

lucaslagrimante@live.com, filippe.jabour@ifsudestemg.edu.br

Abstract. *This article briefly describes the operation of a blockchain Ethereum and how the concept of DApps (Decentralized Applications) can solve real-life problems without the need for central control. The goal was to develop a solution that seeks to replace a traditional voting / voting system and to do a viability test applied to a group of 8 people for cost and feasibility analysis. The need for a differentiated modeling was necessary and then adopted a method specially directed to Blockchain.*

Resumo. *Este artigo descreve de forma breve o funcionamento de uma blockchain Ethereum e como o conceito dos DApps (Aplicativos Descentralizados) consegue resolver problemas da vida real sem necessidade de um controle central. O objetivo foi desenvolver uma solução que busca substituir um sistema de eleições/votação tradicional e fazer um teste de viabilidade aplicado em um grupo de 8 pessoas para análise de custo e viabilidade. A necessidade de uma modelagem diferenciada foi necessária e então adotado um método especialmente direcionado para a Blockchain.*

1. Introdução

A proposta da Blockchain da moeda digital Bitcoin [Nakamoto 2008] surgiu em 31 de outubro de 2008 com seu *white paper*, documento que descreve de forma detalhada um um problema e detalha seus conceitos, causas e principalmente suas soluções, propondo um sistema eletrônico de troca de fundos *peer-to-peer* sem controle central e também a proposta de acabar com um problema conhecido pelos criptógrafos chamado gasto duplo. Com um *fork* do Bitcoin (cópia de um código fonte para outra linha do tempo independente), o Ethereum [Buterin 2013] surgiu em outubro de 2013 com o propósito de permitir a execução de lógicas programáveis além das conhecidas transações de valores já apresentadas pelo Bitcoin.

Como a maioria das cadeias de blocos (Figura 1) mais confiáveis atualmente, a rede Ethereum possui sua rede principal sustentada por nós (*nodes*) que validam e executam as transações por meio de um sistema de prova de trabalho (*proof-to-work*) que garante a confiabilidade e integridade da rede utilizando *Hashing*. Essa rede também é responsável por executar contratos inteligentes e aplicações descentralizadas. Basicamente, os DApps são aplicações executadas em um *back-end peer-to-peer* descentralizado.

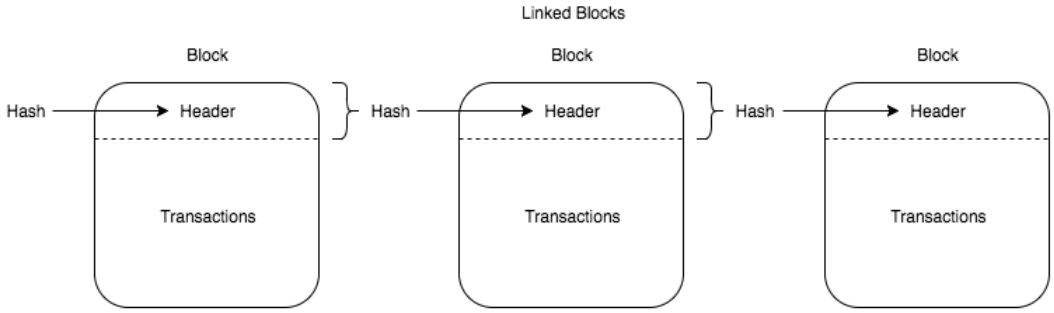

Figura 1. BlockChain Simplificada. Fonte: Plural Sight [1].

Contratos inteligentes são, portanto, aplicativos que funcionam exatamente como programados, sem qualquer possibilidade de tempo de inatividade, censura, fraude ou interferência de terceiros.

A natureza imutável e descentralizada transparente do Blockchain Ethereum permite o desenvolvimento de muitos casos de uso importantes, como organização autônoma, vendas, redes sociais, companhias de seguros, jogos e centenas de outros.

Se uma réplica dessas aplicações acima for feita na plataforma Ethereum descentralizada, isso resultará nas seguintes benefícios:

- Exclusão da possibilidade de um único ponto de falha ou controle.
- Eliminação do tempo ocioso entre respostas das partes.
- Redução de custo, pois os intermediários serão removidos.

1.1. Eleições eletrônicas

Determinar todos os custos de uma eleição nacional é uma tarefa árdua, visto que estão envolvidos nessa cadeia diversos aspectos da política de um país, bem como os diversos níveis de um governo: estados, regiões e municípios.

De acordo com [Ayed 2017], a Estônia foi a primeira nação a adotar um sistema eletrônico de votação para suas eleições nacionais, seguida da Suíça e Noruega, em uma eleição de conselho. Princípios básicos das eleições tradicionais como anonimato, garantia de autenticidade, segurança e não duplicidade de votos devem ser garantidos também na votação eletrônica. Além de assegurar um nível de segurança adicional, também é preciso a certeza de não adulteração das cédulas.

Na Estônia em 2015, segundo [Baltic 2013], os cidadãos utilizaram uma identidade nacional que possuía um circuito integrado protegido por uma chave SHA1 / SHA2. O eleitor precisava baixar um aplicativo, autenticar utilizando seu cartão eletrônico e assim os candidatos se tornavam disponíveis para serem escolhidos e votados. Os votos eram recebidos pelo sistema Estoniano e então computados (Figura 2). Os eleitores poderiam votar mais de uma vez mas apenas o último voto era computado. Isso foi implementado para impedir o voto comprado.

Para [Ayed 2017], um sistema eletrônico de votação deve garantir os seguintes requisitos de sistema:

[1] https://www.pluralsight.com/guides/blockchain-architecture

- Auditoria: As informações são públicas e poderão ser auditadas a qualquer momento.
- Autenticação: Somente eleitores cadastrados poderão realizar votos. O registro geralmente requer a verificação de certas informações e documentos para cumprir as leis atuais, o que não poderia ser feito *online* de maneira segura. Portanto, o sistema deve ser capaz de verificar as identidades dos eleitores em relação a um cadastro prévio.
- Anonimato: A identidade dos eleitores não deve ser nunca ligada à sua cédula de votação.
- Exatidão: os votos devem ser precisos; cada voto deve ser contado e não pode ser alterado, duplicado ou removido.

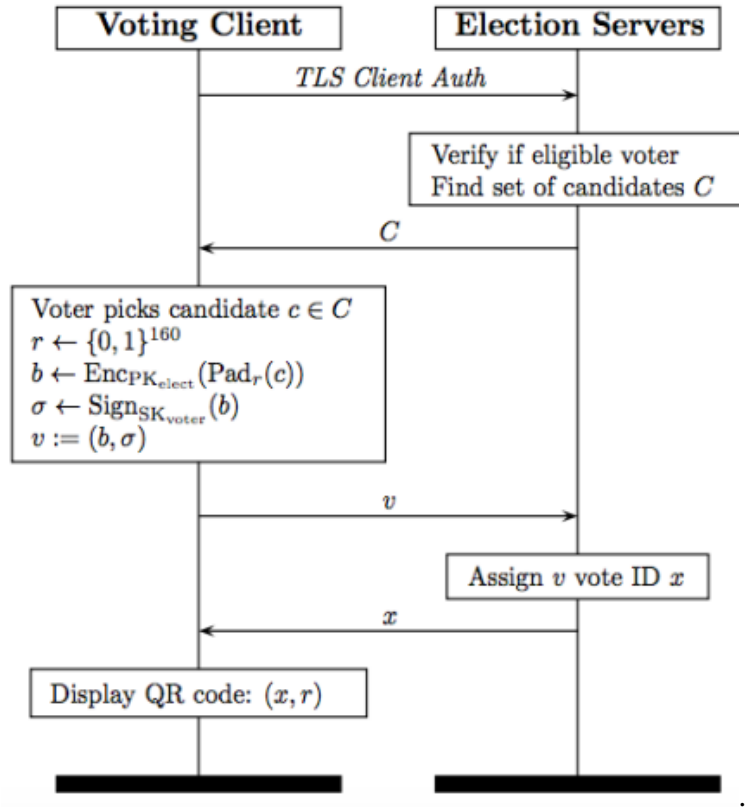

Figura 2. Esquema do Sistema Estoniano de Votação Eletrônica. Fonte: [Ayed 2017].

2. Motivação

Com o surgimento da Blockchain, uma inovadora tecnologia disruptiva, tornou-se necessária a substituição dos métodos clássicos de se transacionar, registrar e armazenar a informação. Possui um livro-razão incapaz de ser violado pelas tecnologias existentes, seguro por design [Reid and Harrigan 2011]. Essa tecnologia se torna a melhor opção para guardar tais informações com total segurança e agilidade de acesso. O protocolo SmartContract (Ethereum) proporciona transações de confiança entre pessoas anônimas, além de garantir que o que foi descrito no contrato será realizado sem a necessidade do intermédio de uma autoridade central.

Não é difícil de imaginar que eleições de qualquer tipo podem e já foram manipuladas, além do tempo de contagem dos votos em casos de eleições com cédulas de papel ser muito elevado. Manipulação seria fisicamente impossível com a rede do Ethereum, já que ninguém possui controle total sobre a rede e todas as mudanças registradas seriam visíveis e públicas, assegurando uma democracia mais transparente e justa.

3. Metodologia

Neste trabalho, foi efetuado o desenvolvimento de uma aplicação baseada na BlockChain do Ethereum que possa sanar alguns dos principais problemas relacionados a eleições com votações presenciais. Levou-se em consideração o custo, a segurança, a remoção das influências externas e a disponibilidade do resultado final da votação.

Também foi executado um teste em laboratório com um grupo de 8 eleitores participantes. Métricas como custo total e velocidade percebida no resultado foram coletadas e analisadas. Também foram coletados dados dos usuários tais como usabilidade, segurança notada, desempenho do sistema e utilidade.

3.1. Levantamento Bibliográfico

Foi executada uma busca de artigos de base científica e foram selecionados artigos relevantes sendo analisados o resumo e título. As fontes da busca foram: Portal Capes/Mec e Google Acadêmico. O idioma dos estudos foi o Inglês e a *string* de busca: (*"blockchain*") OR (*"block chain*") AND (*"election*") OR (*"poll*"). Dez artigos atingiram as metas e foram selecionados.

Segundo [Kubjas 2017], existem alguns grupos que afirmam ter desenvolvido soluções de votação em *blockchain*: Tivi, que afirma ter um sistema seguro, transparente e íntegro, porém não fornece detalhes técnicos; Follow My Vote, que tem alguns detalhes publicados e possui código livre; Blockchain Technologies Corporation, que utiliza a Blockchain do Bitcoin e calcula, por meio de pagamentos, os votos em determinado candidato; BitCongress, que funciona de forma parecida ao anterior, calculando os votos por meio de transações da moeda.

[Zhao and Hubert Chan 2016] propõe uma solução nova, baseada em Bitcoin que, por meio de um esquema de loteria, esconde as escolhas com números aleatórios secretos e, ao final da votação, faz a prova através do algoritmo *zero-knowledge-proofs*. Essa abordagem remove a necessidade de criptografar as cédulas, eliminando a necessidade de uma autoridade central para decodificá-las ao final do período eleitoral.

Já [Lee et al. 2016], descreve a utilização do Bitcoin ou uma BlockChain privada para registro dos votos e um método de autenticação e filtragem dos eleitores com um terceiro confiável.

[Takabatake et al. 2016] usam a moeda ZeroCoin para embaralhar os endereços Bitcoin garantindo que não seja possível ligar as moedas a um endereço específico.

3.2. Tecnologias utilizadas

As tecnologias apresentadas abaixo foram utilizadas no desenvolvimento deste trabalho.

3.2.1. Ethereum

Criado em 2013, o Ethereum surge com um *fork* do código do Bitcoin e se torna a primeira *blockchain* a permitir programação de contratos inteligentes seguros. Os blocos do Ethereum, a medida que são minerados, podem carregar com eles transações simples de valores e, como sabemos, os contratos inteligentes. A leitura dessas informações é gratuita, porém, para desencorajar spam e também como meio de proteção, as informações inseridas são pagas. Esse custo é conhecido como ”gás”, e tem seu preço derivado do Ether (moeda).

Qualquer nó pode participar da proteção da rede através da mineração. Os mineradores competem para encontrar um número aleatório através de uma função de *hash* específica chamada ”Ethash”. Nesse processo, encontramos o *nonce*, que basicamente é um número arbitrário e pode ser usado apenas uma única vez. Nessa caso, funciona como uma possível chave para o enigma de cada bloco. Quando o mineiro acerta o alvo atual, ele é recompensado e transmite através da rede para cada nó validar e adicionar aquele bloco à sua própria cópia da Blockchain. Por a solução do enigma não ser trivial, este método gera custos elevados de energia e poder computacional. Por isso, os mineradores são recompensados com alguma quantidade de Ether, incentivando assim o trabalho e a constante proteção da rede. O processo completo da mineração de cada bloco dura em média 12 segundos.

Um grande problema conhecido nos sistemas digitais de transações de moedas é conhecido como Gasto Duplo. Ele consiste na possibilidade de um usuário conseguir efetuar o gasto de uma mesma moeda duas vezes, algo perfeitamente possível em um sistema digital. Isso poderia se dar se um único usuário com grande poder de mineração (maior que 50% do total), criasse uma bifurcação e ao efetuar um gasto de A para B, enviasse ao mesmo tempo as mesmas moedas de A para C.

Graças ao protocolo de consenso, a probabilidade de uma transação de gasto duplo ser aceita diminui exponencialmente a cada confirmação que uma transação recebe, devido ao fato da maior parte dos nós participantes serem honestos. Deste modo, a única maneira de confirmar a ausência de uma transação anterior é estar ciente de todas as transações anteriores, como aponta [Reid and Harrigan 2011].

3.2.2. Web3 - JavaScript

Para comunicação com as funcionalidades do contrato, utilizamos uma biblioteca escrita em JavaScript chamada Web3. Funcionando de forma distribuída por meio de RPC (*Remote Procedure Call*), ela nos permite fazer chamadas ao contrato diretamente na rede Ethereum. Entretanto, deverá haver um provedor que aprove as transações na rede ”segura”.

3.2.3. Ganache - JavaScript

Para os testes internos de funcionalidade (Figura 3), foi usada a biblioteca Ganache. Ela nos proporciona uma *blockchain* pessoal para desenvolvimento Ethereum e deve ser usada

para rodar testes, executar comandos e inspecionar o status da cadeia de blocos, enquanto controla como as cadeias operam.

```
it('run a complete election and pick a winner', async () => {
  //creating
  await openElection.methods.beAnVoter('Lucas Lagrimante Martinho', '13075988626').send({
    from: accounts[1],
    gas: '3000000'
  });
  await openElection.methods.beAnVoter('Felippe Jabour', '12345678910').send({
    from: accounts[2],
    gas: '3000000'
  });
  await openElection.methods.createCandidate('Candidato1', '13075988626', accounts[3]).send({
    from: accounts[0],//manager
    gas: '3000000'
  });
  await openElection.methods.createCandidate('Candidato2', '12345678910', accounts[4]).send({
    from: accounts[0], //manager
    gas: '3000000'
  });
  assert(await openElection.methods.getNumOfCandidates().call() == 2);
  assert(await openElection.methods.getNumOfVoters().call() == 2);
```

Figura 3. Código de Teste. Fonte: O autor.

3.2.4. Solidity

A linguagem por trás dos DApps é a Solidity, linguagem esta nativa do Ethereum, orientada a objetos de alto nível que implementa os contratos inteligentes. Foi inspirada por C++, Python e JavaScript e possui uma vasta documentação, além de ser *open source*. Nela definimos a versão, variáveis globais, construtores, modificadores e funções do contrato (Figura 4).

```
pragma solidity >=0.4.25;

contract OpenElectionFactory {

    address[] public deployedOpenElections;

    function createOpenElection(
    uint _maxCandidates, uint _maxVoters, bool _onlyAuthenticated, string memory _electionName)
    public
    {
        OpenElection newOpenElection = new OpenElection(
            msg.sender, _maxCandidates, _maxVoters, _onlyAuthenticated, _electionName
        );
        deployedOpenElections.push(address(newOpenElection));
    }

    function getDeployedOpenElections() public view returns (address[] memory) {
        return deployedOpenElections;
    }

    function getDeployedOpenElectionsCount() public view returns (uint256) {
        return deployedOpenElections.length;
    }
}
```

Figura 4. Código Solidity da Fábrica de Eleições. Fonte: O autor.

3.2.5. MetaMask

Uma transação é um conjunto de instruções para modificar o estado do Blockchain. Para assinar transações Ethereum e consumir gás são necessários endereços reais, um endereço público e uma chave privada, ou uma carteira configurada com pelo menos uma conta base desbloqueada para pagar pela transação de votação. A forma mais simples e segura de utilizar as chaves privadas do usuário para assinar as transações é através da extensão MetaMask.

O MetaMask funciona como um provedor para esses endereços, permitindo a interação com a rede sem precisar ter um nó instalado localmente. Ele inclui um cofre de identidade seguro, fornecendo uma interface de usuário para gerenciar suas identidades em sites diferentes e assinar transações Blockchain.

A interação com nossa aplicação foi desenvolvida a partir da utilização dessa extensão instalada no navegador (Figura 5).

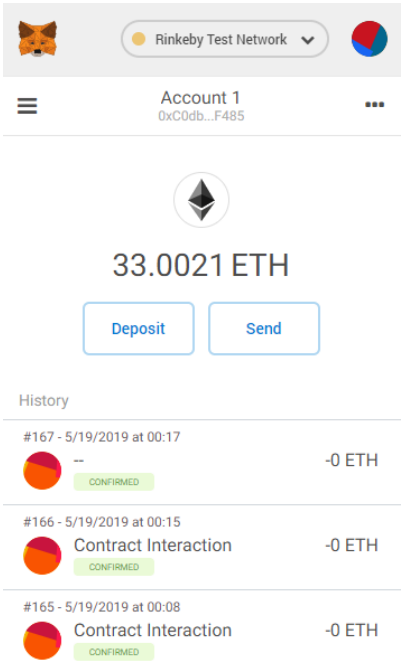

Figura 5. Interface da Extensão MetaMask. Fonte: O autor.

3.2.6. React - Semantic UI

É a biblioteca de construção utilizada para criar a conexão com o usuário. Como definido por seus criadores, React é "uma biblioteca JavaScript declarativa, eficiente e flexível para a criação de interfaces de usuário (UI)". É uma linguagem adaptável, onde tudo se torna componente e reativa, onde os componentes são atualizados em tempo real. Tornando-se assim, ideal para o desenvolvimento dos DApps.

Também foi utilizada uma biblioteca chamada Semantic UI para auxiliar na criação dos componentes visuais.

3.3. Plataforma de execução

Existem diferentes sub-redes Ethereum disponíveis atualmente: Ropsten, Kovan e Rinkeby. As redes não interagem entre si. Porém, um endereço público com sua chave privada coexiste nas três. Cada inserção de dados em uma Blockchain significa literalmente pagar por uma transação. Por isso, foi escolhida a rede Rinkeby que trabalha com o algoritmo *proof-to-authority* e permite adquirir moedas sem custo real e realizar os mais diversos testes como se estivéssemos na rede principal.

4. Descrição do desenvolvimento

4.1. BOS - *Blockchain-Oriented Software*

O campo de pesquisa de métodos de *design*, em geral, de práticas de engenharia voltadas ao software orientado a *blockchain* (BOS) ainda está em sua infância.

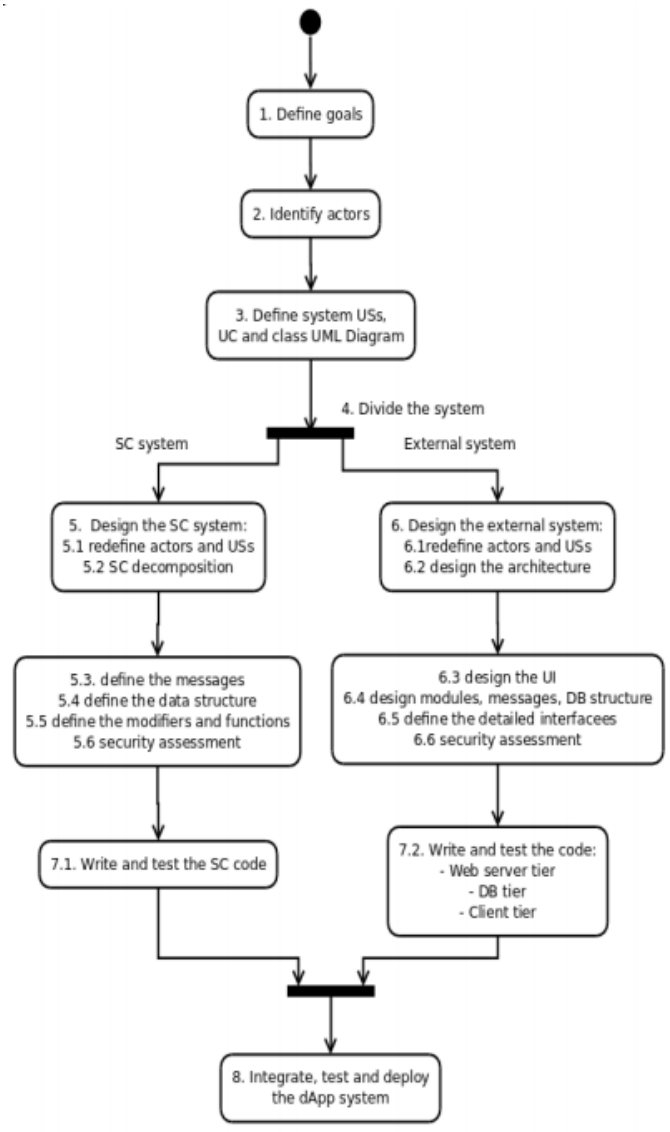

Figura 6. Passos do Processo de Desenvolvimento BOS. Fonte: [Marchesi et al. 2018]

Junto com a crescente demanda por aplicações descentralizadas, segundo [Marchesi et al. 2018], o sentimento da maioria dos engenheiros de software é que o desenvolvimento ”sem regras e apressado”, uma espécie de competição de ”primeiro lugar”, não tem sido suficiente para desenvolver softwares de qualidade e atingir os padrões básicos de engenharia de software. A modelagem proposta buscou seguir uma modelagem baseada no Manifesto ágil, especialmente direcionada aos DApps, com alguns complementos da *Unified Modeling Language* (UML), como também descrito em [Rocha and Ducasse 2018].

O desenvolvimento foi baseado em alguns dos princípios básicos da BOS descritos por [Marchesi et al. 2018] (Figura 6).

4.2. Objetivo

Construir um sistema baseado em Blockchain e no protocolo Smart Contract que gerencie eleições (votações de qualquer tipo, *poll*). Objetivos específicos:

- A autenticidade dos eleitores;
- Que apenas os eleitores permitidos votem;
- Que não haja votos duplicados;
- Que cada voto seja criptografado e anônimo;
- Que tenha resultados públicos.

4.3. Atores

- **Administrador**: Responsável pela criação de uma instância do contrato de eleições, também responsável pelo cadastramento dos candidatos e determinação do início e fim do período de votação, caso não ocorra automaticamente.
- **Eleitor**: Deve se cadastrar em uma eleição e executar os votos.

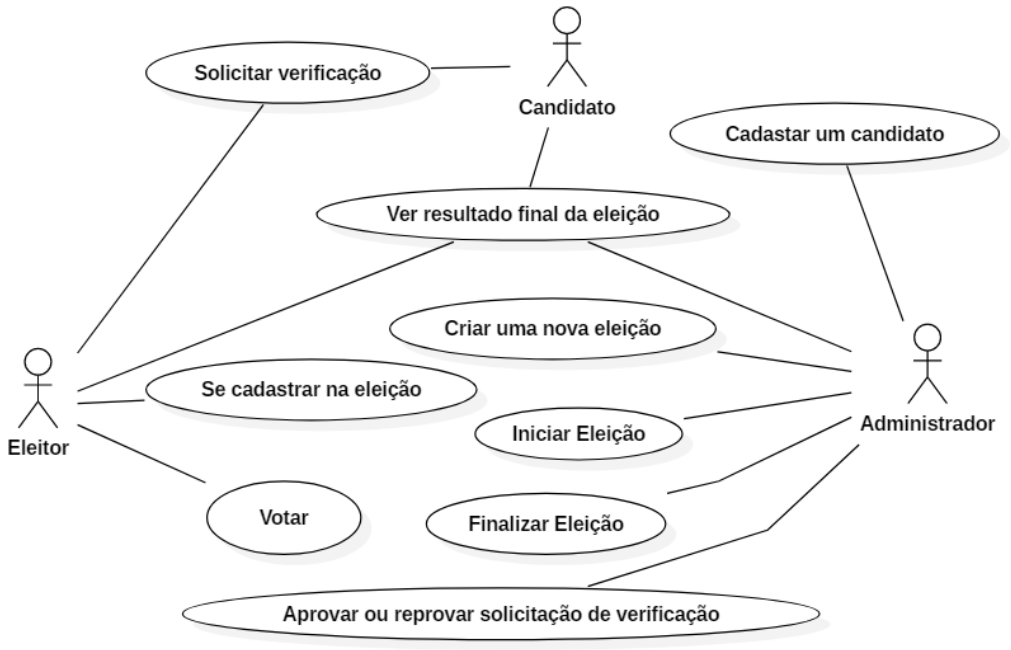

Figura 7. Diagrama de Caso de Uso do Sistema. Fonte: O autor.

4.4. Diagrama de Classes

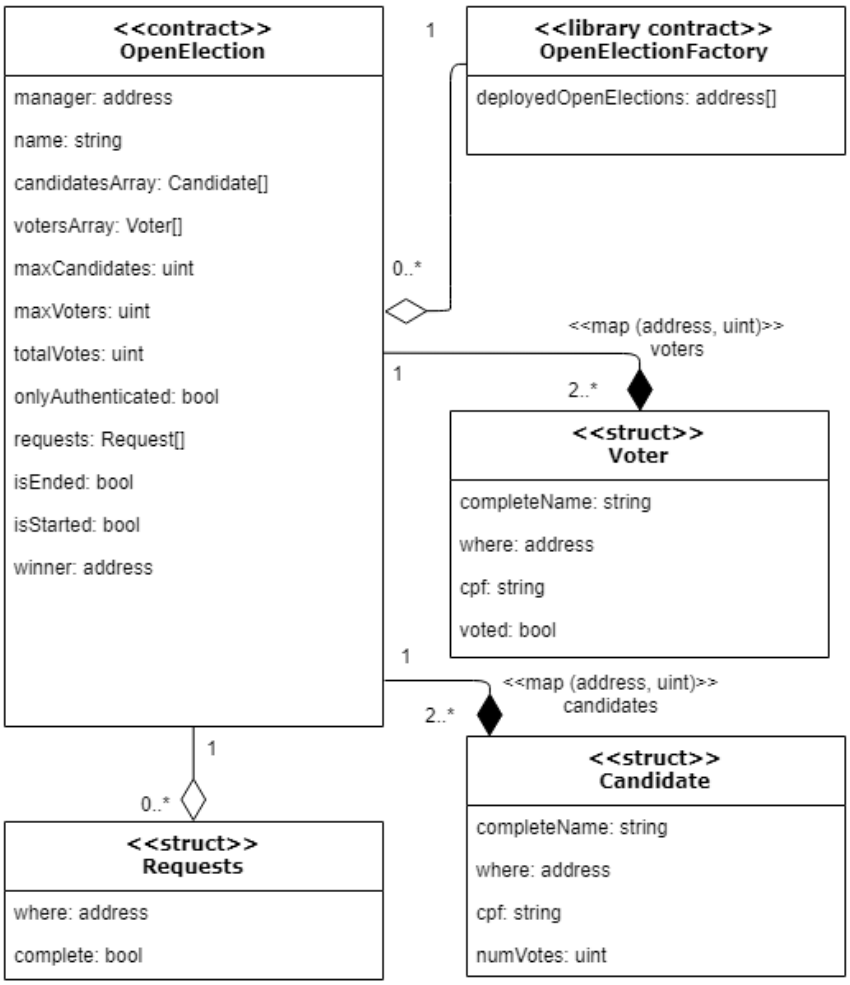

Figura 8. Diagrama de Classes Modificado do Sistema. Fonte: O autor.

4.5. Modificadores e travas de segurança

Os modificadores, como os descritos em Tabela 1, são utilizados como controle de segurança das funções. Além dessa marcação que vai junto à visibilidade da função, o sistema também possui diversas outras validações antes de executar cada função segura do contrato. As travas de segurança da função *vote*, estão descritas abaixo na figura 9.

Modificador	Descrição
restricted()	Obriga que o remetente seja somente o criador/administrador da eleição.
isVoter()	Obriga que o remetente seja um eleitor cadastrado.
isManager()	Obriga que o remetente não seja o criador/administrador.
ended()	Obriga que a eleição não tenha terminado.

Tabela 1. Modificadores. Fonte: O autor.

```
function vote(address _addressOfCandidate)
public isVoter ended
{
    require(isStarted, "Eleição ainda não começou.");
    Voter storage voter = votersArray[voters[msg.sender]];
    require(voter.exists, "Você não é um Eleitor (não existe ainda).");
    require(!voter.voted, "Você não pode votar duas vezes!.");
    if (onlyAuthenticated) {
        require(authenticated[voter.where], "Apenas usuários autenticados.");
    }

    require(candidatesArray[candidates[_addressOfCandidate]].exists, "Candidato não existe.");
    Candidate storage candidate = candidatesArray[candidates[_addressOfCandidate]];
    if (onlyAuthenticated) {
        require(authenticated[candidate.where], "Apenas candidatos autenticados podem ser votados.");
    }
```

Figura 9. Travas de Segurança. Fonte: O autor.

4.6. Funções Públicas do Contrato

As funções públicas representadas na Tabela 2 implementam as funcionalidades descritas nos Casos de Uso do sistema. Para cada função dá-se um nome, possivelmente seguido dos modificadores e parâmetros e uma descrição de sua finalidade.

Função	Modificadores, *Parâmetros*	Ação - Notas
constructor	*address* creator *uint* maxCandidates *uint* maxVoters *bool* onlyAuthenticated *string* electionName	Função criadora da eleição, que é chamada pelo contrato fábrica que gerencia as eleições criadas. Aqui é definido se apenas usuários autenticados pelo administrador poderão votar.
beAnVoter	ended isManager started *string* completeName *string* cpf	Função que pode ser chamada por qualquer pessoa que queira se tornar um eleitor daquela eleição.
createCandidate	restricted ended started *string* completeName *string* cpf *address* candidateAddress	Função que cadastra um novo candidato para a eleição. Pode ser chamada apenas pelo administrador.
vote	isVoter ended *address* addressOfCandidate	Função para votar, apenas eleitores cadastrados podem votar.
createRequest	ended	Função que pode ser chamada por um eleitor ou candidato cadastrado. É necessário caso a variável global onlyAuthenticated = true.
resolveRequest	restricted ended	Onde o administrador aprova ou reprova uma solicitação de verificação.
startOpenElection	restricted ended started	Administrador inicia a eleição caso não ocorra automaticamente.
endOpenElection	restricted ended	Administrador finaliza a eleição caso não ocorra automaticamente.

Tabela 2. Funções Públicas do Contrato. Fonte: O autor.

5. Teste de Campo

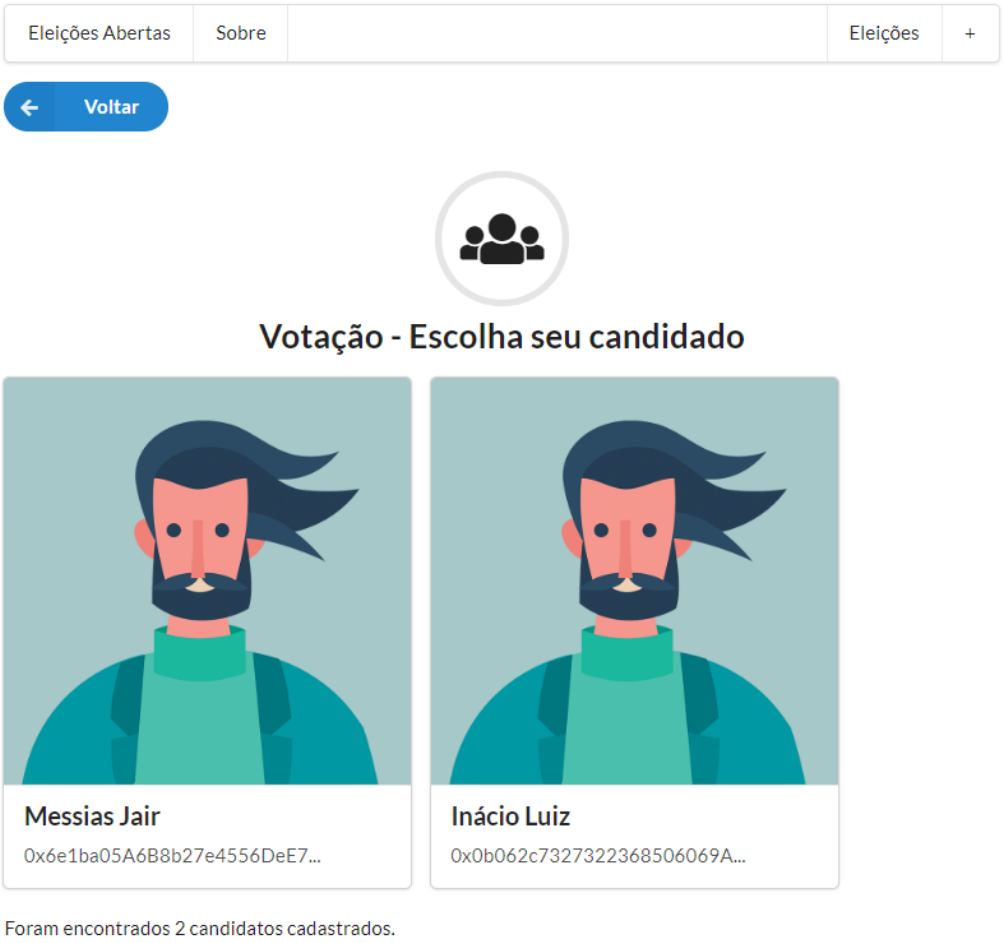

Figura 10. Tela de escolha de Candidatos. Fonte: O autor.

Foi executado um teste com um grupo de 8 alunos do IF Sudeste MG - Campus Juiz de Fora no laboratório da própria instituição. Com o provedor do MetaMask previamente instalado no navegador, os participantes foram instruídos a configurarem em segurança suas carteiras Ethereum para poderem interagir com o sistema de forma segura.

Após essa configuração, foram qualificados a acessar um endereço web onde o sistema está hospedado e efetuarem o cadastro como eleitores.

Diferente da rede principal, os custos na sub-rede Rinkeby não são reais. Consequentemente, é possível adquirir grátis quantidades de moeda em alguns sites na web. Foi então distribuída uma quantidade significativa de 1 Ether para cada participante poder executar as funções.

A eleição foi iniciada e finalizada automaticamente após todos os votos serem computados. O contrato da eleição permanece público e auditável no endereço 0xc24946F06A1dA5472044643a7660F0936d38C0b1.

Os participantes do teste foram solicitados a responder algumas questões referentes a interação com o sistema cuja escala variou de 1 a 5, sendo 1 ruim e 5 bom (demonstrado em figura 11).

Apesar da tecnologia ser complexa e pouco familiar, mais de 60% dos participantes declararam ao mesmo tempo alta sensação de segurança e utilidade na criação de um novo sistema, que possa executar de forma clara e assertiva a contabilização de votos para fazermos decisões democráticas.

Como dito, o tempo de espera de mineração entre cada bloco leva em torno de 12 segundos, e como o sistema precisa processar cada informação, todas as formas de escrita na Blockchain é seguida de um tempo de espera. 50% dos dos participantes não se importaram com esse tempo.

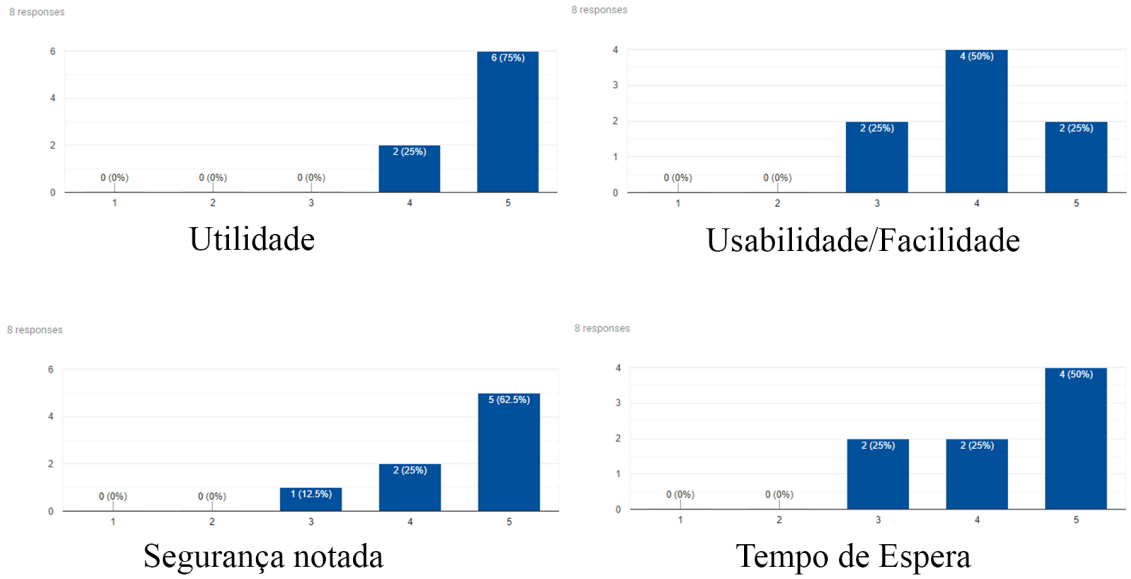

Figura 11. Dados Coletados. Fonte: O autor.

5.1. Considerações

É importante ressaltar que, por questões de custo e manutenção, foi necessário utilizar uma rede de testes para a experiência. O custo e o tempo de cada transação na rede principal são definidos por uma série de questões como capacidade, escalabilidade e utilização em geral.

Na sub-rede Rinkeby utilizada, o preço do combustível (*gas*) é constante, 0.000000001 Ether (1 Gwei), e os blocos são minerados praticamente no mesmo instante. Apesar disso, a rede se assemelha à rede principal e torna viáveis testes e validações de aplicações da Blockchain Ethereum.

Apesar do teste não ter sido executado na rede principal, os principais requisitos são satisfeitos:

- **Anonimato**: a identidade dos participantes não é vinculada ao seu endereço gerado em nenhum momento;
- **Auditoria**: todas as transações são públicas e em tempo real;
- **Exatidão**: todos os requisitos de segurança são validados.

A execução na rede principal acrescenta a proteção que o processo de mineração garante com a prova de trabalho e o consenso coletivo, tornando factível a implementação

real de uma aplicação de votação, em que os endereços não possam ser rastreados sem o uso de informações pessoais ligadas a pessoa que está utilizando o sistema eletrônico.

Por fim, a autenticação dos eleitores com uma base confiável garantiria a impossibilidade de fraudar qualquer tipo de resultado das votações no sistema proposto, já que a exatidão também é garantida pelas travas de segurança.

5.2. Custo

Com o resultado público, qualquer pessoa pode acessar as informações do endereço do contrato inteligente e fazer esse tipo de verificação de gastos e várias outras métricas da rede.

Nosso contrato não possui nenhuma função que faça alguma cobrança do usuário, ou seja, nenhuma interação que gere custos diretos. Entretanto, como as transações de escrita são pagas, apenas a taxa cobrada pelos mineradores está envolvida nos custos gerais.

	Criação do contrato	**Cadastro candidatos**	**Cadastro eleitores e votação**
3 candidatos 8 eleitores	0.002358989 R$ 2,47	0.000440315 R$ 0,46	0.001698432 R$ 1,78
13 candidatos 143,3 milhões de eleitores	0.003576 R$ 3,41	0.00190803166 R$ 2,00	30423.1632 R$ 31.914.585,91

Tabela 3. Comparativo de Gastos. Fonte: O autor.

Acompanhando os valores em Real na Tabela 3, estão os valores em Ether, que representam a quantidade de combustível (*gas*) necessária para execução e inserção dos dados na rede. Na cotação de 3 de junho de 2019, 1 Ether custava aproximadamente R$ 1.049, que foi o valor utilizado de base de cálculo para a relação descrita na tabela.

Segundo dados do Tribunal Superior Eleitoral [2], em 2018, estima-se que cerca de 2 milhões de pessoas atuem como mesários, 4 por seção. Levando em consideração que metade desses mesários são voluntários e que o auxílio alimentação é de até R$ 35, o custo total apenas com mesários seria de R$ 35 milhões. A economia total gerada com o ambiente de testes representa apenas 8,82% desse valor, porém deve-se considerar também os custos gerais com logística, transporte e manutenção das urnas eletrônicas e infraestrutura geral para a ocorrência das eleições gerais brasileiras.

6. Conclusão e trabalhos futuros

Após teste realizado em campo foram efetuados ajustes pontuais no código do contrato e constatada eficácia do sistema proposto (Figura 12).

A criação de um sistema de autenticação de pessoas *online* não foi possível de ser feita de forma segura, portanto, seria necessário um cadastro físico ou alguma verificação das informações cadastradas dos candidatos/eleitores em um banco de dados seguro nacional. Para mais, cada nação possui suas próprias leis e formas de validar quem deve ou não ter um cadastro autenticado em seu sistema eleitoral. Os serviços que operam atualmente com ativos em criptomoedas utilizam um processo de negócio conhecido como

[2]https://g1.globo.com/politica/eleicoes/2018/eleicao-em-numeros/noticia/2018/09/10/cerca-de-2-milhoes-de-mesarios-devem-trabalhar-no-1o-turno-das-eleicoes-2018.ghtml

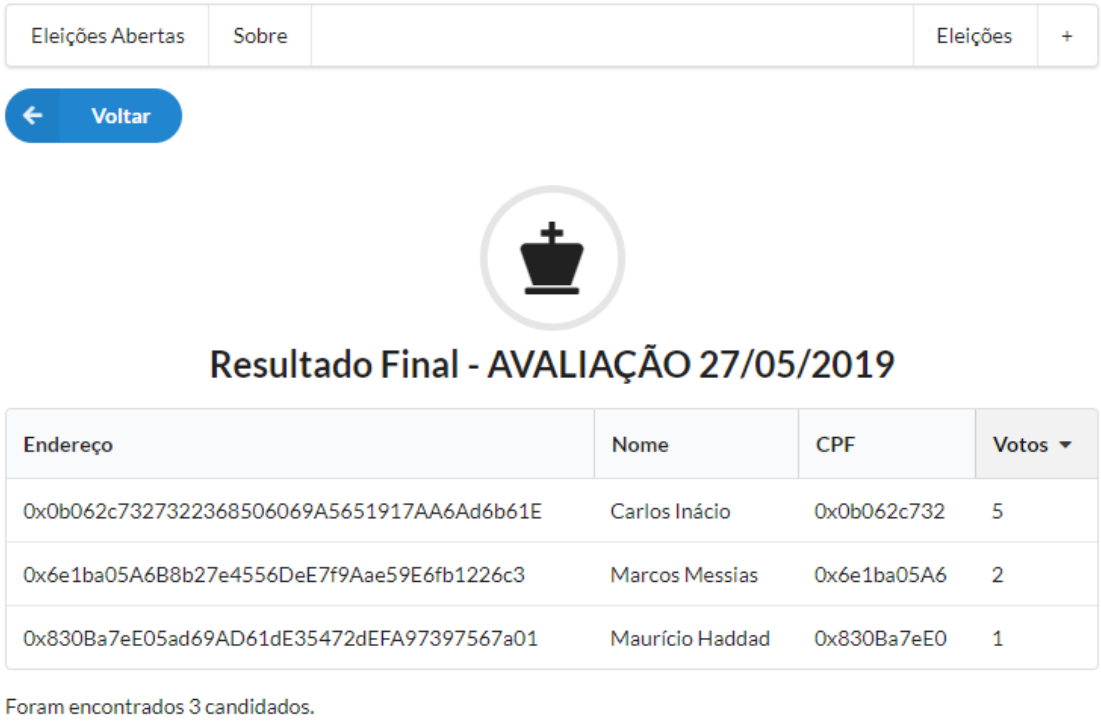

Figura 12. Final da eleição de avaliação. Fonte: O autor.

Know Your Customer, que basicamente é um cadastro que vai determinar se a relação da empresa com o cliente é segura, garantindo seguridade das normas de lavagem de dinheiro e regulamentos bancários.

A lógica que define o ganhador da eleição seleciona apenas um único usuário, caso não haja empate. Como trabalho futuro, deve ser feita a implementação de uma lógica mais robusta que possa permitir a criação de um segundo turno com dois candidatos mais votados.

A tecnologia da Blockchain basicamente é um grande banco de dados distribuído. Entretanto, permite que processos do mundo real que necessitam de transparência, imutabilidade, confiança e integridade dos dados sejam possíveis, graças às aplicações descentralizadas. O problema da fraude ou a demora no resultado de eleições seria facilmente resolvido com implementação Blockchain, representando mais uma grande aplicação para essa tecnologia revolucionária.

Referências

Ayed, A. B. (2017). A conceptual secure blockchain based electronic voting system. *International Journal of Network Security & Its Applications*, 9:01–09.

Baltic, T. (2013). Estonian electronic id – card application specification prerequisites to the smart card differentiation to previous version of esteid card application.

Buterin, V. (2013). Ethereum: The ultimate smart contract and decentralized application platform.

Kubjas, I. (2017). Using blockchain for enabling internet voting.

Lee, K., James, J., Ejeta, T. G., and Kim, H. J. (2016). Electronic voting service using block-chain. *JDFSL*, 11:123–136.

Marchesi, M., Marchesi, L., and Tonelli, R. (2018). An agile software engineering method to design blockchain applications. In *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia*, CEE-SECR '18, pages 3:1–3:8, New York, NY, USA. ACM.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Reid, F. and Harrigan, M. (2011). An analysis of anonymity in the bitcoin system. *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, pages 1318–1326.

Rocha, H. and Ducasse, S. (2018). Preliminary steps towards modeling blockchain oriented software. pages 52–57.

Takabatake, Y., Kotani, D., and Okabe, Y. (2016). An anonymous distributed electronic voting system using zerocoin. *IEICE Technical Report*, 116:127–131.

Zhao, Z. and Hubert Chan, T.-H. (2016). How to vote privately using bitcoin. volume 9543, pages 82–96.