Universidade Estadual do Sudoeste da Bahia
Departamento de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

DESENVOLVIMENTO DE UM SISTEMA DE VOTAÇÃO UTILIZANDO *BLOCKCHAIN*

Vitor Silva Sampaio

Orientador:
Prof. Dr. Marlos Marques

Vitória da Conquista - BA
Agosto - 2019

Universidade Estadual do Sudoeste da Bahia
Centro de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

DESENVOLVIMENTO DE UM SISTEMA DE VOTAÇÃO UTILIZANDO *BLOCKCHAIN*

Vitor Silva Sampaio

Trabalho de conclusão de curso apresentado junto ao curso de Ciência da Computação da Universidade Estadual do Sudoeste da Bahia, na área de concentração de Ciências Exatas, como requisito parcial à obtenção do título de Bacharel.

Vitória da Conquista - BA
Agosto - 2019

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS E ABREVIATURAS

AES (*Advanced Encryption Standard*) - Padrão de Criptografia Avançada.

IDE (*Integrated Development Environment*) - Ambiente de Desenvolvimento Integrado.

PoS (*Proof of Stake*) - Prova de Participação.

PoW (*Proof of Work*) - Prova de Trabalho.

QR Code (*Quick Response Code*) - Código de Resposta Rápida.

RESTFull (*Representational State Transfer*) - Transferência de Estado Representacional.

SD - Sistema Distribuído.

SHA256 (*Secure Hash Algorithm*) – Algoritmo de *Hash* Seguro.

Resumo

O grande sucesso obtido pela *blockchain* no gerenciamento das criptomoedas, despertou o interesse da sua utilização em outras áreas, entre elas, os sistemas de votação. A similaridade entre os princípios de uma votação e as características intrínsecas da *blockchain* fizeram com que ela fosse uma alternativa eficiente para a implementação de um sistema de votação distribuído e seguro. A desconfiança nos atuais sistemas eletrônicos de votação é outro ponto que justifica a implementação de um sistema de votação baseado em *blockchain*. Embora a blockchain seja uma tecnologia muito recente, o fato dela ser, desde o princípio, baseada nos sistemas distribuídos e na criptografia dá a ela, embasamento teórico suficiente para sua utilização em diversas outras áreas, diferentes da sua abordagem inicial. Neste trabalho será proposto um sistema de votação baseado em *blockchain* que visa trazer mais segurança em eleições eletrônicas.

Palavras Chaves: *Blockchain*. Ethereum, Votação.

Abstract

Blockchain's great success in the management of crypto-coins has raised the interest of its use in other areas, including voting systems. The similarity between the principles of a vote and the intrinsic characteristics of the blockchain made it an efficient alternative for the implementation of a distributed and secure voting system. The distrust in the current electronic voting systems is another point that justifies the implementation of a voting system based on blockchain. Although blockchain is a very recent technology, the fact that it is based on distributed systems and encryption from the outset provides enough theoretical background for its use in several other areas, different from its initial approach. In this work, a voting system based on blockchain will be proposed, aiming to bring more security in electronic elections.

Keywords: Blockchain, Ethereum. Voting.

1 INTRODUÇÃO

1.1 Motivação

Votação é um método utilizado por pessoas, máquinas ou organizações para solucionar problemas de desacordos ou realizar escolhas por meio do voto, tem como objetivo representar a decisão de uma maioria. (NURMI, 2014). Para que uma eleição seja válida, é necessário que ela seja íntegra e justa. Além disso, dependendo do objetivo da votação, também pode ser necessário o anonimato e a imutabilidade dos votos. Em um processo de votação, a manifestação do voto pode se dar de diferentes formas como: levantando o braço, registrado em papel ou por uma sequência de *bits* em um computador. O registro em papel foi, e continua sendo, uma forma muito comum de votação. Porém certas características deste modelo são questionáveis e sucessíveis a fraudes (SCHAUREN, 2016).

Com o avanço tecnológico no final do século XX, era esperado que uma solução eletrônica fosse proposta para solucionar os problemas dos métodos baseados em papel (SCHAUREN, 2016). Para eleições presidenciais, por exemplo, alguns países como Brasil, Índia e Romênia, propuseram alternativas que tentaram solucionar os problemas do método anterior. Embora as soluções eletrônicas propostas trouxessem melhorias na segurança e audibilidade do processo, estes sistemas ainda são questionados e segundo Schauren (2016) ainda são propensos a fraudes.

Casos e suspeitas de fraudes em eleições continuaram sendo comumente noticiadas, organizações estatais e privadas ainda relatam com certa frequência estes acontecimentos. Alternativas que buscam agregar mais segurança nos processos de votação foram, e continuam sendo propostas mundo a fora. Porém, o que muitos destes métodos propostos têm em comum é a interferência humana em fases críticas do processo de votação, sendo assim necessário que haja idoneidade de todos os envolvidos, algo que é dificilmente garantido.

Neste cenário de incertezas e insegurança surgiu a *blockchain*. Uma tecnologia bastante poderosa e segura que obteve grande sucesso no processamento de transações financeiras, sendo pioneiramente aplicada no gerenciamento de criptomoedas. Baseada na computação distribuída e utilizando conceitos de criptografia e *hash*, logo foram vislumbradas outras

aplicações para a tecnologia, como por exemplo, uma possível solução para a implementação de um sistema de votação confiável, seguro e distribuído.

A motivação para o desenvolvimento deste trabalho surgiu pela fragilidade dos métodos atuais de votação, pela necessidade de mecanismos mais seguros de votação e pela interessante interseção existente entre os princípios de um sistema de votação e as características intrínsecas da *blockchain*.

1.2 Objetivos

O objetivo deste trabalho é desenvolver um sistema de votação que utilize conceitos e práticas de *blockchain* e segurança da informação, buscando agregar mais segurança em um processo de votação, além disso, a implementação de um aplicativo *mobile* busca trazer mais comodidade ao eleitor, fornecendo uma forma mais prática de votação.

1.3 Estrutura do Trabalho

O trabalho está dividido em 6 capítulos, os três primeiros capítulos apresentam uma introdução ao tema, a motivação, os objetivos do trabalho, as definições e o estado da arte da ciência da área. O quarto capítulo apresenta o sistema que foi desenvolvido, mostrando detalhadamente os passos que foram seguidos. Nos dois últimos capítulos são feitas as considerações finais, a apresentação dos resultados obtidos e sugestões de trabalhos futuros.

A implementação do sistema de votação desenvolvido neste trabalho foi dividida em 3 fases. Na primeira fase foi implementado o *smart contract* que roda na *blockchain*. Na segunda fase foi desenvolvido o servidor RESTFul. E na última fase foi implementada a aplicação *mobile* utilizada pelos eleitores.

2 REFERENCIAL TEÓRICO

2.1 Votação: história e princípios

Uma votação é um processo na qual indivíduos, máquinas e organizações expressam seus desejos e opiniões sobre um determinado assunto por meio do voto. Votações auxiliam processos de escolha onde há discordância entre as partes envolvidas (NURMI, 2014).

Dependendo do objetivo da votação, o processo pode assumir diferentes arranjos (exemplificados mais adiante), mas mantendo sua ideia central, que é: "*Como agregar as preferências individuais em uma única escolha coletiva de maneira racional e justa?* ".

O primeiro registro de uma eleição é datado de 500 a.C., na Grécia Antiga, introduzido por Clístenes, um político grego. Na ocasião, os gregos votavam, (normalmente utilizando cacos de cerâmica) no político que eles menos gostassem, e caso algum deles recebessem mais de 6000 votos, o mesmo era exilado por 10 anos. Caso houvesse grande dispersão entre os votos, nenhum candidato "ganharia", desta forma, um candidato só seria exilado se fosse muito impopular e rejeitado pelos eleitores (O'CONNOR E ROBERTSON, 2002).

Em um sistema de votação, alguns princípios são intrínsecos e necessários, enquanto outros são apenas desejáveis, e sua necessidade depende do objetivo da votação. O anonimato, por exemplo, é um princípio desejável, mas não inerente a uma votação. No Brasil, na era imperial, o anonimato do voto não era necessário, e o eleitor poderia votar e declarar sua escolha em voz alta, à vista de todos os presentes (BRASIL, 2016).

Em sistemas baseados em *blockchain*, por sua vez, o anonimato e a praticidade representam um *trade-off*[1], optar por mais praticidade nestes sistemas implica, normalmente, na perda de anonimato e vice-versa. Encontrar um viés aceitável entre o anonimato e a praticidade costuma ser um tópico importante e difícil de ser resolvido nestes sistemas.

Um princípio também desejável em uma eleição é a imutabilidade, que, somente é necessária quando o processo é suscetível a auditorias futuras. Como normalmente é interessante manter os registros históricos das eleições, este princípio costuma ser agregado na maioria das eleições.

[1] *Trade-off* é uma situação na qual ocorre conflito de escolha entre duas ou mais opções, onde a escolha de uma opção implica na perda da(s) outra(s).

Outros dois princípios, por sua vez, inerentes a uma eleição são a integridade e o conceito de justo. A integridade de um dado é definida por Stallings (2013) como sendo a garantia que as informações de um sistema somente sejam alteradas de maneira específica e autorizada. Como uma eleição é formada por um conjunto de dados, a integridade de uma eleição, depende da integridade dos seus dados.

Um sistema de votação é tido como justo quando seu resultado expressa o desejo dos eleitores. Uma votação pode ter uma quantidade *n* de candidatos. Quando *n = 2*, a eleição precisa, obrigatoriamente, obedecer a 3 premissas para ser considerada justa (MAY, 1952):

1. Todos os votos têm o mesmo peso;
2. Todos os candidatos são tratados igualmente;
3. Se um candidato receber mais votos, ele deve ser o ganhador.

Quando uma eleição tem *n > 2*, a situação torna-se mais complicada, e vários sistemas de votação diferentes foram propostos, cada um com características e peculiaridades distintas para tentar produzir uma eleição justa (O'CONNOR E ROBERTSON, 2002).

Um sistema de votação com *n > 2* candidatos é dito como justo quando o vencedor da eleição obtém mais de 50% dos votos em todos os confrontos binários com cada um dos *n - 1* outros candidatos. Vários arranjos diferentes de votação foram propostos e várias formas de contabilização, distribuição e interpretação do voto foram criadas para garantir um sistema justo.

Uma solução óbvia para este problema é a seguinte: “Cada eleitor vota em um único candidato, o candidato mais votado ganha”. Esta solução é denominada *Winner Takes All* (ou, o vencedor leva tudo). Por ser simples e intuitiva, este método costuma ser amplamente utilizado. O problema desta solução ocorre quando existem muitos candidatos e há uma grande dispersão dos votos, desta forma, um candidato pode ganhar mesmo sendo rejeitado pela maioria. Supondo uma votação com 4 candidatos, e suas respectivas porcentagens de votos conquistados: A = 26%, B = 20%, C = 30%, D = 24%, o candidato C ganha a votação mesmo não sendo escolhido por 70% dos eleitores, mostrando que a votação não satisfez a maioria dos votantes.

Um outro sistema de votação proposto por Llull (13--, *apud* O'Connor e Robertson, 2002), seguia à risca o conceito de justo, definido acima. Neste sistema, cada um dos *n* candidatos deveriam disputar uma votação binária com todos os outros *n - 1* candidatos, exigindo $\frac{n(n-1)}{2}$ confrontos binários. O problema deste sistema é que a eleição poderia não ter um candidato vencedor.

Além desses, vários outros sistemas foram propostos para tentar resolver este problema (CUSA, 1443, *apud* O'CONNOR E ROBERTSON, 2002) e (CONDORCET, 1785, *apud* O'CONNOR E ROBERTSON, 2002), até que em 1951, o matemático Nobel em economia, Kenneth Arrow, provou pelo Teorema da Impossibilidade de Arrow que nenhum sistema de votação é justo no sentido de produzir um resultado no qual o candidato vencedor é o preferido dentre todos os candidatos que disputaram a votação.

Mesmo provando a impossibilidade de um sistema justo, Arrow (1951) também afirmou que: “A maior parte dos sistemas não funcionará mal o tempo todo, tudo que provei é que eles podem funcionar mal às vezes”.

Desta forma, a utilização de qualquer um desses arranjos de votação é facultativo e pode variar dependendo do objetivo da votação. Como é provado que nenhum deles funcionará corretamente em todos os casos, a utilização de qualquer um deles é válida. Neste trabalho será usado o método mais comum de votação, o *Winner Takes All*.

2.2 Sistema Eletrônico de Votação

Durante a história, o voto em papel foi amplamente utilizado em processos de votação. Tal característica injetava diversos problemas no sistema, visto a fragilidade intrínseca do papel, juntamente com a sua suscetibilidade à adulteração.

Problemas como a dualidade do voto, a falsificação de cédula de votação e a grande interferência humana no processo de votação, fizeram com que votações que utilizavam esta técnica perdessem credibilidade.

Segundo Schauren (2016, p. 10-11), o modelo baseado em papel apresentava diversos problemas e vícios inerentes ao processo de votação. Ainda segundo ele, o modelo baseado em papel não era capaz de garantir a segurança necessária no processo, além disso, o modelo dependia de fiscalização por todas as partes interessadas, assim como garantia de idoneidade absoluta entre os envolvidos.

Com a evolução tecnológica das últimas décadas, era esperado que uma solução digital fosse proposta para sistemas de votação. Em 1960, o brasileiro Sócrates Ricardo Puntel, propôs uma solução mecânica para sistemas de votação, denominado, pelo inventor, *urna mecânica*. Devido a limitações de resistência, acessibilidade e transporte o projeto foi descontinuado, embora tenha servido como precursor da urna eletrônica brasileira (Brasil, 2016).

Vários países desenvolveram métodos eletrônicos de votação. A Figura 2.1 mostra distribuição da utilização dos mecanismos eletrônicos de votação ao redor do mundo.

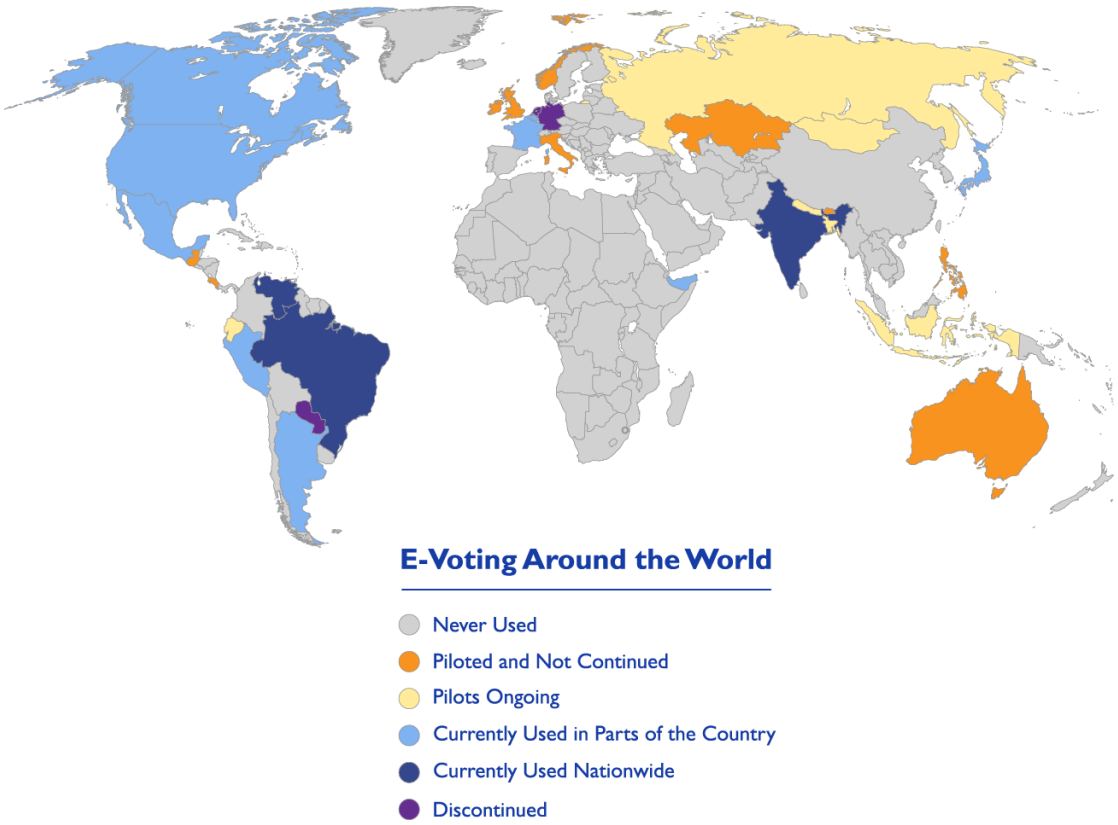

Figura 2.1 Votação eletrônica no mundo. Fonte: National Democratic Institute (2018).

Embora, hodiernamente, a implementação de um sistema eletrônico de votação não seja um processo altamente complexo, ainda se observa, de acordo com a Figura 2.1, que poucos países implementaram estes sistemas. Uma das explicações para esta rejeição foi dada pela corte alemã ao julgar a necessidade da implementação de um sistema eletrônico de votação no país. De acordo com a corte, utilizar um método de votação que não fosse possível de ser auditado por um cidadão comum, seria uma quebra dos princípios eleitorais do pais. Ainda de acordo com a corte, apenas pessoas com um certo nível técnico seriam capazes de entender o funcionamento do sistema, fazendo com que o processo perdesse credibilidade ao olhar de um cidadão comum (SCHAUREN, 2016).

2.3 Sistemas Distribuídos

Um sistema distribuído é definido por Tanenbaum e Steen (2007) como um conjunto de computadores independentes, direto ou indiretamente conectados, que se apresentam a seus usuários como um único e transparente sistema. A principal função de um SD (Sistema Distribuído) é distribuir recursos computacionais entre diversos computadores, de forma que seus usuários não tenham conhecimento desta distribuição.

Esta distribuição de recursos implica redundância e segmentação dos dados, que em um sistema de votação são características importantes para manter a segurança do processo. Embora na maioria dos sistemas distribuídos a segmentação dos recursos implique na perda de segurança, na *blockchain*, a dispersão dos dados é um fator crucial para a garantia da segurança.

A transparência, a escalabilidade e o consenso são caraterísticas necessárias em um SD. A transparência e a escalabilidade, se referem, respectivamente, a capacidade do sistema se apresentar a seus usuários como um único e consistente sistema, e a facilidade de adaptação diante de variações no número de usuários, sem que haja prejuízos na sua utilização.

O outro princípio importante em um SD é o consenso, que, de acordo com Coulouris *et al.* (2013) consiste em fazer com que um conjunto de processos independentes entrem em acordo sobre algum valor, após um ou mais processos terem sugerido qual deveria ser esse valor.

2.4 Consenso em Sistemas Distribuídos

Em SD, o consenso é utilizado para tratar diversos problemas de coordenação e acordo entre processos. Problemas como exclusão mútua, seleção de líder, sincronização e *reliable multicast* são problemas clássicos de consenso em sistemas distribuídos.

Vários algoritmos foram propostos para tratar cada um desses problemas clássicos. Para tratamento de exclusão mútua, por exemplo, pode-se utilizar uma solução baseada em um servidor central, que gerencia as entradas/saídas das regiões críticas, ou uma estratégia, totalmente distribuída, baseada em anel (COULOURIS, 2013).

Para problemas de seleção de líder, uma possível solução é o algoritmo de Chang e Roberts (1979), que também organiza os processos em um anel lógico. No algoritmo, cada

processo é identificado unicamente, os processos trocam mensagens entre si, e no final, o processo com maior identificador é definido como o líder.

2.5 Blockchain

A *blockchain*[2] é uma estrutura de dados distribuída que guarda informações sobre o histórico de transações realizadas pela rede. Sua origem foi nas criptomoedas, onde tinha como objetivo guardar o histórico das transações financeiras realizadas pelos nós da rede e manter a integridade, imutabilidade e autenticidade das informações nela presente (LEWIS, 2015). Como a base da *blockchain* são os sistemas distribuídos, ela herda boa parte das características desses sistemas.

A *blockchain* foi criada com o objetivo de eliminar a necessidade de um terceiro-confiável. Nela, a segurança dos dados é mantida pela própria rede, não há a necessidade de uma pessoa ou órgão responsável que mantenha a integridade, imutabilidade e a autenticidade dos dados nela presentes.

Para realizar transações na *blockchain*, os usuários devem ter um endereço que os identifiquem unicamente (*wallet address*) e todas as transações realizadas pelos nós da rede são públicas, o anonimato é garantido pelo fato de não existir nenhuma ligação entre a pessoa que realizou a transação e seu endereço na rede. A Figura 2.2Figura 2.2 mostra a comparação de como o anonimato é garantido no sistema bancário e como ele é garantido na *blockchain*, por exemplo.

[2] A palavra *blockchain* pode significar duas coisas diferentes, mas interligadas. A *blockchain* pode ser considerada apenas a estrutura de dados mantida por cada nó, ou toda a rede em si. A literatura não faz nenhuma distinção entre a homonímia.

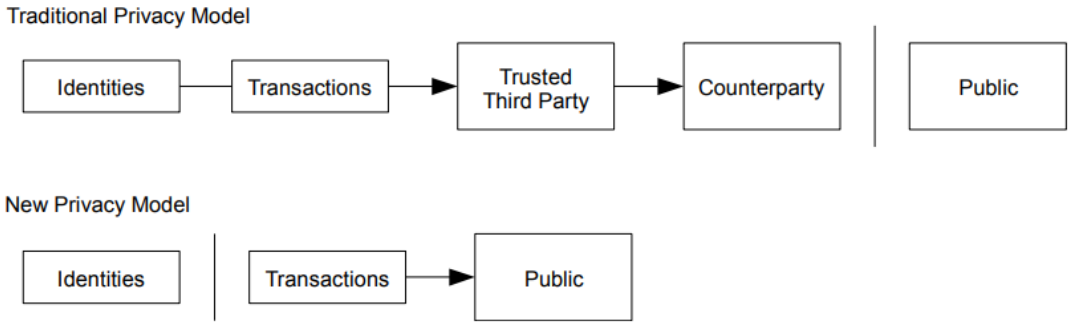

Figura 2.2: Anonimato no sistema bancário e na *blockchain*. Fonte: Nakamoto, 2008, p. 6.

A *blockchain* é composta por blocos interligados por referência, de forma análoga a uma lista encadeada. Cada bloco b_n conhece seu bloco anterior b_{n-1}, guardando uma referência para esse bloco, essa referência é o $hash(b_{n-1})$. Cada bloco tem a mesma estrutura com diversos campos, cada um com funções específicas. A Figura 2.3 mostra a estrutura básica de uma *blockchain*:

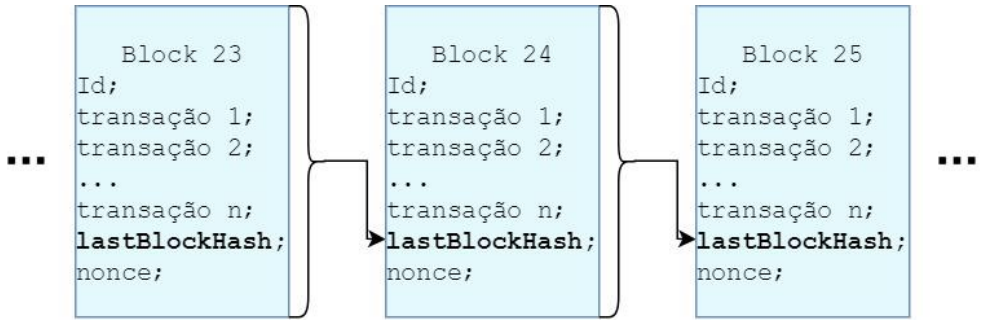

Figura 2.3: Estrutura básica de uma *blockchain*. Fonte: elaborada pelo autor.

Na Figura 2.1, a funções dos campos são:

- Id: responsável por identificar a ordem dos blocos. Dado de controle,
- Transações: representam as transações recebidas pela rede e já mineradas. Dado útil,
- *Lastblockhash*: guarda o *hash* do bloco anterior, dessa forma, um bloco com id $n + 1$, não pode ser minerado antes do bloco com id n. Dado de controle,
- *Nonce*: um dado de controle que será explicado mais a seguir.

Antes de avançar na explicação é importante citar algumas premissas importantes que devem ser consideradas na *blockchain*:

- Nenhum nó da rede é confiável,

- Cada nó da rede deve ter a mesma cópia de toda a *blockchain*,
- Cada nó da rede, deve fazer esforços para ter sempre a versão mais atualizada da *blockchain*,
- Uma transação é um evento acontecido na rede que precisa ser guardado,
- Um bloco guarda uma ou mais transações, além de outros campos de controle,
- Quando um nó recebe uma transação, ele precisa enviá-la, imediatamente, a todos os outros nós da rede.

A *blockchain* é mantida por uma rede descentralizada composta por vários nós. Cada nó da rede é responsável por manter a integridade e imutabilidade e autenticidade dos dados ali contidos.

Para manter a integridade da *blockchain*, os nós precisam verificar a integridade de cada uma das transações submetidas à rede. Uma vez verificada a integridade de uma transação (ou várias delas), um nó pode colocá-la (s) em um bloco e enviar para todos os outros nós da rede, indicando que aquele bloco é válido e que todos os outros nós devem inseri-lo também na sua *blockchain*.

Na maioria das implementações de *blockchains*, uma remuneração é pega para cada nó que coloque um novo bloco na *blockchain*. Para evitar que os nós da rede coloquem blocos indefinidamente na *blockchain*, cada bloco precisa ser minerado antes de ser colocado na *blockchain*.

Minerar um bloco *b* consiste em encontrar o *hash(b)* que satisfaça alguma condição pré-estabelecida pela rede. Para conseguir variar a saída da função *hash(b)*, o campo *nonce* do bloco é variado e o *hash*(b) é recalculado até que a condição seja satisfeita. O campo *nonce* é o único campo que pode ser variado durante o processo de mineração, sua função no bloco é justamente essa: ser variado para mudar a saída da função *hash*(b).

Uma vez minerado, o bloco pode ser enviado para a rede. Como nenhum nó da rede é confiável, cada nó que recebe um bloco, verifica a integridade do bloco antes do colocá-lo na sua *blockchain*.

O processo de mineração de blocos costuma ser um problema NP, mas a verificação de integridade é, normalmente, um problema polinomial, o que facilita a verificação, e dificulta a mineração.

A *blockchain* tenta atender princípios de transparência, escalabilidade e consenso utilizando técnicas variadas. A transparência é garantida pela redundância, como cada nó tem a versão completa e mais atual da *blockchain*, qualquer usuário conectado a um nó arbitrário

do sistema, tem acesso a todas as informações da *blockchain*. A redundância e a segurança implicam, à *blockchain*, uma perda de escalabilidade, atualizações demoradas e "incertas" são um dos grandes problemas desta tecnologia.

Para implementar o consenso na *blockchain*, dois algoritmos são comumente utilizados, o PoW (*Proof of Work*, ou Prova de Trabalho) e o PoS (*Proof of Stake*, ou Prova de Participação). No PoW, a importância de um nó é definida pelo seu poder de processamento, quanto mais processamento um nó tem, maior a sua autoridade nas escolhas da *blockchain*.

Como diferentes nós podem ter o mesmo poder de processamento, também é válida a de ideia de "*longest chain wins*", ou seja, o nó que tiver a maior *blockchain* tem direito sobre as decisões (NAKAMOTO, 2008). Como o processo de adição de blocos na *blockchain* depende também da PoW, para um nó ser líder unânime da *blockchain*, ele precisa ter um poder de processamento maior que todo o resto da rede, como isso é impraticável, a escolha de líder costuma ser um processo estocástico.

No algoritmo PoS, a validação de transações é feita sem a necessidade de mineração de blocos. Um algoritmo pseudorrandômico escolhe o nó que deve colocar o próximo bloco na *blockchain* e todos os outros nós devem estar de acordo com a escolha. Uma vez sorteado, o nó gera o próximo bloco, recebe uma recompensa (*stake*), e envia o bloco para os outros nós da *blockchain*. A probabilidade de escolha de um nó, depende da quantidade de *stakes* (recursos), que o nó tem. Desta forma, quanto mais blocos um nó já validou, maior a chance de ele conseguir o direito de validar outros futuros blocos.

Visto o sucesso da *blockchain* no gerenciamento de transações financeiras, novas abordagens desta tecnologia surgiram. *Blockchains* privadas, federais e *turing*-completas foram criadas para atender necessidades variadas (BLOCKCHAIN HUB, 2018).

Uma dessas inovações foi a *blockchain turing*-completa que pode ser vista como um computador com processamento, armazenamento e memória distribuídos, e que mantém todas as propriedades já citadas da *blockchain* convencional.

Exemplos de *blockchains turing*-completas são o Ethereum e a Neo (BLOCK GEEKS, 2018). A *blockchain turing*-completa desacopla as regras de negócio da implementação da *blockchain*. Tal melhoria faz com que a *blockchain* deixe de somente armazenar históricos de transações e passe a executar código-fonte. Esses códigos são denominados *smart contracts* (contratos inteligentes).

Contratos inteligentes são códigos-fonte escritos em alguma linguagem de programação que são executados dentro da própria *blockchain*. A sua função é desacoplar as restrições e regras de negócio da implementação da *blockchain*. Esta característica faz com que uma *blockchain*

possa ser utilizada como uma substituição de contratos físicos e terceiros-confiáveis, e também, em sistemas de votação.

3 ESTADO DA ARTE

A grande valorização financeira das criptomoedas ocorrida no final do ano de 2017 fez aflorar o interesse de organizações públicas e privadas na sua tecnologia base, a *blockchain*. Diversas foram as aplicações propostas, dentre elas, os *sistemas de votação*. A implementação de um sistema de votação utilizando *blockchain* é um tópico de interesse de diversas organizações, uma plataforma pioneira, confiável e escalável de votação com *blockchain* poderia ser a chave para solucionar problemas envolvendo eleições em todo o mundo.

Por ainda ser uma tecnologia recente, pouco explorada e instável, o desenvolvimento de sistemas que utilizem *blockchain* costuma ser um processo custoso e demorado. Esta dificuldade no desenvolvimento fez com que poucas organizações realmente desenvolvessem aplicações com *blockchain*. Abaixo serão brevemente descritas algumas plataformas de votação, que de acordo com seus desenvolvedores, utilizam *blockchain*.

3.1 Polys

O Polys é uma plataforma online de votação que utiliza a *blockchain* da criptomoeda Ethereum para contabilizar os votos e realizar o controle e gerenciamento de uma votação. As Figura 3.1 e a Figura 3.2 mostram, respectivamente, a tela inicial do *site* e a tela de gerenciamento das eleições.

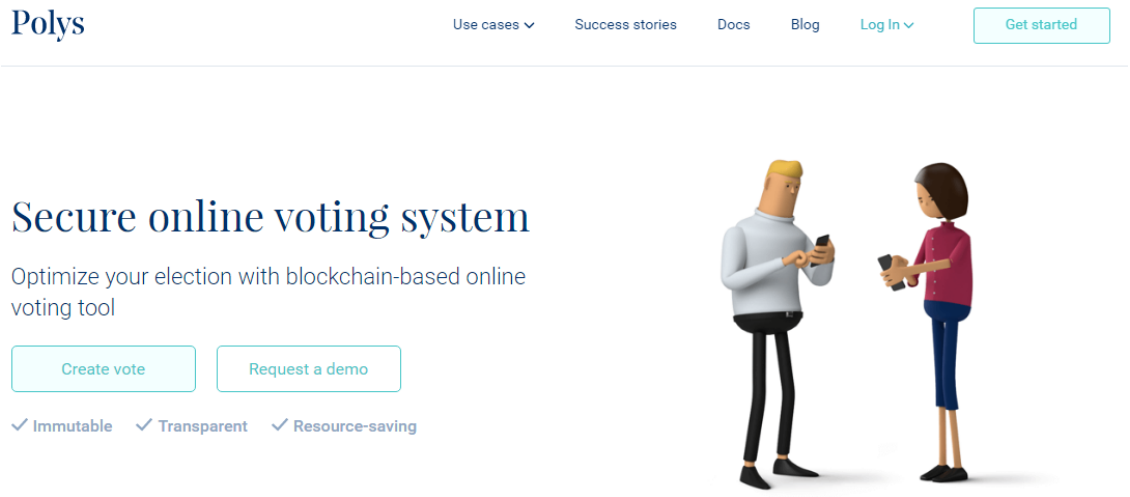

Figura 3.1 *Homepage* da plataforma Polys. Fonte POLYS (2019).

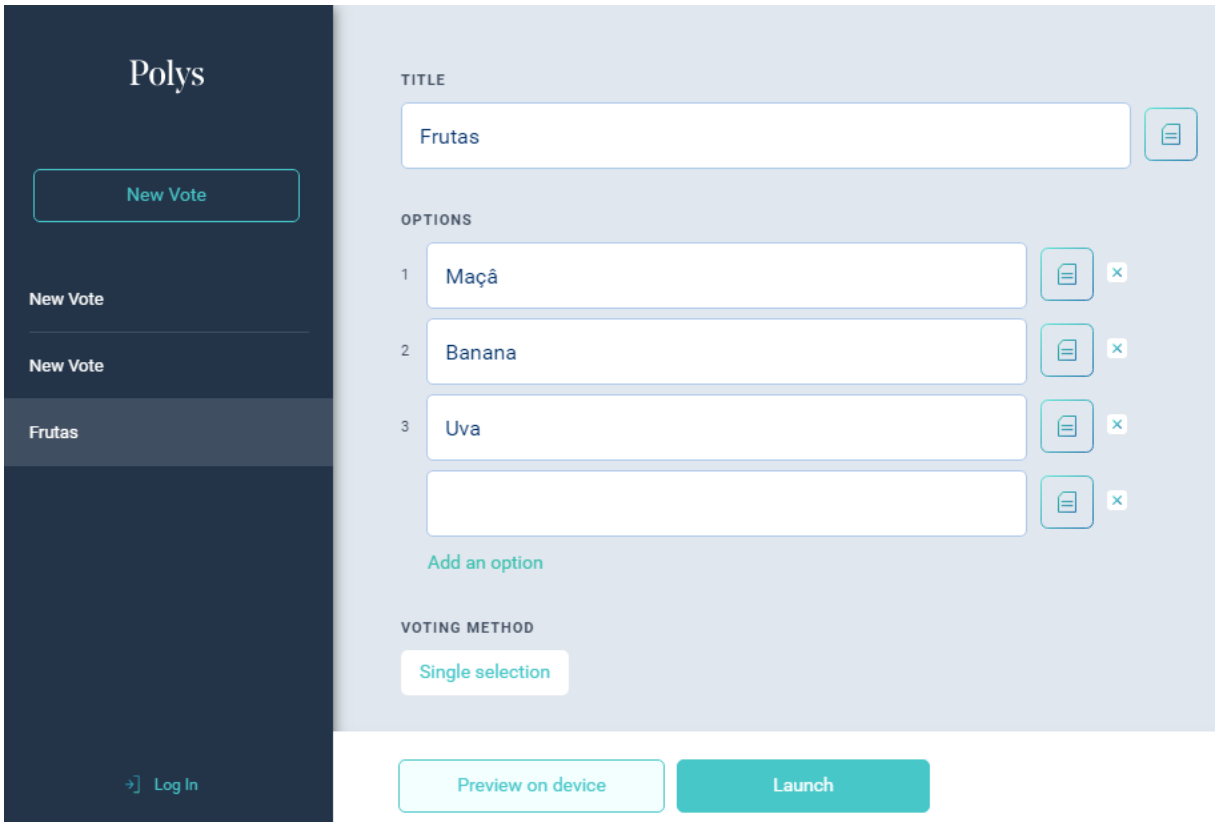

Figura 3.2 Tela para criar eleições e visualizar as eleições já criadas. Fonte POLYS (2019).

Os desenvolvedores do Polys afirmam que utilizam um contrato inteligente para fazer o controle dos eleitores que podem votar na eleição e impedir que um mesmo eleitor vote mais de uma vez na mesma eleição. Ainda de acordo com os desenvolvedores, cada eleitor é representado no sistema por um *token* que é renovado a cada novo *login*, impedindo que o *token* do eleitor seja rastreado, o que poderia quebrar o anonimato do voto.

O Polys permite controlar os eleitores que podem votar em uma eleição utilizando o e-mail do eleitor ou um código *single-use*. O sistema também permite deixar a eleição pública para qualquer eleitor com o *link* da eleição. Mais informações sobre esta plataforma e suas especificações porem ser encontradas em POLYS (2019).

3.2 Follow My Vote

O Follow My Vote é um *software*, que utiliza *blockchain* para gerenciar e garantir a segurança em eleições. O *site* do *software* fornece poucas informações referentes a como o

sistema funciona e como ele foi implementado. Para testar o sistema, é necessário enviar um formulário requisitando uma versão de teste do sistema, o que foi feito, porém não foi obtida resposta. A Figura 3.3 mostra a tela inicial do *site*, onde é possível entrar em uma lista de colaboradores, ou realizar uma doação para os desenvolvedores. Mais informações podem ser encontras em FOLLOW MY VOTE (2019).

Figura 3.3 *Homepage* do *site* Follow My Vote. Fonte: FOLLOW MY VOTE (2019).

3.3 Agora

O Agora é um ecossistema de votação baseado em *blockchain* que permite que qualquer pessoa em qualquer lugar vote *online* a partir de um dispositivo digital de maneira fácil e segura. A Figura 3.4 mostra a tela inicial do site do Agora.

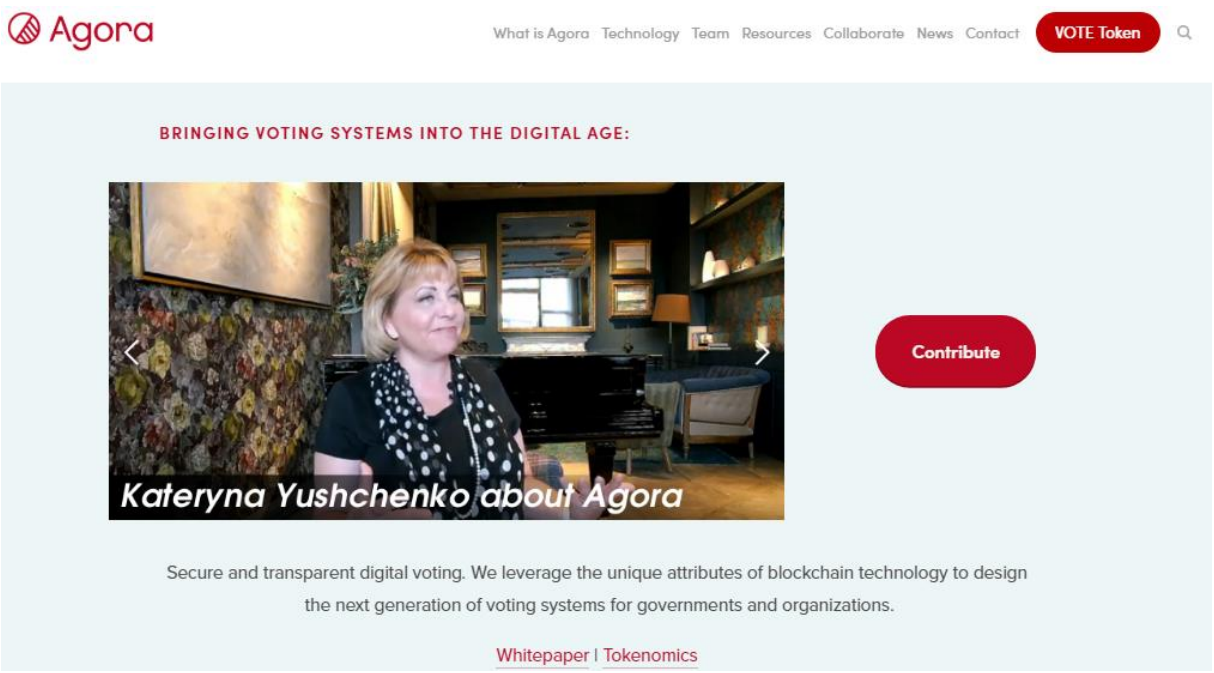

Figura 3.4 Homepage do site do Agora. Fonte: AGORA (2019).

De acordo com notícias publicadas em alguns *sites*, o Agora foi utilizada nas eleições presidenciais de Serra Leoa, um país da África Ocidental. Porém, dias após a realização das eleições, as autoridades do país negaram a utilização do *software*, alegando que a empresa responsável pelo Agora apenas acompanhou a eleição em algumas sessões eleitorais.

O *site* do Agora não fornece uma forma gratuita de testar as funcionalidades do seu sistema, para utilizá-lo é necessário adquirir os *tokens* de votação que são comercializados no próprio site, o que não possível devido às limitações financeiras do trabalho. Mais informações podem ser encontradas em AGORA (2019).

4 DESENVOLVIMENTO DE UM SISTEMA DE VOTAÇÃOUTILIZANDO *BLOCKCHAIN*

Com o sucesso obtido pela *blockchain*, novas versões desta tecnologia, melhoradas e com características específicas para um determinado problema, foram desenvolvidas. A ramificação da *blockchain* utilizada neste trabalho será a *Turing*-Completa.

As *blockchains Turing*-Completas têm como característica principal permitir que códigos-fonte sejam executados pelos nós que compõem a rede. Estes códigos rodam na *blockchain* de forma transparente, desta forma, o programador não se preocupa com detalhes relacionados à distribuição dos recursos manipulados por sua aplicação.

Os códigos-fonte que rodam na *blockchain* são denominados *contratos inteligentes*. Estes contratos, rodando na *blockchain*, só podem ser acessados por uma aplicação cliente. O conjunto formado pelo contrato inteligente e a aplicação cliente é denominado Decentralised Application ou, somente, DApps.

Várias criptomoedas desenvolveram suas *blockchains* Turing-Completas para permitir que terceiros consigam desenvolver aplicações para rodarem dentro de suas *blockchains*. Neste trabalho será usada a *blockchain Turing*-Completa da criptomoeda *Ethereum*.

A *blockchain* do Ethereum permite o desenvolvimento de contratos inteligentes em 3 linguagens de programação diferentes, e fornece uma API para que as aplicações clientes consigam acessar os contratos implantados na *blockchain***.** As linguagens aceitas pelo Ethereum são:

- Solidity: baseada no JavaScript,
- Like Lisp Language (LLL): baseada no Lisp,
- Serpent: baseada no Python.

A API fornecida pelo Ethereum é a Web3, que é escrita a manipulada em JavaScript. Uma API isomorfa a Web3 é a Web3j, que, por sua vez, é escrita e manipulada em Java.

A Web3j tem as mesmas funcionalidades da API em JavaScript, porém o processo de compilar e implantar o contrato inteligente na *blockchain* é diferente, mas gera o mesmo resultado.

Neste trabalho foi desenvolvido um DApp para gerenciar uma votação, desta forma é necessário um contrato para rodar na *blockchain* e uma aplicação cliente para acessar o contrato, além disso, foi necessário um servidor para validar os eleitores e servir como *rendezvous point*[3] para as aplicações clientes. Os componentes do DApp desenvolvido são mostrados na Figura 4.1.

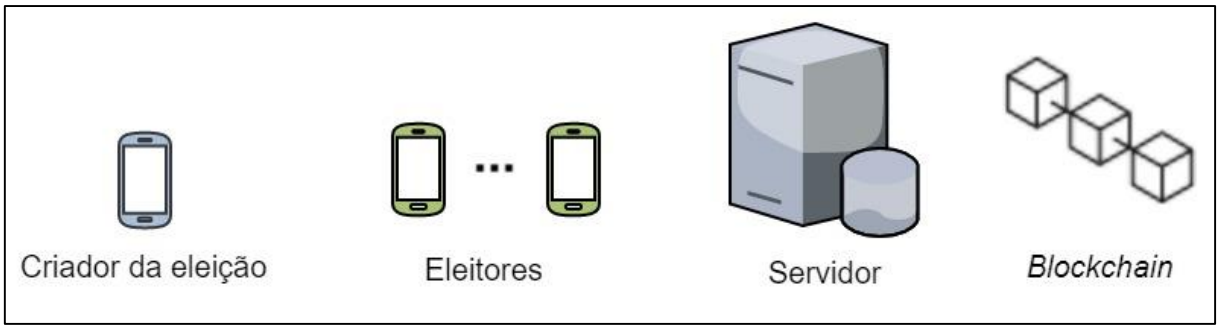

Figura 4.1: Componentes do sistema. Fonte: elaborada pelo autor, 2019.

A aplicação desenvolvida fornece duas arquiteturas diferentes para uma eleição. Cada uma com suas características, malefícios e benefícios. O que difere uma arquitetura da outra é o nível de dependência da aplicação no servidor. Na primeira forma, o sistema é bastante dependente do servidor, impossibilitando que seja realizada uma eleição sem o servidor. A segunda forma é mais independente e permite que uma eleição seja realizada praticamente sem a necessidade do servidor.

Para realizar uma eleição é necessário, primeiro, implantar o contrato inteligente na *blockchain*, segundo, informar para os eleitores o endereço onde o contrato está implantado, e terceiro, o criador da eleição precisa acessar a *blockchain* e autorizar o voto de cada um dos eleitores. Na primeira arquitetura proposta, todas as três etapas são realizadas pelo servidor, na segunda arquitetura a primeira e terceira etapas são feitas pela aplicação cliente e a segunda deve ser feita pelo usuário.

[3] Em SD, um *rendezvous point* é um nó responsável por enviar informações uteis para os novos nós que desejam ingressar da rede em questão.

4.1 Arquitetura de Votação com Servidor

Na Figura 4.2 é mostrada a arquitetura do sistema de votação utilizando o servidor para gerenciar o processo. Na etapa 1, o dispositivo criador da eleição envia para o servidor duas informações: a lista de candidatos da votação e o retorno da função *hash(key)*[4]. A *key* é a chave que um eleitor precisa saber para poder votar em uma determinada eleição com servidor. O servidor não recebe qual é a *key* escolhida pelo criador da eleição, apenas o *hash(key)*.

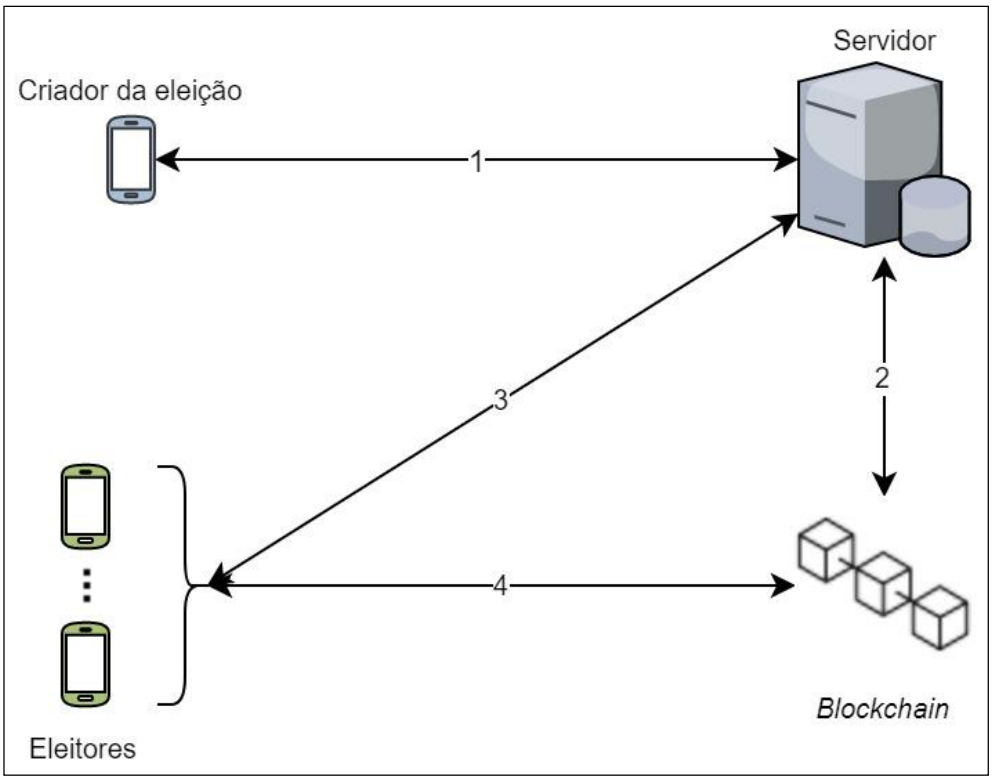

Figura 4.2: Arquitetura de votação utilizando o servidor. Fonte: elaborada pelo autor, 2019.

Na etapa 2, o servidor acessa a *blockchain* e implanta o contrato inteligente, recebendo como retorno o endereço onde o contrato foi implantando. Tendo em posse o *hash*(*key)* da eleição e o endereço onde ela está implantada, o servidor persiste estas duas informações no seu banco de dados.

Na etapa 3 os eleitores devem acessar o servidor para configurar seu acesso à *blockchain*. Cada eleitor deve enviar ao servidor o *hash(key)* da eleição que votar deseja votar

[4] A função *hash(str)* calcula e retorna o *hash* da *string* recebida como parâmetro usando o algoritmo *SHA256* (*Secure Hash Algorithm*, ou Algoritmo de *Hash* Seguro).

e o endereço da carteira que ele irá utilizar para votar. O servidor acessa novamente a *blockchain*, autoriza o voto do cliente na eleição e envia fundos para a carteira que o usuário irá utilizar.

Na etapa 4, o eleitor acessa diretamente a *blockchain* e realiza seu voto, já sem a interferência do servidor. A inserção do servidor é uma característica negativa no *DApp*, pois, é um ponto crítico de falha e sua queda impossibilita que eleições sejam realizadas utilizando este método.

Na Figura 4.4 , o servidor tem a responsabilidade de informar aos clientes o endereço onde o contrato está implantando, e também, autorizar o voto do cliente na *blockchain*. O protocolo utilizado para a comunicação entre aplicação cliente e servidora é o RESTful (*Representational State Transfer*, ou Transferência de Estado Representacional). A aplicação RESTful, rodando no servidor, aceita basicamente 3 funções: *registro*, *login* e *config*.

A função registro recebe como atributo a *string hash(concat(login, senha)*[5]*)* e insere esta entrada no banco de dados do servidor. A vantagem de receber somente o *hash* é que, o servidor, em nenhum momento, tem conhecimento do *login* e da senha do usuário que está usando o sistema, garantindo o anonimato.

A função *login* também recebe a *string hash(concat(login, senha))* como parâmetro e retorna *true* se a *string* estiver no banco de dados do servidor, e *false* caso a *string* não esteja no banco de dados.

A função *config* também recebe como parâmetro a *string hash(concat(login, senha))* para poder validar o usuário, além disso, a função também deve receber o endereço da carteira que o usuário vai utilizar para realizar o voto. Ao receber esta requisição, o servidor, insere fundos na carteira e autoriza o voto do usuário na *blockchain*. O retorno do servidor é o endereço onde o contrato está implantado, caso o usuário seja autenticado, e *null* caso o usuário não seja autenticado.

A função de configuração retorna todas as informações necessárias para o cliente poder acessar a *blockchain* e votar. Depois de configurado, o cliente já pode votar sem precisar do intermédio do servidor, acessando diretamente a *blockchain*.

Para um melhor entendimento da ordem em que as operações são realizadas, a Figura 4.4 demonstra cronologicamente a sequência em que as operações são realizadas na arquitetura de votação com servidor.

[5] A função *concat(args)* retorna a concatenação das *strings* recebidas como parâmetro.

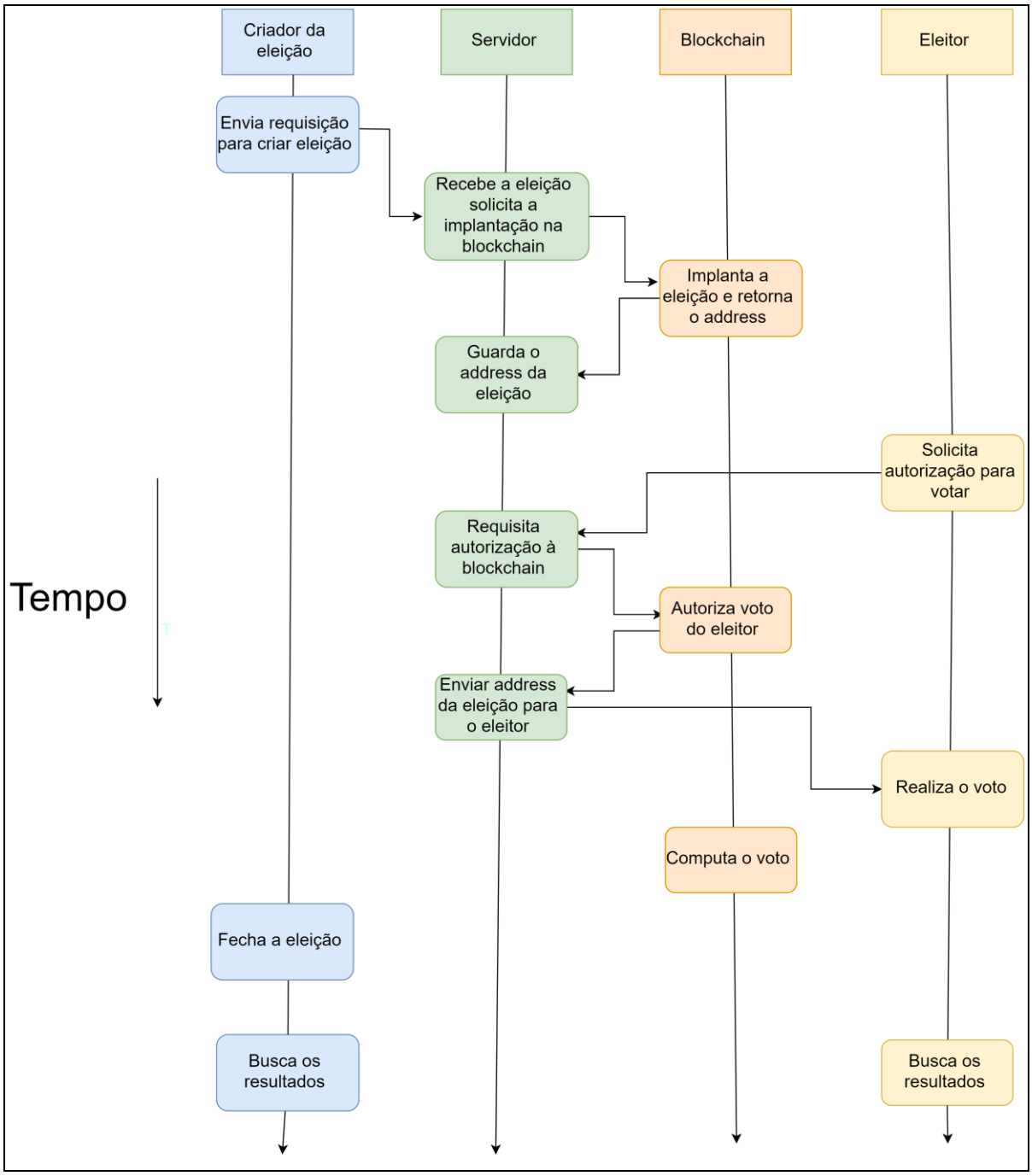

Figura 4.3: Ordem cronológica das operações na arquitetura de votação com servidor. Fonte: elaborada pelo autor, 2019.

4.2 Arquitetura de Votação sem Servidor

Na segunda arquitetura de votação, foi proposta uma técnica na qual o servidor é praticamente indispensável, proporcionando mais independência do sistema, além de maior tolerância a falhas. A segunda arquitetura de votação é mostrada na Figura 4.4. Na etapa 1, o próprio criador da eleição acessa a *blockchain* e implanta o contrato inteligente. Na etapa 2, deve haver uma comunicação manual entre o criador da eleição e os eleitores. O criador da eleição deve informar aos eleitores o endereço onde o contrato da eleição está implantado e autorizar o voto de cada um dos eleitores.

Na etapa 3 os eleitores já podem acessar diretamente a blockchain e votar, sem a necessidade do servidor e do criador da eleição.

A segunda arquitetura precisa de uma condição para que funcione totalmente sem a necessidade do servidor: o criador da eleição precisa ter fundos para enviar para seus eleitores.

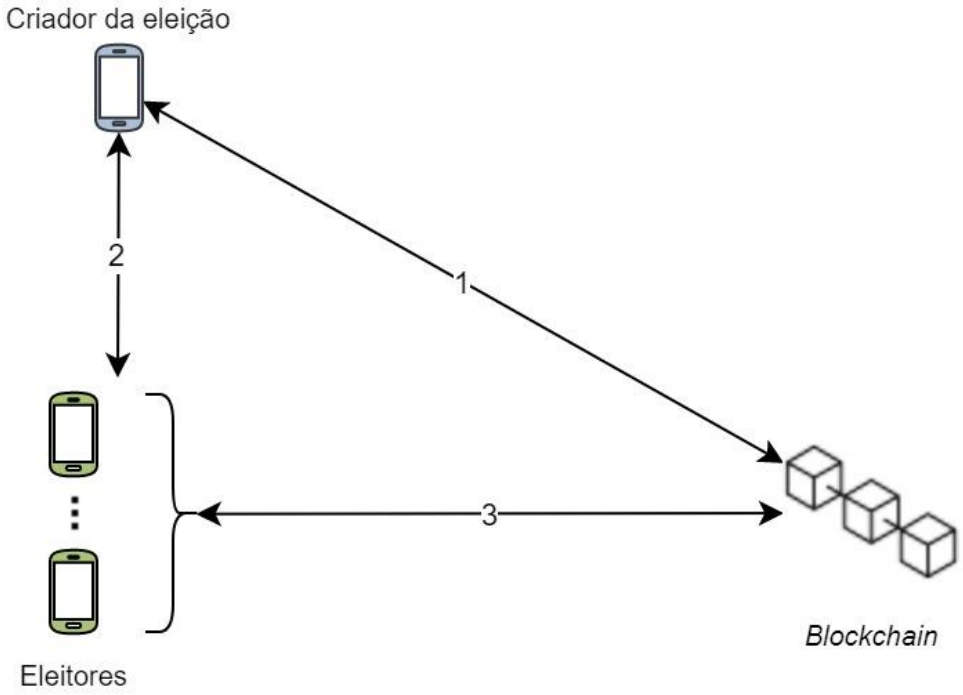

Figura 4.4 Arquitetura de votação sem o servidor. Fonte: elaborada pelo autor, 2019.

Nas Figuras Figura 4.2 e Figura 4.4, dentro da *blockchain*, está rodando um contrato inteligente escrito em Solidity. Neste contrato estão implementadas todas as regras relativas à integridade da votação. O contrato não deve permitir irregularidades na votação, todas as suas regras precisam ser idóneas, sem *backdoors* ou erros na lógica da votação. A integridade de

votação depende, integralmente, da idoneidade do contrato, sendo ele, a única parte que tem sua segurança garantida pela *blockchain*.

Uma vez implantado na *blockchain*, um contrato não pode mais ser substituído, e quando sua substituição é permitida, o evento de substituição é salvo imutavelmente na *blockchain*, permitindo a fácil auditoria e detecção de possíveis fraudes de adulteração de contrato.

Para um melhor entendimento da ordem em que as operações são realizadas, a Figura 4.5 demonstra cronologicamente a sequência em que as operações são realizadas na arquitetura de votação sem servidor.

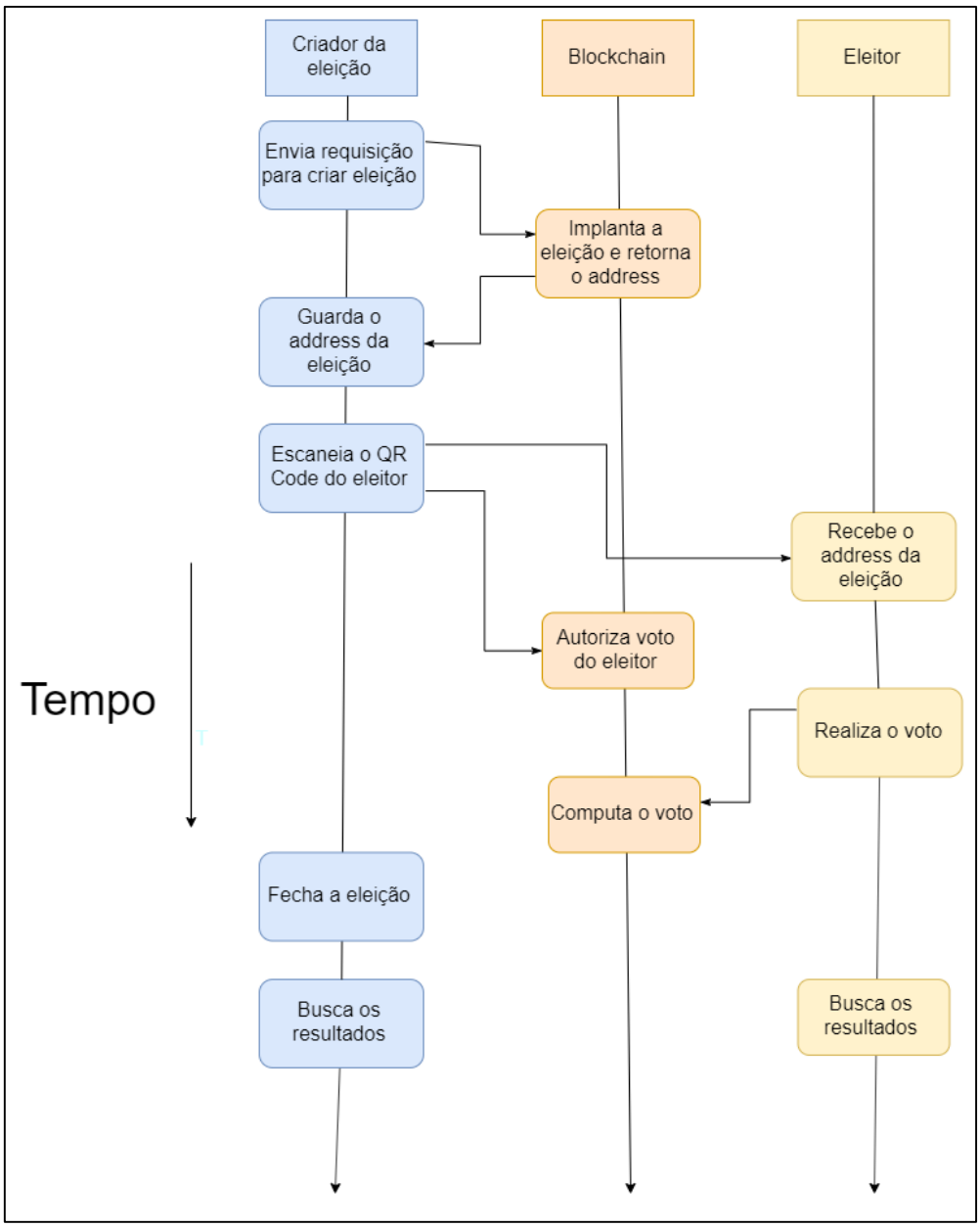

Figura 4.5: Ordem cronológica das operações na arquitetura de votação sem servidor. Fonte: elaborada pelo autor, 2019.

4.3 Aplicativo Móvel

Um dos componentes essenciais do sistema é o Vochain, aplicativo móvel utilizado pelos usuários. Sua função é se comunicar com a *blockchain* e com o servidor, permitindo que os usuários criem uma nova eleição (com ou sem servidor), autorizem o voto dos eleitores, vejam seu histórico de eleições criadas e votadas entre outras funcionalidades. O aplicativo foi desenvolvido exclusivamente para a plataforma Android, utilizando a plataforma de desenvolvimento Android Studio.

A utilização do aplicativo possibilita que os usuários tenham mais comodidade durante a votação, permitindo que os eleitores votem mesmo estando geograficamente separados. Por ser desenvolvido para a plataforma Android, o aplicativo, no Brasil, é compatível com cerca de 84% dos *smartphones* do País (STATISTA, 2018). No aplicativo o usuário pode realizar as seguintes operações

1. Criar uma nova conta,
2. Realizar *login* utilizando uma conta já criada, ou *login* anônimo,
3. Votar,
4. Criar uma nova eleição,
5. Visualizar as eleições criadas,
6. Visualizar as eleições votadas,
7. Solicitar fundos ao servidor,
8. Consultar o saldo atual da sua carteira Ethereum,
9. Autorizar o voto de um eleitor em uma determinada eleição,
10. Fechar uma eleição,
11. Ver o resultado final de uma eleição.

O aplicativo utiliza o servidor para guardar as informações do usuário, como não se pode garantir a idoneidade do servidor, o aplicativo sempre envia os dados criptografados para o servidor, utilizando o algoritmo AES (*Advanced Encryption Standard*, ou Padrão de Criptografia Avançada) e *cancat(login, senha)* como chave no algoritmo.

A disponibilidade do servidor também é um ponto que deve ser levado em conta pelo aplicativo. Como o *app* permite um método de votação que minimiza o uso do servidor (denominado aqui Arquitetura Sem Servidor), caso haja uma "queda" do servidor, o sistema não é capaz de acessar as informações de um determinado usuário, impossibilitando sua

autenticação. Mesmo que a autenticação fosse possível, o servidor necessitaria de algum mecanismo de sincronização para ordenar cronologicamente as operações feitas por um mesmo usuário, em dispositivos diferentes, caindo em um complicado problema de *sincronização*.

A solução proposta para este problema é permitir apenas *logins* anônimos quando o servidor não estiver disponível. Os dados de um usuário anônimo são armazenados apenas localmente.

Esta solução implica que um determinado usuário, que fizer *login* anônimo, só poderá gerenciar uma eleição pelo mesmo dispositivo onde a eleição foi criada. Embora, por uma perspectiva, isto seja um ponto negativo, por outra, esta característica aumenta consideravelmente o anonimato nas votações que utilizem esta arquitetura de votação.

4.4 Contrato Inteligente

Para o desenvolvimento de um *DApp*, o contrato inteligente costuma ser o primeiro código desenvolvido, porém, ele só pode ser realmente testado quando a aplicação cliente estiver desenvolvida, por tal motivo, optou-se por comentar sobre o desenvolvimento do contrato inteligente apenas após os capítulos referentes ao desenvolvimento da aplicação cliente.

Para a implementação do contrato inteligente foi utilizada a IDE (*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado) Remix. O Remix já traz consigo um conjunto modesto de ferramentas que ajudam no desenvolvimento de contratos inteligentes para a plataforma Ethereum.

O contrato inteligente utilizado neste trabalho é baseado em um dos códigos de exemplo fornecidos pelo Remix mesclado com algumas melhorias e adaptações. Os métodos principais do contrato e suas respectivas funções são:

- `vote(int proposal)`: vota para o candidato recebido como parâmetro,
- `giveRightToVote(string address)`: autoriza o voto do eleitor recebido como parâmetro. Apenas o criador da eleição consegue executar esta função,
- `proposal(int proposal)`: retorna o nome e a quantidade de votos do candidato recebido como parâmetro, se a eleição já estiver fechada,
- `close()`: fecha a eleição, impossibilitando novos votos. Apenas o criador da eleição consegue executar esta função,

- `chairPerson()`: retorna o *address* do criador da eleição,
- `quantCandidatos()`: retorna a quantidade de candidatos.

O contrato inteligente é a única parte do sistema que roda dentro da *blockchain*, nele devem ser implementadas as regras de negócio da eleição. No contrato são implementadas as restrições que impedem dualidade do voto, o fechamento não autorizado da eleição, o voto de eleitores sem permissão, entre outras restrições. A Figura 4.6 mostra a implementação da função `vote()`, expondo as verificações que são aplicadas antes de se computar um voto de um eleitor:

```
// acao executada quando um eleitor vota em um candidato
function vote(uint proposal) public {
    require ( // permite o voto apenas se a eleição não estiver fechada
        closed == false,
        "Eleicao ja encerrada."
    );
    Voter storage sender = voters[msg.sender]; // recupera o eleitor do vetor
    require(!sender.voted, "Already voted."); // verifica se o eleitor não votou
    sender.voted = true; // indica que o eleitor ja votou
    sender.vote = proposal;

    proposals[proposal].voteCount += sender.weight; // computa o voto
}
```

Figura 4.6 Implementação da função `vote()`. Fonte: elaborada pelo autor, 2019.

5 AVALIAÇÃO E RESULTADOS

5.1 Estudo de Caso

Para demonstrar o funcionamento do sistema desenvolvido, será apresentado um estudo de caso onde será criada uma eleição com servidor e outra eleição sem servidor, em seguida um eleitor irá votar nas duas eleições e posteriormente a eleição será fechada e os resultados serão recuperados. As capturas de tela e as demais explicações serão apresentadas nas sessões seguintes.

5.1.1 Criação das Eleições

Nesta etapa foram criadas as eleições com servidor e sem servidor. A Figura 5.1 mostra a criação de uma eleição com servidor. A eleição com servidor tem 3 candidatos e a chave é: **cripto**.

Figura 5.1 Criação de uma eleição com servidor. Fonte: elaborada pelo autor, 2019.

A Figura 5.2 mostra a criação de uma eleição sem servidor. A eleição sem servidor tem 3 candidatos e não possui chave.

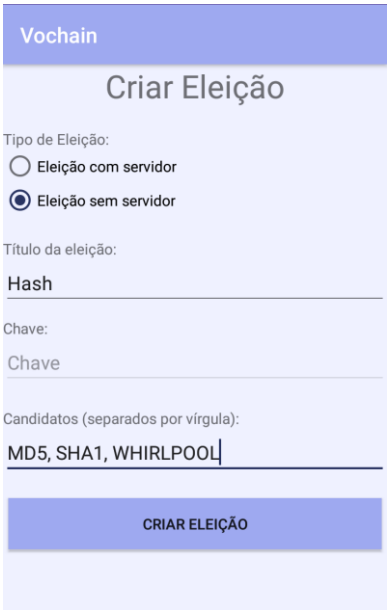

Figura 5.2 Criação de uma eleição sem servidor. Fonte: elaborada pelo autor, 2019.

5.1.2 Realização do Voto

Para realizar o voto com servidor, o eleitor deve inserir o título e posteriormente, a chave da eleição. A Figura 5.3 mostra a tela após o eleitor inserir o título: **Criptomoedas** e a chave: **cripto**.

Figura 5.3 Votação em uma eleição com servidor. Fonte: elaborada pelo autor, 2019.

Após inserir o título e a chave corretos, o eleitor é direcionado para a tela de votação. A Figura 5.4 mostra tela para escolher o candidato. Após selecionar o candidato, e clicar em votar, o voto é submetido à *blockchain*, e posteriormente, computado.

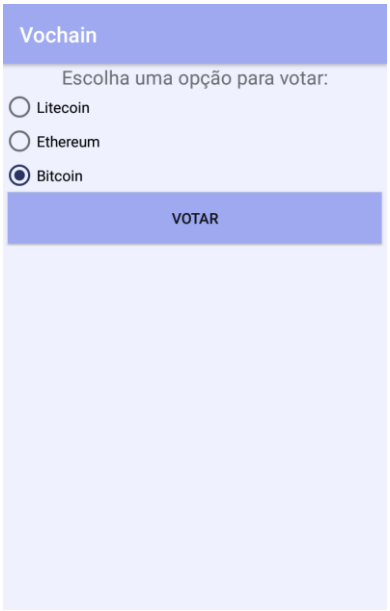

Figura 5.4 Tela de seleção de candidato. Fonte: elaborada pelo autor, 2019.

Para realizar o voto em uma eleição sem servidor, o eleitor deve inserir o *address* da eleição, ou escanear um QR Code (*Quick Response Code*, ou Código de Resposta Rápida) que contenha um *address* válido, em seguida o eleitor deve ser ativado pelo criador da eleição. Para ativar um eleitor é necessário que o criador da eleição escaneie o *address* do eleitor que deseje ser ativado. A Figura 5.5 mostra a tela onde o *address* é inserido, e onde se verifica se existe uma eleição naquele *address*.

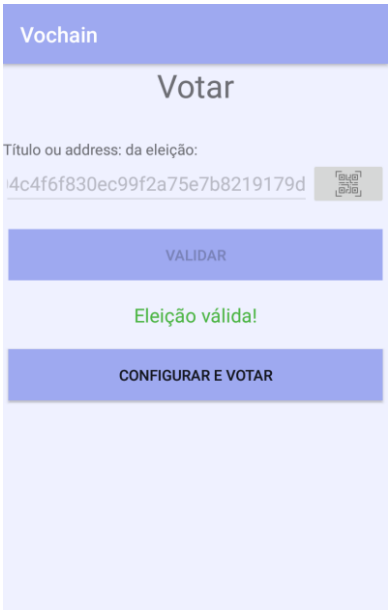

Figura 5.5 Tela para inserção do *address* da eleição. Fonte: elaborada pelo autor, 2019.

A ativação do eleitor é feita na tela mostrada na Figura 5.6, nela é possível ativar um ou mais eleitores de uma única vez.

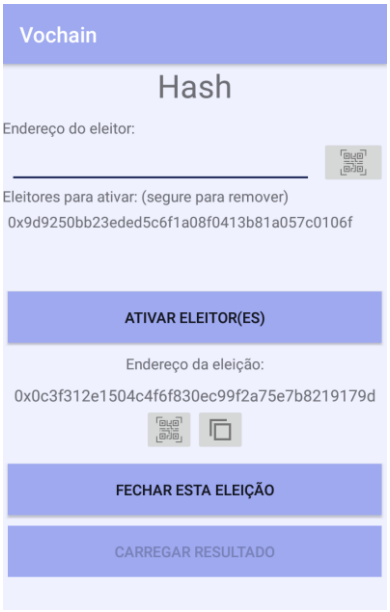

Figura 5.6 Tela para ativação dos eleitores. Fonte: elaborada pelo autor, 2019.

Após um eleitor ser ativado, ele já pode votar, o voto é feito na tela mostrada na Figura 5.7.

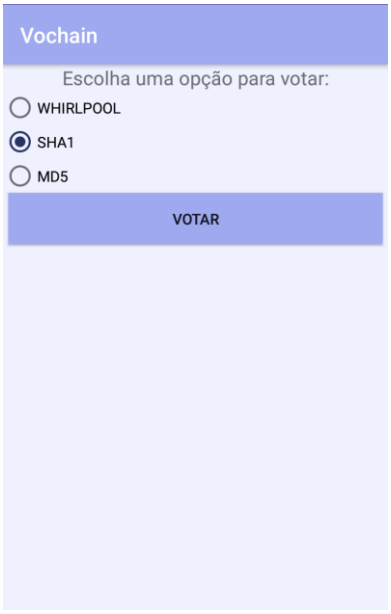

Figura 5.7 Voto na eleição sem servidor. Fonte: elaborada pelo autor, 2019.

5.1.3 Fechamento da Eleição e Recuperação dos Resultados

O fechamento de uma eleição só pode ser realizado pelo seu criador, e após fechada, uma eleição não poderá mais receber votos, e os resultados já podem ser consultados pelo criador ou por qualquer eleitor que tenha votado na eleição em questão. A Figura 5.8 mostra o fechamento da eleição com servidor e sem servidor, respectivamente. A Figura 5.9 mostra os resultados de ambas as eleições.

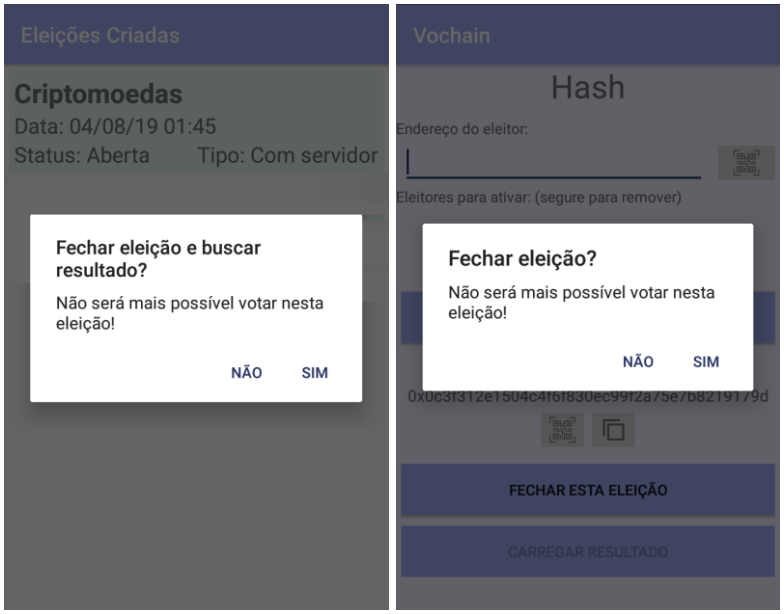

Figura 5.8 Fechamento da eleições com servidor e sem servidor, respectivamente. Fonte: elaborada pelo autor, 2019.

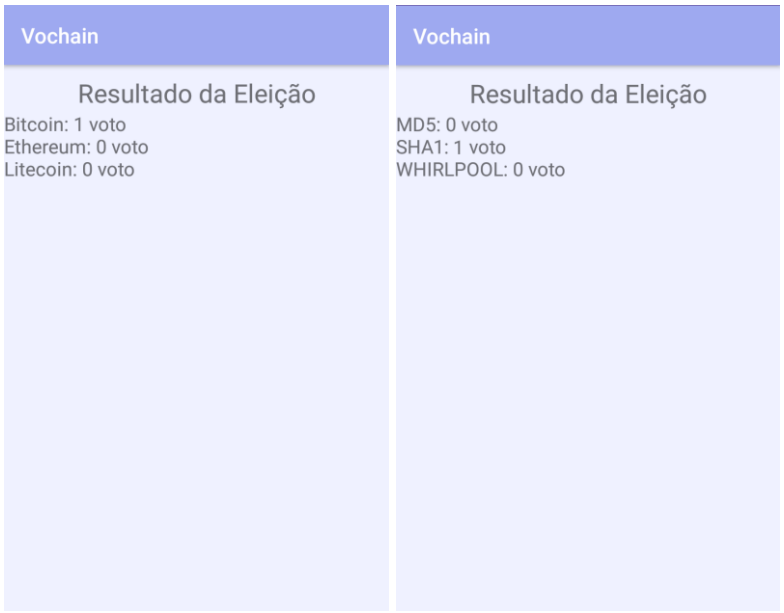

Figura 5.9 Resultados obtidos em ambas as eleições. Fonte: elaborada pelo autor, 2019.

5.2 Análise de Custo

As *blockchains* Turing-Completas permitem que códigos-fonte sejam executados dentro da própria *blockchain* e essa funcionalidade tem seu custo. Cada criptomoeda implementa uma forma diferente de monetizar a execução dos códigos-fonte. A criptomoeda Ethereum cobra uma taxa que varia de acordo com a quantidade de recursos computacionais que serão alocados durante a execução de uma instrução, desta forma, a execução de funções que contenham grandes laços de repetição ou cálculos matemáticos complexos devem ser evitadas para diminuir os custos de execução.

O ”combustível” utilizado na *blockchain* do Ethereum é o *ether*. Quando aplicações interagem com um contrato, enviam recursos ou realizam qualquer outra operação que envolta a *blockchain*, a aplicação deve pagar por esta computação. Esse pagamento é calculado em gás, e o gás é sempre pago em *ethers* (ETH GAS STATION, 2019).

A aplicação paga pelo cálculo que está sendo realizado na *blockchain*, independentemente se a transação obteve êxito ou não. Mesmo quando a execução de uma transação falha, os mineradores devem validar e executar a transação, o que requer poder computacional e consequentemente, custo (ETH GAS STATION, 2019).

A quantidade de *ethers* oferecida para executar uma transação deve refletir a rapidez com que se deseja que a transação seja minerada. Caso um preço muito baixo seja oferecido, a transação pode demorar muito para ser executada ou até mesmo não ser executada. Embora oferecer um preço mais alto do possa acelerar a confirmação de uma transação, há um limite para a aceleração, pois oferecer um preço mais alto do que o necessário dificilmente acelera o tempo da transação em circunstâncias normais Eth Gas Station (2019).

Para evitar custos demasiados, toda transação tem um limite de gasto. O limite de gasto é utilizado para evitar que uma determinada transação entre em *loop* infinito e consuma todos os recursos da carteira que executou a transação. Caso o custo de uma transação atinja o limite, a sua execução é parada, mas os recursos gastos não são reembolsados. Quando a execução de uma transação é finalizada, caso existem recursos excedentes, eles são devolvidos.

O contrato inteligente do Vochain foi desenvolvido com a ideia de reduzir execuções de código desnecessárias e verificações redundantes para evitar gastos supérfluos. Os custos de uma eleição variam de acordo com a quantidade de eleitores, a Figura 5.10 mostra o crescimento linear do custo total de uma eleição em relação a sua quantidade de eleitores.

O custo de uma eleição é diretamente proporcional à quantidade de eleitores que participam da eleição, quanto mais eleitores, maior o custo.

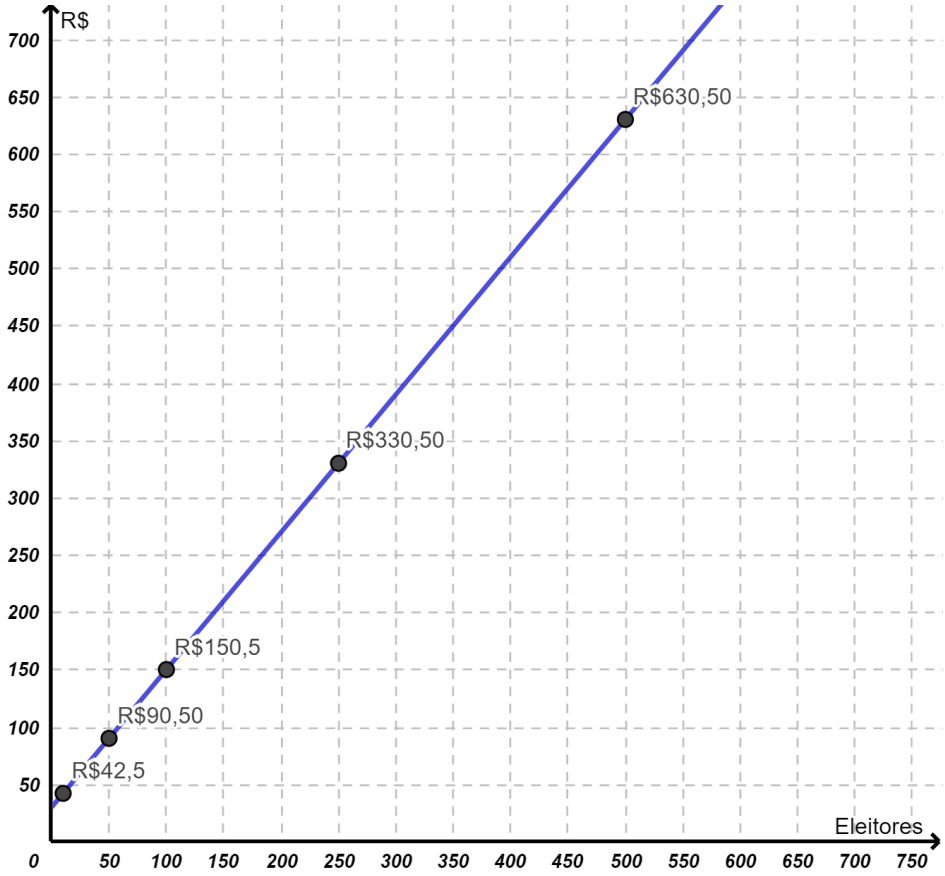

Figura 5.10 Função de custo para eleições com 10, 50, 100, 250 e 500 eleitores, respectivamente. Fonte: elaborada pelo autor, 2019.

A análise de custo mostrada na Figura 5.10 leva em consideração a cotação do Ethereum no dia 20/08/2019, no qual 1 ETH era equivalente a R$797, os custos de uma eleição real podem variar de acordo com a cotação da moeda e com as taxas cobradas para execução das funções no momento da realização da eleição. A função abaixo pode ser utilizada para calcular o custo médio de uma eleição:

$$C(n) = k_i + k_f + k_a * n + k_v * n$$

Onde, k_i é o custo de implantação da eleição na *blockchain*, k_f é o custo para fechar a eleição, k_a é o custo para a ativação de um eleitor, k_v é o custo para um eleitor votar e n é a quantidade de eleitores. Mais informações sobre as taxas do Ethereum podem ser encontradas em Eth Gas Station (2019).

Implantar a eleição consiste em colocar o contrato inteligente em um *address* na *blockchain*. Ativar um eleitor significa acessar o contrato implantado na *blockchain* e autorizar o voto do eleitor. A operação de ativação precisa ser realizada para cada eleitor de forma individual. Fechar uma eleição significa acessar o contrato, na *blockchain* e alterar o valor de uma variável booleana responsável por autorizar novos votos e liberar os resultados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Com a finalização do trabalho, foi possível observar o grande poder e as inúmeras possibilidade que a *blockchain* fornece. Os sistemas de votação são umas das aplicações que tem grande potencial e podem ser a substituição das votações em papel e das urnas eletrônica em todo o mundo.

O trabalho atingiu seu objetivo, o sistema desenvolvido foi capaz de realizar eleições utilizando *blockchain*. As duas arquiteturas de votação propostas pelo sistema funcionaram como esperado e serviram para analisar os prós e contras relacionados com arquiteturas que utilizam ou não servidores para gerenciar transações.

O Vochain foi testado na *blockchain* de teste de Ethereum. A *blockchain* de teste é de menor porte, portanto não fornece as garantias fornecidas pela *blockchain* de produção da criptomoedas. Para a realização de uma eleição com maior grau de segurança, é necessário colocar o sistema na *blockchain* de produção.

Embora o sistema desenvolvido tenha atingido seu objetivo, algumas melhorias e adaptações são necessárias para colocar o sistema em produção em eleições reais de médio ou grande porte. Melhorias de usabilidade e interface deixariam o sistema mais fácil de ser utilizado pelos seus usuários.

A demora nas requisições feitas à *blockchain* deixaram o sistema lento em algumas partes, encontrar formas de acelerar as respostas da *blockchain* ou uma *blockchain* mais rápida agilizaria o processo de votação.

Referências

AGORA. **Bringing voting systems into the digital age**. Disponível em: < https://www.agora.vote/>. Acesso em: 09 set. 2019.

Arrow, Kenneth. **Social choice and individual values**. John Wiley & Sons, Nova York. 1951. Disponível em: <https://cowles.yale.edu/sites/default/files/files/pub/mon/ m12-all.pdf>. Acesso em: 15 ago. 2018.

BLOCK GEEKS. **Smart contracts**: the blockchain technology that will replace lawyers. Disponível em: <https://blockgeeks.com/guides/smart-contracts/>. Acesso em: 13 ago. 2018.

BLOCKCHAIN HUB. **Blockchains & distributed ledger technologies**. Disponível em: <https://blockchainhub.net/blockchains-and-distributed-ledger-technologies-in-general/>. Acesso em: 13 ago. 2018.

BRASIL. Tribunal Superior Eleitoral **Urna eletrônica**: 20 anos a favor da democracia. Brasília. 2016.

COULOURIS, G. et al. **Sistema distribuídos**: Conceitos e projeto. 4. ed. Porto Alegre: Bookman, 2013. 1055 p.

GORENDER, Sérgio; MACÊDO, Raimundo. **Consenso distribuído eficiente no modelo síncrono particionado**. Universidade Federal da Bahia, Salvador, 2011.

ETH GAS STATION. **Consumer oriented metrics for the Ethereum gas market**. Disponível em: <https://ethgasstation.info/>. Acesso em: 09 set. 2019.

FOLLOW MY VOTE. **Introducing a secure and transparent online voting solution for the modern age**. Disponível em: <https://followmyvote.com/>. Acesso em: 09 set. 2019.

LEWIS, Antony. **A gentle introduction to blockchain technology**. Brave new coin, New Zealand, 2015. Disponível em: <https://bravenewcoin.com/assets/reference-papers/a-gentle-introduction/a-gentle-introduction-to-blockchain-technology-web.pdf>. Acesso em: 12 ago. 2018.

NAKAMOTO, Satoshi. **Bitcoin**: a peer-to-peer electronic cash system, [S.L.], 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: 10 ago. 2018.

NATIONAL DEMOCRATIC INSTITUTE. **Electronic voting and counting around the world**. Disponível em: <https://www.ndi.org/e-voting-guide/electronic-voting-and-countingaround-the-world>. Acesso em: 11 set. 2018.

NURMI, Hannu. **Voting systems for social choice**. Academy of Finland and University of Turku, Turku - Finlândia, 2014. Disponível em: <https://www.utu.fi/fi/yksikot/soc/yksikot/pcrc/huippu/julktoiminta/tyopaperit/documents/nurmi4[1]_12.pdf>.Acesso em: 11 set. 2018.

O'CONNOR, J J; ROBERTSON, E F. **The history of voting**. Mactutor history of mathematics, [S.L], 2002. Disponível em: <http://www-history.mcs.st-andrews.ac.uk/histtopics/voting.html>. Acesso em: 10 set. 2018.

POLYS. **Online voting system**. Disponível em: <https://polys.me/>. Acesso em: 09 set. 2019.

SCHAUREN, Luís Fernando. **Segurança no sistema brasileiro de votação eletrônica**. 2016. 93 f. TCC (Graduação) - Curso de Ciência da Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2016. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/151030/ 001009822.pdf?sequence=1>. Acesso em: 11 set. 2018.

STALLINGS, William. **Criptografia e segurança de redes**: princípios e práticas. 6 ed. São Paulo: Pearson Education do Brasil, 2015.

STATISTA. **Market share of mobile operating systems in Brazil from January 2012 to December 2018**. Disponível em: < https://www.statista.com/statistics/262167/market-share-held-by-mobile-operating-systems-in-brazil/>. Acesso em: 15 mar. 2019.

TANENBAUM, Andrew Stuart; STEEN, Maarten Van. **Sistemas distribuídos**: princípios e paradigmas. 2 ed. São Paulo: Pearson Education do Brasil, 2007.