ESTUDO DA APLICAÇÃO DE BLOCKCHAIN, ETHEREUM E SMART CONTRACTS EM SISTEMAS DE VOTAÇÃO

Guilherme Floriani Baron, Aurélio Faustino Hoppe – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

floriani.guilherme@gmail.com, aurelio.hoppe@gmail.com

Resumo: *Apesar de todo o furor causado pelo Bitcoin, em termos tecnológicos o que se sobressai da moeda virtual é a arquitetura conhecida como blockchain. Com intuito de trazer as características que tornam seguras as transações de criptomoedas para um cenário diferente, este artigo apresenta um sistema de votação online utilizando a plataforma Ethereum e smart contracts. Este trabalho também aborda conceitos técnicos pertinentes ao funcionamento de uma rede de blockchain e detalha o desenvolvimento de um contrato inteligente capaz de gerenciar um processo de votação online. De maneira geral, o sistema mostrou-se promissor para votações relativamente pequenas, porém a sua utilização em cenários de larga escala como as eleições de um país deve ser abordada com cautela, já que muitos outros fatores devem ser levados em consideração, incluindo performance, escalabilidade e custo.*

Palavras-chave*: Votação. Blockchain. Smart contracts. Ethereum.*

1 INTRODUÇÃO

Em um momento onde o mercado financeiro mundial passava pela maior crise desde a virada do século, o Bitcoin surgia com uma proposta ousada que poderia mudar a economia como conhecíamos até então. Segundo Ferreira (2017), mesmo que a ideia de criar uma moeda virtual totalmente independente do sistema monetário existente fosse inovadora por si só, a arquitetura conhecida como blockchain utilizada por Satoshi Nakamoto para fazer sua implementação teria tanto potencial quanto para causar grande impacto tecnológico na sociedade. Arquitetura esta, que foi a chave para o sucesso do Bitcoin, tornando suas transações seguras e transparentes sem depender de qualquer instituição financeira.

Swan (2015, p. 10) descreve o blockchain como livro-razão que registra cada uma das transações realizadas com a criptomoeda. Essas transações são persistidas em blocos e novos blocos vão sendo criados e adicionados constantemente ao blockchain de forma linear e cronológica. O blockchain é visto como principal inovação tecnológica no Bitcoin porque cria um mecanismo de prova confiável de todas as transações da rede. Ao realizar uma transação, os usuários podem confiar no livro-razão armazenado mundialmente em nós descentralizados mantidos por "contadores-minerados", o oposto de ter que criar e manter confiança em uma contraparte (outra pessoa) ou terceira parte (como um banco) (SWAN, 2015, p. 10). Desta forma, qualquer transação que faça uma mudança nos dados armazenados no blockchain deve passar por todos os nós participantes da rede, necessitando ser aceita pelo consenso geral, o que torna a fraude destas informações pouco provável.

Ainda que o conceito da arquitetura pareça específico para o controle de uma moeda virtual, segundo Koç et al. (2018), o blockchain também tem potencial para resolver diversos outros problemas existentes na sociedade, pois a arquitetura torna muito difícil a fraude ou manipulação dos dados armazenados, dados estes que podem ser muito mais complexos que apenas os saldos dos usuários. Algumas das aplicações que surgiram vão além do mercado financeiro, como emissão de certificados acadêmicos, controle de direitos autorais, armazenamento de dados médicos, controle da procedência de alimentos, além de aplicações no setor público, como registro único de cidadãos e sistemas de votação para fins democráticos. Dentre as aplicações citadas, o cenário democrático sempre pareceu um dos mais promissores e interessantes para validar a aplicabilidade do blockchain. Segundo uma pesquisa realizada pela Avast (2018), 91,84% dos brasileiros acreditam que o sistema eletrônico de votação possa ser violado, o que demonstra a insegurança dos usuários com a forma como as votações são realizadas, diante de tamanha a importância de seus resultados. Conforme aponta a pesquisa, mesmo no Brasil, onde utiliza-se a urna eletrônica, os eleitores têm receio de acreditar nos resultados pois não há nada que comprove efetivamente a veracidade dos mesmos.

Levando estas informações em consideração, este trabalho apresenta a proposta de um sistema online de votação utilizando uma arquitetura de blockchain como base. Inicialmente a abordagem tem como objetivo gerenciar processos de votação em escala relativamente pequena, o que deve proporcionar uma perspectiva real sobre a efetividade da utilização da arquitetura na solução de problemas reais e identificar as limitações que devem ser observadas na implementação de uma solução que atenda cenários de larga escala como as eleições de um país.

A próxima seção apresenta a fundamentação teórica em que se baseia o artigo, explicando os principais conceitos envolvidos no ecossistema de uma rede de blockchain. Já no terceiro capítulo é descrito a arquitetura da solução proposta, assim como detalhes do *smart contract* desenvolvido e recursos utilizados. Em seguida, o quarto capítulo apresenta os

resultados dos testes realizados, além de uma análise de performance e custo da aplicação. Por fim, são apresentadas as conclusões e limitações levantadas a partir do estudo realizado, assim como sugestões de extensão para futuros trabalhos.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção expõe os conceitos mais importantes relacionados ao blockchain e sua arquitetura, assim como detalhes referentes a sua utilização no Bitcoin e os contrastes com a Ethereum. Além disso são apresentados os mecanismos utilizados pela Ethereum para permitir a criação de aplicações baseadas em blockchain dentro da sua plataforma, os *smart contracts*. Por fim, são apresentados os trabalhos correlatos.

2.1 BLOCKCHAIN

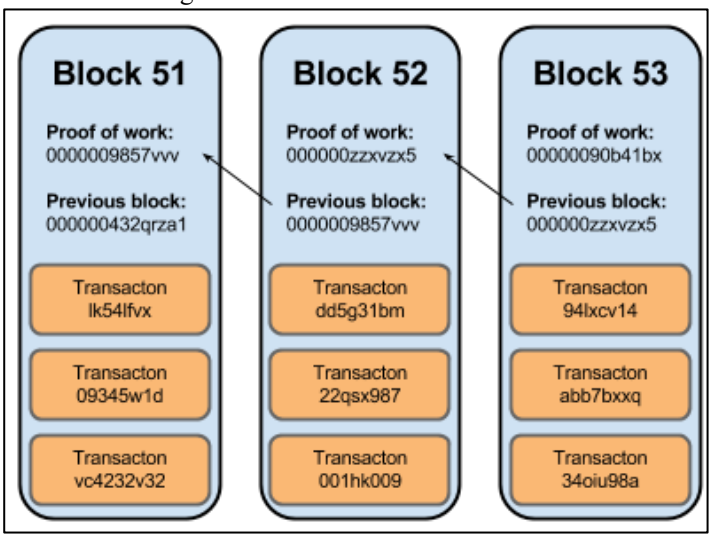
Segundo Antonopoulos (2014), o Bitcoin consiste em um conjunto de conceitos e tecnologias que formam a base para o ecossistema de uma moeda virtual segura ou criptomoeda como é popularmente conhecida. Moeda essa, que é chamada de bitcoin, armazenada em carteiras digitais e transmitida pela internet através de um protocolo próprio entre seus usuários. Ao contrário de outras moedas, o bitcoin não possui forma física e é armazenado somente em um registro virtual de transações confiável conhecido como blockchain (ANTONOPOULOS, 2014).

Ainda que o termo blockchain fosse desconhecido antes da proposta do Bitcoin por Satoshi Nakamoto em 2008, na década de 90, Dave Bayer, W. Scott Stornetta e Stuart Haber já vinham discutindo uma solução que fazia uso dos mesmos princípios para aumentar a confiabilidade de documentos digitais. A ideia consistia basicamente em usar funções de *hash* criptográficas para garantir não apenas a ordem das informações do documento, mas também que informações anteriores não fossem adulteradas (FERREIRA, 2017). Porém somente após a aplicação bem-sucedida no Bitcoin, o termo blockchain tornou-se amplamente popular, fazendo com que diversas criptomoedas alternativas surgissem, assim como muitas pesquisas relacionadas a sua utilização em outros cenários que vão muito além do mercado financeiro (KOÇ et al., 2018).

O conceito do termo blockchain é amplo e relativamente novo, sendo assim é possível encontrar diferentes definições para o mesmo. Analisando-o a partir de uma perspectiva técnica, Faour (2018) o descreve como uma base de dados descentralizada funcionando sobre uma rede *peer to peer*. Já de um ponto de vista mais superficial, Swan (2015) define a arquitetura, no contexto das criptomoedas, como livro-razão que registra todas as transações ocorridas entre os participantes.

De acordo com Ferreira (2017), a arquitetura proposta pelo criador do Bitcoin é composta por uma rede que utiliza os mesmos princípios do que conhecemos por *peer to peer*, onde cada nó possui uma cópia completa do blockchain, que em termos de estrutura de dados, é como o próprio nome sugere, uma lista encadeada de blocos. Cada bloco possui um *header*, que armazena informações importantes como tamanho do bloco, momento de criação, o seu *hash* de identificação assim como do seu antecessor e um conjunto de dados, que normalmente se refere a uma lista de transações como é possível observar na Figura 1 (FERREIRA, 2017).

Figura 1 – Lista encadeada de blocos



Fonte: Faggart (2015).

Segundo Antonopoulos (2014), a criação de cada novo bloco se dá pelo processo popularmente conhecido como **mineração**, onde os vários nós participantes da rede competem pelo direito de adicionar o novo bloco a cadeia e ganhar uma pequena bonificação por seu trabalho. Durante este processo, o minerador precisa utilizar um algoritmo conhecido

como ***proof of work*** para encontrar o *hash* de identificação do bloco e assim poder submetê-lo a aprovação dos outros nós através do **mecanismo de consenso (**ANTONOPOULOS, 2014).

O processo de mineração e tudo que ele engloba é um dos pontos fundamentais para que o Bitcoin seja considerado seguro, evitando transações fraudulentas ou inválidas. Segundo Swan (2015), os mineradores fornecem seu poder de processamento computacional para a rede do Bitcoin, em troca da possibilidade de eventualmente serem recompensados com novas unidades da criptomoeda. Antonopoulos (2014) menciona que esse mecanismo é chamado assim pois as recompensas diminuem com o tempo, da mesa forma que geralmente acontece com a mineração de metais preciosos.

Segundo Antonopoulos (2014), sempre que uma transação de bitcoin é realizada entre duas carteiras, ela é transmitida para todos os nós mineradores da rede, que por sua vez a adicionam em um *pool* de transações pendentes. Cada minerador ordena estas transações considerando prioridade e tempo de espera, para então agrupar um conjunto delas em um novo bloco. Uma vez que se tenha organizado as transações e gerado um bloco no formato esperado, o próximo passo é encontrar um *hash* de identificação para o bloco através do algoritmo de *proof of work* **(**ANTONOPOULOS, 2014).

Diferentemente das moedas comuns que podem ser impressas ilimitadamente gerando inflação monetária, o bitcoin foi projetado para que apenas uma determinada quantidade de unidades da moeda seja "minerada" a cada intervalo de tempo, que gira em torno de 10 minutos **(**ANTONOPOULOS, 2014). Ainda segundo o autor, independentemente da quantidade de nós ou poder computacional empregado na mineração, o tempo de geração de um novo bloco e consequentemente de novos bitcoins será aproximadamente este, isto ocorre porque a própria rede regula a dificuldade necessária para que a saída da função de *hash* utilizada no algoritmo de *proof of work* seja um valor válido.

Segundo Ferreira (2017), pode-se definir as funções de *hash* como uma função que aceita um valor de tamanho indefinido e retorna um valor de tamanho fixo, que passamos a chamar de *hash*. No caso do algoritmo de *proof of work* do Bitcoin, a função utilizada é a SHA-256, que gera um *hash* de 256 bits **(**ANTONOPOULOS, 2014). Uma propriedade importante deste tipo de função é que ela sempre irá apresentar a mesma saída para uma mesma entrada. No campo da criptografia estas funções são geralmente de rápida execução e não permitem que se possa deduzir o valor do parâmetro de entrada conhecendo apenas o *hash* resultante e a própria função (FERREIRA, 2017).

Em linhas gerais, o algoritmo de *proof of work* consiste em gerar um *hash* para o *header* do novo bloco repetidas vezes até que a saída seja uma sequência de caracteres alfanumericamente menores que um valor estipulado pela rede, conhecido como *target* **(**ANTONOPOULOS, 2014). Conforme mencionado anteriormente, uma função de *hash* criptográfica sempre gera a mesma saída para uma mesma entrada, portanto para que o *hash* gerado seja diferente da tentativa anterior, um valor do *header* do bloco é alterado, este valor é chamado de *nonce*. Ou seja, em sua maioria, o processo de mineração consiste em encontrar um *nonce* que resulte em um *hash* menor que o *target* definido pela rede, e como o resultado de uma função de *hash* criptográfica é imprevisível, a única forma existente é através de tentativa e erro **(**ANTONOPOULOS, 2014). Segundo Antonopoulos (2014), a dificuldade para encontrar um valor de *nonce* que atenda o *target* estipulado é definida pela própria rede regularmente usando a data de geração dos blocos para encontrar um nível de dificuldade que faça com que somente 2016 blocos sejam criados a cada 2 semanas, ou um bloco a cada 10 minutos.

De acordo com Antonopoulos (2014), o primeiro minerador que encontrar um *hash* válido segundo as especificações da rede e publicá-lo será o vencedor e como recompensa terá o direito de incluir no bloco uma transação conhecida como *generation transaction* ou *coinbase transaction*, que transfere uma certa quantidade de bitcoins para sua própria carteira. Porém, para que isso aconteça, seu novo bloco contendo o *nonce* calculado deve ser validado e aceito por mais da metade dos nós da rede, o que é conhecido como mecanismo de consenso **(**ANTONOPOULOS, 2014).

Segundo Antonopoulos (2014), quando um novo bloco é minerado, os outros nós realizam uma série de validações antes de propagá-lo para garantir que o bloco gerado é válido e atende a uma série de requisitos como:

a) estrutura sintática do bloco está correta;
b) o *hash* gerado é alfanumericamente menor que o *target* estipulado (*proof of work*);
c) o tamanho do bloco está dentro do limite;
d) a primeira transação é uma *generation transaction*, e somente a primeira;
e) todas as transações incluídas no bloco são válidas.

Após validar e propagar o bloco pela rede, os nós que recebem o novo bloco param de fazer a mineração do mesmo pois isto significa que outro nó conseguiu faze-la antes, iniciando então a geração do próximo bloco. Segundo Antonopoulos (2014), neste momento as transações incluídas pelo minerador do bloco são efetivadas no blockchain.

2.2 ETHEREUM

Todo o furor causado pelo Bitcoin fez com que eventualmente a utilização de blockchain na solução de outros problemas fosse discutida. Porém, segundo Koç et al. (2018) a rede criada por Nakamoto não foi projetada com o intuito

de suportar outras aplicações, tendo assim poucos recursos para isso. Diante disto algumas alternativas de plataforma surgiram, dentre elas a que mais se destacou foi a Ethereum.

A Ethereum é uma plataforma de computação distribuída que usa blockchain para armazenar não somente o estado das contas de seus usuários, mas também código fonte e seu estado associado (SCHÜPFER, 2017). Sendo proposta em 2013 por Vitalik Buterin, um antigo programador envolvido no projeto do Bitcoin, a plataforma tem como objetivo permitir que qualquer um possa criar e executar aplicações distribuídas baseadas em blockchain, tendo assim acesso a todas as características inerentes a arquitetura, porém sem a complexidade de criar sua própria rede (FERREIRA, 2017).

Segundo Schüpfer (2017), no contexto da Ethereum, as **contas** são objetos que podem armazenar o balanço de um usuário ou o próprio estado de um *smart contract*. A primeira é chamada de Externally Owned Account (EOA) ou conta externa, pois pertence a um usuário externo, permitindo a ele enviar mensagens e realizar transações assinadas com sua chave privada. Já a segunda é chamada de *contract account* e seu estado é controlado pelo código fonte de um contrato (SCHÜPFER, 2017). De acordo com Schüpfer (2017), sempre que uma conta contrato recebe uma mensagem, seu código fonte é executado e tem permissão para enviar mensagens a outros contratos ou realizar transações com outras contas.

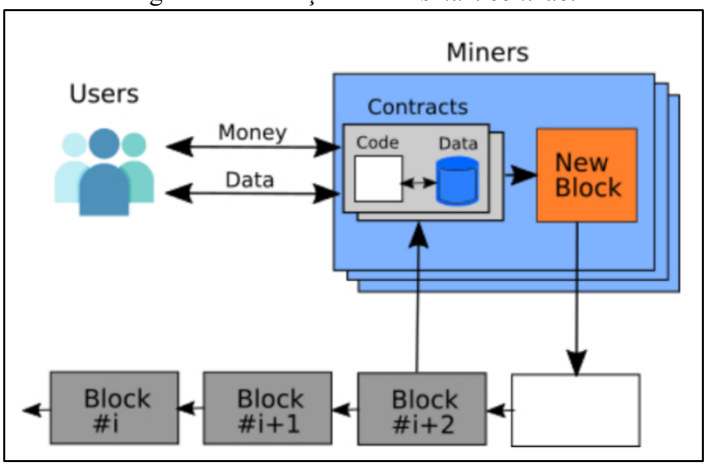
Da mesma forma que no Bitcoin, as **transações** na Ethereum realizam a transferência de uma quantidade de *ether* de uma conta para outra, mas que além disso podem criar novos contratos ou executar funções de um já existente (SCHÜPFER, 2017). Segundo Schüpfer (2017), elas são um pacote de dados contendo informações do remetente, destinatário, quantidade de *ether*, limite e preço do *gas*, além de um campo opcional de dados. Além disso, há a possibilidade de que um contrato envie uma **mensagem** para outro invocando uma de suas funções, o que funciona de forma muito parecida com a chamada de uma função em uma linguagem de programação de alto nível (SCHÜPFER, 2017).

A unidade de moeda corrente na Ethereum é o ***ether***, de acordo com Schüpfer (2017), além de ser movimentada através das transações, ela também é utilizada como taxa de pagamento para a criação ou execução de algum contrato. Assim como o bitcoin, o *ether* pode ser obtido em casas de câmbio de criptomoedas, conhecidas como *cryptocurrency exchanges*, além de também poder ser divido em partes menores, sendo a mais comum delas o *gwei*, que equivale a 10^9 *ether* (SCHÜPFER, 2017).

Com intuito de evitar ataques de negação de serviço assim como outros ataques de *spam*, a Ethereum utiliza um mecanismo chamado de ***gas*** (abreviação de gasolina em inglês). Essa unidade é usada internamente pela plataforma e não é propriamente uma moeda (FERREIRA, 2017). Ainda segundo o autor, cada unidade de *gas* representa uma computação realizada por um contrato, ou seja, quanto mais complexo seu código fonte, maior será a quantidade de *gas* necessária para executá-lo. De acordo com Ferreira (2017), o próprio usuário que está executando o contrato decide quanto ele está disposto a pagar em ether por cada unidade de *gas*, além do limite que ele pretende gastar, isto torna os ataques inviáveis, pois eles se tornariam muito caros caso o valor pago pelo *gas* fosse alto demais ou nem mesmo seriam executados caso fosse baixo demais.

De acordo com Koç et al. (2018) a Ethereum se distingue do Bitcoin principalmente por fazer uso do conceito de ***smart contracts***. Conforme mencionado anteriormente, estes contratos inteligentes são um tipo especial de conta, que além do balanço em ether, armazena também código fonte e seu estado associado (SCHÜPFER, 2017). Segundo Schüpfer (2017), outros usuários interagem com o *smart contract* enviando *ether* através de transações ou executando alguma função como é possível observar na Figura 2. Informações como nome da função a ser executada e os parâmetros passados são enviados através do campo de dados da transação mencionado anteriormente (SCHÜPFER, 2017).

Figura 2 – Interação com o *smart contract*



Fonte: Schüpfer (2017).

Da mesma forma que o blockchain, a ideia de criar contratos virtuais automatizados também não é completamente nova, o conceito de *smart contracts* foi introduzido originalmente por Nick Szabo em *Building Blocks for Digital Markets* no ano de 1996:

> Eu chamo esses novos contratos de "inteligentes" porque são muito mais funcionais que seus inanimados ancestrais de papel. O uso de inteligência artificial não é necessário. Um contrato inteligente é um conjunto de promessas, especificadas de forma digital, incluindo protocolos executados pelas partes quando essas promessas são cumpridas (SZABO, 1996, tradução nossa).

De acordo com Ferreira (2017), ainda que o Bitcoin já permitisse que as unidades da moeda fossem gerenciadas por *scripts* automatizados, a linguagem baseada em pilhas utilizada para criação dos mesmos tem recursos muito limitados. Esse era um dos principais argumentos nas sugestões de melhoria propostas por Buterin durante sua participação no projeto do Bitcoin, porém suas ideias nunca tiveram muito apoio. Os contratos inteligentes da Ethereum por sua vez têm seu código fonte compilado para executar em uma máquina virtual conhecida como Ethereum Virtual Machine (EVM). Isto permite que eles sejam desenvolvidos utilizando linguagens de alto nível, dentre elas a mais utilizada chama-se Solidity (FERREIRA, 2017).

A Solidity é uma linguagem de alto nível orientada a contratos com influência de outras linguagens como Python, C++ e Javascript. Além de usar tipos estáticos ela suporta herança, bibliotecas e tipos mais complexos definidos pelo próprio usuário. No Quadro 1 é possível observar o código fonte de um simples *smart contract* escrito em Solidity para gerenciar uma moeda virtual.

Quadro 1 – Exemplo de *smart contract* escrito em Solidity

```
contract Moeda {
    // variáveis de estado do contrato
    address public admin;
    mapping (address => uint) public saldos;

    // constructor define o criador do contrato como admin
    function Moeda() public {
        admin = msg.sender;
    }

    // admin pode gerar novas moedas
    function gerarMoedas(address destinatario, uint valor){
        if (msg.sender != admin) return;
        saldos[destinatario] += valor;
    }

    // transfere um montante para outro usuário
    function transferir(address destinatario, uint valor){
        if (saldos[msg.sender] < valor) return;
        saldos[msg.sender] -= valor;
        saldos[destinatario] += valor;
    }
}
```

Fonte: elaborado pelo autor.

Segundo Schüpfer (2017), o processo de mineração na Ethereum é muito similar ao do Bitcoin, com a diferença de que os blocos não armazenam somente uma lista de transações e sim o estado de todos os contratos da rede, que se refere as variáveis públicas de cada um, como é possível observar no Quadro 1. Outra grande diferença é que a Ethereum utiliza outro algoritmo de *proof of work* chamado de *ethash*, o que faz com que seus blocos sejam criados a cada 12 segundos em média, que é relativamente menor comparado aos 10 minutos do Bitcoin (SCHÜPFER, 2017). De acordo com Schüpfer (2017), isto faz com que as transações sejam efetivadas mais rapidamente no blockchain, porém também faz com que muitas vezes múltiplos mineradores criem o novo bloco simultaneamente, sendo então necessário que a plataforma gerencie a forma de compensação pelo trabalho de cada um.

De acordo com Koç et al. (2018), todo o processo de mineração e consenso é abstraído pela própria plataforma, de forma que os desenvolvedores somente tenham que se preocupar em desenvolver as regras do seu contrato e estarem dispostos a despender o *ether* necessário para executá-los. Diante disto os *smart contracts* e a própria plataforma da Ethereum se mostraram um grande recurso na criação de aplicações baseadas em blockchain, tornando acessíveis todas as vantagens e características inerentes a arquitetura sem a necessidade de gerenciar toda a sua complexidade.

2.3 TRABALHOS CORRELATOS

Em seguida são descritos trabalhos que possuem conteúdo relacionado ao tema proposto e podem auxiliar na compreensão dos conceitos relacionados a blockchain e suas aplicações. O Quadro 2 apresenta o trabalho de Faour (2018), onde o autor descreve uma solução descentralizada para um sistema de votação utilizando blockchain. Já no Quadro 3, é descrito o trabalho de Water (2017), onde o mesmo apresenta algumas abordagens para solução do problema utilizando

uma arquitetura de blockchain, tendo como cenário as eleições de seu país, a Nova Zelândia. Por fim, o Quadro 4 apresenta o trabalho de Koç et al. (2018), onde os autores propõem a implementação de um sistema online de votação utilizando *smart contracts* na Ethereum.

Quadro 2 – Transparent Voting Platform Based on Permissioned Blockchain

Referência	Faour (2018)
Objetivos	O autor apresenta a proposta de uma aplicação que utiliza uma plataforma de blockchain para atender cenários de votação pequenos principalmente no meio corporativo, como por exemplo a aprovação da união de duas empresas ou novos investimentos, escolha de novos diretores ou ainda aprovação de planos de remuneração.
Principais funcionalidades	Alguns requisitos que o autor visa atender incluem a capacidade do usuário realizar seu voto anonimamente, além de poder verificá-lo posteriormente, assim como permitir que os resultados da votação possam ser conferidos.
Ferramentas de desenvolvimento	O autor utilizou uma plataforma descentralizada e open source para aplicações blockchain chamada de Waves, onde cada unidade da moeda chamada de *wave* seria contabilizada como um voto. De acordo com Faour, o processo ocorre através da transferência de valores entre os usuários votantes e os candidatos.
Resultados e conclusões	A aplicação desenvolvida visa atender processos de votação relativamente pequenos, mas mostra a efetividade da utilização do blockchain para resolver problemas de transações seguras entre usuários sem a utilização de mediadores de acordo com o autor.

Fonte: elaborado pelo autor.

Quadro 3 – Blockchain Ballot: Electoral Enhancement or Danger to Democracy?

Referência	Water (2017)
Objetivos	O autor tem como objetivo analisar quais são as possíveis abordagens utilizando blockchain para implementar um sistema eleitoral a nível nacional no seu país de origem, a Nova Zelândia, apontando assim quais seriam os impactos e problemas de cada uma delas.
Principais funcionalidades	Water discorre inicialmente sobre alguns fatores que segundo ele são imprescindíveis para que eleições sejam realizadas de forma livre e justa. Sendo que os principais deles estão relacionados a capacidade do usuário realizar seu voto de forma anônima, assim como não permitir que nenhum usuário possa acompanhar a contabilização dos votos antes de efetivar seu voto, o que segundo o autor pode comprovadamente influenciar a sua escolha.
Ferramentas de desenvolvimento	Visando atender os requisitos para uma eleição livre e justa apontados inicialmente pelo autor, ele divide a proposta em duas abordagens. Na primeira ele descreve uma implementação utilizando a rede de blockchain pública da Ethereum e *smart contracts*, o que fere o requisito de anonimato e contagem dos votos, pois qualquer participante poderia consultá-los já que o blockchain é público. Na segunda abordagem o autor utiliza uma plataforma *open source* de redes blockchain privadas, a Hyperledger Fabric, que também permite a criação de uma espécie de contrato chamado de *chaincode*. Esta por sua vez resolve o problema da contagem de votos e anonimato dos usuários, mas traz de volta um problema solucionado pelo blockchain que é a necessidade de os usuários confiarem em uma instituição terceira para gerenciar todo o processo sem que possam auditar o que está acontecendo. Ao fim o autor propõe uma solução que mistura alguns elementos utilizados pelas duas abordagens anteriores, onde utilizando uma rede de blockchain pública, funções de *hash* e chaves assimétricas ele consegue garantir que os votos sejam anônimos. Porém esta abordagem faria com que os usuários tivessem que comparecer a sessão de votação duas vezes, uma para efetivar seu voto criptografado e outra para revelar o mesmo depois que todos os outros usuários já tivessem completado a primeira etapa.
Resultados e conclusões	Segundo o autor, nenhuma das abordagens realizadas consegue atender todos os requisitos de forma completa e eficiente, além de que considerando o tempo de criação de cada bloco no blockchain, as eleições em uma país como a Nova Zelândia atualmente iriam necessitar de dois dias para serem registradas por completo. Water aponta que uma nova abordagem ou reformulação do processo de votação em si aliado a provável evolução da arquitetura do blockchain podem fazer com que no futuro eleições possam ser realizadas utilizando esta tecnologia como base. Porém atualmente ainda não é possível realizar todo o processo de votação e apuração das eleições de um país utilizando a plataforma segundo o autor.

Fonte: elaborado pelo autor.

Quadro 4 – Towards Secure E-Voting Using Ethereum Blockchain

Referência	Koç et al. (2018)
Objetivos	A publicação tem como objetivo propor um *smart contract* para um sistema de votação online utilizando a rede de blockchain pública da Ethereum.

Principais funcionalidades	Os autores apontam que o contrato desenvolvido visa atender processos de votação considerando requisitos como autenticidade e transparência, permitindo que os usuários criem propostas e votem nas existentes.
Ferramentas de desenvolvimento	Segundo os autores, ainda que o Bitcoin permitisse criar scripts para transações, eles são muito limitados como mencionamos anteriormente, sendo assim eles utilizaram a plataforma da Ethereum, o que os permitiu fazer uso de todo o potencial e versatilidade dos *smart contracts*.
Resultados e conclusões	O contrato desenvolvido é capaz de gerenciar processos de votação simples, porém segundo os autores, para que algo dessa natureza seja utilizado em um cenário de larga escala, problemas como escalabilidade e autenticação a nível pessoal utilizando biometria ou algo do tipo devem ser levados em consideração. Ainda de acordo com os autores, em um mundo onde a tendência é que cada vez mais aparelhos do nosso dia a dia estejam conectados a internet, recursos como o blockchain e sua arquitetura descentralizada possuem potencial para permitir a criação de aplicações complexas que podem resolver muitos dos problemas da nossa sociedade.

Fonte: elaborado pelo autor.

3 DESCRIÇÃO DA APLICAÇÃO

Sendo o principal objetivo deste artigo a implementação em escala relativamente pequena de um sistema online de votação utilizando uma arquitetura de blockchain, o foco desta seção é descrever a especificação realizada, assim como os aspectos técnicos relevantes e abordagem utilizada durante o desenvolvimento da solução.

3.1 ESPECIFICAÇÃO

Para que a aplicação possa gerenciar completamente um processo de votação online e seja possível analisar a viabilidade da utilização de uma arquitetura de blockchain. Para isto, a proposta deve contemplar os requisitos descritos a seguir, assim como as funcionalidades descritas no diagrama de caso de uso apresentado na Figura 3. Os requisitos funcionais (RF) a serem atendidos e sua relação com cada uma das funcionalidades mencionadas no caso de uso são apresentados no Quadro 5. Além disso, também são apresentados os requisitos não funcionais (RNF) no Quadro 6.

Quadro 5 - Matriz de rastreabilidade de RF e UC

Requisito funcional	**Caso de uso**
RF01: permitir o cadastro de votações	UC01
RF02: gerar um *token* para cada usuário apto a votar	UC01
RF03: permitir o voto somente de usuários aptos a votar	UC02
RF04: permitir somente um voto por usuário	UC02
RF05: permitir encerrar a votação	UC03
RF06: permitir visualizar as votações existentes	UC04
RF07: permitir a apuração dos votos	UC05

Fonte: elaborado pelo autor.

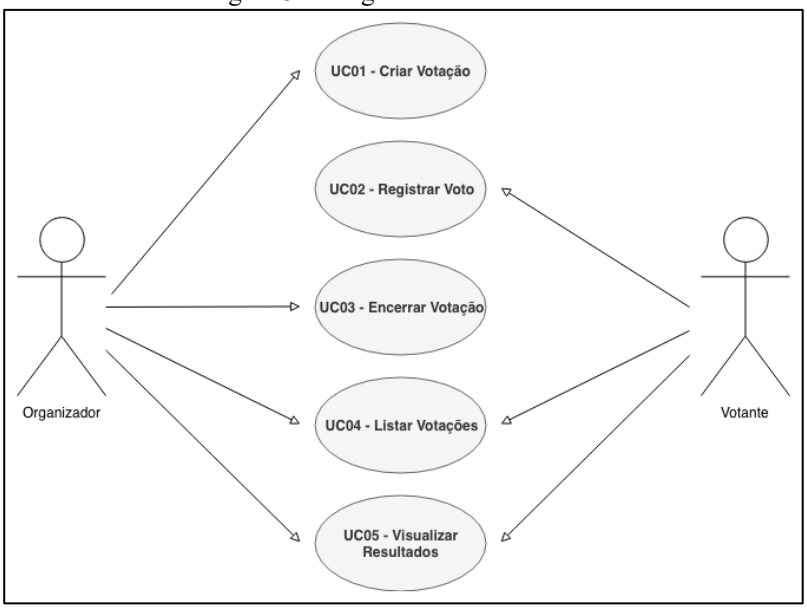
Quadro 6 - Requisitos não funcionais (RNF)

Requisito não funcional
RNF01: utilizar a plataforma de blockchain da Ethereum
RNF02: escrever o *smart contract* utilizando a linguagem Solidity
RNF03: a aplicação web deve rodar em qualquer browser que suporte Javascript

Fonte: elaborado pelo autor.

O diagrama de caso de uso apresentado na Figura 3, introduz os dois principais atores identificados durante a especificação da solução, sendo eles o `Organizador` da votação e o `Usuário` votante. O primeiro é responsável por criar a votação (`UC01`) e distribuir os *tokens* gerados para os usuários votantes, permitindo assim que eles possam registrar individualmente seus votos (`UC02`) antes que a votação seja encerrada (`UC03`). Ambos usuários têm acesso a lista de votações existentes (`UC04`) e podem visualizar os resultados (`UC05`) de cada uma assim que estiver encerrada.

Figura 3 – Diagrama de Caso de Uso

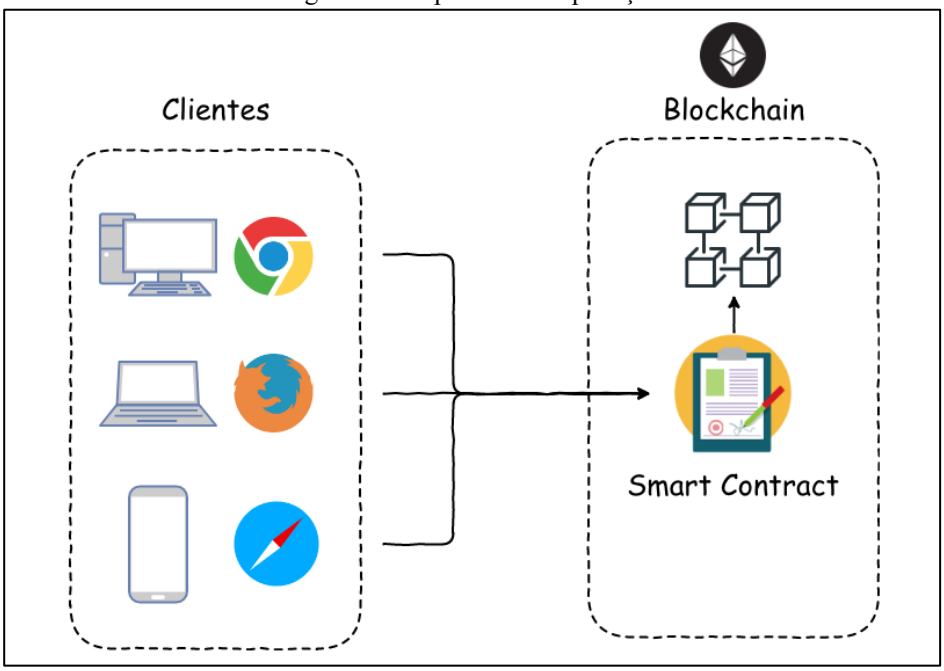


Fonte: elaborado pelo autor.

3.2 ARQUITETURA

Assim como em uma aplicação web cliente-servidor, pode-se dividir a arquitetura da solução proposta em duas grandes partes conforme é possível observar na Figura 4. A primeira é o *front-end* ou interface, onde os clientes ou usuários, através de seus *browsers* executam uma aplicação web capaz de comunicar-se com o blockchain da Ethereum para executar as funções disponíveis no *smart contract*. Este por sua vez, compõe a segunda parte da arquitetura e faz o papel de servidor, armazenando e gerenciando as informações das votações no blockchain, eliminando assim a necessidade de haver qualquer outra estrutura no *back-end*.

Figura 4 – Arquitetura da aplicação



Fonte: elaborado pelo autor.

Nas próximas seções detalha-se melhor as duas partes, onde a primeira apresenta o desenvolvimento do contrato como um todo, assim como características técnicas pertinentes ao seu desenvolvimento usando a linguagem Solidity. Já a seção seguinte descreve a interface da solução, assim como apresenta detalhes referentes a comunicação entre as páginas web e o *smart contract* publicado na Ethereum.

3.3 CONTRATO

A utilização da Ethereum traz uma série de recursos que ajudam no desenvolvimento de aplicações baseadas em blockchain, um deles é a possibilidade de criar todas as regras de negócio desenvolvendo apenas um *smart contract*, o que em uma aplicação comum normalmente dependeria de um *web server* ou arquitetura de *back-end* mais complexa. Fazendo uso desse recurso, todas as regras necessárias para a implementação da solução proposta foram escritas em um contrato usando a linguagem Solidity.

Conforme apresentado nas seções anteriores, para que os nós mineradores da Ethereum não desperdicem poder de processamento com a execução de contratos que não tenham valor real para alguma aplicação, a plataforma usa o mecanismo de *gas*, onde o usuário ou aplicação executando um *smart contract* deve pagar por isto. Sendo assim, durante a etapa de desenvolvimento e testes da aplicação foi utilizado um utilitário chamado de Ganache que cria uma rede de blockchain local simulando o mesmo comportamento da rede oficial da Ethereum.

Alguns fatores precisam ser levados em consideração para que a aplicação seja capaz de controlar completamente um processo de votação online, um dos mais importantes é a **integridade**, o que implica na garantia de que nenhuma informação seja manipulada ou alterada de outra forma, que não seja através da própria aplicação. Sendo assim, as informações relacionadas aos processos de votação já realizados e principalmente aos que estão em andamento devem ser persistidas no blockchain, assim como quais propostas estão em votação e o número de votos recebido por cada uma delas até o momento. Conforme mencionado anteriormente, para que o blockchain mantenha todas estas informações na cadeia de blocos é necessário que elas estejam armazenadas nas variáveis de estado do *smart contract*, o que pode ser feito utilizando estruturas de dados complexas, suportadas pela linguagem Solidity através da palavra-chave `struct` como é possível observar no Quadro 7.

Quadro 7 – Definição de estruturas de dados e estado do contrato

```
contract Voting {
    struct Votante {
        bool votou;
        bool valido;
    }

    struct Proposta {
        bytes32 descricao;
        uint contadorVotos;
    }

    struct Votacao {
        bool encerrada;
        string descricao;
        mapping (uint => Proposta) propostas;
        mapping (bytes32 => Votante) votantes;
    }

    // estado do contrato armazena uma lista de votações
    mapping (uint => Votacao) internal _votacoes;

    // ... demais funções
}
```

Fonte: elaborado pelo autor.

Como é possível observar no Quadro 7, cada usuário apto a votar é identificado por uma *string* de 32 *bytes* chamada de *token*. Este *token* é um identificador único gerado pela interface da aplicação utilizando uma biblioteca Javascript chamada *uuid-token-generator* e disponibilizado para o organizador da votação no momento da sua criação. É de responsabilidade do organizador distribuir os tokens de forma segura entre os usuários votantes. Vale ressaltar que esta etapa é um ponto de falha conhecido do processo, pois qualquer usuário que de alguma forma obter o *token* de outro poderá realizar o voto no seu lugar. Porém, considera-se esta etapa um subproblema do assunto abordado neste artigo, portanto está fora do escopo proposto.

Para que a lista de votações e seus valores se mantenham íntegros, a única forma de alterá-los é através das funções definidas no próprio contrato. Assim como é possível observar no trecho de código apresentado no Quadro 8, a criação de uma nova votação é feita utilizando a função `criarVotacao`, que recebe como parâmetros, sua descrição, propostas e quais os *tokens* gerados para os usuários. Este tipo de função que faz alterações no estado do contrato gera uma nova transação no blockchain da Ethereum, fazendo com que essa mudança seja incluída na cadeia de blocos principal durante o processo de mineração do próximo bloco.

Quadro 8 – Função responsável por criar a votação

```
function criarVotacao(string descricao, bytes32[] propostas, bytes32[] tokens)
public {
    require(propostas.length > 1, "Pelo menos duas propostas são necessárias");
    require(tokens.length > 1, "Pelo menos dois votantes são necessários");

    uint index = // ... lógica para encontrar um índice disponível
    _votacoes[index] = Votacao(false, descricao, propostas.length);

    for (uint i = 0; i < propostas.length; i++) {
        _votacoes[index].propostas[i] = Proposta(propostas[i], 0);
    }
    for (uint j = 0; j < tokens.length; j++) {
        _votacoes[index].votantes[tokens[j]] = Votante(false, true);
    }
}
```

Fonte: elaborado pelo autor.

A função `votar` apresentada no Quadro 9 demonstra como um voto é realizado apenas incrementando o valor do atributo `contadorVotos` da proposta escolhida. Nesta etapa o fator **elegibilidade** deve ser considerado, pois somente usuários aptos a votar devem poder realiza-lo. Assim como na função responsável por criar a votação apresentada anteriormente, a função `votar` realiza algumas validações utilizando a palavra-chave `require` para não permitir que um voto inválido seja realizado. Como é possível observar no Quadro 9, as validações realizadas pela função votar não permitem que uma votação encerrada recebe novos votos, assim como também não permitem que um usuário possuindo um token inexistente ou que já tenha votado, vote novamente.

Quadro 9 – Função responsável por realizar o voto

```
function votar(bytes32 token, uint indexVotacao, uint indexProposta)
public {
    Votacao storage votacao = _votacoes[indexVotacao];
    Votante storage votante = votacao.votantes[token];
    Proposta storage proposta = votacao.propostas[indexProposta];

    require(votacao.encerrada == false, "Votação está encerrada");
    require(votante.valido == true, "Token inválido");
    require(votante.votou == false, "Votante já votou");

    proposta.contadorVotos = proposta.contadorVotos + 1;
    votante.votou = true;
}
```

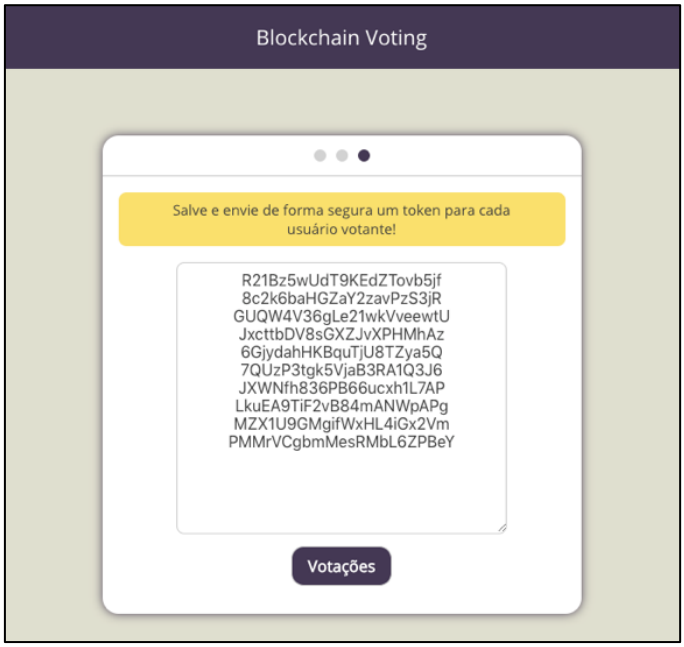
Fonte: elaborado pelo autor.

Além das funções `criarVotacao` e `votar` apresentadas nos Quadros 8 e 9, o *smart contract* ainda possui uma função `encerrarVotacao` que como o próprio nome sugere, finaliza a votação não permitindo que novos votos sejam registrados. Além disso, o contrato ainda disponibiliza algumas funções que retornam os valores armazenados no seu estado, permitindo que a interface da aplicação possa acessar estas informações e exibi-las para o usuário conforme necessário.

3.4 INTERFACE

Para que o usuário final da aplicação possa interagir com as funcionalidades expostas pelo *smart contract*, a solução conta com uma interface web conforme mencionado anteriormente. Esta interface ou *front-end* consiste em algumas páginas desenvolvidas utilizando tecnologias como HTML, CSS, Javascript e uma biblioteca chamada React que auxilia na construção de interfaces web interativas. Porém, diferentemente de uma aplicação web comum onde estas páginas se comunicariam com um *webservice*, que por sua vez iria buscar e armazenar informações em um banco de dados, neste caso cada página se comunica diretamente com o *smart contract* publicado no blockchain da Ethereum conforme apresentado anteriormente na Figura 4. Essa comunicação é realizada para obter os dados que devem ser apresentados ao usuário ou executar alguma função do contrato. A Figura 5 demonstra um destes exemplos, onde após receber os dados de entrada do usuário, a página responsável por criar votações envia uma mensagem ao contrato solicitando que a função `criarVotacao` apresentada no Quadro 8 seja executada e ao final exibe os *tokens* gerados para que o organizador da votação possa enviá-los aos usuários conforme descrito anteriormente.

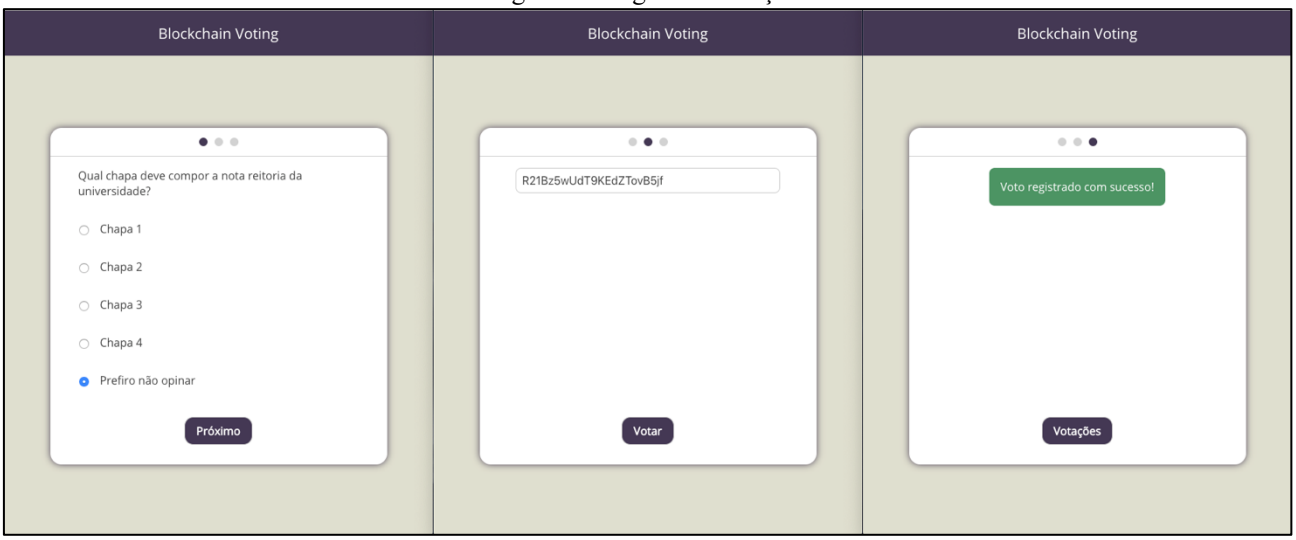
Figura 5 – *Tokens* gerados ao criar uma votação



Fonte: elaborado pelo autor.

De modo geral, as demais páginas da aplicação são visualmente semelhantes. Na Figura 6 é possível observar como ocorre o processo de votação pela perspectiva do usuário votante. O formulário de votação está divido em três etapas, sendo que a primeira delas permite ao usuário selecionar a proposta desejada, já a segunda solicita que ele informe o *token* que recebeu do organizador da votação e na terceira o voto é efetivamente registrado, produzindo um *feedback* visual para o usuário indicando se a chamada realizada para a função `votar` apresentada anteriormente no Quadro 9 ocorreu com sucesso.

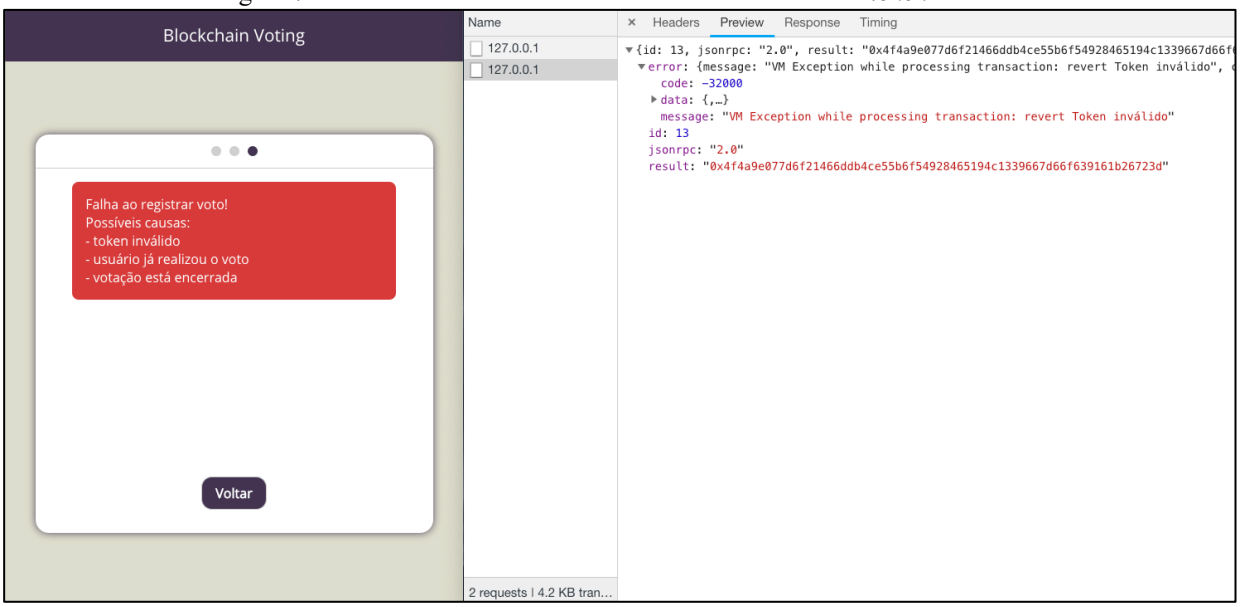
Figura 6 – Página de votação



Fonte: elaborado pelo autor.

A comunicação entre as páginas web e o *smart contract* publicado na Ethereum ocorre através de um protocolo de chamada de procedimento remoto ou Remote Procedure Call (RPC), que funciona sobre o Hypertext Transfer Protocol (HTTP). Para realizar esta comunicação, a Ethereum disponibiliza uma biblioteca chamada de web3.js, que permite acessar todas as funções do contrato diretamente no *browser* utilizando Javascript, e de forma transparente a biblioteca faz as chamadas remotas para a rede de blockchain. Na Figura 7 é apresentado o exemplo de uma destas chamadas, onde utilizando uma ferramenta que permite inspecionar as requisições feitas por uma página web é possível observar o que o contrato retorna quando a função `votar` apresentada anteriormente no Quadro 9 é chamada com um *token* inválido.

Figura 7 – Chamada remota realizada ao votar utilizando um *token* inválido



Fonte: elaborado pelo autor.

O processo de votação se encerra quando a função `encerrarVotacao` do contrato é chamada, isto ocorre através de um botão disponível na interface da aplicação. A partir do momento em que a votação está encerrada não serão mais registrados novos votos conforme descrito anteriormente e será possível acessar a página de resultados para que o organizador da votação ou qualquer outro usuário possa conferir a contagem de votos recebida por cada proposta.

4 RESULTADOS E DISCUSSÕES

De forma geral, dentro do escopo deste artigo a solução proposta consegue gerenciar votações de pequena escala utilizando uma rede de blockchain e *smart contracts*, o que traz inúmeras vantagens relacionadas a segurança e transparência do processo como apontado anteriormente. Porém, os contrastes entre o ambiente de desenvolvimento e um cenário real nos levam a considerar alguns fatores importantes que poderiam se tornar um problema. Um deles é o **custo** em termos monetários, pois a Ethereum utiliza o mecanismo de *gas* para evitar que a rede sofra ataques de negação de serviço como descrito anteriormente, ou seja, qualquer transação necessita que uma quantidade de *ether* seja enviada para pagar por esta taxa. A quantidade de *gas* a ser paga depende exclusivamente do número de computações que serão realizadas ao executar a ação solicitada, que no contexto desta proposta pode ser a publicação do contrato, criação de uma votação, registrar um voto ou encerrar uma votação.

Assim como apontado anteriormente, o *gas* faz com que a realização de testes utilizando a rede oficial da Ethereum se torne algo caro dependendo do volume de transações, mas ainda que este tipo de taxa não exista em uma rede de desenvolvimento, é possível identificar a quantidade de *gas* e custo em *ether* que seria necessário para cada uma das operações realizadas. Utilizando esta metodologia, a Tabela 1 apresenta as estimativas para cada uma destas operações e valor correspondente em Reais utilizando a cotação corrente na data deste artigo (R$ 201,78 por unidade).

Tabela 1 – Custo médio estimado para execução do contrato na Ethereum

Operação	*Gas*	Total em ether (ETH)	Conversão para Reais (R$)
Publicar Contrato	277462	0.0006659	0,33
Criar Votação	528846	0.0012692	0,63
Registrar Voto	35201	0.0000845	0,04
Encerrar Votação	43435	0.0001042	0,05

Fonte: elaborado pelo autor.

De acordo com os dados apresentados na Tabela 1, é possível estimar por exemplo quanto custaria em média uma votação envolvendo cem usuários votantes, ao qual, somando os totais em reais e multiplicando o valor estimado para a transação responsável por registrar o voto pelo número de votantes, teríamos um total de R$ 5,01, o que é um valor relativamente baixo. Porém outra característica importante neste caso, é o **tempo** necessário para que a transação seja efetivada no blockchain, e conforme mencionado anteriormente, esse tempo tem relação direta com o preço ofertado por cada unidade de *gas*, ou seja, as transações que estiverem oferecendo um pagamento maior, naturalmente vão ter a preferência dos mineradores.

Sendo o blockchain da Ethereum público, é possível estimar a relação custo x tempo para uma transação ser incluída na cadeia de blocos baseado no histórico das últimas milhares de transações e quanto se pagou pelo *gas* em cada

uma delas. Segundo o mecanismo de cálculo de taxas de ETH Gas Station (2018), é possível identificar por exemplo que a transação "Criar Votação" apresentada na Tabela 1 levaria aproximadamente 1657 segundos para ser efetivada, o que é um espaço de tempo relativamente longo se tratando de uma aplicação web que precisa apresentar um *feedback* para o usuário. Portanto, foram realizadas outras simulações da mesma operação aumentando o preço pago pelo *gas*. Os tempos resultantes e totais são apresentados na Tabela 2.

Tabela 2 – Custos e tempos para a operação responsável por criar votação

Total em ether (ETH)	Conversão para Reais (R$)	Tempo (segundos)
0.0012692	0,63	1657
0.0015865	0,78	452
0.0021154	1,04	39
0.0216827	10,63	31

Fonte: elaborado pelo autor.

Como é possível observar através dos dados apresentados na Tabela 2, o aumento do preço pago pelo *gas* influencia consideravelmente o tempo que a transação precisa para ser efetivada, principalmente entre a primeira e terceira linha, onde o valor não se altera tanto, mas o tempo passa de quase 30 minutos para 39 segundos. Já entre as duas últimas linhas é possível constatar o oposto, o valor aumenta consideravelmente enquanto o tempo não. Foram realizados testes aumentando ainda mais o valor pago, porém o tempo não é mais alterado, o que significa que provavelmente o tempo apresentado na última linha deve ser o mínimo necessário para a criação de um bloco atualmente.

Desta forma, é plausível pressupor que em um cenário de larga escala como as eleições de um país por exemplo, tenha que se levar em consideração possíveis problemas relacionados ao tempo para registrar os votos, assim como analisar qual impacto o evento teria na rede da Ethereum para garantir que os tempos estimados em um estado normal da rede não sofreriam grande alteração.

5 CONCLUSÕES

O desenvolvimento deste artigo possibilitou uma análise mais concreta sobre a viabilidade da utilização de uma rede de blockchain como base para uma plataforma de votação online. O uso da arquitetura trouxe consigo características como confiabilidade e transparência, que solucionam muitos dos problemas pertinentes aos sistemas legados de votação, porém outros fatores como autenticação a nível pessoal estão fora do escopo do blockchain e devem avançar tecnologicamente de forma independente. Mesmo tendo como foco processos de votação em escala relativamente pequena, o estudo proporciona uma perspectiva sobre possíveis problemas de performance e escalabilidade em cenários de larga escala.

A plataforma Ethereum e os *smart contracts* se mostraram uma das maiores evoluções desde a criação a Bitcoin, permitindo o desenvolvimento de aplicações baseadas em blockchain enquanto abstraem muita de sua complexidade. Não se encontrou grandes problemas no desenvolvimento do contrato, porém vale ressaltar que a execução dele em uma rede de desenvolvimento local pode omitir problemas relacionados ao tempo de efetivação das transações quando utilizando a rede oficial da Ethereum conforme foi possível identificar nos testes realizados.

De forma geral, o uso de blockchain se mostrou um grande recurso na implementação da solução, porém deve ser abordado com cautela quando se fala das eleições de um país por exemplo, já que algumas características inerentes a um blockchain público como a transparência podem tornar difícil manter em sigilo as opções escolhidas por cada participante, assim como a utilização de um blockchain privado pode trazer de volta a necessidade de confiar em uma entidade externa para gerenciar todo o processo. Para que a proposta possa avançar no sentido de se tornar viável nestes cenários de larga escala, este trabalho pode ser estendido identificando os requisitos legais envolvidos em uma eleição nacional, quais alterações seriam necessárias no contrato, assim como realizar uma análise dos impactos em termos de performance, escalabilidade e custo.

A solução proposta possui ainda algumas limitações quando se trata da autenticação dos usuários, pois somente os usuários votantes são autenticados através dos *tokens*, sendo assim não é possível permitir que somente o organizador da votação possa encerrá-la por exemplo, já que ele não pode ser identificado. A arquitetura da proposta pode evoluir neste sentido com outra possível extensão deste trabalho, definindo uma forma de autenticação para os organizadores das votações, assim como propor uma forma segura e automatizada de distribuição dos *tokens* evitando que o organizador tenha que fazê-la manualmente.

Além da aplicação proposta, o artigo se mostrou relevante pois aborda com maiores detalhes alguns dos conceitos chave para o funcionamento de uma rede de blockchain e o processo de mineração, o que deve contribuir com a base de conhecimento científico local e facilitar o desenvolvimento de futuras pesquisas envolvendo a arquitetura não somente em cenários democráticos, mas em qualquer aplicação que contribua com a sociedade de alguma forma.

REFERÊNCIAS

ANTONOPOULOS, Andreas. **Mastering Bitcoin**. United States of America: O'Reilly Media, 2014.

AVAST. **91,84% dos brasileiros acreditam que o sistema eletrônico de votação pode ser violado nas eleições**. [S.l.], 2018. Disponível em: <https://press.avast.com/pt-br/9184-dos-brasileiros-acreditam-que-o-sistema-eletrônico-de-votação-pode-ser-violado-nas-eleições>. Acesso em: 1 dez. 2018.

ETH GAS STATION. **Tx Calculator**. [S.l.], 2018. Disponível em: <https://ethgasstation.info/calculatorTxV.php>. Acesso em: 30 nov. 2018.

FAGGART, Evan. **Thoughts On Bitcoin Block Size Economics**. [S.l.], 2015. Disponível em: <http://bitcoinist.com/thoughts-bitcoin-block-size-economics>. Acesso em: 27 mai. 2018.

FAOUR, Nazim. **Transparent Voting Platform Based on Permissioned Blockchain**. 2018. 49 f. Master Thesis (Faculty of Computer Science) - Department of Software Engineering, Higher School of Economics (National Research University), Moscow, Russia.

FERREIRA, Frederico Lage. **Blockchain e Ethereum**: Aplicações e Vulnerabilidades. 2017. 36 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo (USP), São Paulo, Brasil.

KOÇ, A. K. et al. Towards Secure E-Voting Using Ethereum Blockchain. In: Conference: International Symposium on Digital Forensic and Security (ISDFS), 6, 2018, Antalya, Turkey. **Proceedings…** Antalya: Institute of Electrical and Electronics Engineers, 2018. p. 143-149.

SCHÜPFER, Florian. **Design and Implementation of a Smart Contract Application**. 2017. 129 f. Master Thesis (Communication Systems Group) – University of Zurich, Lucern, Suíça.

SWAN, Melaine. **Blockchain Blueprint for a New Economy**. United States of America: O'Reilly Media, 2015.

SZABO, Nick. Smart Contracts: Building Blocks for Digital Markets. **Extropy**, [S.l.], v.1, n. 16, 1996.

WATER, Gijs van de. **Blockchain Ballot**: Electoral Enhancement or Danger to Democracy?. 2017. 60 f. Master Thesis (Information Management) - Tilburg University, Tilburg, Nova Zelândia.