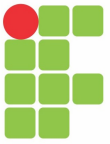


Instituto Federal de Educação, Ciência e Tecnologia de Alagoas
Campus Maceió
Coordenação de Informática
Curso Superior de Bacharelado em Sistemas de Informação

Arthur Denner Oliveira Santos
Djalma do Nascimento Júnior

Um Sistema de Eleição Antifraude Baseado em *Blockchain* Permissionada com o *Hyperledger Fabric* e *Composer*

Maceió
2019

Arthur Denner Oliveira Santos
Djalma do Nascimento Júnior

Um Sistema de Eleição Antifraude Baseado em *Blockchain* Permissionada com o *Hyperledger Fabric* e *Composer*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Alagoas como requisito parcial para obtenção do título de bacharel em Sistemas de Informação.

Orientador: MSc. Breno Jacinto Duarte da Costa

Maceió
2019

INSTITUTO
FEDERAL
Alagoas

Dados Internacionais de Catalogação na Publicação
Instituto Federal de Alagoas
Campus Maceió
Biblioteca Benevides Monte

S237s *Santos*, Arthur Denner Oliveira.
Um sistema de eleição antifraude baseado em blockchain Permissionada com o hyperledger Fabric e Composer / Arthur Denner Oliveira Santos, Djalma do Nascimento Júnior. - Maceió : IFAL, 2019.
64 f. : il.
1 CD-ROM: il., col.; (1 arquivo : 3.405 kilobytes)..

Orientador: Profº Me. Breno Jacinto da Costa.
Trabalho de Conclusão de Curso (Bacharelado em Sistema de Informação) - Instituto Federal de Alagoas, Campus Maceió. Maceió, 2019.
.
CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico, acondicionada em caixa acrílica (12,5 cm x 14 cm).

1. Sistema de Informação. 2. Tecnologia Blockchain. 3. Sistema de eleições Online. 4. Hyperledger Fabric e Composer – Ferramenta. I. Nascimento Júnior, Djalma. II. Título.

*CDD**: 005.436***

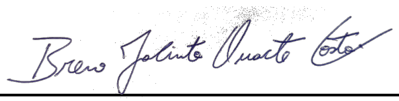
Nalva Maria Amaral
Bibliotecária – CRB-4/989

Arthur Denner Oliveira Santos
Djalma do Nascimento Júnior

Um Sistema de Eleição Antifraude Baseado em *Blockchain* Permissionada com o *Hyperledger Fabric* e *Composer*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Alagoas como requisito parcial para obtenção do título de bacharel em Sistemas de Informação.

BANCA AVALIADORA



MSc. Breno Jacinto Duarte da Costa
(Orientador)
IFAL - Campus Maceió

MSc. Jailton Cardoso da Cruz
IFAL - Campus Maceió

PhD. Fábio Paraguaçu Duarte da Costa
UFAL - Campus Maceió

Maceió
2019

AGRADECIMENTOS

ARTHUR DENNER OLIVEIRA SANTOS

Agradeço primeiramente a Deus por proporcionar ótimas oportunidades na minha vida e permitir que eu cresça cada dia mais.

À minha mãe por confiar e sempre acreditar em mim, além de fornecer um apoio incondicional em momentos muito importantes da minha vida.

A todos os meus familiares por sempre desejarem o meu melhor, por me ensinarem bons valores e pelos ótimos momentos proporcionados.

Aos meus amigos do "Nóis Quatro" por toda a ajuda no decorrer do curso e pelos momentos de descontração essenciais para enfrentar os desafios do curso.

Ao meu amigo Djalma pela sua contribuição e esforço neste trabalho, além de toda a troca de conhecimento realizada enquanto trabalhamos juntos.

Ao meu amigo e orientador, Breno Jacinto Duarte da Costa, por me proporcionar uma chance para ingressar na área e mudar completamente de vida, por todos os bons conselhos e por sempre torcer por mim.

A todos os professores do curso de Sistemas de Informação pelos seus trabalhos, incentivos e ensinamentos, dentro e fora das salas de aula.

A todos os amigos que conheci durante o curso, na faculdade e no exercício da profissão e que, direta ou indiretamente, ajudaram a formar quem eu sou hoje.

AGRADECIMENTOS

DJALMA DO NASCIMENTO JÚNIOR

Agradeço primeiramente a Deus, pelo dom da vida, pela sua graça à humanidade e por sustentar aqueles que clamam por sua presença com sinceridade (Salmos 145:18).

À minha mãe, Rosa Elisa, por todo seu esforço ao criar/educar minha irmã e a mim... Posso afirmar que não foi fácil :-)!

À minha esposa, Leidiane, por todo o seu suporte em todos os momentos. Te amo <3!

À minha família, pelos momentos prazerosos de conversas.

Ao meu amigo Arthur Denner pela sua contribuição e esforço neste trabalho.

Ao Instituto Federal de Alagoas (IFAL) pela oportunidade concedida para realização da graduação.

Ao meu amigo e orientador, Breno Jacinto Duarte da Costa, pelos vários anos de boa conversa, conselhos e inspirações.

A todos os professores do curso de Sistemas de Informação, pelos seus trabalhos e incentivos.

"Mera mudança não é crescimento.
Crescimento é a síntese de mudança e continuidade,
e onde não há continuidade não há crescimento."
(C. S. Lewis)

RESUMO

Blockchain é uma tecnologia com mais de 10 anos de existência e que está sendo bastante explorada nos dias atuais. Tendo suas primeiras aplicações em criptomoedas, hoje a tecnologia está sendo testada em diversos cenários públicos e privados, desde a área de saúde e educação até sistemas de votação. Tomando o último exemplo, a tecnologia *blockchain* ajuda a reduzir os níveis de insatisfação e desconfiança dos eleitores com os métodos existentes de votação. Este trabalho, portanto, teve como objetivo a criação de um sistema de eleições *online* utilizando *blockchain* e, com isso, foi realizada uma análise comparativa entre esse sistema e um outro com modelagem cliente-servidor tradicional. Foi utilizada a metodologia *Design Science Research* (DSR) para a condução deste trabalho. Além disso, foram utilizados o *framework Hyperledger Fabric* e o conjunto de ferramentas *Hyperledger Composer* para a criação da rede *blockchain*.

Palavras-chaves: blockchain, distributed ledger technology, election, vote

ABSTRACT

Blockchain is a technology with more than 10 years of existence and is being explored a lot these days. Having its first applications in cryptocoins, today the technology is being tested in several public and private environments, from the health and education area to voting systems. Taking the last example, blockchain technology helps reduce voter dissatisfaction and distrust with existing voting methods. This work aimed to create an online election system using blockchain and make a comparative analysis between the proposed system and another one made in traditional client-server modeling. Design Science Research (DSR) methodology was used to conduct this work. In addition, the Hyperledger Fabric framework and the Hyperledger Composer toolkit were used to create the blockchain network.

Keywords: blockchain, distributed ledger technology, election, vote

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DSR	Design Science Research
DLT	Distributed Ledger Technology
PoW	Proof of Work
PoS	Proof of Stake
BFT	Byzantine Fault Tolerance
PBFT	Practical Byzantine Fault Tolerance
SBFT	Simplified Byzantine Fault Tolerance
API	Application Programming Interface
REST	Representational State Transfer
DDoS	Distributed Denial of Service
PKI	Public Key Infrastructure
MD	Message Digest
SHA	Secure Hash Algorithm
DLT	Distributed Legder Technologies
MSP	Membership Service Provider
CA	Certification Authorities
RA	Registration Authority
VA	Validation Authority
LDAP	Lightweight Directory Access Protocol
CRUD	Create, Read, Update and Delete
CLI	Command Line Interface
BNC	Business Network Card
URL	Uniform Resource Locator
QP	Questões de Pesquisa

StArt	State of the Art through Systematic Reviews
LAPES	Laboratório de Engenharia de Software
UFSCAR	Universidade Federal de São Carlos
ID	Identity
FA	Fluxo Alternativo
FE	Fluxo de Exceção

Sumário

1 INTRODUÇÃO

Blockchain é uma tecnologia com mais de 10 anos de existência (NAKAMOTO et al., 2008) e que está sendo bastante explorada nos dias atuais. Tendo suas primeiras aplicações somente em criptomoedas, hoje a tecnologia está sendo testada em diversos cenários, públicos e privados, de saúde (METTLER, 2016) a educação (CHEN et al., 2018), passando por áreas antes dominadas pela arquitetura cliente-servidor tradicional.

O motivo desta explosão de interesse é que, com a tecnologia *blockchain*, aplicações que antes só funcionavam com um intermediário confiável, agora podem operar de maneira descentralizada, sem a necessidade de uma autoridade central, e alcançar as mesmas funcionalidades com a mesma certeza. *Blockchain* permite redes *trustless* porque os participantes podem fazer transações entre si mesmo que não confiem um no outro (CHRISTIDIS; DEVETSIKIOTIS, 2016).

Um clássico cenário onde há uma entidade atuando como intermediária nos processos – e uma alta onda de desconfiança por parte dos integrantes – é o de eleição.

Atualmente, muitos sistemas de votação sofrem de um sério problema de *design*: são controlados por um intermediário que controla o código, o banco de dados, as saídas do sistema e fornece as ferramentas de monitoramento ao mesmo tempo. A falta de transparência na implementação das regras desses sistemas dificulta a aquisição de confiabilidade exigida pelos eleitores e organizadores de eleições (NOIZAT, 2015). Essas preocupações existem em eleições públicas e privadas, realizadas dentro de uma organização.

O rápido desenvolvimento da *blockchain* como plataforma em diferentes indústrias e a enorme demanda de suas características e soluções tecnológicas levantaram a necessidade da visão ampla do caso de uso oferecido pela tecnologia no contexto dos negócios (KONSTANTINIDIS et al., 2018).

Hoje é possível escrever aplicações com camadas de permissão bem definidas (permissionadas), customização na validação das transações, entre outras características exigidas pelo negócio, mantendo todas as características de descentralização da tecnologia.

1.1 Justificativa

Dentre seus diversos projetos, a empresa Audora tem um sistema de eleições que é comercializado com um de seus clientes. Todos os requisitos da eleição estão atualmente no servidor da empresa, numa aplicação escrita com a linguagem *Java*. Mesmo com seus clientes confiando na empresa, a gerência tem como um de seus objetivos melhorar seus produtos e torná-los mais seguros, além de manter-se atualizada com o que o mercado

tem a oferecer.

Visando explorar uma tecnologia em ascensão, a partir da percepção dos autores, foi iniciado um estudo sobre *blockchains* para entender melhor o estado da arte da tecnologia, suas aplicações voltadas para negócios, quais ferramentas existem para tal finalidade e como a tecnologia *blockchain* pode ajudar a melhorar o sistema de eleições da Audora.

A escolha do *blockchain* como tecnologia, para o modelo proposto neste trabalho, deu-se pelas suas características de permitir uma estrutura confiável em uma rede não-confiável (a *Internet*), a ausência de uma autoridade central para o controle de informações, a possibilidade de automatizar alguns fluxos através de *smart contracts* e pelo desafio de usar uma nova tecnologia em um cenário relativamente incomum.

1.2 Objetivo Geral e Específico

O objetivo principal deste trabalho é fazer uma análise comparativa entre o atual sistema de eleição de modelagem cliente-servidor tradicional – da empresa Audora – e um feito em *blockchain* – proposto neste trabalho, destacando as vantagens e desvantagens da abordagem atual e da abordagem proposta, além de mostrar uma das possíveis aplicações da *blockchain* em soluções empresariais.

Com base no objetivo principal, os objetivos específicos são apresentados:

- Descrever os casos de uso para o sistema de eleições;
- Descrever a arquitetura do sistema atual com diagrama de classes e de atividades;
- Descrever a modelagem para o sistema *blockchain* proposto;
- Fazer uma análise comparativa entre as duas arquiteturas;
- Destacar as vantagens e desvantagens da abordagem atual e da abordagem proposta;
- Analisar se a abordagem proposta pode substituir a atual;
- Avaliar o resultado da análise comparativa.

1.3 Procedimentos Metodológicos

O projeto de pesquisa proposto adota o método *Design Science Research* (DSR) para a condução do trabalho, especificamente, o método DSR que (DRESCH; LACERDA; JR, 2014) propõe. Se trata, portanto, de uma pesquisa prescritiva. O diagrama das fases deste trabalho se encontra na Figura 1.

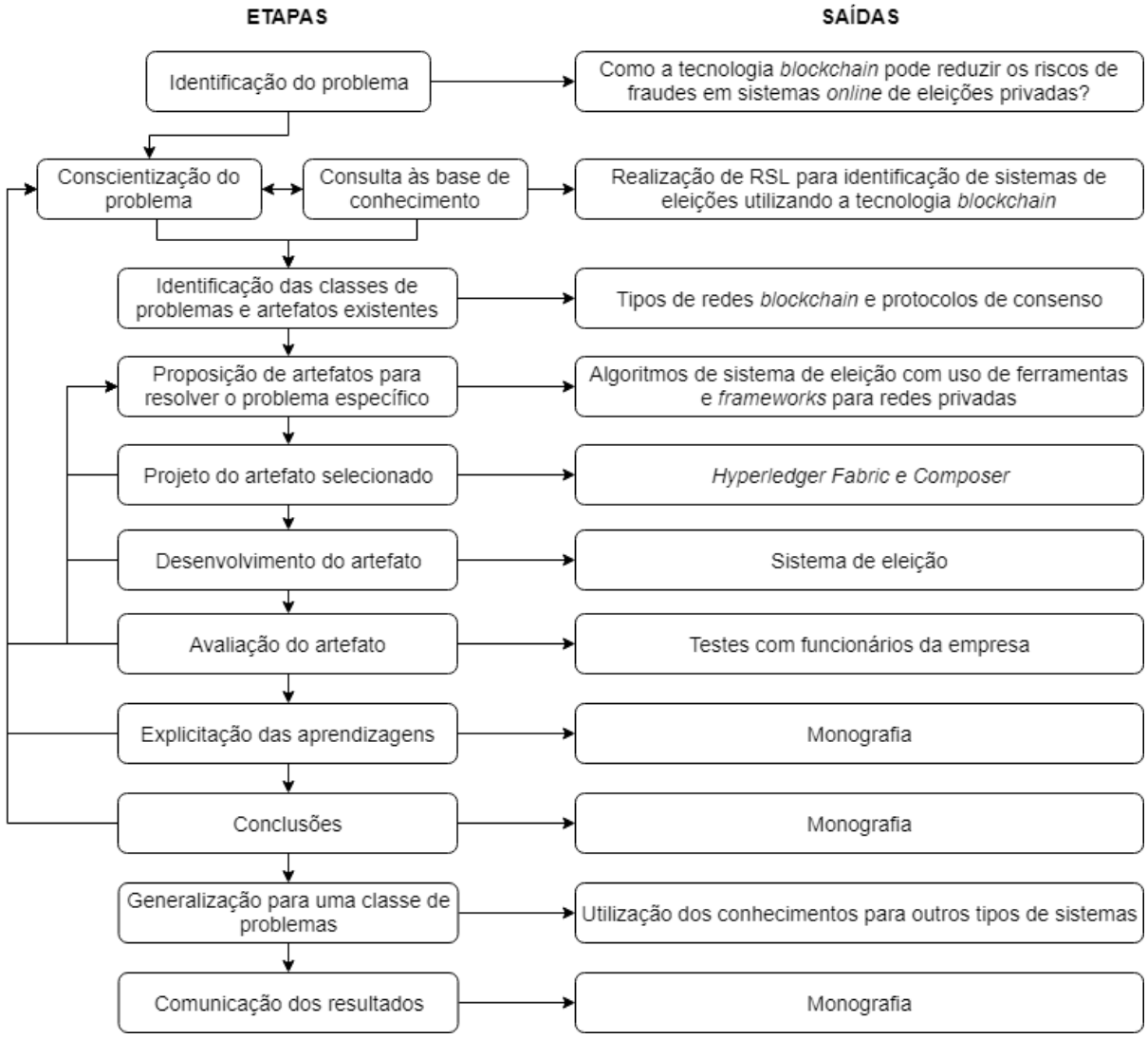


Figura 1: Estrutura da metodologia utilizando DSR

Fonte: Elaborado pelos autores.

1.4 Organização do Trabalho

Este trabalho está dividido em cinco capítulos. O capítulo 1 apresentou a contextualização do problema abordado neste trabalho, assim como os objetivos. No capítulo 2, o conteúdo necessário para o entendimento do trabalho, mostrando os conceitos básicos de *blockchain* e das tecnologias utilizadas. O capítulo 3 apresentará uma revisão sistemática da literatura, contendo os trabalhos relacionados. No capítulo 4, será descrito o desenvolvimento do sistema de eleição em *blockchain* e a comparação com sua contraparte tradicional. No capítulo 5, uma análise e discussão dos resultados e, por fim, no capítulo 6, será apresentado um resumo do que foi produzido neste trabalho, suas contribuições e possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Eleições

Eleições públicas são uma das bases sobre as quais a democracia é construída, portanto, é de suma importância que governos e organizações sejam capazes de realizar eleições não fraudulentas com sucesso (MOURA; GOMES, 2017).

Muitos países hoje aplicam diferentes mecanismos de voto. Os mecanismos tradicionais de voto, por exemplo, são limitados por fatores como distância dos locais de votação, área de registro eleitoral ou se o eleitor apresenta identificação adequada no ato da votação. Se o método de votação requer operação manual, controvérsias ou abusos podem ocorrer.

Por conta disso, vários países, como Califórnia e Austrália, desenvolveram sistemas eletrônicos de votação, acrescentando tecnologia de criptografia ao seu voto eletrônico para melhorar a segurança e a credibilidade desses sistemas (WU; YANG, 2018). Além de minimizar o erro humano, o voto eletrônico permite análise rápida dos dados.

Como bem observado por (WU; YANG, 2018), atualmente, muitos sistemas de voto eletrônico empregam servidores centralizados para processamento de dados; mesmo assim, servidores centralizados são suscetíveis a ataques maliciosos e ataques *DDoS*[1], então a arquitetura distribuída deve ser usada em vez disso, para reduzir o risco de desligamento pós-ataque.

Os métodos existentes de votação, eletrônicos ou não, fornecem níveis insatisfatórios de transparência, no sentido de que pode ser muito difícil ou impossível para um eleitor assegurar que o seu voto, que ele realizou durante a eleição, seja o mesmo contado nos resultados (MOURA; GOMES, 2017).

No sistema eleitoral brasileiro, com exceção da contagem de votos, nenhum dado de uma eleição é tornado público, o que significa que apenas funcionários do governo podem recontar os votos, se necessário, sem fornecer à população qualquer garantia de que não houve interferência externa no processo eleitoral. Para aumentar ainda mais a falta de confiança, já se foi provado que o software da urna eletrônica brasileira tem vulnerabilidades (MOURA; GOMES, 2017).

Como apontado por (ZHANG et al., 2018) e (ADIPUTRA; HJORT; SATO, 2018), um sistema de votação *online* ideal tem as seguintes características desejáveis mais importantes:

- **Disponibilidade:** Um sistema de voto eletrônico deve permanecer disponível durante

[1] <https://en.wikipedia.org/wiki/Denial-of-service_attack>

toda a eleição e deve servir eleitores conectando-se a partir de seus dispositivos;

- **Elegibilidade:** Apenas eleitores elegíveis devem poder votar; e apenas um voto por contagem de eleitores;
- **Integridade:** Os votos não devem ser modificados, forjados ou excluídos sem detecção;
- **Auditoria:** Deve ser possível verificar se todos os votos foram corretamente contabilizados;
- **Anonimato:** A conexão entre o voto de um usuário e o próprio usuário não deve ser rastreável sem sua ajuda (e de preferência nem mesmo com a ajuda dele);
- **Justiça:** Os resultados (parciais) devem ser secretos até a finalização da contagem;
- **Corretude:** Os resultados das eleições devem ser apropriadamente contados e corretamente publicados;
- **Robustez:** O sistema deve ser capaz de tolerar (alguns) votos falsos.

É nesse cenário que *blockchain* parece se encaixar perfeitamente, pois esses requisitos são inerentes à própria natureza dessa tecnologia.

2.2 Blockchain

Em 2008, um artigo intitulado *Bitcoin: A Peer-to-Peer Electronic Cash System* (NAKAMOTO et al., 2008) foi escrito por alguém sob o pseudônimo *Satoshi Nakamoto*, onde ele introduziu o termo *chain of blocks* (cadeia de blocos). O termo cadeia de blocos evoluiu ao longo dos anos para a palavra *blockchain*. A partir de então, a tecnologia *blockchain* vem se mostrando útil para uma infinidade de aplicações que podem ser implementadas em vários setores econômicos.

A tecnologia *blockchain* foi originalmente usada para a moeda digital *Bitcoin*, mas agora, ela está sendo utilizada para muitos outros fins. Assim como um banco de dados tradicional, uma rede *blockchain* pode, em princípio, ser usada para representar transações ou informações em qualquer tipo de domínio.

2.2.1 Definição

Blockchain é um *ledger* (livro-razão[2,3]) distribuído, *peer-to-peer* (ponto-a-ponto), criptograficamente seguro, *append-only* (apenas gravações), imutável e atualizável somente

[2] Se trata de um livro que mantêm registros de todos os dados em aberto além das contas a pagar ou a receber de uma organização.

[3] <https://osayk.com.br/livro-razao-e-livro-diario-o-que-sao-e-para-que-servem/>

através de consenso ou acordo entre pares (BASHIR, 2018). Observando os pontos em mais detalhes:

- **Ponto-a-ponto:** significa que não há controlador central na rede e todos os participantes falam diretamente entre si. Isso permite que transações em dinheiro, por exemplo, sejam trocadas diretamente entre os pares sem um envolvimento de terceiros, como um banco.
- ***Ledger* distribuído:** significa que o *ledger* é distribuído pela rede entre todos os pares e cada par mantém uma cópia do *ledger*.
- **Criptograficamente seguro**: significa que a criptografia tem sido usada para fornecer segurança, tornando esse *ledger* seguro contra adulterações e uso indevido.
- **Apenas gravação:** significa que os dados só podem ser adicionados ao *blockchain* em ordem sequencial, ordenada pelo tempo. Uma vez que os dados são adicionados ao *blockchain*, é quase impossível alterar esses dados e pode ser considerado praticamente imutável.
- **Atualizável via consenso:** significa que nenhuma autoridade central está no controle de atualizar o *ledger*. Em vez disso, qualquer atualização solicitada ao *blockchain* é validada contra critérios estritos definidos pelo protocolo *blockchain* e adicionada à rede somente após um consenso ter sido alcançado entre todos os participantes.

Um *ledger*, em ciência da computação, é um software que armazena transações. Um banco de dados é diferente de um *ledger*, de forma que em um banco de dados podemos adicionar, remover e modificar registros, enquanto em um *ledger* só podemos adicionar, mas não excluir ou modificar registros (PRUSTY, 2018). Nesse sentido, *blockchain* é uma estrutura de dados para implementar um *ledger* descentralizado.

2.2.2 Ilustração

Por questões de organização, empresas que fabricam dispositivos móveis, por exemplo, registram suas transações comerciais, relacionadas à seus ativos, em um *ledger*. Este *ledger* contém registros das transações de seus ativos com outras empresas ou entidades. Ao fabricar um novo dispositivo, o mesmo é registrado nesse *ledger*. Eventualmente, a empresa irá transferir a propriedade do dispositivo móvel para a sua revendedora. Essa transferência é registrada no *ledger*. A revendedora, por sua vez, também tem seu próprio *ledger* e registra o fato de que recebeu o dispositivo móvel da fabricante. Em algum momento, a revendedora irá transferir a propriedade do dispositivo para um cliente. Esse fato também será registrado no *ledger* da revendedora (Figura 2).

Figura 2: *Ledgers* individuais de entidades.

Fonte: Adaptado de (SAKHUJA, 2016)

Em essência, um *ledger* é um livro de registros que é mantido por cada uma das entidades envolvidas na transação de seus ativos. Essas entidades não têm visibilidade do *ledger* da outra. Geralmente, a empresa fabricante não cria nem constrói todas as peças necessárias para a fabricação de um ativo, mas adquire as partes de fornecedores. Cada uma dessas empresas fornecedoras tem seus próprios *ledgers*.

Se um cliente quisesse, por exemplo, saber o histórico de um dispositivo móvel desde a sua origem, teria que começar a obter essa informação da empresa que vendeu-lhe esse dispositivo; em seguida, de seu fabricante e depois de seus fornecedores. Mas e se esse dispositivo móvel tiver sido revendido para outras pessoas? As coisas rapidamente ficam complicadas pois os próximos compradores não possuem nenhum *ledger* para registrar essas transações. A Figura 3 ilustra esse cenário.

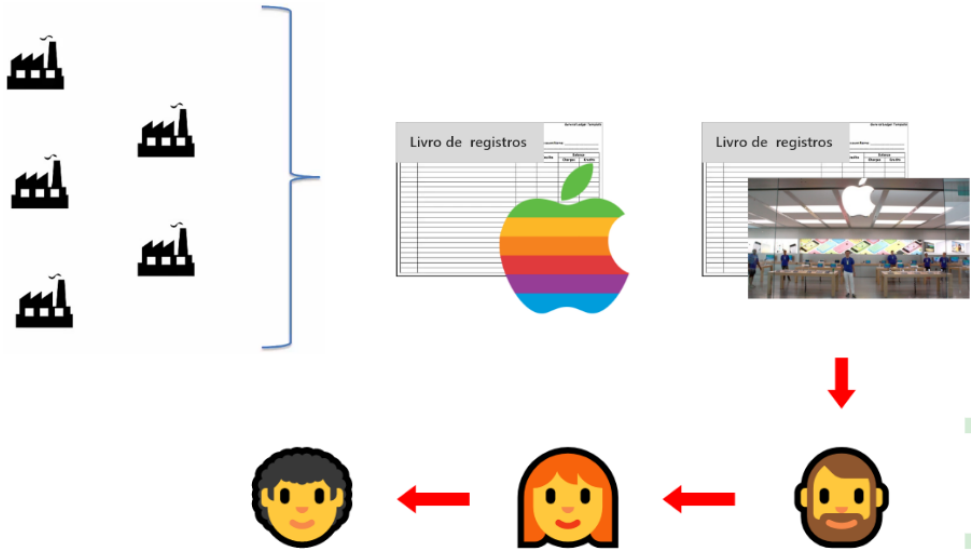


Figura 3: Entidades envolvidas na transação de um dispositivo móvel.

Fonte: Adaptado de (SAKHUJA, 2016).

Aprofundando o exemplo acima, se um cliente quiser saber a fonte de cada parte de um dispositivo móvel, teria que obter essa informação dos fornecedores que, por sua vez, possuem outros fornecedores – como os de matérias-primas (Figura 4).

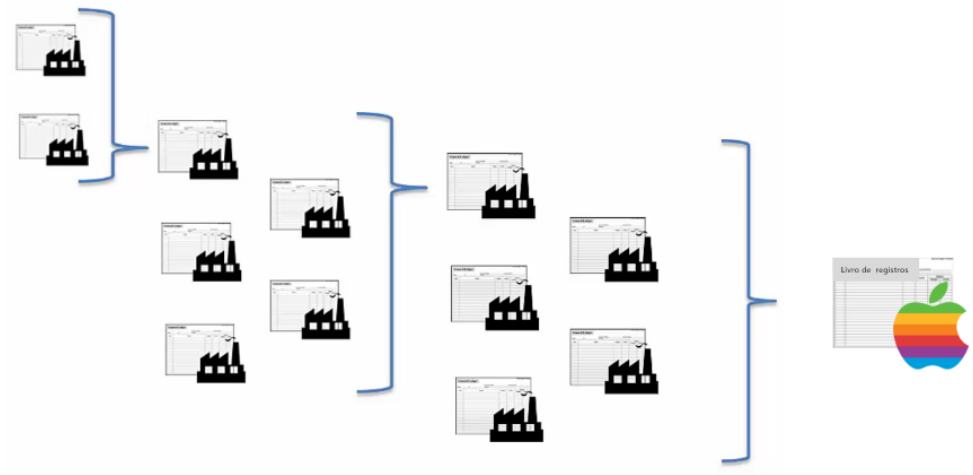


Figura 4: *Ledgers* de múltiplos fornecedores.

Fonte: Adaptado de (SAKHUJA, 2016).

Em alguns setores, é importante, e às vezes crítico, ter uma boa visibilidade desse tipo de informação. Dois exemplos claros são a indústria de fabricação de aeronaves e a indústria alimentícia. Se houver um acidente aéreo, devido à alguma falha, investigações serão realizadas para saber quem foi a parte responsável pelo acidente. Esta é uma questão sensível ao tempo, pois a peça defeituosa poderá estar presente em outros aviões, o que pode causar mais acidentes. Da mesma forma, na indústria alimentícia, é importante rastrear a fonte de alimentos contaminados no menor tempo possível. Rastrear essas

informações pelos vários *ledgers* das entidades envolvidas não é a maneira mais eficiente de se fazer isso. Um *ledger* distribuído fornece uma maneira mais eficaz de solucionar esse problema (Figura 5).

Figura 5: *Ledger* distribuído.

Fonte: Adaptado de (SAKHUJA, 2016).

Na Figura 5, cada entidade envolvida possui uma réplica do mesmo *ledger* – referido como *Distributed Ledger*. Todas as transações são registradas, pelas várias entidades envolvidas, nesse *ledger* distribuído. Por exemplo, uma transação para criação de um dispositivo móvel é iniciada pela empresa fabricante e refletida na cópia do *ledger* da revendedora. Da mesma forma, a transferência de propriedade do dispositivo da revendedora para o cliente também é registrada na cópia do *ledger* da fabricante. Com isso, é possível obter a rastreabilidade de um produto de sua fabricação até a sua revenda – mesmo que seja realizada por usuários finais (Figura 6).

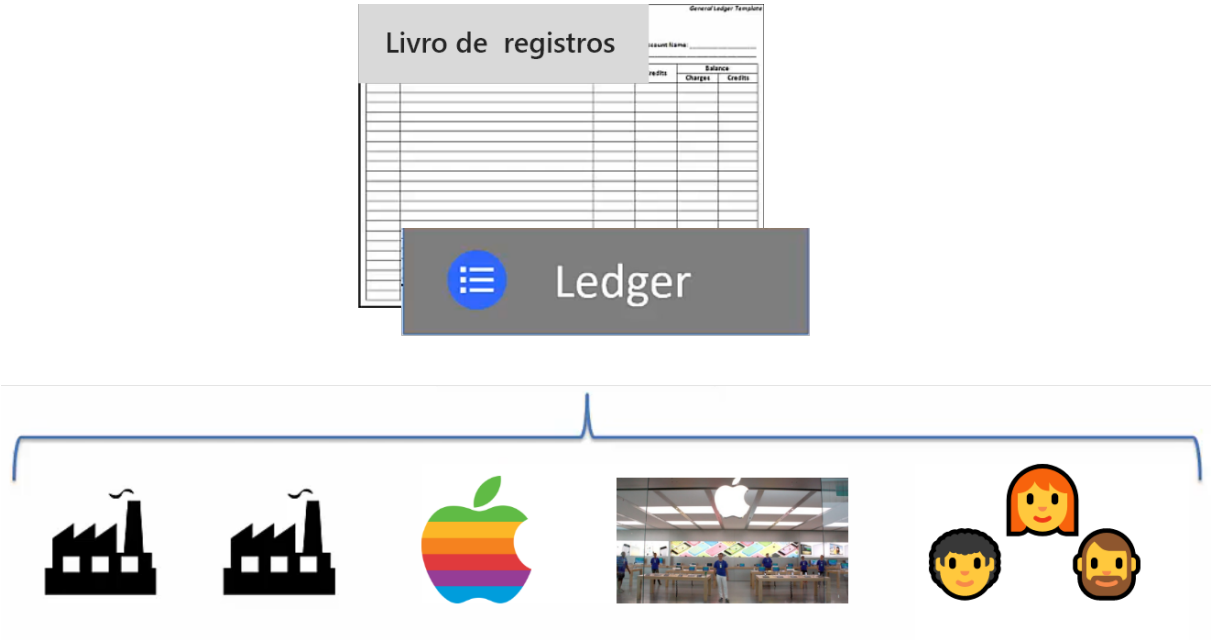


Figura 6: *Ledger* distribuído.

Fonte: Adaptado de (SAKHUJA, 2016).

2.2.3 Estrutura

Para atender ao cenário da subseção anterior (entre outros), uma rede *blockchain* é uma tecnologia de *ledger* distribuído (*Distributed Ledger Technology* ou DLT) que implementa uma cadeia de blocos conectados uns aos outros. Cada bloco contém uma lista de transações e alguns outros metadados – como quando foi criado, o bloco anterior, o número do bloco, quem é o criador do bloco e assim por diante. Cada bloco mantém um *hash* do bloco anterior, criando assim uma cadeia de blocos ligados entre si. Cada nó da rede deve conter uma cópia completa do estado da rede *blockchain* e, quando um novo nó entrar, ele solicitará e baixará esse estado de outros nós (PRUSTY, 2018). A Figura 7, ilustra essa situação.

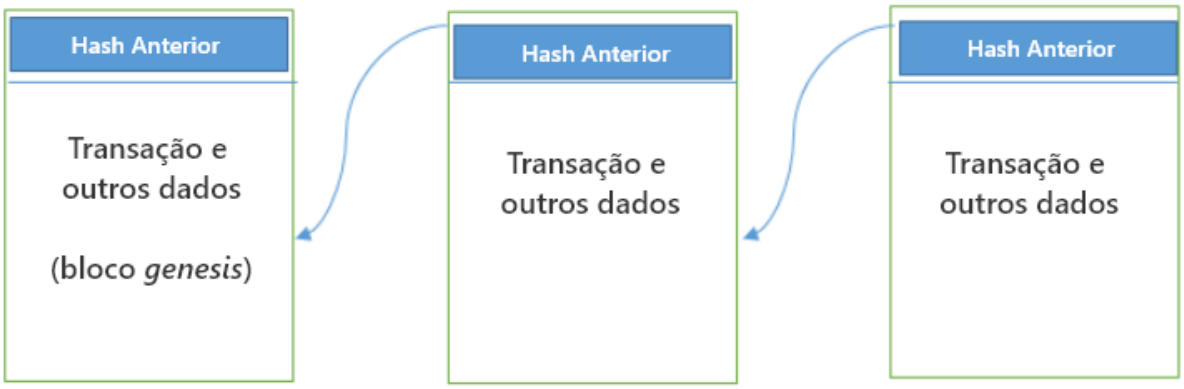


Figura 7: Estrutura genérica de uma rede *blockchain*.

Fonte: adaptado de (BASHIR, 2018)

Blockchain atua como um histórico de transações no qual todos os nós eventualmente concordam. É essencialmente um banco de dados público com o potencial de armazenar e transferir informações de ativos tangíveis (propriedades físicas como carros, imóveis, etc.) e intangíveis (como votos, dados genômicos, reputação, software e até ideias) (SWAN, 2015 apud CORRALES; FENWICK; HAAPIO, 2019).

De forma simplificada, a tecnologia por trás do *blockchain* poderia ser comparada a de um banco de dados, cujo modo de interação difere por ser mantido e atualizado por uma rede de computadores, em vez de um único computador de uma empresa ou organização (MOUGAYAR, 2019 apud CORRALES; FENWICK; HAAPIO, 2019).

2.2.4 Descentralização

A descentralização é uma das características essenciais fornecida pela tecnologia *blockchain*, oferecendo um meio para a eliminação de intermediários. Ela pode ser aplicada em vários graus, desde um modelo semi-descentralizado a um totalmente descentralizado, dependendo dos requisitos e circunstâncias. A Figura 8 mostra os diferentes tipos de sistemas que existem atualmente: centralizado, descentralizado e distribuído.

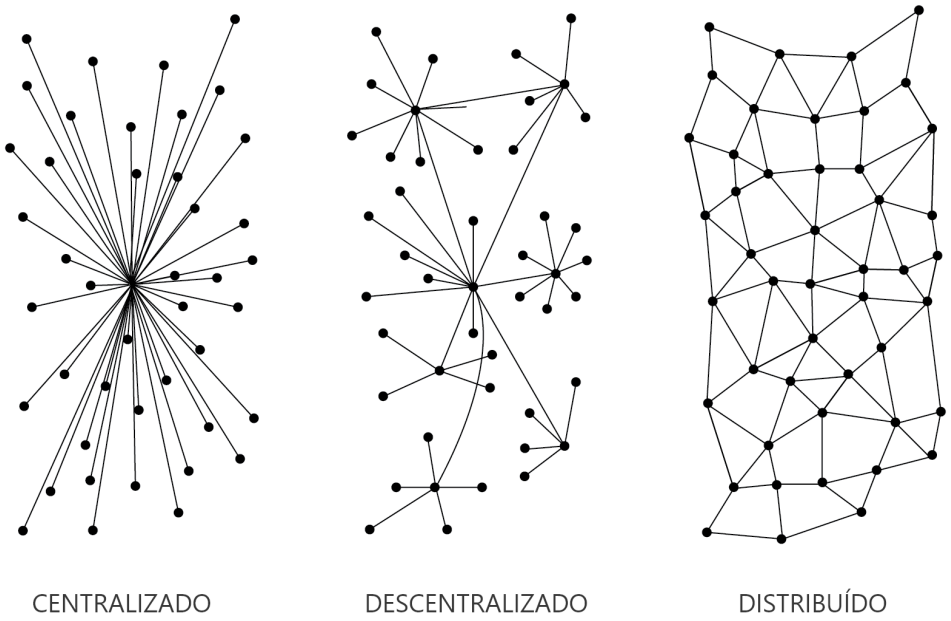


Figura 8: Tipos de redes/sistemas.

Fonte: adaptado de (BASHIR, 2018)

Os sistemas centralizados convencionais (cliente-servidor) são aqueles onde existe uma única autoridade que controla o sistema, sendo o único responsável por todas as operações nele. Todos os usuários de um sistema centralizado dependem de uma única fonte de serviço (BASHIR, 2018).

Em um sistema descentralizado, os nós não são dependentes de um único nó mestre; em vez disso, o controle é distribuído entre muitos nós. Isso é análogo a um modelo em que cada departamento de uma organização é responsável por seu próprio servidor de banco de dados, tirando assim a energia do servidor central e distribuindo-o aos subdepartamentos que gerenciam seus próprios bancos de dados (BASHIR, 2018).

Em um sistema distribuído, os dados são distribuídos por vários nós na rede. A diferença crítica entre um sistema descentralizado e um sistema distribuído é que, em um sistema distribuído ainda existe uma autoridade central que governa todo o sistema; enquanto que, num sistema descentralizado, não existe tal autoridade (BASHIR, 2018).

2.2.5 Smart Contracts

As transações armazenadas em um *blockchain* podem ser mais do que simples registros de um *ledger* - os sistemas *blockchains* emergentes também permitem que códigos sejam armazenados e executados como parte das transações no *ledger*. Esses códigos são frequentemente chamados *smart contracts*[4], embora esses códigos normalmente não sejam muito inteligentes e muitas vezes não relacionados a contratos legais.

[4] Do inglês, contratos inteligentes

Smart Contracts podem ser definidos como programas implementados como dados no *ledger* e executados em transações. Eles podem manter e transferir ativos digitais gerenciados pelo *blockchain* e podem invocar outros *smart contracts* (XIWEI, 2019).

O *Bitcoin* permite apenas formas muito simples de *smart contracts*, mas outros *blockchains*, como o *Ethereum*, permitem que códigos mais complexos sejam escritos (XIWEI, 2019). Esta capacidade expande significativamente o poder dos sistemas *blockchains* e aumenta sua gama de uso e potencial de inovação.

Os *smart contracts* podem ser usados para administrar ativos como, por exemplo, criptomoedas. Embora os *smart contracts* nem sempre sejam usados para contratos legais, às vezes eles podem ser usados para automatizar ou monitorar a execução de partes de contratos legais (XIWEI, 2019). *Smart contracts* também podem ser usados para implementar jogos, apostas ou loterias, bem como protocolos de interação entre diferentes partes, como em um processo de negócios colaborativo entre empresas (XIWEI, 2019).

2.2.6 Permissões

Blockchains podem ser permissionados ou não-permissionados. Em um *blockchain* permissionado, cada participante tem uma identidade exclusiva, que permite o uso de políticas para restringir a sua participação na rede e o acesso aos detalhes das transações.

Blockchains permissionados também são mais eficazes no controle da consistência dos dados que são adicionados à sua rede. Com a capacidade de restringir o acesso aos detalhes das transações, mais detalhes das transações podem ser armazenados no *blockchain*, e os participantes podem especificar as informações das transações que desejam permitir que outras pessoas visualizem.

Em redes permissionadas, alguns participantes podem ser autorizados a ver apenas certas transações, enquanto outros, como auditores, pode ter acesso a um número mais amplo de transações. Em um *blockchain* público, o nível de detalhes das transações pode ser limitado com o objetivo de proteger a confidencialidade e o anonimato dos participantes.

2.2.7 Criptografia

Criptografia é um dos componentes mais importantes da *blockchain*. Ela, por si só, é um campo de pesquisa e baseia-se em técnicas matemáticas avançadas. A criptografia tem por objetivo manter as informações confidenciais, contudo, este não é o seu único objetivo. Alguns de seus outros usos são: confidencialidade, integridade de dados, disponibilidade, autenticidade e não-repúdio (SINGHAL, 2018) – que são os cinco pilares básicos da Segurança da Informação (TELECO, 2019).

Qualquer informação na forma de texto, dados numéricos ou programa de computador pode ser chamado de *texto simples* (*plain text*). A ideia é criptografar o texto simples

usando um algoritmo de criptografia e uma chave que produz o texto cifrado. O texto cifrado pode então ser transmitido para o destinatário que o decifrar usando um algoritmo de decifragem e a chave para obter o texto simples (SINGHAL, 2018). Um exemplo desse fluxo pode ser visto na Figura 9.

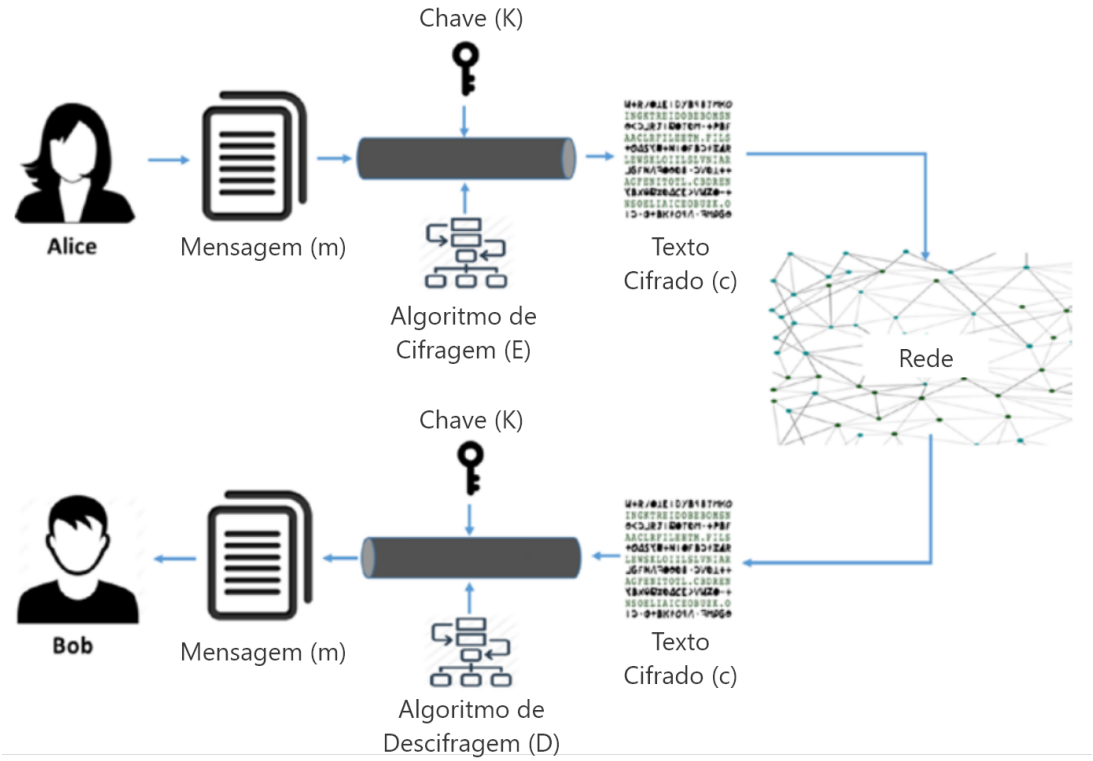


Figura 9: Exemplo de um fluxo de criptografia simétrica.

Fonte: adaptado de (SINGHAL, 2018)

Na Figura 9, Alice envia uma mensagem `(m)` para Bob. Se ela apenas enviar a mensagem como está, qualquer invasor pode facilmente interceptar a mensagem e a confidencialidade fica comprometida. Então Alice envia a mensagem usando um algoritmo de criptografia `(E)` e uma chave secreta `(K)` para produzir a mensagem criptografada chamada *texto cifrado*. O invasor tem que estar ciente tanto do algoritmo `(E)` quanto da chave `(K)` para interceptar a mensagem. Quanto mais forte for o algoritmo e a chave, mais difícil é para o invasor atacar.

Em geral, existem dois tipos de criptografia: *chave simétrica* e *chave assimétrica*.

2.2.7.1 Chave Simétrica

Se a mesma chave for usada para cifragem e descifragem, essa criptografia é chamada de chave simétrica. No caso da Figura 9, Alice e Bob têm que concordar com uma chave `(K)` chamada *chave secreta compartilhada* (*shared secret key*) antes de trocarem o texto cifrado. O processo é o seguinte:

- **Remetente (Alice)** – Criptografa a mensagem (`m`) usando um algoritmo de criptografia (`E`) e chave secreta (`K`) e envia o texto cifrado (`c`): (`c = E(K, m)`);
- **Receptor (Bob)** – Descifra o texto cifrado (`c`) usando o algoritmo de descifragem (`D`) e a mesma chave secreta (`K`) para obter a mensagem (`m`): (`m = D(K, c)`).

2.2.7.2 Chave Assimétrica

Criptografia de chave assimétrica, também conhecida como "criptografia de chave pública" é um conceito introduzido por Diffie e Hellman (SINGHAL, 2018), com o objetivo de resolver o problema de distribuição de chaves em um sistema de criptografia simétrico, através da introdução de assinaturas digitais. Um exemplo desse fluxo pode ser visto na Figura 10.

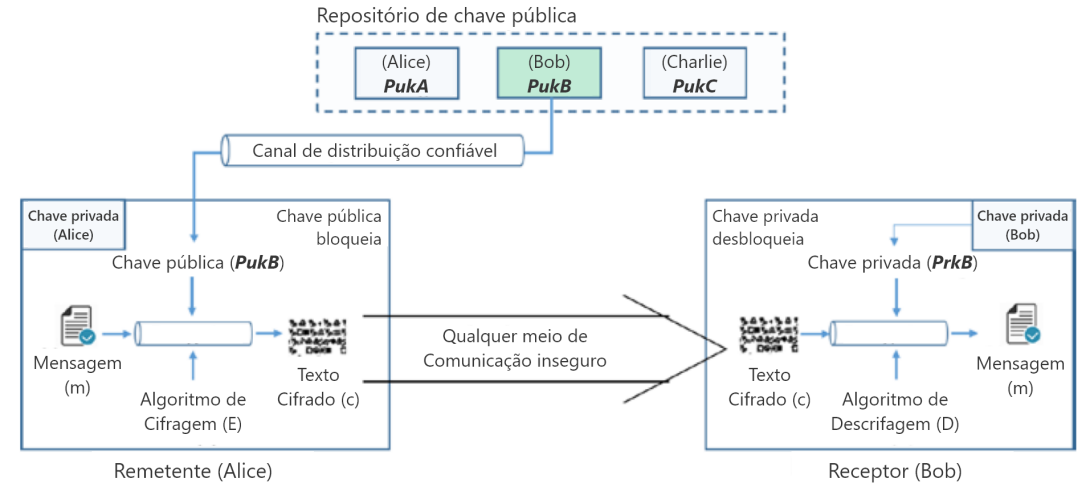


Figura 10: Exemplo de um fluxo de criptografia assimétrica para confidencialidade.

Fonte: adaptado de (SINGHAL, 2018)

Na Figura 10, Alice deseja enviar uma mensagem (`m`) para Bob. Para isso, os seguintes passos são tomados:

- **Remetente (Alice)** – Criptografa a mensagem (`m`) usando um algoritmo de criptografia (`E`) e a chave pública de Bob ($\texttt{Puk}_{\texttt{Bob}}$) e envia o texto cifrado (`c`): ($\texttt{c = E(Puk}_{\texttt{Bob}}\texttt{, m)}$);
- **Receptor (Bob)** – Descifra o texto cifrado c usando o algoritmo de descifragem (D) e sua chave privada ($\texttt{Prk}_{\texttt{Bob}}$) para obter a mensagem (`m`): ($\texttt{m = D(Prk}_{\texttt{Bob}}\texttt{, c)}$).

A criptografia de chave pública também fornece meios para autenticação. O receptor, Bob, pode verificar a autenticidade da origem da mensagem (`m`) da mesma forma que o fluxo anterior. A Figura 11 ilustra essa situação.

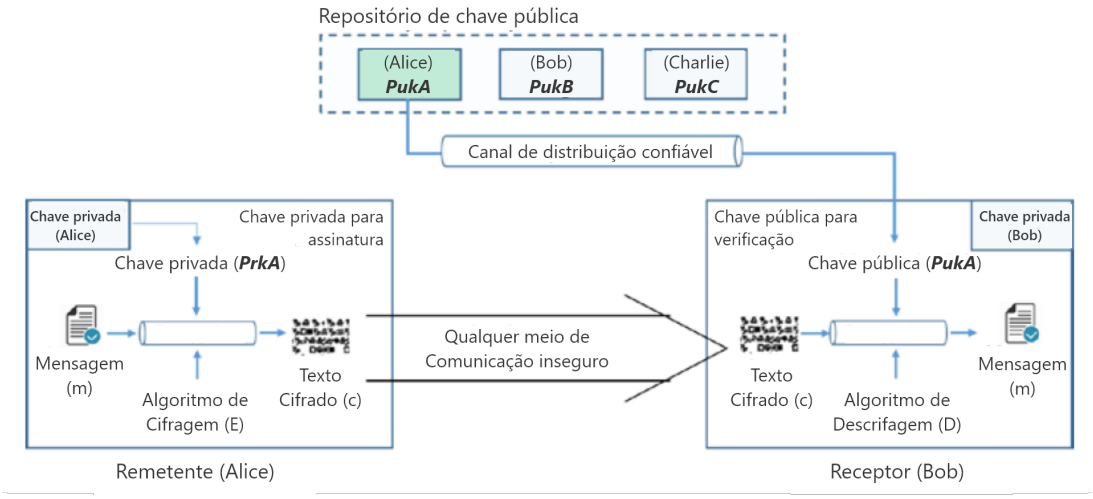


Figura 11: Exemplo de um fluxo de criptografia assimétrica para autenticação.

Fonte: adaptado de (SINGHAL, 2018)

No exemplo da Figura 11, a mensagem (`m`) foi criptografada usando o chave privada de Alice ($\texttt{Prk}_{\texttt{Alice}}$), servindo como uma assinatura digital. Para que a confidencialidade e autenticidade sejam alcançadas, a criptografia de chave pública tem que ser usada duas vezes. Primeiramente, a mensagem deve ser criptografada com a chave privada do remetente para fornecer uma assinatura digital. Em seguida, deve ser criptografada com a chave pública do destinatário para fornecer a confidencialidade. Esse cenário é representado nas equações 2.1:

$$\begin{aligned} c &= E(Puk_{\text{Bob}}, E(Prk_{\text{Alice}}, m)) \\ m &= D(Puk_{\text{Alice}}, D(Prk_{\text{Bob}}, c)) \end{aligned} \tag{2.1}$$

Nas equações 2.1, a criptografia de chave pública é usada quatro vezes: duas vezes para a criptografia e duas vezes para descifragem. Um exemplo dessa abordagem acontece nas lojas de aplicativos móveis, como o *Google Play* ou a *App Store*, onde os aplicativos são assinados digitalmente antes de serem publicados (SINGHAL, 2018).

Em suma, chaves públicas são conhecidas e acessíveis a todos. Elas podem ser usadas para criptografar a mensagem ou para verificar assinaturas. Chaves privadas são particulares e são usados para descifrar uma mensagem ou para criar assinaturas digitais. Nesse tipo de criptografia, não há o problema de distribuição de chaves; no entanto, existe um desafio com essa abordagem: como se garante que a chave pública, utilizada para criptografar a mensagem, é realmente a chave pública do destinatário e não a de um intruso? Para resolver isso, a noção de uma terceira parte confiável, chamada de infra-estrutura de chave pública (*Public Key Infrastructure*, PKI), é introduzida. Através das PKIs, a autenticidade de chaves públicas são asseguradas. A maneira como as PKIs

operam é que elas fornecem chaves públicas verificadas incorporando-os em um certificado de segurança, assinando-os digitalmente (SINGHAL, 2018).

2.2.7.3 Funções de *Hash*

Funções de *hash* são funções unidirecionais que convertem dados de entrada, de tamanho variável, em uma saída de tamanho fixo. A saída é geralmente denominada "valor de *hash*" (SINGHAL, 2018). A Figura 12 ilustra essa definição:

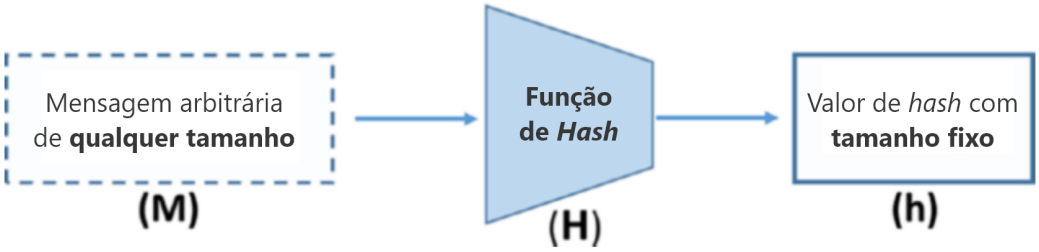


Figura 12: Forma básica de uma função de *hash*.

Fonte: adaptado de (SINGHAL, 2018)

Funções de *hash* devem possuir as seguintes características:

- Ter entrada de qualquer tamanho e saída de comprimento fixo – por exemplo, uma saída de 256 bits ou 512 bits;
- Ter um cálculo de valor de *hash* eficiente para qualquer entrada;
- Ser determinista, no sentido de que a mesma entrada – para uma mesma função de *hash* – produza o mesmo valor de *hash* todas as vezes;
- Tenha o processo reverso inviável – embora não impossível;
- Qualquer pequena alteração na mensagem deve influenciar no valor de *hash* de saída;
- Deve ser inviável encontrar dois valores de *hash* iguais para duas entradas diferentes.

Uma das funções de *hash* mais antigas são da família *Message Digest* (MD) – como MD5. Outra família de funções de *hash* bem conhecidas é a *Secure Hash Algorithm* (SHA) – como SHA-256 e SHA-512 (SINGHAL, 2018).

2.2.8 Protocolo de Consenso

Em uma rede *blockchain*, registros são adicionados através de um protocolo distribuído executado pelos nós dessa rede. Cada bloco contém um *hash* do bloco anterior

(*parent block*), incorporando uma representação segura do histórico completo da cadeia em cada bloco (Figura 7) – o primeiro bloco é chamado bloco *genesis*. No entanto, nós individuais podem falhar, se comportar de maneira mal-intencionada, agir contra o objetivo comum ou a comunicação da rede ser interrompida.

Em uma rede não-permissionada, qualquer nó da rede pode gravar registros no *ledger*. Redes permissionadas, em contrapartida, possuem meios para identificar os nós que podem controlar e atualizar o *ledger* e, muitas vezes, também têm maneiras de controlar quem pode realizar transações.

Tendo em vista este cenário, um dos aspectos cruciais em *blockchain* é a definição de mecanismos para a atualização do estado da rede. Esses mecanismos são chamados de *Protocolos de Consenso*. Obter consenso numa rede distribuída é um problema clássico de sistemas distribuídos e é citado no trabalho de (LAMPORT; SHOSTAK; PEASE, 1982) com o "problema dos generais bizantinos"[5].

O problema dos generais bizantinos foi um problema enfrentado pelo exército bizantino ao atacar uma cidade. A situação era que várias facções desse exército, comandadas por generais separados, cercaram uma cidade para tomá-la. A única chance de vitória é quando todos os generais atacam a cidade juntos. No entanto, o problema é como chegar a um consenso. Isso implica que todos os generais devem atacar ou todos devem recuar. Se alguns deles atacam e alguns recuam, então as chances de perder a batalha são maiores. Essa situação poderia ser mais complicada:

- E se houver mais de um traidor?
- Como deve ocorrer a coordenação de mensagens entre generais?
- E se um mensageiro for pego/morto/subornado pela cidade?
- Como encontrar os generais que são honestos e que são traidores?

Tendo em vista essa problemática, cada tipo de rede *blockchain* – permissionada ou não-permissionada – possuem protocolos que procuram resolver tal problema.

2.2.8.1 Redes Não-Permissionadas

Em uma rede não-permissionada, os protocolos de consenso mais amplamente usados são *Proof of Work* (PoW)[6] e *Proof of Stake* (PoS)[7], que geralmente aparecem acoplados a criptomoedas (CACHIN; VUKOLIĆ, 2017).

[5] The byzantine generals problem, em inglês
[6] Do inglês, prova de trabalho
[7] Do inglês, prova de participação

PoW é uma estratégia de consenso usada na rede *Bitcoin* (NAKAMOTO et al., 2008). Numa rede descentralizada, algum nó tem que ser selecionado para registrar as transações. A maneira mais fácil é via seleção aleatória. No entanto, a seleção aleatória é vulnerável a ataques. Então, se um nó quiser publicar um bloco de transações, um trabalho muito grande deve ser feito para provar que esse nó não irá atacar a rede. Geralmente, esse trabalho significa cálculos de computador (ZHENG et al., 2017).

Nesse protocolo, cada nó da rede está calculando um valor de *hash* do cabeçalho do bloco – nós que calculam esses valores são chamados de mineradores. Quando um nó atinge o valor alvo, ele transmite o bloco para outros nós e todos os outros nós devem confirmar mutuamente a exatidão do valor de *hash*. Se o bloco a ser registrado é validado, outros mineradores anexarão este novo bloco a sua própria estrutura. Esse procedimento de PoW é chamado de "mineração" em *Bitcoin* e requer muito cálculo computacional (ZHENG et al., 2017).

PoS é uma alternativa ao recurso computacional gasto em PoW. Mineradores em PoS têm que provar seu montante de moedas. A probabilidade de um nó produzir um novo bloco é proporcional à sua participação na rede; quanto maior o seu montante, maior é sua chance de validar um novo bloco de transações. Um minerador só precisa provar que possui uma certa porcentagem de todas as moedas disponíveis em um determinado estado da rede. Em tese, pessoas com mais moedas seriam menos prováveis de atacar a rede. A seleção com base no saldo da conta é injusta porque uma pessoa mais moedas acaba sendo a mais dominante na rede. Como resultado, muitas soluções foram propostas para decidir qual minerador deve adicionar o próximo bloco (ZHENG et al., 2017).

2.2.8.2 Redes Permissionadas

Em redes permissionadas, um dos protocolos mais conhecitos é o Practical Byzantine Fault Tolerance[8] (*Byzantine Fault Tolerance* ou PBFT). Ele é uma replicação do algoritmo de tolerância a falhas bizantinas BFT (CASTRO; LISKOV et al., 1999 apud ZHENG et al., 2017).

Para a versão v1 do *Hyperledger Fabric*, o *Apache Kafka* é fornecido como uma implementação de referência (HYPERLEDGER, 2017). Vários outros protocolos estão sendo desenvolvidos para esse *framework*, como: BFT *Smart*, *Simplified Byzantine Fault Tolerance* (SBFT) *Honey Badger of* BFT e SOLO[9] (este último está em fase de testes, na data deste trabalho).

[8] Do inglês, tolerância prática a falhas bizantinas

[9] <https://github.com/hyperledger/fabric/tree/master/orderer>

2.2.9 Oracles

Da perspectiva da arquitetura de software, um *blockchain* pode ser visto como um componente dentro de um sistema de software maior. No caso do *blockchain* ser usado como um banco de dados distribuído para propósitos mais gerais, que não sejam serviços puramente baseados na tecnologia, os aplicativos construídos em um *blockchain* precisarão interagir com outros sistemas externos. Assim, a validação de transações no blockchain dependerá desses sistemas externos (XIWEI, 2019).

O ambiente de execução de um *blockchain* é autônomo. Informações presentes nos dados e transações no *blockchain* só estão na sua rede. *Smart contracts* executados em um *blockchain* são funções puras por *design*. O estado dos sistemas externos não é diretamente acessível aos *smart contracts*. No entanto, as chamadas de função nos *smart contracts* às vezes precisam acessar o estado do mundo externo (XIWEI, 2019).

Para conectar o ambiente de execução fechado de um *blockchain* com o mundo externo, um *oracle* é introduzido para avaliar as condições que não podem ser expressas em um *smart contract*. Um oracle é uma *third-party*[10] confiável que fornece *smart contracts* com informações sobre o mundo externo. Quando a validação de uma transação depende do estado externo, o *oracle* é solicitado a verificar e fornecer o resultado ao validador (minerador), que leva em conta o resultado fornecido ao validar a transação (XIWEI, 2019).

O *oracle* pode ser implementado, dentro de uma rede *blockchain*, como um *smart contract* com estado externo, sendo injetado periodicamente por um injetor fora da rede. Outros *smart contracts* podem acessar os dados do *smart contract* do *oracle*. Um *oracle* também pode ser implementado como um servidor fora do *blockchain* (XIWEI, 2019).

Um benefício do uso de *oracles* é a conectividade, pois os aplicativos baseados em *blockchain* podem acessar estados externos, através deles, e usar esses estados externos em sua execução. Em contrapartida, uma das desvantagens é a desconfiança, pois os estados externos injetados nas transações não podem ser totalmente validados por outros mineradores. O *oracle* selecionado, para verificar ou fornecer o estado externo, precisa ser confiável para todos os participantes envolvidos, em transações relevantes. Assim, quando os mineradores validam transações, eles confiam no *oracle* (XIWEI, 2019).

2.3 Hyperledger

O *Hyperledger*[11] é considerado um "guarda-chuva" *open source* de projetos de *blockchains* e ferramentas relacionadas, com colaboração global. Ele é mantido pela *Linux Foundation* desde 2016[12]. Atualmente, há mais de 270 organizações que suportam essa

[10] Uma desenvolvedora de terceiros ou *third-party* não é diretamente ligada ao produto primário que um consumidor está usando.

[11] <https://www.hyperledger.org/>

[12] <https://www.hyperledger.org/about>

iniciativa e 13 projetos em andamento, incluindo o *Hyperledger Fabric*, o *Hyperledger Composer* e outras ferramentas e frameworks voltados para DLT (*Distributed Legder Technologies*).

Hyperledger Fabric[13] é um framework para desenvolvimento de *blockchains* de negócios, servindo como uma base para o desenvolvimento de aplicações *blockchain* com uma arquitetura modular. Os dados podem ser armazenados em vários formatos e vários algoritmos de consenso podem ser configurados.

Hyperledger Composer[14] é um conjunto de ferramentas para desenvolvimento de aplicações em cima de frameworks DLT. É feito em JavaScript e tem como objetivo permitir a criação de soluções empresariais de maneira rápida e simples. As abstrações providas permitem que os requisitos sejam atendidos e testados de forma bastante robusta e alinhada com o negócio.

2.3.1 Permissão

O *Hyperledger Fabric* (ou apenas *Fabric*), permite a criação de *blockchains* permissionadas. Membros de tais redes precisam se inscrever através de um provedor de serviços. Todos os participantes dessas redes possuem identidades conhecidas. Chaves públicas são usadas como certificados criptografados vinculados a organizações, componentes de rede e usuários finais. O controle de acesso a dados é aplicado no nível de rede (SAKHUJA, 2016).

Fabric supera algumas das limitações encontradas nos primeiros desenvolvimentos de *blockchains* permissionadas (ANDROULAKI et al., 2018), tais como:

- O consenso era *hard-coded*[15] na plataforma, o que se opunha ao princípio de que não pode haver somente um validador na rede;
- *Smart contracts* precisavam ser escritos numa linguagem específica, o que limitava a adoção da tecnologia e deixava o programador mais propenso a causar erros;
- A execução sequencial de todas as transações causava problemas de performance e medidas complexas eram necessárias para prevenir ataques *DDoS*.

2.3.2 Arquitetura

Antes do *Fabric*, todos os sistemas de *blockchain*, permissionados ou não, seguiam a arquitetura *order-execute*, ou seja, um protocolo de consenso primeiro ordena as transações

[13] <https://www.hyperledger.org/projects/fabric>
[14] <https://www.hyperledger.org/projects/composer>
[15] Prática de incorporar dados diretamente no código-fonte de um programa.

e as propaga para todos os *peers* e, depois, cada *peer* executa as transações sequencialmente (ANDROULAKI et al., 2018). Essa arquitetura é ilustrada na Figura 13.

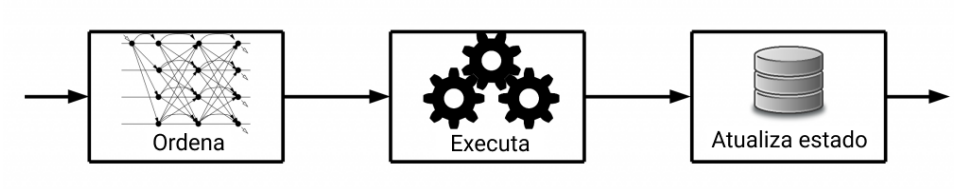


Figura 13: Arquitetura *order-execute*.

Fonte: adaptado de (ANDROULAKI et al., 2018)

Blockchains permissionadas baseadas em PoW, como a *Ethereum*, combinam consenso e execução de transações da seguinte maneira: cada nó participante monta um bloco contendo transações válidas já pré-executadas; O nó tenta resolver um desafio de PoW; Se o nó resolver o desafio, dissemina o bloco na rede e todos os nós que recebem o bloco validam a solução para o desafio e todas as transações no bloco. Em seguida, todos os nós repetem a execução do primeiro nó. Dessa forma, todos os nós executam as transações sequencialmente (ANDROULAKI et al., 2018).

Essa arquitetura é conceituavelmente simples e, portanto, bastante usada. No entanto, ela tem muitos problemas quando usada em *blockchains* permissionadas com propósito geral (ANDROULAKI et al., 2018), como:

- Execução sequencial, que acarreta em perda de performance e limita o rendimento que pode ser alcançado pela *blockchain*;
- Confidencialidade da execução, pois muitos sistemas permissionados executam os *smart contracts* em todos os nós. No entanto, muitos casos de uso pretendidos para *blockchains* permissionadas exigem confidencialidade, ou seja, que o acesso à lógica do *smart contract*, aos dados de transação ou estado do *ledger* possa ser restrito.

Fabric introduz a arquitetura *execute-order-validate*, que consiste de duas partes: *chaincode* e política de endosso. A *chaincode* é um *smart contract* que implementa a lógica da aplicação e roda durante a fase de execução. A *chaincode* é a parte central de uma aplicação feita com *Fabric* (ANDROULAKI et al., 2018). A política de endosso é avaliada durante a fase de validação e não pode ser modificada por pessoas sem permissão. Uma típica política de endosso permite que a *chaincode* especifique os *endorsing peers* necessários para endosso com uma expressão lógica como "três de cinco" (ANDROULAKI et al., 2018).

A arquitetura *execute-order-validate* funciona da seguinte maneira: um cliente envia transações para os nós especificados pela política de endosso. Cada transação é executada

por esses nós e seu resultado é armazenado. Depois da execução, as transações entram na fase de validação, que usa um protocolo de consenso para produzir uma sequência de transações endossadas agrupadas em blocos. Esses blocos são transmitidos para todos os nós. Cada nó valida as alterações de estados das transações endossadas na mesma ordem e a validação é determinística (ANDROULAKI et al., 2018). Essa arquitetura é ilustrada na Figura 14.

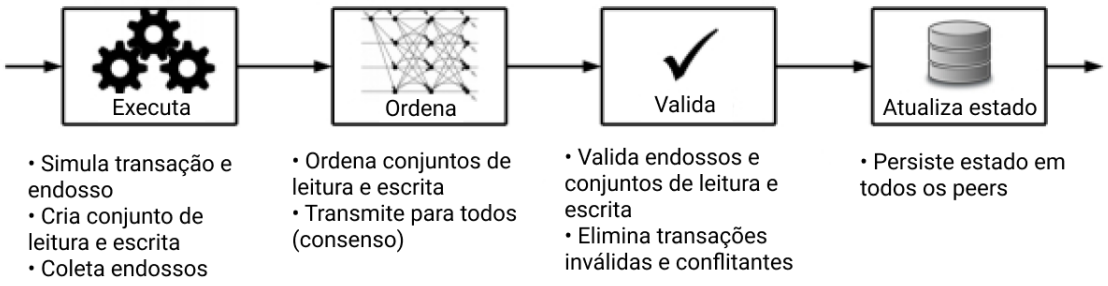


Figura 14: Arquitetura *execute-order-validate*.

Fonte: adaptado de (ANDROULAKI et al., 2018)

2.3.3 Tipos de nós

Como o *Fabric* é permissionado, todos os nós participantes de uma rede têm uma identidade, fornecida por um Provedor de Serviço de Assinatura (*Membership Service Provider* - MSP[16]). Os nós da rede podem assumir três papéis (ANDROULAKI et al., 2018):

- ***Clients*:** submetem transações para execução, ajudam a orquestrar a fase de execução e transmitem transações para ordenação;
- ***Endorsing Peers*:** executam e validam transações. Todos os *peers* têm um *ledger* contendo os registros e o estado da rede, mas nem todos executam transações, somente alguns deles, chamados de *endorsing peers* ou *endorsers* o fazem, conforme especificado pela política de endosso;
- ***Orderers*:** estabelecem a ordem das transações, via consenso, no *Fabric*, onde cada transação contém atualizações de estado e dependências calculadas durante a fase de execução, juntamente com assinaturas criptográficas dos *endorsers*.

A figura 15 ilustra a comunicação entre esses três tipos de nós.

[16] Definição na próxima subseção.

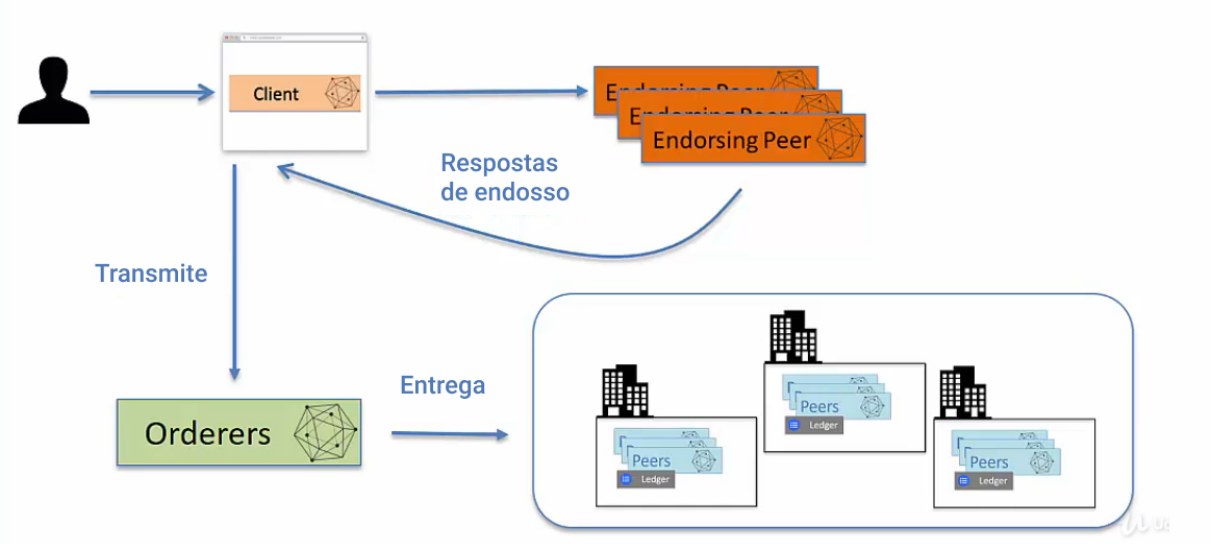


Figura 15: Fluxo de comunicação entre os tipos de nós do *Fabric*.

Fonte: adaptado de (SAKHUJA, 2016)

2.3.4 Identificação dos nós

O MSP mantém as identificações de todos os nós na rede e é responsável por emitir as credenciais que são usadas para autenticação e autorização. Como o *Fabric* é permissionado, todas as interações entre os nós ocorrem através de mensagens autenticadas, normalmente com assinaturas digitais (SAKHUJA, 2016).

O MSP é uma abstração passível de customização. A implementação padrão no *Fabric* lida com métodos envolvendo PKI para autenticação e pode acomodar autoridades de certificação comercial (SAKHUJA, 2016).

Certificação comercial (*Certification Authorities* – CA) são responsáveis por homologar a identidade de uma entidade assinando o certificado contendo a chave pública da entidade. Há duas CAs no contexto do *Fabric*: *Registration Authority* (RA) e *Validation Authority* (VA) (SAKHUJA, 2016).

2.3.4.1 Emissão de certificados

O processo começa com o *client* levantando uma solicitação para emissão de identidade ou assinatura do certificado. O solicitante envia a documentação apropriada para a RA. Uma vez validada a identidade, a RA informa a CA para emitir um certificado. A CA emite o certificado assinando-o e, em seguida, enviando-o de volta ao solicitante. Uma outra coisa que a CA faz é informar a VA sobre o novo certificado para que qualquer pessoa possa verificar com a VA se o certificado emitido é válido ou não (SAKHUJA, 2016). O fluxo descrito pode ser visualizado na Figura 16.

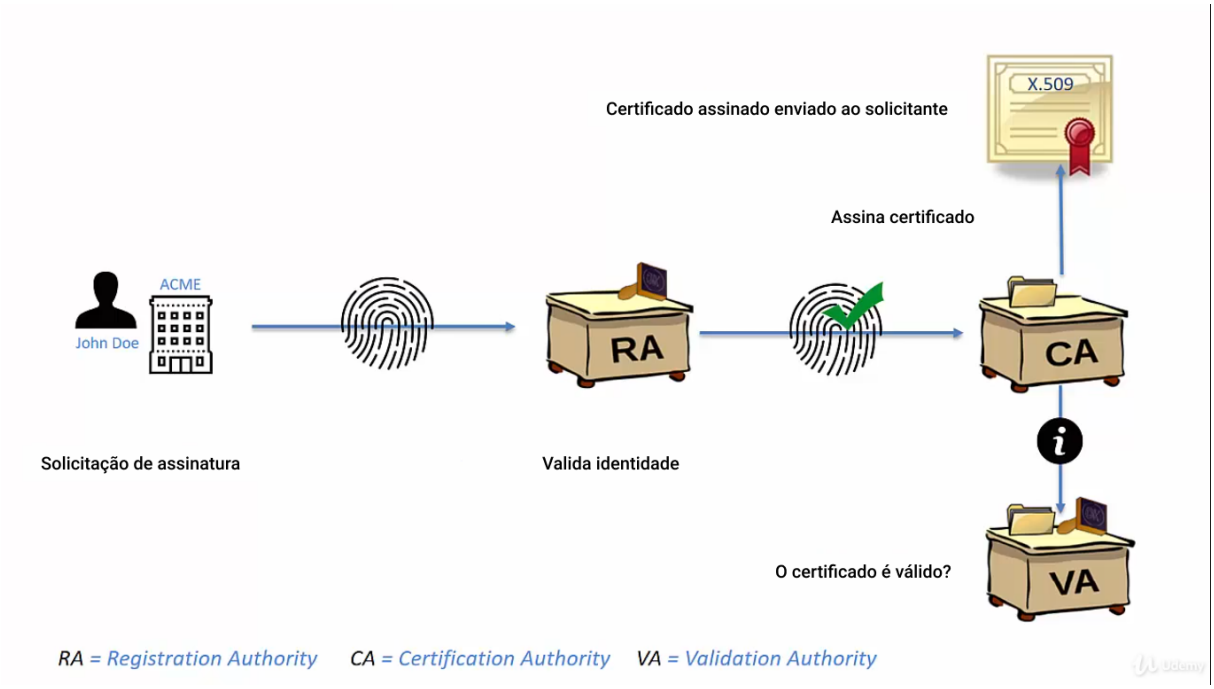


Figura 16: Fluxo de emissão de um certificado.

Fonte: adaptado de (SAKHUJA, 2016)

No *Fabric*, usa-se múltiplas CAs, caso contrário, o processo de emissão de certificados seria ineficiente e não muito efetivo. Portanto, a forma como o *Fabric* funciona é que um *certificado raiz* é emitido para cada membro da rede e, em seguida, o certificado raiz pode ser autorizado a emitir novas identidades para que os membros da rede possam gerenciar as identidades em suas organizações (SAKHUJA, 2016).

As CAs gerenciam identidades e certificados em banco de dados, sendo sua implementação padrão com SQLite[17], mas podem ser configuradas para usar MySQL[18] ou PostgreSQL[19] para suportar o protocolo LDAP[20].

2.3.5 Implementação do *ledger*

No *Fabric*, há duas partes que constituem o *ledger*: *transaction log* e *state database*. O *transaction log* registra todas as transações de *assets* e o *state database* representa os *assets* no momento atual. Enquanto o *log*[21] é imutável e somente as operações de criação e leitura estão disponíveis, o *database* é mutável e tem as quatro operações de CRUD[22] disponíveis (SAKHUJA, 2016).

17 <https://www.sqlite.org/>
18 <https://www.mysql.com/>
19 <https://www.postgresql.org/>
20 <https://br.ccm.net/contents/271-o-protocolo-ldap>
21 Processo de registro de eventos relevantes.
22 *Create, Read, Update, Delete*, do inglês, criar, ler, atualizar e "deletar".

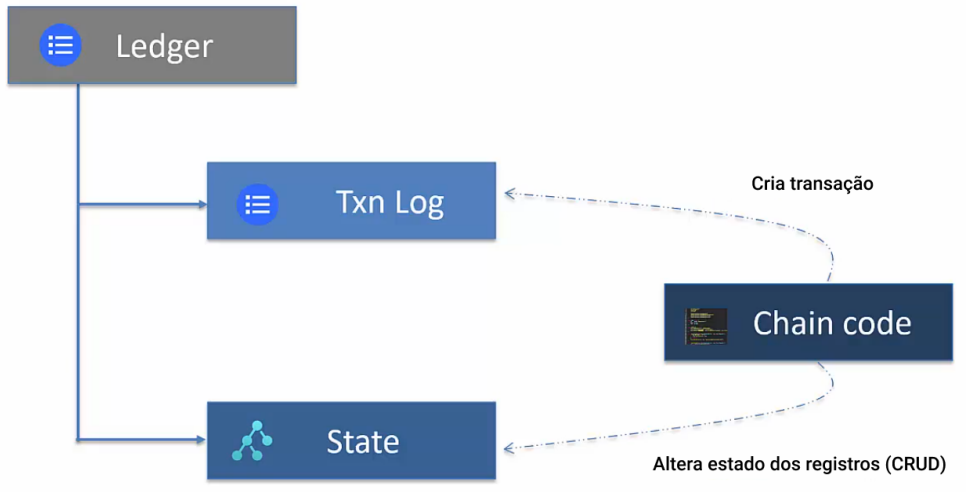


Figura 17: Partes do *ledger* no *Fabric*.

Fonte: adaptado de (SAKHUJA, 2016)

O *log* e o *database* são implementados com LevelDB[23], mas o *database* pode ser configurado para usar CouchDB[24], que são bancos *key-value*[25] rápidos (SAKHUJA, 2016).

2.3.6 Criação de modelos

Uma linguagem orientada a objetos e altamente tipada é usada para definir os modelos da rede. A linguagem é bem descritiva e parece bastante com uma língua falada. Um modelo define a representação dos *participants*, *assets*, *transactions* e *events* da rede. Os modelos são declarados em arquivos com extensão `.cto` (SAKHUJA, 2016).

Cada modelo precisa de um *namespace*[26] que será aplicado para todos os recursos declarados naquele arquivo. O *namespace* deve ser único para cada arquivo ou isso causará um erro na hora de compilar a rede (SAKHUJA, 2016).

Um *asset* representa algo que tenha valor no negócio, sendo tangível (como um avião) ou não (como um voto). *Participants* agem sobre esses *assets* alterando seu estado através de *transactions*. Dentro da lógica de uma *transaction*, *events* podem ser emitidos e esses podem conter dados (SAKHUJA, 2016).

2.3.6.1 Registros de recursos

Um registro de recurso gerencia instâncias de recursos durante a execução da aplicação. É criado um registro separado para cada tipo de recurso e cada instância de

[23] <https://github.com/google/leveldb>
[24] <http://couchdb.apache.org/>
[25] <https://database.guide/what-is-a-key-value-database/>
[26] Um espaço de nomes ou *namespace* é um delimitador abstrato que fornece um contexto para os itens que ele armazena, o que permite uma desambiguação para itens que possuem o mesmo nome mas que residem em espaços de nomes diferentes.

asset ou *participant* tem um identificador próprio, declarado no modelo pelas palavras-chave *identified by*. *Transactions* e *events* também têm um identificador próprio, mas esse é gerado automaticamente durante as operações, juntamente com um *timestamp*[27] (SAKHUJA, 2016).

2.3.6.2 Relacionamentos

É possível declarar relacionamentos unidirecionais entre os recursos e esses são identificados pelo uso do símbolo especial `->`. A característica unidirecional dos relacionamentos garante que a remoção de um recurso que tenha vínculo com outro, não acarreta na remoção do recurso vinculado (SAKHUJA, 2016).

2.3.6.3 Validação e configuração de campos

É possível validar campos do tipo *String* usando expressões regulares. Campos do tipo *Integer* podem ser validados através de um *range* de valores. Na definição do modelo, também é possível marcar campos como opcionais e atribuir valores padrão para eles (SAKHUJA, 2016).

2.3.6.4 Papéis de *participants* e *Business Network Cards*

Os usuários da rede podem realizar ações baseadas em seu papel e as informações necessárias de cada usuário para conectar-se com a aplicação ficam armazenadas no *Business Network Card*, ou somente BNC. Há dois papéis de administrador na rede. O primeiro papel é o de *peer administrator*, responsável por liderar atividades a nível de infraestrutura e é criado durante a configuração de ambiente. O segundo é o de *network administrator*, responsável por liderar atividades a nível de aplicação e é criado por *peer admins*. (SAKHUJA, 2016)

Um *peer admin* pode criar *network admins*, que por sua vez, podem criar *participants* e autorizar *participants* a criar outros *participants*. Não é necessário que exista um *peer admin* para cada aplicação, podendo existir um para várias redes. (SAKHUJA, 2016)

O BNC contém as credenciais do usuário, as chaves e certificados emitidos para ele e um *connection profile*. O usuário pode ter diversos BNCs configurados em sua máquina para se conectar com várias redes. Dentro do *connection profile* estão as informações e URLs para se comunicar com os CAs e como alcançar os *peers* e *orderers*. Esses *cards* são gerenciados no sistema de arquivos do usuário (SAKHUJA, 2016).

[27] Uma marca temporal, estampa de tempo ou *timestamp* é uma cadeia de caracteres denotando a hora ou data que certo evento ocorreu.

3 TRABALHOS RELACIONADOS

Este capítulo aborda o cenário atual da tecnologia *blockchain* em sistemas de eleição. Na seção 3.1, é apresentada uma Revisão Sistemática da Literatura.

3.1 Revisão Sistemática da Literatura

Para levantar as principais necessidades da área pesquisada, elaboraram-se questões de pesquisa (QP) de modo a listar os principais sistemas *blockchain* relacionados ao tema abordado neste trabalho. Dessa forma pretende-se responder à questão de pesquisa principal:

Quais as aplicações ou soluções utilizadas com blockchain para sistemas de eleição?

Com base na questão principal de pesquisa acima, outras questões de pesquisa secundárias foram levantadas:

- QP1: Quais as principais contribuições desses trabalhos?
- QP2: Quais trabalhos são voltados para soluções empresariais/privadas?

3.2 Bases e String de Busca

As fontes digitais selecionadas foram: *Science Direct*[1], *Springer Link*[2], *IEEE Xplore Digital Library*[3] e a *ACM Digital Library*[4], cobrindo dessa forma, os trabalhos mais relevantes da área. Para responder às questões de pesquisa, a seguinte *string* de busca foi utilizada:

(blockchain OR "smart contract"OR "distributed ledger") AND (election OR vote OR voting)

3.3 Critérios de Inclusão e Exclusão

Após a definição das questões, o passo seguinte foi definir os critérios de seleção de busca. Esses critérios se encontram no Quadro 1.

[1] <https://www.sciencedirect.com/>
[2] <https://link.springer.com/>
[3] <https://ieeexplore.ieee.org/Xplore/home.jsp>
[4] <https://dl.acm.org/dl.cfm>

Quadro 1: Critérios de Inclusão e Exclusão

Critérios de Inclusão	**Critérios de Exclusão**
Artigos a partir de 2014.	Artigos não relacionados com o tema da pesquisa.
Artigos relacionados a *blockchain* ou eleições.	Artigos que não sejam da área de Ciência da Computação.
Artigos que apresentem propostas de aplicações/soluções.	Artigos que não contenham as palavras-chave no título, resumo/*abstract* ou palavras-chave.
Artigos publicados somente na língua inglesa.	Revisões ou mapeamentos sistemáticos, surveys.
Artigos que respondam, ao menos, uma questão de pesquisa.	Artigos duplicados (considerando o mais recente).

O critério "Artigos publicados somente na língua inglesa" justifica-se pelo seu caráter universal, tendo em vista que a maior parte dos trabalhos publicados nas principais bases de dados na área de Ciência da Computação são escritos na língua inglesa, inclusive artigos brasileiros e portugueses.

Da busca, 649 artigos foram retornados, sendo 334 da base *Springer Link*, 252 da *Science Direct*, 56 da *IEEE* e 7 da *ACM*. Foram identificados 19 artigos duplicados por intermédio do software *StArt* e por meio de revisão manual (neste último caso, 1 artigo estava duplicado). Ao final do processo, foram selecionados 16 artigos para a extração dos dados.

A extração de dados se deu em forma de leitura total dos artigos selecionados – os mesmos se encontram no rodapé desta página[5]. A Figura 18 mostra o fluxo do processo de busca desta revisão, bem como o número de artigos identificados em cada etapa. Na próxima seção, os dados extraídos serão mostrados em forma de quadros.

[5] <https://1drv.ms/x/s!AofQ8hR8AgfwlIY4FRGW8FZ8_8g5cw>

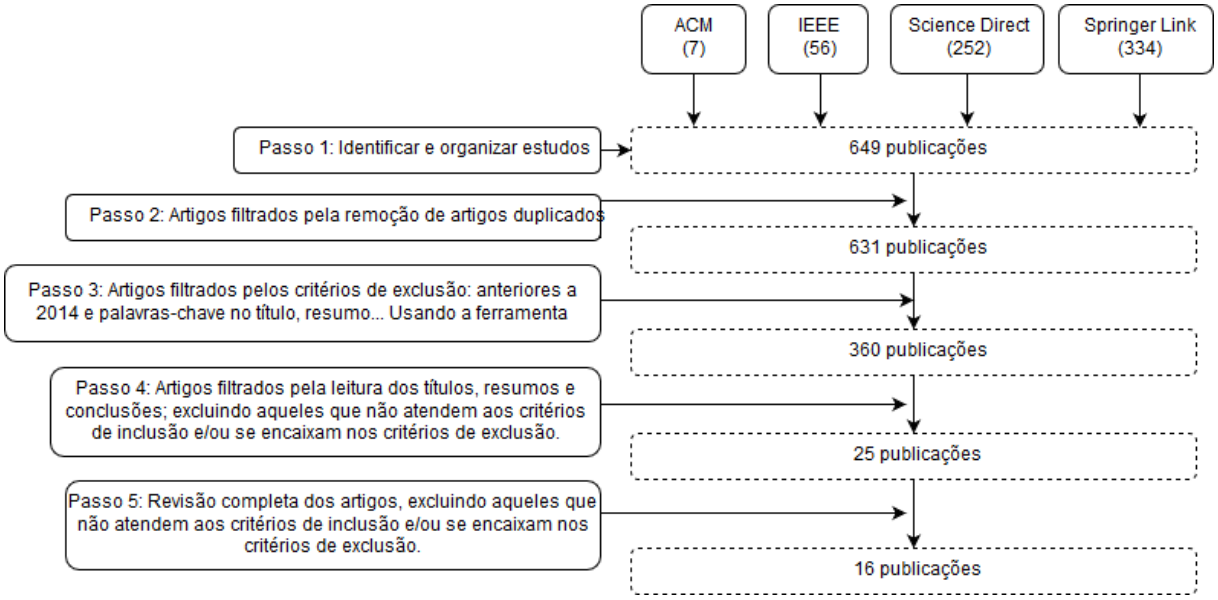


Figura 18: Fluxograma de seleção de artigos

Fonte: Elaborado pelos autores

3.4 Resultados da Revisão

Esta seção exibe os resultados obtidos desta revisão da literatura, com o objetivo de identificar quais os trabalhos utilizam *blockchain* em sistemas de eleição. Para isso, foram coletadas informações sobre as soluções propostas.

Para a organização e análise dos trabalhos relacionados, foi utilizada a ferramenta *StArt*[6] (*State of the Art through Systematic Reviews*) na versão 3.0.3 desenvolvida pelo Laboratório de Engenharia de Software (LAPES) da Universidade Federal de São Carlos (UFSCAR).

3.4.1 QP1: Quais as principais contribuições desses trabalhos?

Basicamente em todos os trabalhos, as seguintes contribuições eram propostas: a) sistema de eleição com ou sem o uso de plataforma/*framework* (*MultiChain* ou *Ethereum*) e *smart contracts* e b) protocolos para sistemas de eleição.

3.4.2 QP2: Quais trabalhos são voltados para soluções empresariais/privadas?

Com exceção do trabalho de ID 80871, todos os trabalhos procuraram atender o formato de eleições públicas através de uma rede *blockchain* não-permissionada.

[6] <http://lapes.dc.ufscar.br/tools/start_tool>

3.5 Limitações da Revisão

Esta revisão é passível de exclusão de algum trabalho relevante, devido a não correspondência aos critérios de inclusão definidos no protocolo, especificamente na aplicação dos critérios de escopo do título, resumo/*abstract* e palavras-chave. Outra limitação refere-se às decisões subjetivas ocorridas no decorrer do processo de análise, em razão de algumas publicações não apresentarem uma descrição clara, dificultando a execução dos critérios e, posteriormente, a sua análise. Em relação à busca, como ela foi realizada em apenas quatro bases de dados – consideradas as principais da área – num intervalo de tempo dos últimos cinco anos, é possível que estudos relevantes não tenham sido incluídos.

3.6 Conclusão

Como observado, muitos estudos foram realizados com um ou mais dos seguintes objetivos: a) propor um sistema de eleição com ou sem o uso de plataforma/*framework* e b) propor protocolos para sistemas de eleição. Entre os 16 estudos selecionados, apenas 1 propôs um sistema *blockchain* voltado para redes privadas.

Ao analisar os artigos selecionados, nota-se a falta de soluções satisfatórias que facilitem a construção de redes *blockchain* privadas. Tendo em vista esse resultado, o próximo capítulo apresenta a construção de uma rede *blockchain* privada e, para facilitar esse processo, o *framework Hyperledger Fabric* e o conjunto de ferramentas *Hyperledger Composer* foram utilizados.

4 MATERIAIS E MÉTODOS

Neste capítulo serão descritos os recursos que serviram de base para elaboração do trabalho.

4.1 Especificação do sistema

4.1.1 Atores

Só existe um ator no sistema, o usuário Audora (representado pelos autores). O ator acessa a eleição, analisa os candidatos e vota em um deles.

4.1.2 Diagrama de Caso de Uso

Para melhor entendimento do papel do usuário com as suas respectivas tarefas, foi construído um diagrama de caso de uso (Figura 19) que descreve as principais funcionalidades da aplicação e a interação dessas funcionalidades com o ator (usuário):

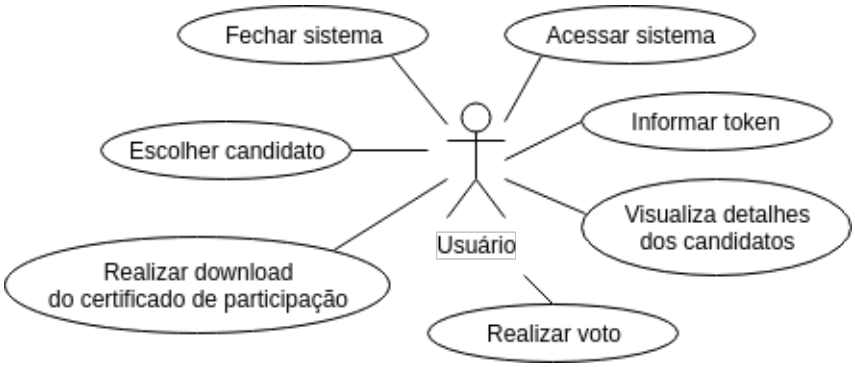


Figura 19: Diagrama de Caso de Uso do sistema.

Fonte: Elaborado pelos autores.

4.1.3 Especificações de Casos de Uso

4.1.3.1 Realizar Votação

O caso de uso "Realizar Votação" tem por objetivo permitir que o usuário acesse a eleição vigente após se identificar, visualize os candidatos e vote em algum deles após confirmar sua identidade e é apresentado a seguir:

- Atores:
  - Usuário do sistema Audora.
- Pré-condições:

  - O usuário é colaborador da entidade responsável pela eleição;
  - O usuário está logado no sistema Audora;
  - O usuário se registrou para participar da eleição (fora do sistema de eleições);
  - Há somente uma eleição vigente;
  - Os candidatos da eleição vigente foram registrados anteriormente (fora do sistema de eleições);
  - A eleição vigente tem, no mínimo, um candidato cadastrado.

- Fluxo principal:
  - O usuário acessa o sistema Audora;
  - O sistema solicita o token do usuário;
  - O usuário informa seu token;
  - O usuário acessa o sistema de eleições;
  - O sistema exibe os candidatos da eleição vigente;
  - O usuário seleciona um dos candidatos;
  - O sistema solicita o token do usuário;
  - O usuário recebe confirmação de que seu voto foi registrado;
  - O usuário faz download de seu certificado de participação;
  - O usuário fecha o sistema de eleições.

- Fluxo alternativos:
  - **FA1:** O usuário fecha o sistema antes de confirmar sua decisão sobre o voto. Nada é registrado no sistema;

- Fluxos de exceção:
  - **FE1:** O usuário informa um token inválido ao tentar acessar o sistema de eleições. O sistema Audora notifica sobre o erro e solicita o token novamente;
  - **FE2:** O usuário informa um token inválido ao tentar votar. O sistema de eleições notifica sobre o erro;
  - **FE3:** O usuário tenta votar uma segunda vez informando seu token no sistema de eleições. O sistema notifica que o usuário já realizou seu voto;

4.1.4 Diagrama de Atividade

A Figura 20 exibe o diagrama de atividade que representa as ações, que o usuário pode tomar, no caso de uso "Realizar Votação":

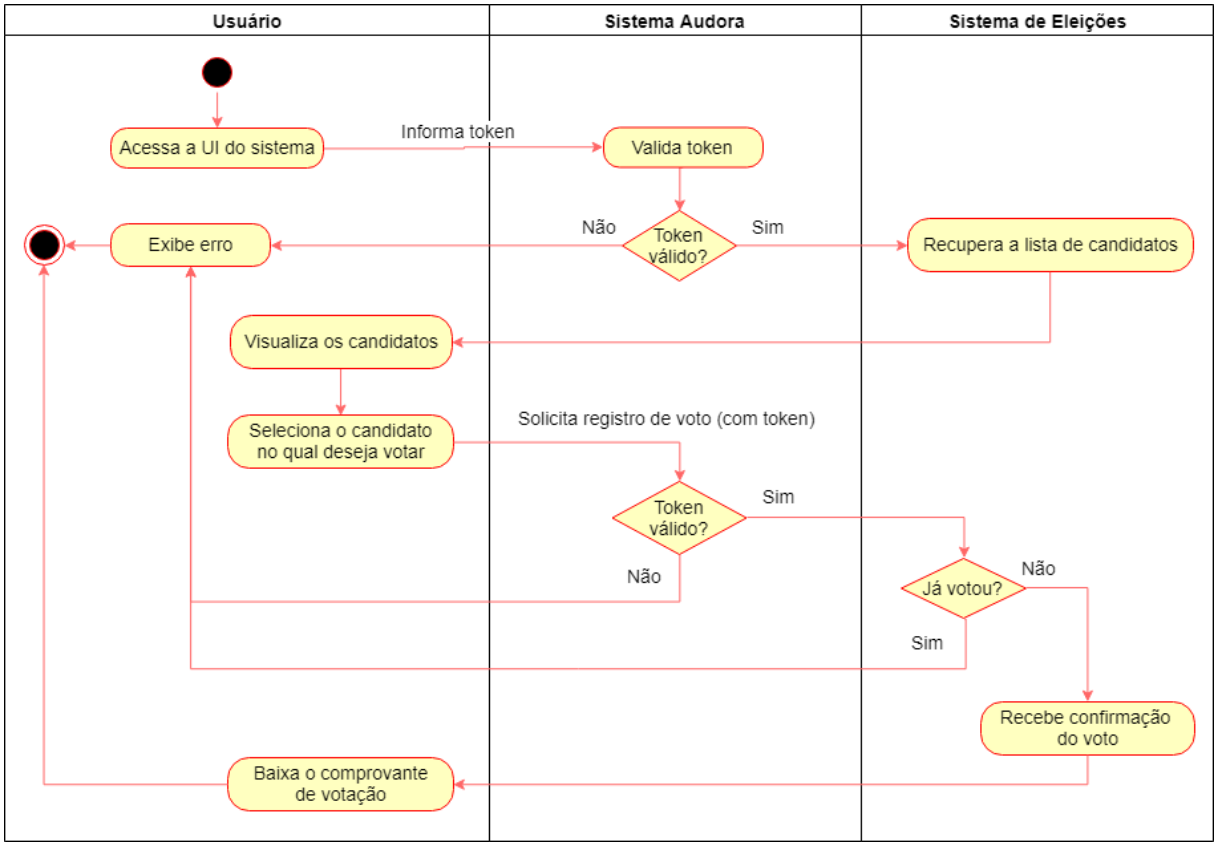


Figura 20: Diagrama de Atividade do sistema.

Fonte: Elaborado pelos autores.

4.2 Modelo Relacional Atual

O modelo relacional, na Figura 21, apresenta como as tabelas do banco de dados estão estruturadas e como elas se relacionam no banco de dados do sistema atual.

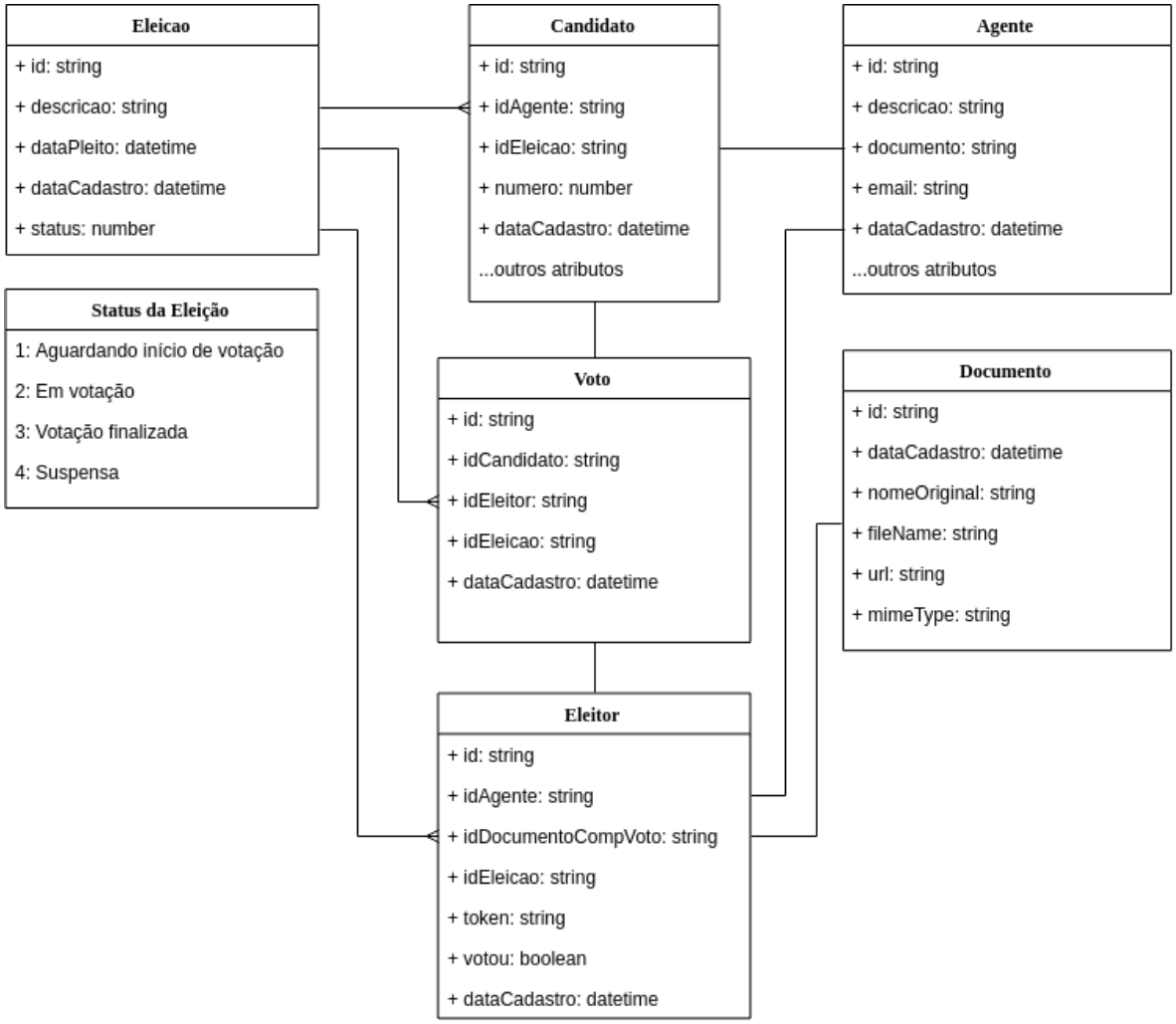


Figura 21: Modelo relacional do sistema.

Fonte: Elaborado pelos autores.

4.3 Modelo Relacional Proposto

O modelo proposto neste trabalho não elimina a existência do modelo atual, mas integra-se a ele de forma que o caso de uso para o sistema pode continuar o mesmo. Os modelos na *blockchain* refletem apenas parte dos dados que existem no modelo relacional, pois a rede só ficará responsável por garantir que os requisitos da eleição sejam atendidos.

Os modelos foram declarados em três arquivos e podem ser visto abaixo:

models/org.audora.eleicoes.candidato.cto

```
1 namespace org.audora.eleicoes.candidato
2
3 participant Candidato identified by candidatoId {
4   o String candidatoId
5   o String eleicaoId
```

```
  o Integer totalVotos
}
```

models/org.audora.eleicoes.eleitor.cto

```
namespace org.audora.eleicoes.eleitor

participant Eleitor identified by eleitorId {
  o String eleitorId
  o String eleicaoId
  o Boolean votou default=false
}
```

models/org.audora.eleicoes.voto.cto

```
namespace org.audora.eleicoes.voto

import org.audora.eleicoes.candidato.Candidato

asset Voto identified by votoId {
  o String votoId
  o String eleitorId
  o String candidatoId
  o String eleicaoId
}

transaction Votar {
  o String eleitorId
  o String candidatoId
}

event VotoRegistrado {
  o String votoId
  o String eleitorId
  o String candidatoId
  o String eleicaoId
}

event VotacaoEncerrada {
  o String eleicaoId
  o Candidato[] candidatos
}
```

O modelo `Candidato` representa a tabela de mesmo nome no modelo relacional, mas contém somente um identificador do tipo *string* - exigido pelo *Fabric*, uma *string* com o `id` da tabela `Eleicao` do modelo relacional, ao qual se refere, e um *integer* com o total de votos. Este modelo representa um *participant* da rede.

O modelo `Eleitor` representa a tabela de mesmo nome no modelo relacional, mas contém somente um identificador do tipo *string*, uma *string* com o `id` da tabela `Eleicao` do modelo relacional e um *boolean* que representa se o eleitor já votou. Este modelo representa um *participant* da rede.

O modelo `Voto` representa a tabela de mesmo nome no modelo relacional e, além de um identificador do tipo *string*, contém uma *string* com o `id` da tabela `Eleicao` do modelo relacional, uma *string* que representa o `id` do `Eleitor` está votando e em qual `Candidato` ele está votando. Este modelo representa um *asset* da rede.

Junto ao modelo `Voto` está a única *transaction* da rede, `Votar`, cujas propriedades são o `id` de `Eleitor` e de `Candidato`. Os dois eventos da rede também estão descritos junto ao modelo `Voto` - um representando o registro de um voto e outro o encerramento da votação, quando o último eleitor vota.

A lógica para a *transaction* `Votar` encontra-se no seguinte código:

lib/script.js

```
async function votar(votoEleitor) {
  const eleitorRegistry = await getParticipantRegistry(
    'org.audora.eleicoes.eleitor.Eleitor'
  );
  const eleitor = await eleitorRegistry.get(votoEleitor.eleitorId).catch(() => {
    throw new Error('O eleitor especificado não existe');
  });

  if (eleitor.votou) {
    throw new Error('Você já realizou seu voto');
  }

  const candidatoRegistry = await getParticipantRegistry(
    'org.audora.eleicoes.candidato.Candidato'
  );
  const candidato = await candidatoRegistry
    .get(votoEleitor.candidatoId)
    .catch(() => {
      throw new Error('O candidato especificado não existe');
    });

  if (eleitor.eleicaoId !== candidato.eleicaoId) {
    throw new Error('O candidato especificado não está na eleição')
  }

  const votoId = Date.now().toString();
  const factory = getFactory();
  const voto = factory.newResource('org.audora.eleicoes.voto', 'Voto', votoId);

  voto.eleitorId = votoEleitor.eleitorId;
```

```
31    voto.candidatoId = votoEleitor.candidatoId;
32
33    candidato.totalVotos += 1;
34    eleitor.votou = true;
35
36    const votoRegistry = await getAssetRegistry('org.audora.eleicoes.voto.Voto');
37
38    await candidatoRegistry.update(candidato);
39    await eleitorRegistry.update(eleitor);
40    await votoRegistry.add(voto);
41
42    const eventoRegistroVoto = factory.newEvent(
43      'org.audora.eleicoes.voto',
44      'VotoRegistrado'
45    );
46
47    eventoRegistroVoto.eleitorId = votoEleitor.eleitorId;
48    eventoRegistroVoto.candidatoId = votoEleitor.candidatoId;
49    eventoRegistroVoto.eleicaoId = eleitor.eleicaoId;
50    eventoRegistroVoto.votoId = votoId;
51
52    emit(eventoRegistroVoto);
53
54    const todosEleitores = await eleitorRegistry.getAll();
55    const outrosEleitores = todosEleitores.filter(
56      e => e.eleitorId !== votoEleitor.eleitorId
57    );
58
59    if (outrosEleitores.every(e => e.votou)) {
60      const eventoEncerramentoVotacao = factory.newEvent(
61        'org.audora.eleicoes.voto',
62        'VotacaoEncerrada'
63      );
64      const todosCandidatos = await candidatoRegistry.getAll();
65
66      eventoEncerramentoVotacao.eleicaoId = eleitor.eleicaoId;
67      eventoEncerramentoVotacao.candidatos = todosCandidatos.map(c =>
68        c.candidatoId === votoEleitor.candidatoId ? candidato : c
69      );
70
71      emit(eventoEncerramentoVotacao);
72    }
73  }
```

O fluxograma da lógica da *transaction* `Votar` se encontra na Figura 22.

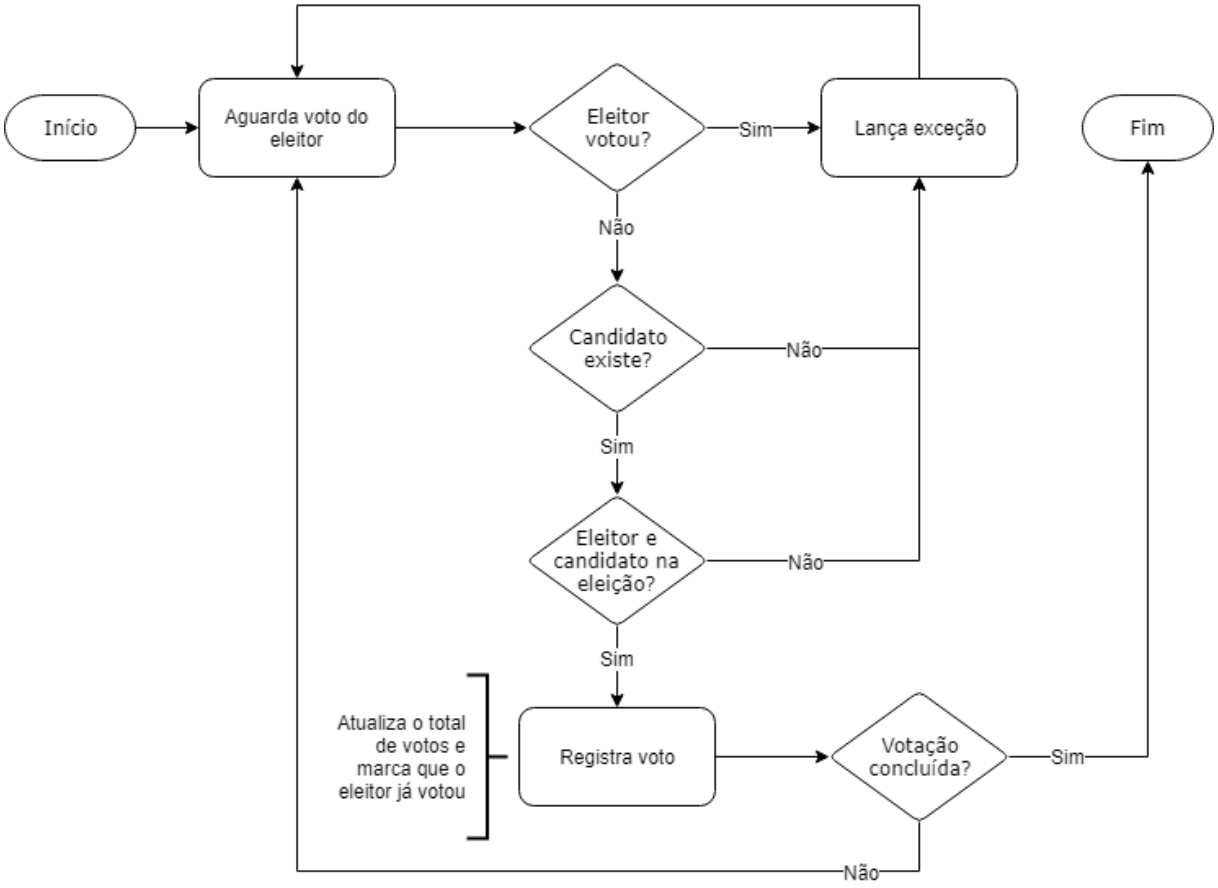


Figura 22: Fluxograma da *transaction* `Votar`.

Fonte: Elaborado pelos autores.

Aplicando as regras do fluxograma anterior, os requisitos da eleição são atendidos na rede e cria-se formas de manter o modelo relacional em sincronia com a rede *blockchain*. Além disso, por haver uma conexão via *WebSocket* entre o cliente e servidor da *blockchain*, as interfaces de usuário de todos os clientes se mantêm atualizadas em tempo real.

Em caso de erro das operações realizadas na *transaction*, o *Fabric* desfaz as operações bem-sucedidas e não persiste as alterações de estado, tirando essa responsabilidade do desenvolvedor (SAKHUJA, 2016).

4.4 Diagrama de Sequência do Modelo Proposto

A Figura 23 descreve como as três partes do modelo proposto – servidor *blockchain*, servidor da Audora e cliente – interagem entre si.

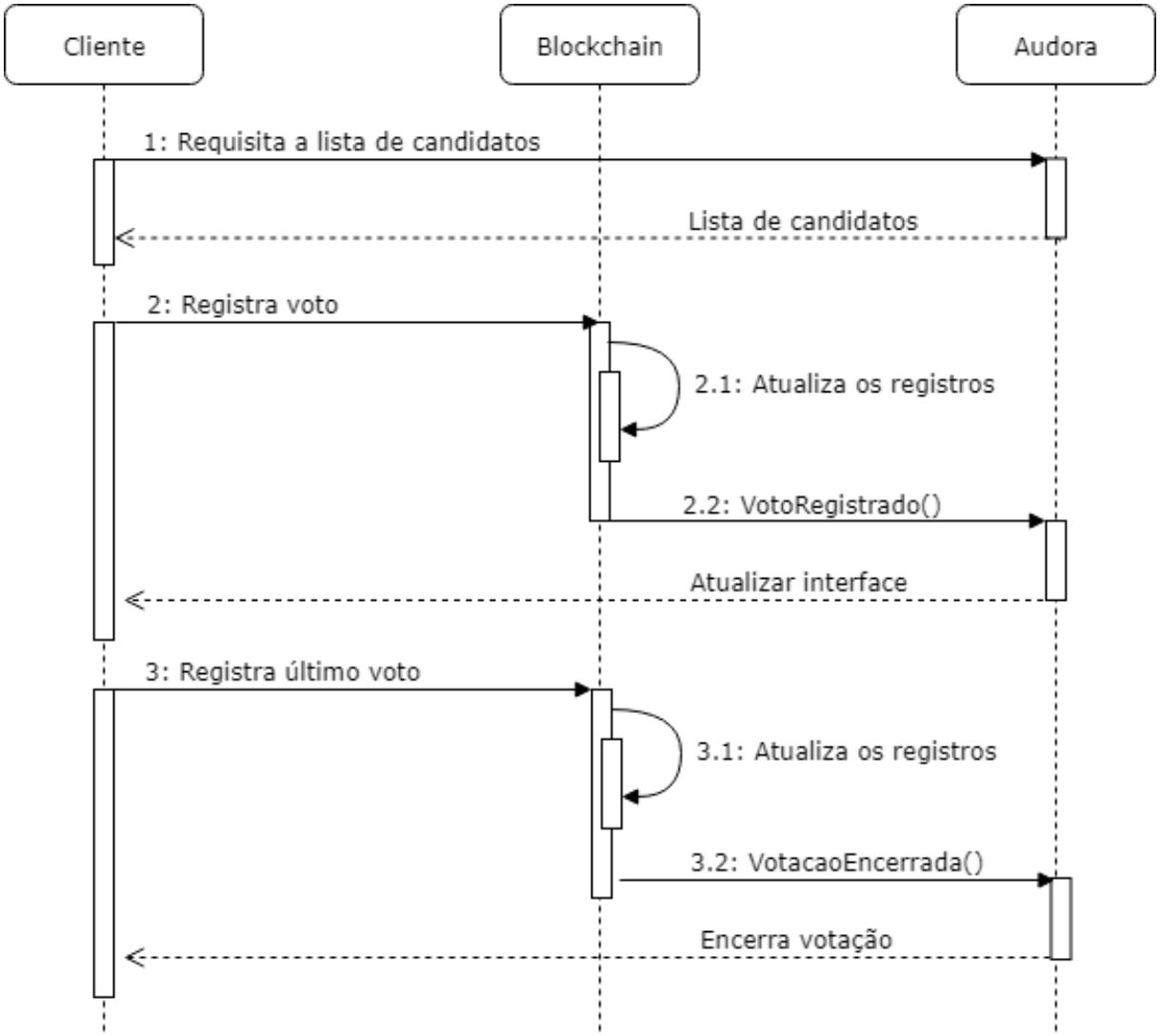


Figura 23: Diagrama de sequência do modelo proposto.

Fonte: Elaborado pelos autores.

4.5 Interação com a *blockchain*

Dentre as ferramentas que compõem o *Hyperledger Composer* está a interface de linha de comandos (*Command Line Interface* - CLI) *composer-rest-server*[1]. Ao iniciar uma rede com essa CLI, uma API REST[2] é disponibilizada para consultar e alterar o estado dos recursos da rede, junto com uma conexão *WebSocket*[3] na mesma porta, por onde os *events* são emitidos, permitindo que aplicações sejam notificadas e reajam às operações na *blockchain* em tempo real.

Essa API REST permite mais que uma implementação de *oracle*, uma vez que

[1] <https://hyperledger.github.io/composer/v0.19/reference/rest-server>
[2] <https://restfulapi.net/>
[3] <https://developer.mozilla.org/pt-BR/docs/WebSockets>

outros serviços podem injetar dados na rede, consultá-los e até mesmo editá-los, uma vez que tenham permissão para tal. A API REST é automaticamente documentada usando *Swagger*[4], o que ajuda muito numa possível integração com outros serviços. A Figura 24 mostra a API REST do modelo proposto.

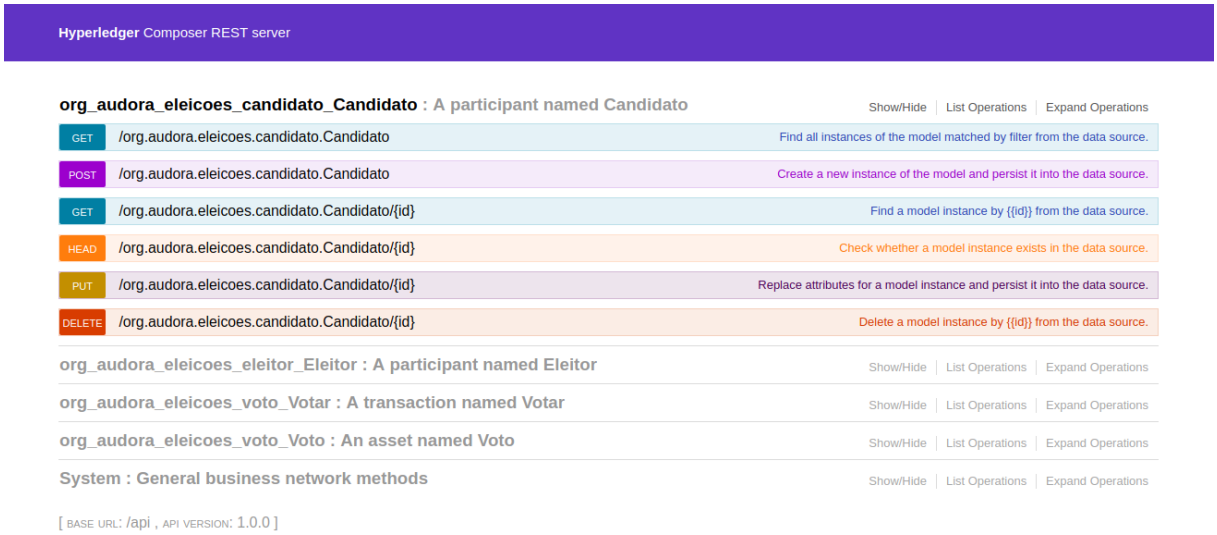


Figura 24: Especificação da API REST disponibilizada pelo *Composer*.

Fonte: Elaborado pelos autores.

4.6 Gerenciamento da rede

Usando a CLI *composer*[5], outra ferramenta do *Hyperledger Composer*, é possível compilar uma rede em um arquivo de extensão `.bna`, que significa *Business Network Application*, usando o comando *composer archive create*. Com o comando *composer network install* é possível instalar a rede na infraestrutura do *Fabric* e com o comando *composer network start* é possível iniciá-la (SAKHUJA, 2016).

A CLI permite a atualização da rede através do comando *composer network upgrade* e possibilita a realização de um *ping* para saber se o *deploy* foi bem-sucedido através do comando *composer network ping* (SAKHUJA, 2016).

4.7 Desenvolvimento e teste

Dentre as ferramentas do *Hyperledger Composer* está a CLI *composer-playground*[6], que fornece uma interface para configuração, *deployment* e teste de uma rede. Os re-

[4] <https://swagger.io/>
[5] <https://hyperledger.github.io/composer/v0.19/reference/commands>
[6] <https://hyperledger.github.io/composer/v0.19/playground/playground-index>

cursos avançados do *Playground* permitem que os usuários gerenciem a segurança da rede, convidem *participants* para redes e conectem-se a várias redes *blockchain*. Ao executar o comando *composer-playground*, a referida interface torna-se disponível na máquina do usuário. Uma versão *online* dessa interface está disponível no link <https://composer-playground.mybluemix.net/>. As Figuras 25, 26, 27 e 28 mostram as principais telas do *Playground*.

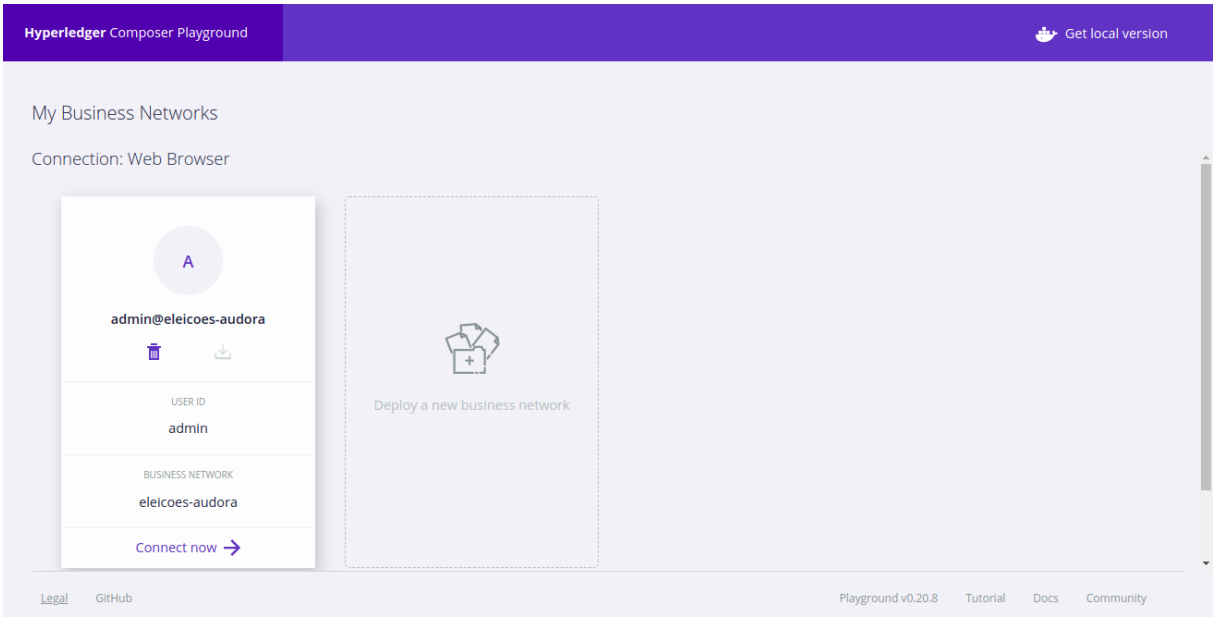


Figura 25: Interface do *Playground* para criar ou instalar redes

Fonte: Elaborado pelos autores.

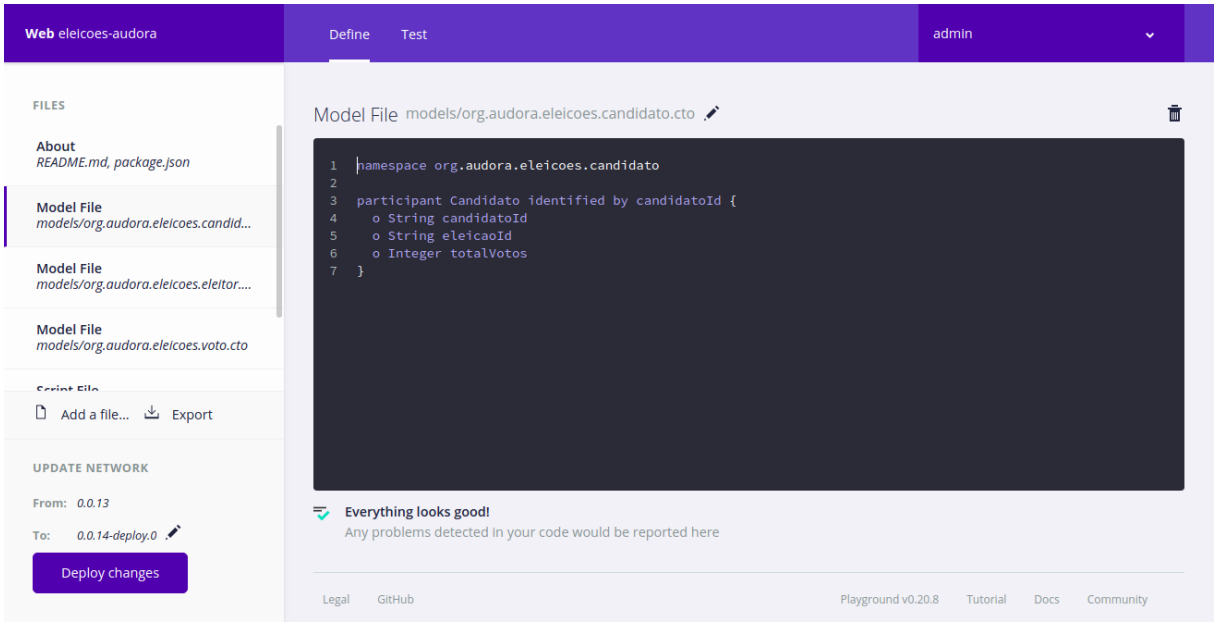


Figura 26: Interface do *Playground* para definição de modelos e lógica

Fonte: Elaborado pelos autores.

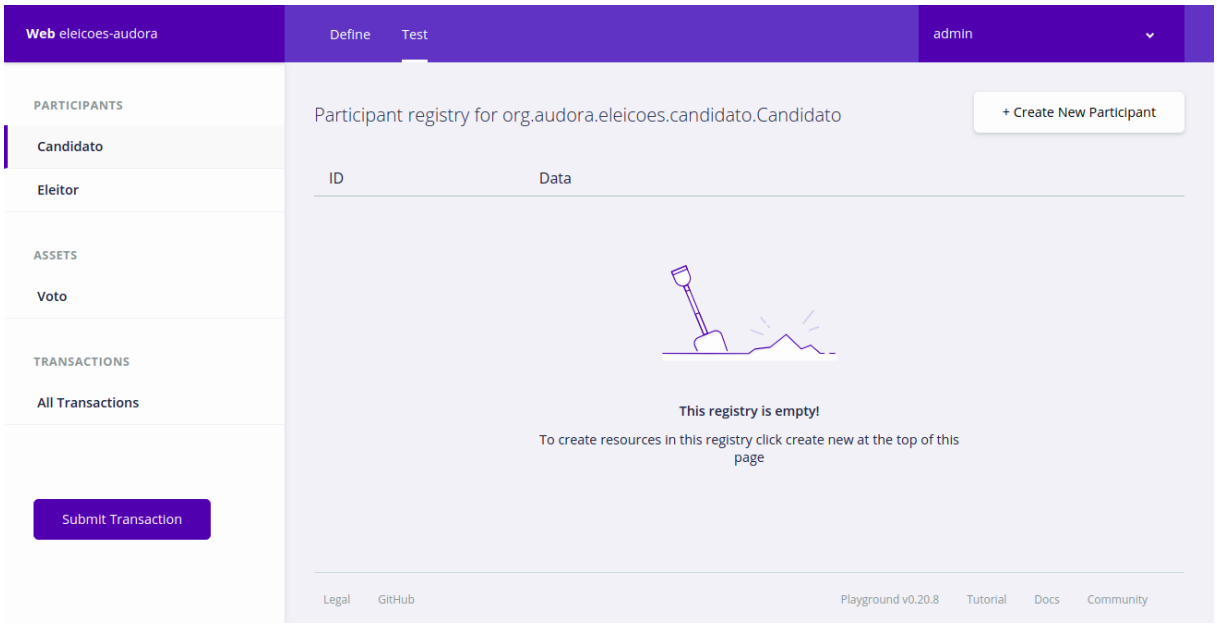


Figura 27: Interface do *Playground* para criar *participants* e *assets*

Fonte: Elaborado pelos autores.

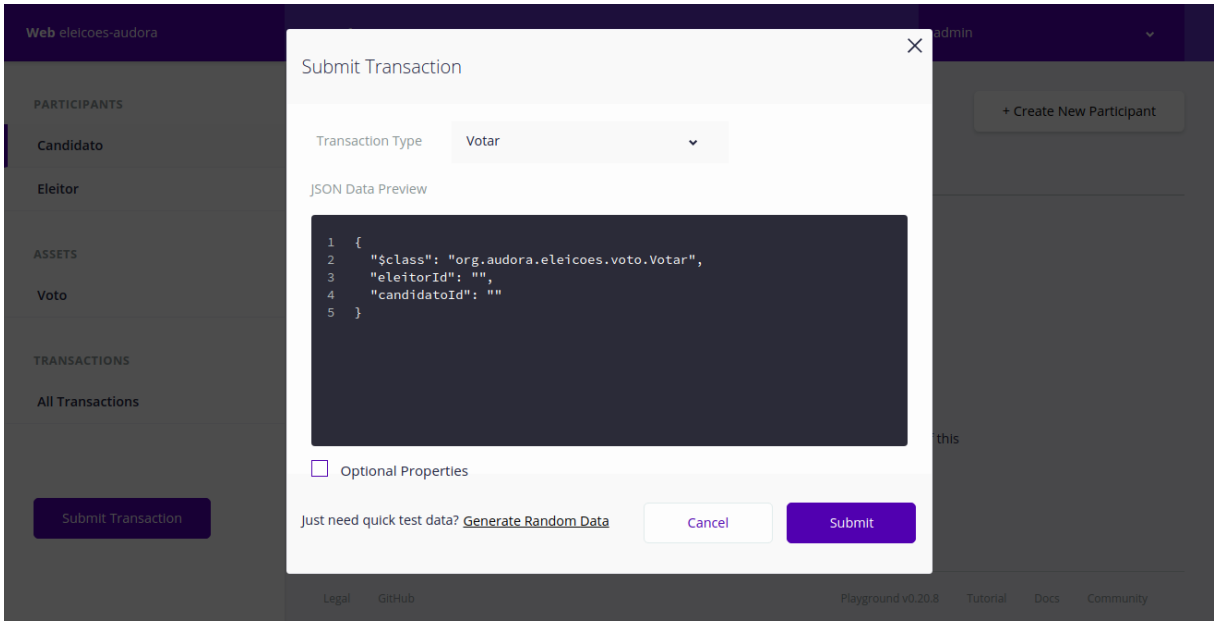


Figura 28: Interface do *Playground* para submeter uma transação

Fonte: Elaborado pelos autores.

5 ANÁLISE E DISCUSSÃO DOS RESULTADOS

5.1 Análise da abordagem atual

Desde 2004, várias pesquisas foram voltadas para os desafios em sistemas de eleição centralizados, protocolos de votação eletrônica e votação descentralizada. Como visto nos capítulos 1 e 2, as principais dificuldades relacionadas a votação eletrônica são o anonimato dos eleitores, a segurança na obtenção dos votos e como evitar fraude no processo de votação.

Na última década, muitos resultados de eleições foram fraudados de diversas formas. Seja através de votação em papel ou de forma eletrônica, muitos são os riscos envolvidos numa eleição. Não importa a forma de votação, há sempre um parte em quem os eleitores devem confiar para realizar a contagem dos votos e não há forma de validar se os resultados estão corretos ou não.

No caso deste trabalho, tal parte é a empresa Audora, responsável pela construção do sistema de eleições. Não estamos assumindo que a empresa teria parte numa possível manipulação dos dados, mas pela arquitetura na qual o sistema atual está construído, é possível que alguém mal-intencionado consiga alterar as regras escritas no *back-end* e permita que alguma fraude seja realizada: voto duplo, remoção do registro de voto, entre outras.

5.2 Análise da abordagem proposta

A abordagem proposta neste trabalho é de uma rede *blockchain* permissionada, onde todos os nós presentes foram previamente autorizados a ingressar na rede e sabem quem faz parte da mesma.

Fazendo uso de *blockchain*, eliminamos a necessidade de uma parte responsável pela segurança da eleição e eliminamos quaisquer suspeitas que possam existir sobre a imparcialidade da Audora no processo de votação.

Os nós podem confiar uns nos outros porque as regras de confiança, conformidade, autoridade, governança, contratos, leis e acordos estão na tecnologia e é praticamente impossível a manipulação das informações, pois pode ser necessária uma grande quantidade de poder computacional para modificar ou alterar algum registro da rede.

Enquanto a abordagem com *blockchain* eliminaria qualquer desconfiança sobre a Audora e daria confiabilidade a uma rede *trustless*, alguns *trade-offs* devem ser considerados, como: a introdução de uma nova tecnologia e/ou linguagem de programação, no ambiente de trabalho, implica em novos desafios para os desenvolvedores e um ritmo de desenvolvimento

mais lento.

Contudo, a escolha do *Fabric* reduz essas dificuldades, pois esse *framework* permite a escrita de modelos em uma linguagem altamente descritiva e de fácil entendimento, além de permitir que a lógica das transações sejam escritas em *Go*, *Java* e *JavaScript* – linguagens de programação altamente difundidas e de fácil entendimento. Além disso, a documentação desse *framework* é bem documentada e sua comunidade cresce a cada dia, sendo fácil encontrar a solução para muitos problemas encontrados durante o desenvolvimento de uma rede *blockchain*.

Comparada com a abordagem atual, a abordagem proposta com *blockchain* seria menos performática, pois todos os nós realizam a mesma tarefa, não permitindo uma execução paralela (*sharding*). Tal situação é perceptível em *blockchains* que têm *smart contracts*, mas é muito mais grave em redes públicas, como relatado por (Christidis; Devetsikiotis, 2016). No caso específico deste trabalho, o fato da rede possuir apenas alguns nós, responsáveis pela validação das transações (*endorsing peers*), o impacto na performance não é percebido pelo usuário.

A natureza da *blockchain* é ser imutável, mas o *Fabric* permite que alterações sejam feitas nos registros por quem tem permissão. Esta característica pode ser um contra-argumento à adoção desse *framework* para implementação de sistemas.

6 CONCLUSÃO

Tendo suas primeiras aplicações em criptomoedas, a tecnologia *blockchain* tem evoluído bastante nos últimos anos e hoje é explorada por diversas áreas, criando oportunidades de novos negócios ou evoluindo negócios existentes, tanto no setor público quanto no privado.

Uma de suas principais características é a ausência de uma entidade central – responsável por realizar, monitorar e relatar todas as transações existentes no sistema – um cenário comum em quase todos os negócios.

A presença de uma entidade central obriga todos os integrantes daquele negócio a confiarem nela, mas a maioria dos sistemas centralizados não são transparentes com o usuário final, o que dificulta a construção da confiança. Neste trabalho, foi abordado um cenário de eleição, onde a desconfiança por partes dos eleitores existe, e uma possível manipulação dos resultados de uma eleição tem impacto imensurável.

O sistema de eleições, construído sob uma arquitetura cliente-servidor tradicional, faz parte do ERP da empresa Audora. Neste trabalho, procurou-se analisar como a tecnologia *blockchain* pode ser utilizada para substituir esse sistema de eleições, ajudando a fortalecer o relacionamento e confiança entre os usuários finais e a Audora.

Uma Revisão Sistemática de Literatura foi realizada para levantar as soluções *blockchain* empresariais/privadas voltadas para eleições. Como muitos trabalhos propuseram sistemas de eleições voltados, principalmente, para *blockchains* públicas (usando *frameworks* ou não), observou-se que apenas um trabalho procurou atender o formato de eleições privadas.

Neste trabalho, com o uso do *framework Hyperledger Fabric*, em conjunto do *Hyperledger Composer*, foi possível contribuir com um sistema de eleições *blockchain*, fornecendo a flexibilidade e controle exigidos por um negócio empresarial.

A possibilidade de customização na implementação de seus protocolos, a granularidade da camada de permissões e a facilidade da integração da rede com outros serviços são pontos favoráveis ao uso do *Hyperledger Fabric* para a criação de redes permissionadas.

Como trabalho futuro, tem-se uma maior integração entre o servidor Audora e a rede *blockchain*, utlizando as ferramentas fornecidas pelo *Hyperledger Fabric* e *Hyperledger Composer*.

REFERÊNCIAS

ADIPUTRA, C. K.; HJORT, R.; SATO, H. A proposal of blockchain-based electronic voting system. In: IEEE. *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. [S.l.], 2018. p. 22–27. Citado na página 18.

ANDROULAKI, E. et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In: *Proceedings of the Thirteenth EuroSys Conference*. New York, NY, USA: ACM, 2018. (EuroSys '18), p. 30:1–30:15. ISBN 978-1-4503-5584-1. Disponível em: <http://doi.acm.org/10.1145/3190508.3190538>. Citado 3 vezes nas páginas 35, 36 e 37.

BASHIR, I. *Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained, 2nd Edition*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78883-904-4. Citado 3 vezes nas páginas 20, 25 e 26.

CACHIN, C.; VUKOLIĆ, M. Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017. Citado na página 32.

CASTRO, M.; LISKOV, B. et al. Practical byzantine fault tolerance. In: *OSDI*. [S.l.: s.n.], 1999. v. 99, p. 173–186. Citado na página 33.

CHEN, G. et al. Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*, SpringerOpen, v. 5, n. 1, p. 1, 2018. Citado na página 15.

CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and smart contracts for the internet of things. *IEEE Access*, Institute of Electrical and Electronics Engineers (IEEE), v. 4, p. 2292–2303, 2016. Disponível em: <https://doi.org/10.1109/access.2016.2566339>. Citado na página 15.

Christidis, K.; Devetsikiotis, M. Blockchains and smart contracts for the internet of things. *IEEE Access*, v. 4, p. 2292–2303, 2016. ISSN 2169-3536. Citado na página 60.

CORRALES, M.; FENWICK, M.; HAAPIO, H. (Ed.). *Legal Tech, Smart Contracts and Blockchain*. Springer Singapore, 2019. Disponível em: <https://doi.org/10.1007/978-981-13-6086-2>. Citado na página 25.

DRESCH, A.; LACERDA, D. P.; JR, J. A. V. A. *Design science research: A method for science and technology advancement*. [S.l.]: Springer, 2014. Citado na página 16.

HYPERLEDGER. *Hyperledger Architecture, Volume 1*. 2017. Disponível em: <https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf>. Citado na página 33.

KONSTANTINIDIS, I. et al. Blockchain for business applications: A systematic literature review. In: _____. [S.l.: s.n.], 2018. p. 384–399. ISBN 978-3-319-93930-8. Citado na página 15.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ACM, v. 4, n. 3, p. 382–401, 1982. Citado na página 32.

METTLER, M. Blockchain technology in healthcare: The revolution starts here. In: IEEE. *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*. [S.l.], 2016. p. 1–3. Citado na página 15.

MOUGAYAR, W. *Understanding the blockchain*. 2019. Disponível em: <https://www.oreilly.com/ideas/understanding-the-blockchain>. Citado na página 25.

MOURA, T.; GOMES, A. Blockchain voting and its effects on election transparency and voter confidence. In: *Proceedings of the 18th Annual International Conference on Digital Government Research*. ACM Press, 2017. Disponível em: <https://doi.org/10.1145/3085228.3085263>. Citado na página 18.

NAKAMOTO, S. et al. Bitcoin: A peer-to-peer electronic cash system. Working Paper, 2008. Citado 3 vezes nas páginas 15, 19 e 33.

NOIZAT, P. Chapter 22 - blockchain electronic vote. In: CHUEN, D. L. K. (Ed.). *Handbook of Digital Currency*. San Diego: Academic Press, 2015. p. 453 – 461. ISBN 978-0-12-802117-0. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780128021170000229>. Citado na página 15.

PRUSTY, N. *Blockchain for enterprise : build scalable blockchain applications with privacy, interoperability, and permissioned features*. Birmingham, UK: Packt Publishing, 2018. ISBN 978-1-78847-974-5. Citado 2 vezes nas páginas 20 e 24.

SAKHUJA, R. Udemy, 2016. Disponível em: <https://www.udemy.com/hyperledger/>. Citado 11 vezes nas páginas 21, 22, 23, 24, 35, 38, 39, 40, 41, 53 e 55.

SINGHAL, B. *Beginning Blockchain : a beginner's guide to building Blockchain solutions*. New York, NY: Apress, 2018. ISBN 978-1-4842-3443-3. Citado 5 vezes nas páginas 27, 28, 29, 30 e 31.

SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: O'Reilly Media, Inc., 2015. Citado na página 25.

TELECO. *teleco.com.br*. 2019. Disponível em: <https://www.teleco.com.br/tutoriais/tutorialitil/pagina_2.asp>. Citado na página 27.

WU, H.-T.; YANG, C.-Y. A blockchain-based network security mechanism for voting systems. In: IEEE. *2018 1st International Cognitive Cities Conference (IC3)*. [S.l.], 2018. p. 227–230. Citado na página 18.

XIWEI. *Architecture for blockchain applications*. Cham, Switzerland: Springer, 2019. ISBN 978-3-030-03034-6. Citado 2 vezes nas páginas 27 e 34.

ZHANG, W. et al. A privacy-preserving voting protocol on blockchain. In: IEEE. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. [S.l.], 2018. p. 401–408. Citado na página 18.

ZHENG, Z. et al. An overview of blockchain technology: Architecture, consensus, and future trends. In: IEEE. *2017 IEEE International Congress on Big Data (BigData Congress)*. [S.l.], 2017. p. 557–564. Citado na página 33.

A ANEXOS

TERMO DE AUTORIZAÇÃO

Neste ato, a empresa Audora Tecnologia e Serviços Ltda., inscrita no CNPJ 06.101.150/0001-78, localizada na Avenida José Pontes de Magalhães 70, JTR, sala 603, ed. Espanha, Maceió/AL. AUTORIZA o uso de sua imagem, bem como a permissão da divulgação de modelagem parcial de seu sistema de eleições, para ser utilizada neste Trabalho de Conclusão de Curso (TCC), elaborado por Arthur Denner Oliveira Santos e Djalma do Nascimento Júnior, alunos matriculados no curso de Sistemas de Informação do Instituto Federal de Alagoas (IFAL), e orientado por Breno Jacinto Duarte da Costa, professor do IFAL, SIAPE 1746551. A presente autorização é concedida a título gratuito. Por esta ser a expressão da vontade da empresa citada, a mesma autoriza o uso acima descrito sem que nada haja a ser reclamado.

Maceió, 17 de Junho de 2019.

(assinatura do representante da empresa)