UNIVERSIDADE FEDERAL DE OURO PRETO

INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS

DEPARTAMENTO DE COMPUTAÇÃO

Anderson Vieira Machado

UM ESTUDO DA TECNOLOGIA *BLOCKCHAIN* UTILIZANDO O *FRAMEWORK* HYPERLEDGER FABRIC E SUA IMPLEMENTAÇÃO

Ouro Preto, MG
2022

Anderson Vieira Machado

UM ESTUDO DA TECNOLOGIA *BLOCKCHAIN* UTILIZANDO O *FRAMEWORK* HYPERLEDGER FABRIC E SUA IMPLEMENTAÇÃO

Monografia II apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti

Ouro Preto, MG
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M149e Machado, Anderson Vieira.
Um estudo da tecnologia blockchain utilizando o framework Hyperledger Fabric e sua implementação. [manuscrito] / Anderson Vieira Machado. - 2022.
27 f.: il.: color., tab..

Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalvanti.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Blockchain. 2. Hyperledger. 3. Hyperledger Fabric. 4. Smart Contracts. I. Marcelo da Cunha Cavalvanti, Carlos Frederico. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

ANDERSON VIEIRA MACHADO

Um estudo da tecnologia blockchain utilizando o framework Hyperledger Fabric e sua implementação

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 01 de Julho de 2022.

Membros da banca

Carlos Frederico M. da Cunha Cavalcanti (Orientador) - Doutor - Universidade Federal de Ouro Preto
Ricardo Augusto Rabelo Oliveira (Examinador) - Doutor - Universidade Federal de Ouro Preto
Célio Márcio Soares Ferreira (Examinador) - Msc. - Linux Place

Carlos Frederico M. da Cunha Cavalcanti, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 01/07/2022.

Documento assinado eletronicamente por **Carlos Frederico Marcelo da Cunha Cavalcanti**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 29/07/2022, às 16:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0370302** e o código CRC **36635AB5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Agradecimentos

A Deus, por ter permitido que eu tivesse saúde e determinação para não desanimar durante a realização deste trabalho.

Aos meus pais e irmão, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

Aos amigos, que sempre estiveram ao meu lado, pela amizade incondicional e pelo apoio demonstrado ao longo de todo o período de tempo em que me dediquei a este trabalho.

Ao professor Carlos Frederico, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

Aos professores, pelas correções e ensinamentos que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.” (Arthur Schopenhauer)

Resumo

Os sistemas de informação trabalham com o modelo arquitetural conhecido como cliente x servidor. Porém, não há uma forma de auditoria segura onde os dados possam ser verificados de forma perene. Com o advento do *Bitcoin*, o mundo veio a conhecer de forma mais ampla o conceito de *blockchain*. Este trabalho propõe o uso de uma rede *blockchain*, utilizando o *framework Hyperledger Fabric* para controle dos ativos de uma organização, através de *smart contracts* com o objetivo de manter os dados e registros de forma perene e de maneira incorruptível.

Palavras-chave: Blockchain. Smart contracts. Hyperledger. Hyperledger Fabric.

Abstract

Information systems work with the architectural model known as client x server. However, there is no form of secure auditing where data can be permanently verified. With the advent of *Bitcoin*, the world became more aware of the concept of *blockchain*. This work proposes the use of a *blockchain* network, using the *framework Hyperledger Fabric* to control an organization's assets, through *smart contracts* to keep data and records in an everlasting and incorruptible form.

Keywords: Blockchain. Smart contracts. Hyperledger. Hyperledger Fabric.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
API	*Application Programming Interface*
CAA	*Certification Authority Authorization*
HTTP	*HyperText Transfer Protocol*
UFOP	Universidade Federal de Ouro Preto
IBM	*International Business Machines*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
JSON	*JavaScript Object Notation*
LTS	*Long-term support*
MIT	*Massachusetts Institute of Technology*
MSP	*Managed Services Provider*
NBR	Norma Brasileira Regulamentadora
P2P	*Peer to Peer*
REST	Representational State Transfer
RSA	*Rivest-Shamir-Adleman*
SI	Sistemas de Informação
SHA-256	*Secure Hash Algorithm*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locate*
WEB	*World Wide Web*

Sumário

1 Introdução

Segurança da informação é uma das chaves para a construção de uma sociedade do conhecimento sólida e organizada. A maioria das definições de Segurança da Informação (SI) (BROSTOFF, 2004; MORRIS; THOMPSON, 1979; SIEBERG, 2005; SMITH, 2002) pode ser sumarizada como a proteção contra o uso ou acesso não-autorizado à informação, bem como a proteção contra a negação do serviço a usuários autorizados, enquanto a integridade e a confidencialidade dessa informação são preservadas. Segundo os padrões internacionais (ISO/IEC 17799, 2005), posteriormente, em 2007, incorporada ao novo esquema de numeração como (NBR 27002, 2005), os atributos básicos da segurança da informação são:

- **Confidencialidade:** propriedade que limita o acesso a informação tão somente às entidades legítimas, ou seja, àquelas autorizadas pelo proprietário da informação;
- **Integridade:** propriedade que garante que a informação manipulada mantenha todas as características originais estabelecidas pelo proprietário da informação, incluindo controle de mudanças e garantia do seu ciclo de vida (corrente, intermediária e permanente);
- **Disponibilidade:** propriedade que garante que a informação esteja sempre disponível para o uso legítimo, ou seja, por aqueles usuários autorizados pelo proprietário da informação;
- **Autenticidade:** propriedade que garante que a informação é proveniente da fonte anunciada e que não foi alvo de mutações ao longo de um processo;
- **Irretratabilidade:** propriedade que garante a impossibilidade de negar a autoria em relação a uma transação anteriormente feita;
- **Conformidade:** propriedade que garante que o sistema deve seguir as leis e regulamentos associados a este tipo de processo.

No Brasil, a norma vigente é a NBR ISO/IEC 27002, que é a mesma norma internacional supracitada.

Em 2008 foi publicado um artigo do programador (ou grupo de programadores) que se identificou pelo pseudônimo de Satoshi Nakamoto. O artigo intitulado *Peer-to-Peer Electronic Cash System* (NAKAMOTO, 2008) instaurou um novo conceito para o sistema bancário mundial propondo a criação de uma moeda digital chamada *Bitcoin*. Esta foi considerada a primeira moeda digital em nível mundial que viabilizou um sistema bancário sem a presença de uma agência reguladora centralizada, como os Bancos Centrais, trazendo à tona um novo paradigma na regulação das finanças mundiais.

Vivenciamos um conjunto de tecnologias disruptivas no campo da segurança da informação. O desenvolvimento de algoritmos criptográficos assimétricos, também denominados de "algoritmos de chave pública", foi um marco nesta área. Neste tipo de algoritmo são gerados um par de chaves e, um texto claro criptografado por uma das chaves só é descriptografado com a outra chave. Assim, não é possível realizar a descriptografia de um texto utilizando unicamente a mesma chave que foi realizada criptografia, como nos casos dos algoritmos criptográficos simétricos. A denominação simétrico se deve porque a mesma chave que criptografa um texto claro é a mesma que descriptografa e assimétrica é porque necessita um par de chaves.

Algoritmos criptográficos assimétricos são baseados em função "*on-way*" e o mais emblemático, tanto pela sua popularidade e inovação, é o algoritmo RSA (RIVEST; SHAMIR; ADLEMAN, 1978), nome criado pelo acrônimo da primeira letra do nome de seus inventores, Ron Rivest, Adi Shamir e Leonard Ademan, pesquisadores do *Laboratory for Computer Science no Massachusetts Institute of Technology* - MIT. O RSA foi concebido em 1972.

Os algoritmos criptográficos assimétricos (CAA, do inglês *Cryptography Asymmetric Algorithms*) permitiram um conjunto de aplicações nunca antes experimentadas como, por exemplo, disponibilizar publicamente uma chave na Internet para ser utilizada por qualquer pessoa que queira enviar uma mensagem privada para um determinado destinatário e só permitir que a mesma seja "entendida" (descriptografada) pelo destinatário, legítimo possuidor do outro par da chave. Caso outras pessoas recebam a mensagem e não possuam o outro par da chave (denominada chave privada), a mensagem é completamente ininteligível.

Com o advento de CAA, garantiu-se um novo nível de segurança de nunca foi alcançado anteriormente pois não há mais necessidade da mesma chave criptográfica ser conhecida pelo destinatário e remetente da mensagem.

No atual cenário, os sistemas computacionais provedores de informações são projetados com o modelo cliente-servidor onde um ou mais clientes solicitam de um servidor de aplicações que processa e retorna sua requisição. No entanto, há casos em que é preciso armazenar os dados de forma persistente e confiável para que estes fiquem disponíveis ao longo do tempo. Embora os sistemas gerenciadores de banco de dados (SGDB) gerem arquivos de *log* para registrar as transações processadas, os banco de dados são suscetíveis a falhas e/ou alterações por atos do desenvolvedor ou por terceiros. A forma como a rede *blockchain* é projetada permite evitar com que vários problemas decorrentes de alterações impróprias ou não autorizadas ocorram, permitindo que os dados armazenados possam ser auditados por todos que têm acesso à rede, tornando os eventos executados e registrados totalmente transparentes. Há várias abordagens para construir uma rede *blockchain* atualmente. Neste trabalho será utilizada a abordagem *Hyperledger Fabric* (ANDROULAKI et al., 2018), proposta e mantida pela Linux Foundation e pela IBM.

1.1 Justificativa

A estrutura de dados computacional conhecida como *blockchain* consiste em uma abstração computacional de um *ledger*[1]. Esta robusta estrutura permite não apenas transacionar moedas digitais mas também gravar perenemente o *log* das transações. No caso de *Bitcoin*, as transações são feitas de forma anônima, computacionalmente distribuída e tolerante a falhas de tal forma que qualquer usuário que tenha acesso à rede possui acesso às transações já realizadas e pode validá-las sem a necessidade de uma autoridade central reguladora. *Blockchain* possibilita o registro incorruptível de transações utilizando técnicas de criptografia avançada tais como SHA-256 e RSA, entre outras, ultrapassando o mundo das moedas virtuais. Exemplos em nossa sociedade são inúmeros, desde o registro de um empréstimo de um livro em uma biblioteca, registro inventarial de uma empresa ou companhia, controle da cadeia de suprimentos, alocação e controle de recursos e, até mesmo, registro de um simples sensor de temperatura entre vários nós de uma rede de sensores. *Blockchain* é uma estrutura para ser provida como software como serviço, também conhecido como SaaS[2]

1.2 Objetivos

O objetivo deste trabalho é um estudo do *framework Hyperledger Fabric* em si e a abordagem para implementação de uma rede *blockchain* para controle de ativos e utilização em um ambiente de uma organização qualquer.

1.3 Organização do Trabalho

Este trabalho será organizado da seguinte forma: a revisão da literatura e fundamentação teórica será apresentada no Capítulo 2, a explanação da proposta do trabalho desenvolvido no Capítulo 3. O Capítulo 4 demonstra os resultados obtidos pelo trabalho proposto considerando instâncias de testes realizados. Por fim, o plano de atividades restantes se encontra no Capítulo 5 e a conclusão é apresentada no Capítulo 6.

[1] Palavra inglesa oriunda da área contábil que significa livro razão onde são realizados os registros contábeis das transações feitas.

[2] SaaS é um termo utilizado em computação no qual designa uma aplicação hospedada em nuvem na qual o mesmo é interoperável possibilitando um design disruptivo do processo de operação. Softwares SaaS também estão relacionados ao modelo de negócio, onde o software é mantido por uma empresa que disponibiliza para outros, como um serviço, pagando uma taxa.

2 Revisão Bibliográfica

2.1 Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica para concepção da rede *blockchain* e, em especial, o *framework* escolhido para o desenvolvimento do trabalho, *Hyperledger Fabric*. Na seções seguintes, serão apresentados uma definição da tecnologia *blockchain*, a definição da técnica de consenso da rede, a maneira como o consenso entre os nós da rede é realizado e os algorítimos envolvidos, contratos inteligentes (*Smart Contracts*) e as particularidades do *framework Hyperledger Fabric*. Também são apresentados alguns trabalhos já realizados com a tecnologia *blockchain*.

2.2 O que é a *blockchain*?

Em termos gerais, a *blockchain* se comporta como uma representação de dados conhecida nos meios de computação como lista duplamente encadeada, porém, com alguns artifícios que a torna diferente de uma implementação comum. Esta estrutura é a representação abstrata de um *ledger* (livro razão) que armazena de forma perene dados referentes a transações realizadas, ou contratos inteligentes. A estrutura opera de forma descentralizada, se comportando como um banco de dados descentralizado, permitindo que os dados armazenados sejam distribuídos em outras máquinas ligadas a mesma rede em formato P2P[1]. O objetivo da descentralização dos dados é a manutenção e recuperação de falhas ocasionadas por hardware ou algum possível ataque cibernético, no qual cada bloco que integra a lista contem as informações referentes a transação realizada e cada bloco é encadeado de forma cronológica linear, respeitando uma ordem hierárquica crescente. Cada transação é registrada em um bloco diferente informando quando foi ocorrido, o que foi realizado, e por quem foi realizado. É possível percorrer os blocos da estrutura tanto da esquerda para direita, quanto da direita para esquerda. A *blockchain* foi originalmente concebida no artigo "*Bitcoin: A Peer-to-Peer Electronic Cash System*"atribuído a Satoshi Nakamoto, sendo que este até então permanece desconhecido não sabendo se o mesmo é uma pessoa ou um grupo de pessoas.

[1] P2P é um modelo arquitetural de rede na qual os participantes são conhecidos como pontos ou pares e possuem a mesma hierarquia, podendo compartilhar informações entre os mesmos.

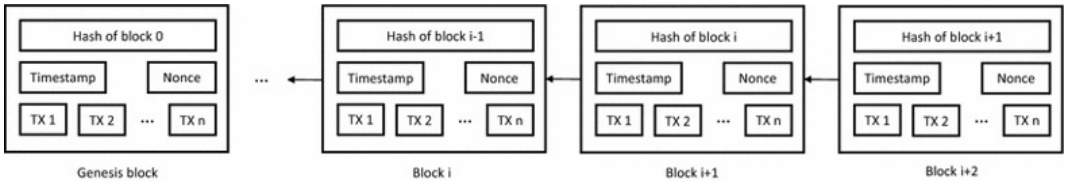

Figura 2.1 – Exemplo de uma rede *blockchain* (ZHENG et al., 2016)

2.2.1 Tipos de *blockchain*

Como apresentado no evento *blockchain Festival*[2], existem 4 tipos principais de *blockchain*, sendo eles abaixo relacionados:

2.2.1.1 *Blockchain* público

É o tipo de *blockchain* mais utilizado pelas criptomoedas, como o *Bitcoin* e o *Ethereum* (WOOD, 2014), originalmente proposto por (BUTERIN et al., 2014). Todas as transações são públicas, os usuários têm direito ao anonimato e qualquer tipo de alteração precisa da validação dos demais participantes. São as cadeias de blocos criptografados mais tradicionais. Foi a partir deles que surgiram todos os demais modelos.

2.2.1.2 *Blockchain* semiprivado

Uma única empresa investe no desenvolvimento de uma rede. Depois, ela fornece acesso a outros usuários, desde que eles atendam a uma lista estabelecida de pré-requisitos. Tecnicamente, essa rede não é descentralizada, já que uma empresa tem controle sobre ela. Suas aplicações podem ser especialmente interessantes para usuários de B2B (business to business), por exemplo.

2.2.1.3 *Blockchain* privado

São áreas restritas, que descaracterizam uma das maiores qualidades do *blockchain*: o fato de não haver um proprietário único da cadeia de informações. Essas redes costumam ser mantidas e utilizadas por uma única empresa. Na prática, são ambientes restritos corporativos tradicionais, que agregam a segurança que as cadeias de blocos criptografados oferecem.

2.2.1.4 Consórcios de *blockchain*

Modelo mais usado neste momento, são os formados por grupos de corporações ou instituições, que dividem o investimento e estabelecem uma lista de pessoas que têm acesso ao sistema. Dependendo do objetivo da rede, o direito de realizar transações e registrá-las nesse sistema pode ser público, ou fechado apenas para os participantes.

[2] O *blockchain Festival* foi um evento ocorrido na cidade de São Paulo, no dia 23/05/2018, onde foi discutido como e por que o *blockchain* vai revolucionar as empresas, as últimas tendências dessa tecnologia em diversos setores, entre outras questões que permeiam este assunto em questão. (Blockchain Festival, 2018)

2.2.2 Benefícios da *blockchain*

Segundo (BASHIR, 2017) existem diversos benefícios que a tecnologia *blockchain* pode trazer para o mercado, tanto para a indústria, comércio, formalização de registros tais como registro imobiliários, cartórios etc. Há alguns principais benefícios, sendo eles:

2.2.2.1 Descentralização

Este é o conceito central que permitiu maior visibilidade à tecnologia *blockchain*. É caracterizado pela não necessidade de uma entidade certificadora para validar um processo ou transação. Assim, o consenso da rede é formado por validações específicas através dos próprios nós da rede. Cada rede *blockchain* utiliza um paradigma para validação e consenso.

2.2.2.2 Transparência e confiança

Em sua concepção, uma rede *blockchain* permite com que todos os usuários da rede tenham acesso aos dados possibilitando transparência e um resultado confiável, pois todos podem observar o que está ocorrendo em cada transação.

2.2.2.3 Imutabilidade

Os dados, quando escritos na *blockchain*, são registrados através de um *log* no qual o mesmo não permite que este seja reescrito ou alterado devido a criptografia empregada, fazendo com que cada transação possua o seu próprio registro em separado por ordem cronológica. No entanto, os dados podem ser alterados sendo controlados por uma forma de versionamento de estados. Isto permite o consenso da rede para verificar se uma transação é correta ou se está viciada.

2.2.2.4 Alta disponibilidade

Como a rede *blockchain* trabalha de forma descentralizada como uma rede P2P, caso algum nó da rede esteja indisponível, isto não ocasiona problema pois todos os *ledgers* (blocos) da rede estão disponíveis a todos os nós permitindo com que a rede se mantenha e se reestruture. A única forma de derrubar a rede *blockchain* seria o não funcionamento de todos os nós em conjunto.

2.2.2.5 Altamente seguro

Todas as transações, e também, os *logs* das transações na rede *blockchain* são criptografados com criptografia assimétrica possibilitando sua integridade.

2.2.2.6 Simplificação de paradigmas

Geralmente, o modelo de controle de finanças e logística das empresas e também na indústria é um tanto confuso e complexo devido a necessidade de integração de mais de uma cadeia de controle ou então, não só por este motivo, mas também pela forma na qual é administrada. Muitas vezes não há o uso de tecnologias para facilitar e ou permitir um controle efetivo da cadeia de produção e administração em geral. A tecnologia *blockchain* permite com que o controle seja realizado de forma objetiva e simplificada, permitindo controlar tanto ativos quanto toda e qualquer transação executada.

2.2.2.7 Negociações mais rápidas

As negociações de ativos e ou qualquer tipo de negociação necessita de um ou mais mediadores para que a mesma ocorra, possibilitando o uso de mais de um sistema de diferentes fabricantes e não só, também, o fato de algumas empresas utilizarem meios não informatizados para verificar, reconciliar, e manter os dados de forma transparente de uma maneira objetiva e prática. O uso da *blockchain* permite que todos os processos da negociação sejam implementados e executados sobre um mesmo paradigma, permitindo a união dos processos fornecendo agilidade e transparência em todo o processo.

2.2.2.8 Economia de custeio

Como não há a necessidade de uma autoridade certificadora que geralmente é alheia a unidade que solicita as transações, todos da rede podem autenticar e verificar a autenticidade das transações ocorridas. Por este motivo, não há a necessidade de mais custos com terceiros para realizar este processo.

2.2.3 Possíveis problemas enfrentados

2.2.3.1 Complexidade

Em relação a um paradigma comum para administração da logística e controle de transações em uma rede, a grande maioria das pessoas está acostumada ao antigo modelo operacional voltado ao papel e documentos certificados em cartório ou entidades certificadoras que validam o processo. A mudança para este novo paradigma requer a aceitação e o treinamento técnico dos participantes envolvidos para que toda a infraestrutura antiga seja deposta para que se aplique este novo conceito.

2.2.3.2 Tamanho da rede

É necessário que haja um número aceitável de nós para que a rede se comporte de forma segura e satisfaça também o custo financeiro para implantação desta tecnologia.

2.2.3.3 Erro humano

Se a rede for utilizada como um banco de dados, é necessário que as operações sejam realizadas por pessoas idôneas para que a mesma se torne de fato confiável. No entanto, como há o consenso entre os participantes para que um dado seja validado, isso pode ser contornado. Uma rede *blockchain* apenas aceita uma transação se e somente se esta for aprovada pelo *smart contract* ao qual ela é submetida. Logo, há a realização de uma operação de consenso na rede para que os nós participantes possam validar a operação que está sendo submetida.

2.2.3.4 Política

Levando em consideração a Lei 13.859 de 08 de julho de 2019 (BRASIL, 2019), conhecida como LGPD - Lei Geral de Proteção de Dados, a obtenção de dados pode se tornar um processo um tanto moroso ou ser dificultado devido a, por muitas vezes, falta de protocolos padrão adequados para interligação entre sistemas, e não só, mas também o fato de o sistema já está consolidado no velho formato de uma entidade terceirizada alheia ao processo ser o certificador ou autenticador das transações. Com a implementação de uma rede *blockchain* permissionada, os dados podem ser armazenados e validados sem a necessidade de uma inconstância no processo de aquisição e ou alteração.

2.3 Características da *blockchain*

Nesta seção serão apresentadas as principais características de concepção e operação da tecnologia *blockchain*.

2.3.1 DLT - *Distributed Ledger Technology*

A tecnologia *blockchain* também é conhecida sob o nome de *Distributed Ledger Technology* devido a mesma operar sobre este conceito. (NAKAMOTO, 2008) apresentou o primeiro artigo descrevendo um uso público para esta tecnologia, propondo uma moeda digital na qual um livro-razão, em inglês *ledger*, que registra as operações financeiras realizadas e que permanece totalmente transparente a todos permitindo uma melhor fluidez e transparência. Isto se deve ao fato de cada bloco da cadeia ter uma cópia de todo o *ledger* e, somado a isto, por estar distribuído, permite a auditagem sem a necessidade de uma entidade certificadora. Ao utilizar este tipo de paradigma, é possível distribuir a aplicação, no caso, a cadeia de *ledgers*, a todos os pontos da rede, que geralmente opera em P2P. Porém é necessário que haja consenso na rede para validar qual ou quais transações são válidas.

2.3.2 Técnica para consenso da rede

Existe um problema extremamente famoso em ciência da computação conhecido como *O problema dos generais bizantinos* (LAMPORT; SHOSTAK; PEASE, 1982). Em (NAKAMOTO, 2008) é apontada uma abordagem extremamente inteligente e elegante para a solução do problema anteriormente citado. Para controle, a inserção de um novo bloco na *blockchain* é assinado por um carimbo de tempo (*timestamp*), (HABER; STORNETTA, 1991; MASSIAS; AVILA; QUISQUATER, 1999), no qual é armazenado de forma cronológica cada registro realizado em cada um dos blocos integrantes da rede. Para validação de um bloco, é necessário que os outros nós validem o novo bloco para que ele possa ser inserido. Caso haja uma bifurcação na sequência de blocos em algum dos nós participantes da rede, é necessário um consenso para que seja decidido qual dos ramos é o ramo válido, e para isso, devido a sincronização ser realizada por meio de um algoritmo de inundação para redes ponto a ponto (P2P), também conhecido como *gossip protocol* (porotocolo de fofoca), todos os nós da rede possuem os blocos participantes já validados. Um dos algoritmos utilizados é o (DEMERS et al., 1987) que replica de forma epidêmica um banco de dados em uma rede P2P. Cada bloco é encadeado na sequência válida de maneira criptografada por uma função resumo, geralmente a SHA-256 devido a sua maior resistência a colisão, ou RSA (RIVEST; SHAMIR; ADLEMAN, 1978), e organizados estruturalmente por uma árvore de Merkle (MERKLE, 1980) que permite uma organização estrutural bem enxuta possibilitando o encurtamento do tamanho dos blocos (NAKAMOTO, 2008) e por consequência, menor espaço para armazenamento da cadeia.

2.3.3 *Smart Contracts*

O conceito de contratos inteligentes foi primariamente definido em (SZABO, 1997) por Nick Szabo. Consiste em um protocolo auto executável que possibilita por fim no modelo tradicional de firmamento de acordos entre duas partes, ou mais que duas partes. É realizado de forma computacional sem a necessidade de um intermediário para certificação ou validação do processo. O processo se estabelece devido a possibilidade de validar as entidades participantes por meio de uma assinatura digital ou por meio de criptografia assimétrica onde os envolvidos realizam suas transações online de forma mais objetiva e prática, diminuindo consideravelmente a morosidade dos sistemas até hoje conhecidos. Um *smart contract* permite com que sejam estabelecidas todas as diretrizes do acordo, tais como quais são os ativos envolvidos, as regras de negócio, e também, as sanções caso o acordo não seja cumprido como previamente estabelecido.

2.4 Por que utilizar a *blockchain*?

O *Bitcoin* foi o primeiro sistema a utilizar a tecnologia *blockchain* como estrutura de construção e administração do sistema. Desde então, é utilizado como exemplo de construção para outros setores que não só o financeiro, tais como o setor de energia (BURGER et al., 2016), (LA-

VRIJSSEN; CARRILO, 2017), cadeia de suprimentos e logística (IANSITI; LAKHANI, 2017), (KORPELA; HALLIKAS; DAHLBERG, 2017), (TITAN, 2016), industria musical (Rethink Music Initiative, 2015), e também no setor de saúde (HOY, 2017).

2.5 O projeto *Hyperledger*

O projeto *Hyperledger* nasceu de um consórcio visando um trabalho colaborativo entre várias empresas, tais como a *Linux Foundation* e a *IBM*. O projeto foi iniciado no ano de 2015 e abraçado pela *IBM* no qual o objetivo é utilizar a tecnologia *blockchain* para suportar transações com mais transparência, controle e segurança. Em 19/05/2015, Brain Behlendorf foi indicado como diretor executivo do projeto (The Linux Foundation Projects, 2016). Como apresentado na figura 2.2, o projeto *Hyperledger* opera como uma incubadora de projetos na qual um dos primeiros projetos a ser incubado foi a (Proposta OpenBlockchain, 2016) da IBM mais tarde tendo seu nome modificado para *Hyperledger Fabric*. Em maio de 2016 o livro razão da Intel também foi incubado sob o nome de *Hyperledger Sawtooth* (Proposta Sawtooth, 2016). Existem outros projetos também incubados como pode-se observar na figura 2.2.

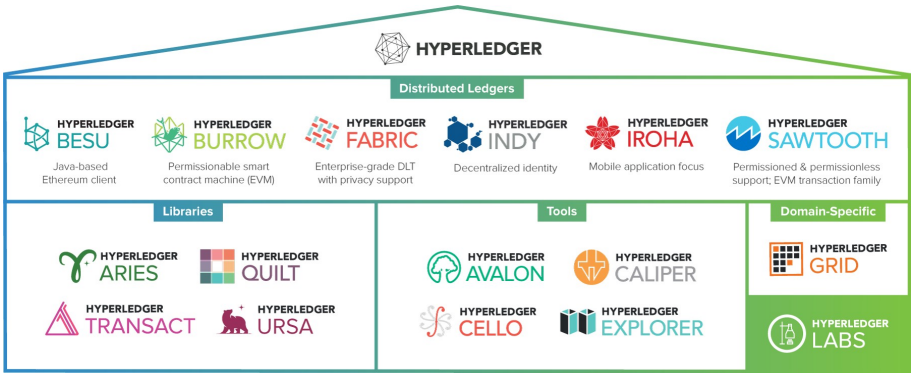

Figura 2.2 – Projeto Hyperledger

2.5.1 O framework *Hyperledger Fabric*

O *Hyperledger Fabric* (ANDROULAKI et al., 2018), consiste em um *framework* criado pela IBM que utiliza o conceito de uma *blockchain* permissionada ou *blockchain* semi-privada. O objetivo é conceder permissões a somente quem possa acessar as transações e ocultar a rede como um todo dos outros nós. Para isso é utilizado o conceito de canal, que nada mais é do que uma forma de ligação entre uma rede e outra, permitindo que as duas se comuniquem. O princípio parte da teoria dos conjuntos onde o conjunto total de nós da rede é subdividido em subconjuntos nos quais cada um deles tem a necessidade específica atendida, ou seja, se comunica e tem permissões apenas onde é necessário. Uma simples analogia para entendimento deste modelo é a de um transmissor Wi-Fi, tal como um roteador *wireless*. São estabelecidas "frequências"nas quais os nós se enxergam e estas são separadas por canais de comunicação permitindo que somente aqueles

que estiverem ligados ao mesmo canal tenham acesso as informações transacionadas. Todos os nós da rede são conhecidos e a cada um é atribuída uma identidade que o identifica de forma única por meio de certificados padrão X.509 (ITU - International Telecommunications Union's, 2017). A rede *blockchain* por meio da implementação do *Hyperledger Fabric* permite que cada companhia que adote o *framework* para controle e automação de seus processos, possua um ou mais MSP[3]. A rede pode ter um ou mais MSPs. Os nós da rede em *Hyperledger Fabric* não se comportam da mesma maneira, e também não são iguais. Existem três tipos diferentes que serão especificados mais tarde. A sincronia dos nós da rede é controlada através de um *middleware*[4] chamado *kafka* (KREPS et al., 2011), hoje incubado pela Linux Foundation Projects.

2.5.2 Conceitos do *Hyperledger Fabric*

Abaixo, serão apresentados e definidos alguns conceitos que integram o *framework*.

2.5.2.1 Cliente

Consiste em uma aplicação desenvolvida com o SDK do *framework*, e é utilizado pelo participante da rede para iniciar a transação. A mesma é enviada ao *endorssing peer* para ser avaliada.

2.5.2.2 *Peer*

Estes são os nós que cuidam de manter o *ledger* em sincronia com a rede. Existem dois tipos de *peer*, sendo eles o *Anchor peer* responsável por criar os outros blocos da rede, e o *Endorsing peer* responsável por realizar a execução da transação, porém, sem alterar os dados de estado. Ele funciona como um verificador ou validador, utilizado para proteger a rede de um possível erro não intencional, ou então de um ataque externo. É possível criar um *cluster* de *anchor peers* ou de *endorsing peers*.

2.5.2.3 *Orderers*

Os nós orderers fornecem o *backbone*[5] de comunicação da rede. Os nós *orderers* são implementados através do *middleware kafka* (KREPS et al., 2011).

2.5.2.4 *Channel*

Um canal é uma estrutura que permite a ligação de uma parte da rede a outra rede, possibilitando a comunicação e a visibilidade entre as duas para que possa haver transações. Um

[3] MSP - Provedor de Serviços Gerenciados. É um nó que tem a permissão de prover uma identidade e atribuição aos nós participantes da rede.

[4] *Middleware* é um tipo de software intermediário que provê funcionalidades além das já fornecidas pelo sistema operacional que executa a aplicação.

[5] *Backbone* é um esquema de ligação central para sustentação de uma rede, se comportando como uma espinha dorsal para ligação de ramificação para redes de menor porte.

canal se comporta de forma privada isolando a comunicação apenas entre os nós participantes.

2.5.2.5 *Assets*

Define o ativo em questão que participa da transação.

2.5.2.6 *Chain code*

É o código em cadeia utilizado para definir a estrutura do ativo que pode ser representado em JSON (CROCKFORD, 2000) ou por representação binária. Também é utilizado para codificar transações que podem ser executadas no ativo. Além disto, está definido no *chain code* as políticas e critérios de validação das transações, e quantos *peers* são necessários para validar a possível transação.

2.5.2.7 *Ledger*

Ledger é uma abstração computacional de um livro razão, onde são armazenadas as informações das transações realizadas, e também, acompanha o estado do ativo. O *ledger* possui duas partes que são o *log* de transações e o banco de dados de estados. O *ledger* é gerenciado no nó *peer* que recebe os dados das transações e então são atualizados o *log* de transação e o banco de dados de estados. Tanto o banco de dados de controle de *log* quanto o de controle de estados são implementados em *levelDB* (DEAN; GHEMAWAT; Google Inc., 2011), que é um banco de dados noSQL[6] que trabalha com dados em forma de chave valor. O *levelDB* não suporta consultas complexas, e por este motivo o mesmo pode ser substituído pelo *couchDB* (Apache Software Foundation, 2008) que também é um banco de dados noSQL, mas que suporta consultas complexas. O registro de cada *log* das transações é imutável, e os estados de cada ativo registrado no banco de estados é alterado por meio de versionamento, registrando versões diferentes a cada transação referente ao ativo em questão. Cada vez que é realizada uma operação com algum dos ativos, no caso, tais como atualização, criação, exclusão, e ou qualquer outro tipo de manipulação nos dados, o *log* é armazenado de forma perene.

2.5.3 Binários do *framework Hyperledger Fabric*

O *framework Hyperledger Fabric* é composto por seis binários, sendo eles:

2.5.3.1 Fabric-Peer

Contêiner que armazena os binários e o código de controle dos *peers*. Como já apresentado no capítulo de fundamentação teórica, são responsáveis pela leitura e escrita na *blockchain*

[6] O termo noSQL foi utilizado pela primeira vez em 1998 para mencionar um banco de dados opensource desenvolvido por Carlo Strozzi, que não possuía uma interface de operação SQL. O autor alega que um banco de dados noSQL nada tem a ver com um banco de dados relacional. (STROZZI, 2007)

2.5.3.2 Fabric-Tools

Contêiner que armazena os binários e o código de controle das ferramentas do cliente *Hyperledger*.

2.5.3.3 Fabric-CA

Contêiner que armazena os binários e o código de controle para registro, renovação e revogação das identidades dos nós participantes. Como já apresentado no capítulo de fundamentação teórica, é responsável pelos nós MSP.

2.5.3.4 Fabric-Orderer

Contêiner que armazena os binários e o código de controle do *middleware* Apache Kafka. Como já apresentado no capítulo de fundamentação teórica, implementa os nós que servem o *backbone* da rede.

2.5.3.5 Fabric-CouchDB

Contêiner que armazena os binários e o código de controle do Apache CouchDB. Como já apresentado no capítulo de fundamentação teórica, é o banco de dados noSQL que controla os dados dos ativos manipulados e transacionados na rede.

2.5.3.6 Chaincode

Contêiner que armazena os binários e o código de controle para geração das regras de negócio para criação, apreciação e validação dos *smart contracts*. Neste trabalho, será implementado na linguagem de programação Golang.

2.6 Funcionamento da rede

O serviço de MSP provê a identificação única para cada usuário da rede, sendo acionada pelo serviço do *Fabric-CA*. Cada usuário é classificado e recebe regras para autorização nas transações na rede. Cada identidade é um certificado digital que é utilizado para assinar as transações e enviá-las para a cadeia de blocos. A vantagem disso é que ao assinar a transação, é possível verificar quem a está submetendo e isso permite com que apenas usuários permissionados na rede possam realizar transações. Existem as operações de leitura, escrita, sendo cada uma delas atribuída a ao usuário de acordo com a política da rede. Caso o usuário não tenha o privilégio de acesso sua transação será negada.

A aplicação cliente pode ser desenvolvida utilizando o SDK do *framework Hyperledger Fabric* em alguma das seguintes linguagens:

- Node.js
- Java
- Python
- Golang

Através da aplicação cliente é possível solicitar as transações na cadeia de blocos da rede.

Dentro de uma rede, uma companhia (empresa) deve possuir mais de um nó *peer* para que possa ser garantida a redundância dos dados e também para que seja tolerante a falhas. A aplicação cliente envia a solicitação de transação à rede, que é recebida por todos nós *peer* que são responsáveis por endossar a transação. Cada *peer* contem uma cópia do *ledger* que registra todas as transações e informações de estado em cada operação. Cada operação é única e distinta não sendo possível alterar a mesma. Caso seja necessária alguma retificação na transação, deverá ser submetida uma nova transação. O *peer* pode ter duas funções diferentes, sendo ser um nó *endorser* responsável por executar o chamado *chaincode*, instalado no mesmo, sendo este o responsável por validar a transação solicitada. O *chaincode* nada mais é que uma sequência de regras pré estabelecidas por um algorítimo escrito em alguma das seguinte linguagens de programação:

- Golang
- Node.js
- Java

A outra função do *peer* é validar cada operação realizada e registrar os estados através de um banco de dados de baixo nível para registro dos eventos da rede. Este banco é integrado a cada *peer*, sendo ele o *levelDB* ou o *couchDB* (sendo este preferido por permitir operações mais complexas com melhor performance). Estes dois bancos realizam o registro das operações de forma perene. Ao realizar toda e qualquer transação, o *peer* que a realizou assina a mesma com a sua identidade da rede, através do certificado que foi outorgado pelo MSP. Apenas assinar a transação não significa que a mesma será adicionada ao *ledger* e por consequência, replicada a cadeia de blocos da rede. Após esta operação, a transação é enviada aos nós *orderers* para que a transação seja confirmada por todos os outros *peers* da rede.

Os nós *orderes* ficam visíveis publicamente fora da rede. Da mesma forma, é interessante que a rede tenha mais de um nó *orderer* instanciado para garantir redundância, tolerância a falha. É recomendado o uso de pelo menos três nós *orderers* para perfeito funcionamento da rede e, quando necessário escalar, utilizar um número ímpar de nós. Eles ordenam as transações através do algoritmo de consenso RAFT (ONGARO; OUSTERHOUT, 2013), implementado através

do *middleware Kafka* que toma conta do serviço de mensageria e despacho das transações e mensagens entre os nós da rede. Essa operação é completamente descentralizada.

A possível transação é enviada a todos os nós *peer* da rede para que seja aclamada e verificada por todos que possam executar o *chaincode*. Isso é necessário para que seja verificada a autenticidade da transação. Cada registro de aprovação e ou reprovação é registrada pelos nós que realizam o *commit* para controle do estado da rede. Caso uma transação submetida seja reprovada, ou seja, não passe no consenso de todos os nós, a mesma não é registrada no *ledger*. Caso aprovada, a transação é enviada novamente aos nós *orderers* para que sejam replicadas entre todos os *peers* e assim registrada nos *ledgers*.

É responsabilidade dos nós *orderers* controlar a ordenação das transações e garantir a consistência da rede transmitindo a todos os *peers* as transações ocorridas. Estes não possuem uma cópia do *ledger* e muito menos acesso ao mesmo, se bastando apenas a realizar o serviço de ordenação e controle das transações para que todos tenham a mesma cópia exata do *ledger*.

O consenso é obtido da seguinte forma:

1. A aplicação cliente envia a solicitação de transação para os *peers*;
2. A transação é submetida a todos os *peers* que podem executar o *chaincode*;
3. Todos os *peers* assinam a transação com a sua identidade da rede;

Quando coletado uma quantidade suficiente de assinaturas para aprovar a transação, a mesma é enviada para os nós *orderers* para que sejam ordenadas. Várias solicitações podem estar sendo enviadas ao mesmo tempo, inclusive solicitando a mudança de estado de um mesmo ativo. Para que a mesma seja validada, cada operação é registrada com um carimbo de tempo (*timestamp*) de forma que as operações sejam registradas exatamente na ordem que foram submetidas. Isso evita com que, por exemplo, ocorra o problema do gasto duplo de um ativo, e ou também, a apropriação de mais de um usuário de um mesmo recurso.

2.6.1 Resumo das fases do fluxo de uma transação

Primeiro é realizado o endosso da transação após a execução através do *smartcontract*, depois a ordenação das transações e então, a validação das transações que foram submetidas. A filosofia empregada de endossar uma transação se dá pelo motivo de apontar quem realmente precisa validar uma específica transação em questão. Tudo é estabelecido na lógica de negócio determinada no *smartcontract*. São eliminadas todas as transações não determinísticas com o objetivo de evitar um estado inconsistente na rede, pois é necessário que todos os *peers* da rede, dentro do mesmo canal, tenham o mesmo *ledger*. É definido na política de endosso quantos nós precisam validar a transação para que ela seja aceita como válida e possa integrar a *blockchain*. Então, as transações válidas são enviadas para serem publicadas nos *ledgers* e é emitido uma

notificação por envio de mensagens de sucesso e falha a aplicação cliente que a solicitou a transação.

2.7 Comparativo entre redes *blockchain*

A escolha do *framework Hypeledger Fabric* se deve ao fato de o mesmo ser concebido para trabalhar de forma modular, permitir desenvolvimento *open-source* e melhor se adequar ao caso de estudo deste trabalho devido ao mesmo permitir o desenvolvimento de seus módulos de forma desacoplada, possibilitando a escolha de mais de uma tecnologia para cada um de seus módulos. O baixo acoplamento e a alta coesão, permitem que o sistema seja mais versátil e flexível, levando em consideração a gama de tecnologias utilizadas em um ambiente industrial.

A seguir, na tabela 2.1 e na tabela 2.2, são apresentadas características de quatro redes que implementam a tecnologia *blockchain*. É possível realizar uma comparação entre as redes *Bitcoin*, *Ethereum*, *Corda* e *Hyperledger*.

Característica	**Bitcoin**	**Ethereum**
Mantenedor	Desenvolvedores Bitcoin	Desenvolvedores Ethereum
Modelo de blockchain	Público	Permissionado Público ou Privado
Descrição	Plataforma blockchain genérica	Plataforma blockchain genérica
Técnica para consenso	Proof of Work	Proof of Work e Proof of Stake
Geração de Smart Contracts	Não	Sim
Moeda	Bitcoin	Ether Tokens por Smart Contract
Linguagem de Programação	C/C++, Simplicity	Solidity, Vyper, Dart, Delphi, .NET, Golang, Java, Javascript, Python, Ruby, Rust, Simplicity
Pagamento por transação	Sim	Sim
Modularidade	Não	Não
Auditável	Sim	Sim

Tabela 2.1 – Comparativo entre redes *blockchain 1*

Característica	**Corda**	**Hyperledger Fabric**
Mantenedor	R3	Linux Foundation
Modelo de blockchain	Permissionado - Privado	Permissionado - Privado
Descrição	Plataforma DLT para industria financeira	Plataforma blockchain modular
Técnica para consenso	Escolhido de acordo com a aplicação	Escolhido de acordo com a aplicação
Geração de Smart Contracts	Sim	Sim
Moeda	Não	Tokens por Smart Contract
Linguagem de Programação	Java, Kotlin	Golang, Node.js (JavaScript) Python, Java
Pagamento por transação	Não	Não
Modularidade	Não	Sim
Auditável	Sim	Sim

Tabela 2.2 – Comparativo entre redes *blockchain 2*

2.8 Trabalhos Relacionados

Em (GARROCHO et al., 2020) é apresentado a abordagem convencional onde a comunicação em uma planta industrial utilizando a IIoT é realizada M2M - *machine to machine*, ou seja, as máquinas se comunicam entre si utilizando protocolos de redes industriais. Os instrumentos, sensores e atuadores, realizam suas leituras e ações diretamente no processo, repassam para PLC que por sua vez, devolve os dados em formato de engenharia para um sistema supervisório que apresenta o estado da planta ao operador. Esta abordagem permite a automação e proteção dos operadores, uma vez que os processos industriais são de certa forma perigosos e muitas vezes repetitivos. É proposto então a inserção da rede *blockchain* com o uso de contratos inteligentes para a criação de redes inteligentes. Com este feito, é possível auditar a rede com mais segurança, pois esta passa a possuir todas as características de uma rede *blockchain*.

Deve ser levado em consideração que é exposto um problema nesta arquitetura, na qual o PLC e os instrumentos de campo apenas operam em uma abordagem original a qual foram concebidos para uso. Isso demanda o uso de de nuvens para comunicação entre os dispositivos IIoT, decorrência direta do uso da *blockchain*, que demanda vários nós para sua operação.

Em (WAN et al., 2019) também é proposto a inserção de uma rede *blockchain* para integrar a arquitetura da IIoT. O modelo é proposto e divido em camadas, sendo elas uma camada de detecção, uma camada de hub para gerenciamento, uma camada de *firmware* e uma camada de aplicação. Nesta abordagem os instrumentos se encontram na camada de detecção e pelo menos um microcomputador com um determinado poder computacional com o objetivo de obter informações e pré-processar os dados coletados da planta. A camada de hub analisa os dados, criptografa, empacota os dados para gerar blocos e os armazena no banco de dados.

Em (STAMATELLIS et al., 2020) os autores propõe uma solução com o nome de PREHEALTH, que representa um prontuário eletrônico onde os dados dos pacientes podem ser preservados de forma segura com privacidade de anonimato. Utilizando-se dos benefícios do *framework Hyperledger Fabric*, a aplicação consegue gerir de forma distribuída e descentralizada os dados sensíveis que tem uma particularidade peculiar que são os dados médico paciente.

Em (MUKNE et al., 2019) os autores propõe uma integração do *framework Hyperledger Fabric* com o IPFS manutenção de registros de terra. O processo é arcaico em muitos países e a forma de garantir autenticidade dos registros gerenciados é uma tormenta enfrentada devido a falta de transparência no sistema, permitindo a adulteração dos registros e de forma indiscriminada. Com o uso do *framework* aliado ao IPFS, é possível garantir a lisura das transações devido as condições providas pela *blockchain*. Com a eliminação de uma entidade reguladora, é possível eliminar a possibilidade de fraude ao criar um histórico imutável de registros, permanentemente vinculado ao sistema, agilizando a documentação e a manutenção de registros.

3 Desenvolvimento

Neste capítulo serão apresentados a descrição da arquitetura a ser utilizada, o levantamento de requisitos, a arquitetura de hardware e software para implementação, modelagem do sistema, o estudo de viabilidade, implementação e manutenção e os experimentos realizados, tal como as suas descrições e métodos empregados.

3.1 Descrição da arquitetura a ser utilizada

Os testes foram realizados em uma máquina local com a seguinte configuração:

- Processador: Core i5 vPro-4590 CPU @ 3.30GHz x 4
- Memória RAM: 8GB DDR3
- HD 500GB
- Sistema Operacional: Ubuntu 20.04.4 LTS

3.2 Levantamento de requisitos

3.2.1 Arquitetura de hardware e software para implementação

Neste trabalho foi utilizada a implementação de uma rede *blockchain* através do *framework Hyperledger Fabric*. Os *smart contracts* foram desenvolvidos na linguagem de programação Go. O sistema operacional no qual foram realizados os experimentos foi o Ubuntu 20.04.4 LTS[1]. A escolha do sistema operacional de *kernel* Linux se deve ao fato de tanto o Docker operar melhor sob este tipo de sistema.

Para implementação de uma rede *Hyeperldger Fabric*, é necessário o preenchimento de alguns pré requisitos que podem ser observados na página oficial do projeto (FABRIC, 2021). Para este trabalho, está sendo utilizada a versão 2.1.

3.2.2 Instalação dos pré-requisitos e dependências

São pré-requisitos para o desenvolvimento e operação do *framework hyperledger Fabric*:

- Instalação do comando cURL;
- Instalação do Node.js[2] na versão 8.9.x ou superior, sendo a versão 9.x não suportada;

[1] LTS - Long Time Support
[2] Disponível em https://nodejs.org/en

- Atualização do gerenciador de pacotes do Node.js, npm;
- Instalação do software de controle de versionamento GIT[3];
- Instalação da linguagem de programação Golang[4] versão 1.18.x ou superior, e exportação da variável de ambiente para acesso ao diretório onde foram instalados os binários;
- Instalação do Docker[5] em sua versão corrente;
- Instalação de um editor de códigos. O escolhido para ser utilizado neste trabalho foi o Visual Studio Code[6] da Microsoft, pois o mesmo possui perfeita integração com o *TypeScript*[7], e também pelo fato de possuir muitos *plugins* que facilitam o desenvolvimento;
- Download através do *script* com a instalação dos contêineres com os binários do *framework Hyperledger Fabric* através do comando:

  curl -sSL https://bit.ly/2ysbOFE | bash -s

Após o download dos binários, deve-se colocar disponível as variáveis de ambiente do *framework* no $PATH do sistema operacional para que possam ser acessíveis em qualquer parte e invocadas quando necessário. As mesma se encontram na pasta */bin*, sendo elas:

- configtxgen;
- configtxlator;
- cryptogen;
- discover;
- idemixgen;
- orderer;
- peer;
- fabric-ca-client;
- fabric-ca-server

[3] Disponível em https://git-scm.com
[4] Disponível em https://go.dev
[5] Disponível em https://www.docker.com
[6] Disponível em https://code.visualstudio.com
[7] TypeScript é um superconjunto da linguagem JavaScript desenvolvido pela Microsoft que permite escrever códigos utilizando o conceito de programação orientada a objetos, e insere na linguagem a tipagem de dados.

3.3 Modelagem do sistema

Originalmente, este *framework* foi concebido para ser trabalhado com a linguagem de programação Golang (linguagem esta que será utilizada para implementação dos *smart contracts* neste trabalho). Porém, foram desenvolvidos outros SDKs que se tornaram oficiais, sendo eles para Node.js, Java, e um ainda não oficial que pode ser utilizado para testes na linguagem Python.

Devido a rede ser projetada para operar sob a tecnologia Docker, é possível escalar o tamanho da mesma levando em consideração que, é necessário apenas instanciar mais contêineres para que os mesmos sejam integrados a rede em questão, apenas alterando o arquivo *docker-compose.yaml*.

3.4 Estudo de viabilidade

O primeiro passo para identificar se uma aplicação possa ser implementada uma rede *blockchain* é observar os processos em si que a constituem. O que você deseja armazenar na *blockchain*? Quais os objetivos você pretende alcançar utilizando uma rede *blockchain*?

É necessário levar em consideração que devido a própria estrutura da rede em si, por ser descentralizada e distribuída, demanda uma infraestrutura computacional específica. É necessário o uso de mais de uma máquina em que todas devem estar ligadas em rede. O mais recomendável é a instalação em nuvem, pois a aquisição de uma infraestrutura própria para gerir um serviço distribuído é relativamente caro e demanda uma série de condições para que o serviço esteja disponível de forma aceitável. É necessário um local adequado com controle de temperatura, uma conexão de internet boa, servidores distribuídos em locais geograficamente espaçados para promover tolerância a falhas, balanceamento de carga e condição de alimentação energética ininterrupta. Além disso, é necessário a verificação da natureza do negócio e de seus requisitos. Deve-se observar também o mercado alvo, principalmente se for um mercado de nicho.

Quando é necessária uma solução onde os dados trafegados precisem ser guardados de forma perene e as transações precisam ser auditadas e ter total confiabilidade, uma rede *blockchain* é extremamente indicada.

3.4.1 Interface cliente

A interface cliente pode ser construída em formato WEB[8] com o *front-end*[9]. O *front-end* deve ser implementado seguindo o padrão de comunicação REST[10] (FIELDING; TAYLOR,

[8] WEB é o acrônimo de *World Wide Web*, que neste caso, foi utilizado como uma metonímia para referenciar a um sistema aplicado a internet.

[9] Front-end é uma definição para determinar a parte do sistema onde o usuário tem acesso direto. É responsável por coletar dados e realizar um pré processamento para envio das informações adequadas para o sistema em si.

[10] REST - Representational State Transfer - consiste em um padrão arquitetural baseado no protocolo HTTP onde os recursos de um sistema são bem definidos possibilitando sua invocação através uma URI - *Uniform resource identifier* estabelecida no URL - *Uniform resource locator* que acessa o sistema.

2002). As chamadas de sistema são realizadas pela integração com o SDK do *framework*.

3.4.2 Back-end

A construção do *back-end* já é pronta, sendo utilizado os contêineres do *Hyperledger Fabric* instanciados pelo gerenciador de contêineres Docker[11]. A escolha do Docker se da pelo fato de a aplicação operar de forma distribuída em rede, e por este motivo, o gerenciamento da aplicação se torna mais prático utilizando esta tecnologia. Caso contrário o controle deveria ser realizado de forma manual devendo instanciar cada binário no sistema operacional em questão. Isso dificulta a escalabilidade da rede. O Docker permite que os contêineres estejam visíveis quando instanciados na mesma rede por se tratar da mesma aplicação e por serem referenciados por um identificador único. A organização e controle da aplicação para que a mesma se mantenha distribuída de forma consistente pode ser realizada pelo Kubernetes[12].

A comunicação com o *font-end* será realizada por meio do protocolo REST.

Uma representação da arquitetura do sistema é apresentada na figura abaixo:

3.5 Implantação e manutenção e segurança da rede

A primeira coisa a ser feita é a configuração do *Hyperledger Fabric CA* em todos os hosts da rede. Deve-se verificar o tempo de expiração dos certificados emitidos para cada participante da rede. O *Hyperledger Fabric* já tem configurado um tempo de expiração padrão, porém, é possível alterar este período de acordo com a política de segurança da informação da organização. Também devem ser informados os hosts que fazem parte da rede. Estas alterações são realizadas nos arquivos *"fabric-ca-client-config.yaml"* e *"fabric-ca-server-config.yaml"*.

O próximo passo é configurar o arquivo *"core.yaml"* de cada *peer* informando onde está o caminho para os certificados. Também é possível neste arquivo, determinar o nome de cada *peer*. Normalmente é utilizado o padrão de nomenclatura *peerX.ORGANIZAÇÃO.com*, sendo X o número identificador do *peer* em questão, seguido pelo nome da organização que ele faz parte. Na hora de colocar o Raft para operar em produção, deve ser observado as variáveis de ambiente *ORDERER_FILE_LOCATION* e *ORDERER_GENERAL_GENESISPROFILE*, no arquivo *orderer.yaml*, sendo elas respectivamente o apontamento de onde o servidor grava os arquivos de rede e a configuração gerada pelo *"configtx.yaml"*.

Para configuração dos *orderers* para ambiente de produção, exitem algumas alterações que devem ser realizadas no arquivo *"configtx.yaml"*. Deve-se apontar onde estão os certificados.

[11] Docker é um gerenciador de contêineres que virtualiza aplicações a nível de sistema operacional, permitindo que cada contêiner invoque chamadas diretamente do *kernel* do sistema operacional onde está instanciado.

[12] Kubernetes é um sistema para controle e orquestração de contêineres open-source que controla o dimensionamento, implantação e automação do processo. Foi inicialmente proposto pela Google e hoje é mantido pela Cloud Native Computing Foundation.

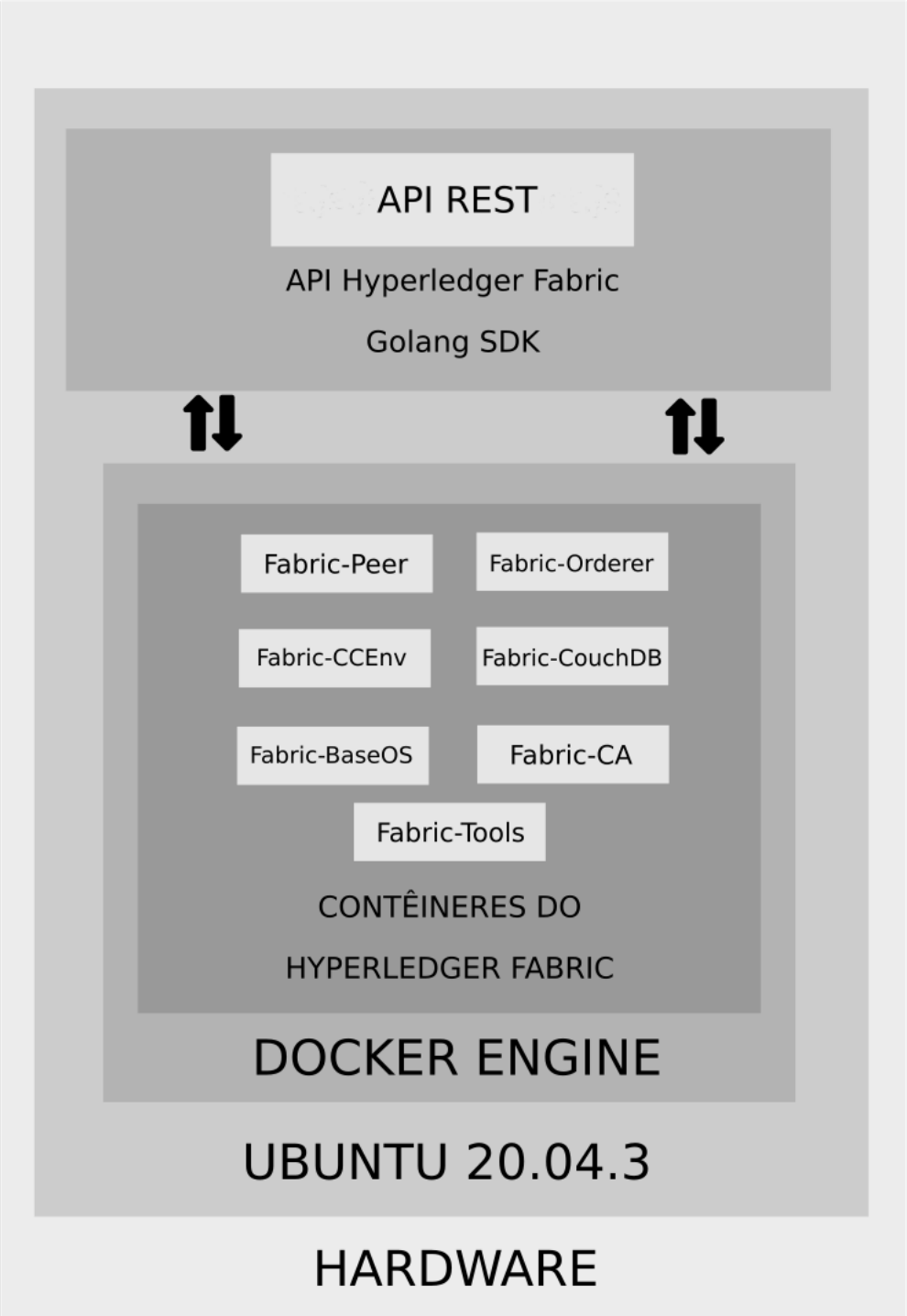

Figura 3.1 – Arquitetura do sistema

É possível e necessário e importante configurar em um ambiente produtivo a monitoração de cada nós, seja ele *peer*, *orderer* ou mesmo os *fabric-CA*. A *blockchain* é uma rede descentralizada e distribuída. Logo, é necessário saber a saúde da rede, observando quais os *peers* estão em funcionamento e quais estão *offline*, quais *orderers* estão em funcionamento e quais estão *offline*. Geralmente é utilizado para coleta de métricas o *Prometheus* (PRO-

METHEUS, 2022) e para visualização é utilizado o *Grafana* (GRAFANA, 2022). Os dois são construídos para dados de séries temporais. O *Prometheus* se destaca na coleta de dados métricos e o *Grafana* promove um ambiente de visualização mais aprimorado. Os dois geralmente são utilizados em conjunto. As variáveis que devem ser configuradas são *ORDERER_GENERAL_OPERATIONS_LISTENADDRESS*, *CORE_OPERATIONS_LISTENADDRESS FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS* para informar o IP e a porta de escuta do respectivo do nó *orderer*, *peer* e *fabric-CA* respectivamente e, para informar o provedor de métricas as variáveis *ORDERER_GENERAL_METRICS_PROVIDER*, *CORE_METRICS_PROVIDER* e *FABRIC_CA_SERVER_METRICS_PROVIDER* para o respectivo do nó *orderer*, *peer* e *fabric-CA*. Pode-se também criar um *script* para ser executado e realizar a coleta de dados do ambiente de rede.

É interessante utilizar o comando *peer chaincode list* para verificar quais os *chaincodes* que estão instalados. O comando *peer chaincode query* para verificar se o *chaincode* está funcionando ou não.

Em ambientes produtivos, instanciar um *chaincode* pode demorar até mais que 10 minutos para que ele seja compilado e instalado no *peer*. Se for necessário reestartar algum *peer* é necessário verificar se o *chaincode* subiu adequadamente. Isso pode ocorrer devido ao conteúdo do arquivo *"package.json"*, logo devemos deixar apenas as dependências que são estritamente necessárias.

A segurança do sistema deve vir sempre em primeiro lugar. Ela depende dos seus parceiros de negócio, pois não adianta você configurar a sua rede de forma adequada e as outra apresentarem margem para ataques. Devem ser verificadas as imagens *Docker* para prevenir falhas de segurança, pois as imagens são um sistema operacional, geralmente montadas com o Alpine Linux (ALPINE. . . , 2022). Então, para garantir que o sistema não tenha conteúdo malicioso é preferível utilizar as imagens que são disponibilizadas pelo *Hyperledger*.

Manter um ambiente de homologação e um ambiente de produção separados. Assim é possível testar os *chaincodes* e observar as formas de operação e seus possíveis efeitos colaterais antes de colocar em produção.

Utilizar o *Kubernets* para poder orquestrar os contêineres do *Docker* e poder escalar a rede de forma mais simples.

Estabelecer os serviços de *peers* e *orderers* de forma distribuída geograficamente. Isso evita com que caso todos estejam armazenados no mesmo *datacenter*, a rede seja derrubada completamente por falha no mesmo.

Criar um sistema de monitoramento externo a rede *blockchain* para verificar como está a saúde de todos os nós integrantes. Assim é possível verificar sobrecargas, queda de algum nó, falta de integração, etc.

Quando a rede é instanciada, o número de *orderer* é sugerido é de pelo menos 3 e quando

escalados, seja em número ímpar. Eles podem ficar junto a apenas uma das organizações que participa da rede, ou estarem distribuídos entre as mesmas para contribuir para que haja resiliência da rede e tolerância a falha devido a rede ser descentralizada e distribuída. Quando uma nova organização entra na rede, ela geralmente instancia um *peer* para que comece a promover suas transações e depois é interessante que sejam instanciados mais *peers* para que haja redundância das informações e o próximo passo é a instanciação de novos *orderers* correspondentes daquela nova organização, aumentando assim a tolerância a falhas e a resiliência da rede.

A resiliência da rede é proporcional a quantidade de elementos que participam da sua rede. Porém, isso leva a uma questão que pode ser encarada como um problema, pois quanto mais *peers* você tem na rede, maior será o tempo para processar uma transação, pois todos devem endossar e validar. Também é necessário avaliar quantos *orderers* em cada região você deve manter para que a rede continue funcional caso haja algum problema. O bacana é subir a rede com alguns *peers* e alguns *orderers* e desligar algumas máquinas observando se a rede continua funcionando.

3.6 Experimentos

Para efeitos de teste e exemplificação, foi criada uma rede de testes com duas organizações, cada uma contendo dois *peers* e uma entidade certificadora cada uma. Para ordenação, foram criados três nós *orderers*.

É apresentado na figura abaixo a arquitetura da rede:

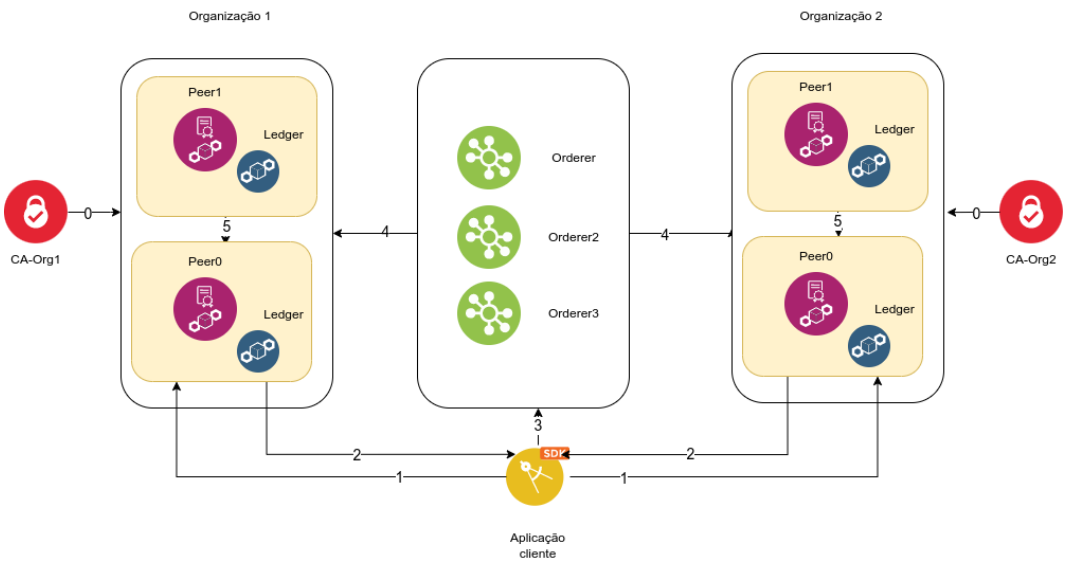

Figura 3.2 – Rede para testes

Para facilitar as operações com os binários do *Hyperledger Fabric*, foram criados arquivos de *shell script* com os comandos necessários para instanciar a rede e seus artefatos. Também foram criados arquivos de configuração e para execução da rede, tal como realizar transações.

4 Resultados

Pôde-se observar que o *framework Hyperledger Fabric* é extremamente versátil, podendo se adequar a condições diversas onde seja necessário a proteção de dados, levando em consideração a abordagem descentralizada e distribuída. Não foi possível a verificação de tolerância a falha, neste trabalho, devido a falta de infraestrutura adequada para realização dos teste e experimentos.

5 Considerações Finais

5.1 Conclusão

A tecnologia *blockchain* permite uma forma disruptiva de abordagem em problemas clássicos, possibilitando a clareza e transparência dos processos. A versatilidade do *framework*, contribui para que sejam aliadas outras tecnologias ao processo e assim, consiga trabalhar como um serviço provido de forma que, a segurança da informação trafegada vem sempre em primeiro lugar.

Por utilizar componentes *plug-and-play*, a integração com sistemas já existentes se torna mais palpável ao fato que, não é necessário substituir toda a arquitetura para que se tenha uma nova abordagem, podendo aproveitar parte dos sistemas legados e algumas formas centralizadas. Isso também permite a escalabilidade em condições favoráveis devido o mesmo operar sobre a tecnologia de contêineres do *Docker* .

5.2 Trabalhos Futuros

Como trabalhos futuros, fica o ensejo de aplicar uma solução dentro da Universidade Federal de Ouro Preto como estudo de caso. É possível vislumbrar várias abordagens possíveis, tais quais o controle de patrimônio, o controle e gerenciamento das bibliotecas, os ensaios e experimentos biológicos, o controle do registro acadêmico entre outros.

Referências

ALPINE Linux. 2022. <https://alpinelinux.org/about/>. Acesso em: 12/06/2022.

ANDROULAKI, E.; BARGER, A.; BORTNIKOV, V.; CACHIN, C.; CHRISTIDIS, K.; CARO, A. D.; ENYEART, D.; FERRIS, C.; LAVENTMAN, G.; MANEVICH, Y.; MURALIDHARAN, S.; MURTHY, C.; NGUYEN, B.; SETHI, M.; SINGH, G.; SMITH, K.; SORNIOTTI, A.; STATHAKOPOULOU, C.; VUKOLIć, M.; COCCO, S. W.; YELLICK, J. 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>.

Apache Software Foundation. *Apache CouchDB*. 2008. <http://couchdb.apache.org>. Acessado em 26/10/2018.

BASHIR, I. *Mastering Blockchain*. [S.l.]: Packt Publishing, 2017.

Blockchain Festival. *Conheça quatro tipos de redes de Blockchain*. 2018. <https://blockchainfestival.com.br/blocknews/conheca-quatro-tipos-de-redes-de-blockchain>. Acessado em 25/10/2018.

BRASIL. Lei nº 13.853, de 08 de julho de 2019. altera a lei nº 13.709, de 14 de agosto de 2018, para dispor sobre a proteção de dados pessoais e para criar a autoridade nacional de proteção de dados; e dá outras providências. *Diário Oficial [da] República Federativa do Brasil*, Brasília, DF, 2019. ISSN 1677-7042. Disponível em: <http://http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>.

BROSTOFF, S. *Improving password system effectiveness*. Tese (Doutorado) — University College London, 2004.

BURGER, C.; KUHLMANN, A.; RICHARD, P.; WEINMANN, J. Blockchain in the energy transition: A survey among decision-makers in the german energy industry. *Deutsche Energie-Agentur GmbH ESMT European School of Management and Technology*, 2016.

BUTERIN, V. et al. A next-generation smart contract and decentralized application platform. *white paper*, 2014.

CROCKFORD, D. *The JSON Data Interchange Syntax - Standart ECMA-404*. [S.l.], 2000.

DEAN, J.; GHEMAWAT, S.; Google Inc. *levelDB*. 2011. <https://github.com/google/leveldb>.

DEMERS, A.; GREENE, D.; HAUSER, C.; IRISH, W.; LARSON, J.; SHENKER, S.; STURGIS, H.; SWINEHART, D.; TERRY, D. Epidemic algorithms for replicated database maintenance. In: *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM, 1987. (PODC '87), p. 1–12. ISBN 0-89791-239-X. Disponível em: <http://doi.acm.org/10.1145/41840.41841>.

FABRIC, H. *Hyperledger Fabric Prerequisites*. 2021. <https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html>. [Online; accessed 16-dezembro-2021].

FIELDING, R. T.; TAYLOR, R. N. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, ACM, v. 2, n. 2, p. 115–150, 2002.

GARROCHO, C. T. B.; KLIPPEL, E.; MACHADO, A. V.; FERREIRA, C. M. S.; CAVALCANTI, C. F. M. da C.; OLIVEIRA, R. A. R. Blockchain-based machine-to-machine communication in the industry 4.0 applied at the industrial mining environment. In: *2020 X Brazilian Symposium on Computing Systems Engineering (SBESC)*. [S.l.: s.n.], 2020. p. 1–8.

GRAFANA. 2022. <https://grafana.com/>. Acesso em: 12/06/2022.

HABER, S.; STORNETTA, W. How to time-stamp a digital document. *Journal of Cryptology*, v. 3, p. 99–111, 1991.

HOY, M. An introduction to the blockchain and its implications for libraries and medicine. *Medical Reference Services Quarterly*, p. 273–279, 2017.

IANSITI, M.; LAKHANI, K. The truth about blockchain. *Harvard Business Review*, 2017.

ISO/IEC 17799. *ISO/IEC 17799*. [S.l.], 2005.

ITU - International Telecommunications Union's. *ISO/IEC 9594-8:2017 - X.509*. [S.l.], 2017.

KORPELA, K.; HALLIKAS, J.; DAHLBERG, T. Digital supply chain transformation toward blockchain integration. *Conference: Hawaii international conference on system sciences (HICSS), At Big Island, Hawai*, v. 50, 2017.

KREPS, J.; NARKHEDE, N.; RAO, J. et al. Kafka: A distributed messaging system for log processing. In: . [S.l.: s.n.], 2011.

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

LAVRIJSSEN, S.; CARRILO, A. Radical innovation in the energy sector and the impact on regulation. *TILEC Discussion*, 2017.

MASSIAS, H.; AVILA, X.; QUISQUATER, J.-J. Design of a secure timestamping service with minimaltrust requirements. *In 20th Symposium on Information Theory in the Benelux*, 1999.

MERKLE, R. C. Protocols for public key cryptosystems. *Symposium on Security and Privacy, IEEE Computer Society*, p. 122–133, 1980.

MORRIS, R.; THOMPSON, K. Password security: a case history. *Communications of the ACM*, v. 22, p. 594–597, 1979.

MUKNE, H.; PAI, P.; RAUT, S.; AMBAWADE, D. Land record management using hyperledger fabric and ipfs. In: IEEE. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [S.l.], 2019. p. 1–8.

NAKAMOTO, S. *Bitcoin: A peer-to-peer electronic cash system," <http://bitcoin.org/bitcoin.pdf>*. 2008.

NBR 27002. *NBR 27002: Tecnologia da Informação - Técnicas de Segurança - Código de prática para gestão da segurança da informação*. [S.l.], 2005.

ONGARO, D.; OUSTERHOUT, J. *In search of an understandable consensus algorithm (extended version)*. [S.l.]: Tech Report. May, 2014. http://ramcloud. stanford. edu/Raft. pdf, 2013.

PROMETHEUS. 2022. <https://prometheus.io/docs/introduction/overview/>. Acesso em: 12/06/2022.

Proposta OpenBlockchain. *Proposta de incubação do projeto OpenBlockchain*. 2016. <https://docs.google.com/document/d/1XECRVN9hXGrjAjysrnuNSdggzAKYm6XESR6KmABwhkE/edit>. Acessado em 23/10/2018.

Proposta Sawtooth. *Sawtooth Lake HIP 1.0*. 2016. <https://docs.google.com/document/d/1j7YcGLJH6LkzvWdOYFIt2kpkVlLEmILErXL6t-Ky2zU/edit>. Acessado em 23/10/2018.

Rethink Music Initiative. Fair music: Transparency and payment flows in the music industry. *Berklee Institute of Creative Entrepreneurship*, 2015.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, ACM, v. 21, n. 2, p. 120–126, 1978.

SIEBERG, D. *Hackers shift focus to financial gain*. 2005. Disponível em: <http://edition.cnn.com/2005/TECH/internet/09/26/identity.hacker/index.html>.

SMITH, R. *The strong password dilemma. Authentication: From Passwords to Public Keys*. [S.l.: s.n.], 2002.

STAMATELLIS, C.; PAPADOPOULOS, P.; PITROPAKIS, N.; KATSIKAS, S.; BUCHANAN, W. J. A privacy-preserving healthcare framework using hyperledger fabric. *Sensors*, v. 20, n. 22, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/22/6587>.

STROZZI, C. *NoSQL: a non-SQL RDBMS*. 2007. <http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page>. Acessado em 23/10/2018.

SZABO, N. Formalizing and securing relationships on public networks. *University of Illinois at Chicago University Library*, v. 2 - Número 9, 1997.

The Linux Foundation Projects. *Founder of the Apache Software Foundation Joins Linux Foundation to Lead Hyperledger Project*. 2016. <encurtador.com.br/enQ34>. Acessado em 30/10/2018.

TITAN, F. An agri-food supply chain traceability system for china based on rfid and blockchain technology. *13th international conference on service systems and service management (ICSSSM)*, 2016.

WAN, J.; LI, J.; IMRAN, M.; LI, D. et al. A blockchain-based solution for enhancing security and privacy in smart factory. *IEEE Transactions on Industrial Informatics*, IEEE, v. 15, n. 6, p. 3652–3660, 2019.

WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. 2014. Acessado em 13/09/2018.

ZHENG, Z.; XIE, S.; DAI, H.-N.; WANG, H. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, v. 1, p. 1–25, 2016.