Capítulo 2

Internet das coisas, blockchain e contratos inteligentes aplicados à saúde

Jauberth Abijaude, Henrique Serra,
Rita Barretto, Aprígio Bezerra,
Péricles Sobreira, Fabíola Greve

Abstract

The Internet of Things aggregates devices able to capture information and interfere in the environment, acting in systems of different domains of application, such as healthcare. These systems need a layer of security to guarantee, among other characteristics, the irrefutability, anonymity, and integrity of the manipulated data. In this sense, an integration with a blockchain, through smart contracts, would meet this need. This chapter, therefore, presents current research using IoT, blockchain, and smart contracts in healthcare. The details for using these technologies in healthcare, the technical challenges and the consensus protocols involved in the main applications will be discussed. This chapter presents a practice that applies knowledge in the health supply chain, building a decentralized application (DApp) that monitors the temperature of vaccines during their storage. In the end, it offers an informative guide that allows participants to design training in this area, including practical exercises.

Resumo

A Internet das Coisas (Internet of Things (IoT)) agrega dispositivos capazes de capturar informações e interferir no ambiente, atuando em sistemas de domínios de aplicações diferentes, como por exemplo o da saúde. Estes sistemas precisam de uma camada de segurança para garantir, dentre outras características, a irrefutabilidade, o anonimato e a integridade dos dados manipulados. Neste sentido, a integração com a blockchain, através dos contratos inteligentes, atenderia a esta necessidade. Este capítulo apresenta, portanto, pesquisas recentes que utilizam IoT, blockchain e contratos inteligentes na área da saúde. Serão apresentados os detalhes para se empregar estas tecnologias na área da saúde, os desafios técnicos e os protocolos de consenso envolvidos nas principais aplicações. Na sequência, apresenta-se uma prática que aplica os conhecimentos abordados na

cadeia de suprimentos para a saúde, construindo uma aplicação descentralizada (DApp) que monitora a temperatura de vacinas durante o seu armazenamento. Ao final, fornece um guia de informações que permite aos interessados a concepção de treinamentos nesta área, contemplando, inclusive, a realização de exercícios práticos.

2.1. Introdução

A Internet das Coisas (IoT - do inglês *Internet of Things*) é capaz de impulsionar várias aplicações médicas, como monitoramento remoto de saúde, programas de condicionamento físico, reabilitação, doenças crônicas e atendimento a idosos [Adibi 2015].

A conformidade com o monitoramento remoto para tratamento e medicação em casa é um potencial importante para a aplicação da telemedicina. Portanto, vários dispositivos médicos, sensores e dispositivos de imagem são essenciais como dispositivos inteligentes ou objetos que constituem uma parte central da IoT para a arquitetura da telemedicina (Kortuem et al., 2010). Assim, o futuro setor de saúde em todo o mundo deve estar preparado para o monitoramento remoto extenso de saúde por meio de IoT e telemedicina (Talalet al., 2019).

Diante dos desafios impostos pela IoT, como segurança e privacidade, a blockchain tem contribuído com a Internet das Coisas Médicas (IoMT) no sentido de melhorar a segurança de dados médicos compartilhados em termos de autenticação de usuário, controle de acesso e privacidade de dados. Também, a mencionada tecnologia tem o potencial de mudar, para descentralizada, a topologia de uma rede de saúde. A blockchain possibilita que os pacientes estejam em um ambiente de ecossistema enquanto aumenta a segurança, a confidencialidade e a interoperabilidade dos dados [Hussien e outros 2021].

Esta seção, de caráter introdutório, tem como objetivo nivelar os conhecimentos básicos nos temas principais a serem abordados nesse capítulo. Ela está dividida nas seguintes subseções: 2.1.1. Internet das Coisas, 2.1.2. Blockchain, 2.1.3. Contratos Inteligentes, 2.1.4. Blockchain e IoT e 2.1.5. Aplicações Distribuídas (DApps).

2.1.1. Internet das Coisas

A IoT é composta por dispositivos físicos com funções de rede, componentes microcomputadorizados e itens incorporados com funções de conectividade [Lao e outros 2020], criando uma classe de objetos inteligentes. Com ela é possível conectar coisas e pessoas a qualquer hora e em qualquer lugar para qualquer serviço. Com o surgimento da IoT, a área da saúde requer uma assistência de outras áreas, principalmente a de Ciência da Computação. Através desta nova classe de objetos inteligentes é possível a aquisição e gerenciamento de registros eletrônicos de saúde para ferramentas que podem auxiliar no diagnóstico e predição de doenças.

A IoT pode ser vista como a combinação de diversas tecnologias, as quais são complementares no sentido de viabilizar a integração dos objetos no ambiente físico ao mundo virtual, ilustrados na Figura 2.1 [Santos e outros 2016].

O bloco de Identificação é um dos componentes mais importantes, pois é fundamental garantir aos objetos inteligentes um meio único de ser reconhecido na internet, como por exemplo um endereço IP. O bloco de Comunicação representa os mecanismos

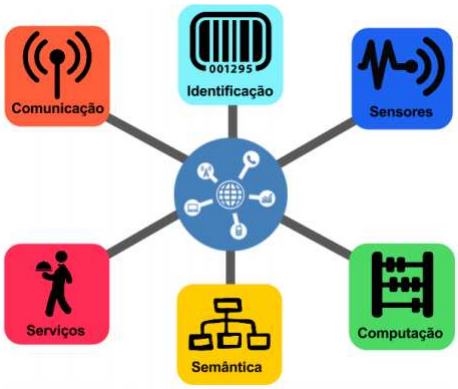

Figura 2.1. Componentes e tecnologias básicas da IoT. *Fonte* [Santos e outros 2016]

para que os dispositivos de IoT possam se comunicar, consideradas as restrições inerentes de potência, processamento e armazenamento. Alguns protocolos usados são o LoraWan, Wi-fi, Bluetooth e Zigbee.

O bloco de sensores representam os sensores e atuadores que coletam informações ou interferem no meio em que se encontram. Os dados capturados são enviados para um middleware, aplicações ou para a blockchain. A computação significa as unidades de processamento capazes de processar algoritmos e disparar as ações de captura de dados ou de acionamento dos atuadores.

Os blocos de serviços e semântica representam, respectivamente, atividades agrupadas em classes (Identificação, Agregação de dados, Colaboração, Inteligência e Ubiquidade) e a habilidade de extrair conhecimentos dos dispositivos de IoT.

De modo geral, a implementação de soluções que abrangem IoT empregam serviços de middleware para abstrair as dificuldades de acesso aos dispositivos devido a heterogeneidade de protocolos e interfaces. A Figura 2.2 ilustra uma organização geral de elementos para Internet das Coisas [Sztajnberg e outros 2018]. Os sensores e atuadores são agrupados em dispositivos e serviços, que por sua vez precisam das interfaces de comunicação para enviar/receber dados das camadas superiores.

Figura 2.2. Organização de elementos na IoT. Fonte [Sztajnberg e outros 2018]

Existe uma variedade de protocolos e padrões de comunicação que podem ser

empregados, entre eles, destacam-se o MQTT, CoAP, AMQP, XMPP, WebSocket, REST, Lorawan, etc.

As práticas deste capítulo utilizam o protocolo REST. Ele permite o uso da infraestrutura do HTTP para acionar ou obter recursos, apenas dando uma interpretação diferente para os métodos GET, PUT, POST e DELETE, e valendo-se da possibilidade de enviar informações adicionais numa mensagem HTTP [Sztajnberg e outros 2018].

2.1.2. Blockchain

A blockchain é uma tecnologia emergente que oferece suporte distribuído confiável para realização de transações entre participantes que não necessariamente têm confiança entre si e que se encontram dispersos numa rede P2P. É considerada uma tecnologia disruptiva e com potencial de substituir entidades certificadoras e centralizadoras das transações de negócios, tais como bancos, governos, cartórios, etc. [Greve e outros 2018].

A primeira rede blockchain, apresentada em 2007, detalhava ao mundo um sistema econômico alternativo com uma moeda digital (Bitcoin) [Nakamoto 2008]. Esta rede permitia transacionar valores digitais através de uma estrutura computacional distribuída. Em 2009, tal rede entra em operação utilizando uma máquina de estado simplificada, com um arranjo até então inédito, que proporcionava eliminar a terceira parte de confiança, necessária para as transações financeiras tradicionais.

Para sustentar esta tecnologia, diversos elementos foram combinados de forma engenhosa e harmoniosa, de forma a pavimentar o caminho para as aplicações descentralizadas. São eles:

- Criptografia: Satisfaz os requisitos de segurança do sistema e das aplicações. Dentre os recursos mais utilizados, destacam-se os resumos criptográficos (funções *hash*) e as assinaturas digitais;
- Consenso distribuído: Permite com que participantes distribuídos coordenem as suas ações, de forma a alcançar decisões comuns, e assim garantir a manutenção da consistência dos seus estados (*safety*) e o progresso do sistema (*liveness*), apesar da existência de falhas [Greve 2005];
- Livro razão distribuído: O livro-razão (*ledger*) é uma estrutura de dados imutável, em que transações são registradas e o estado global do sistema é mantido replicado em todos os nós da rede P2P.

A consequência desta composição garante algumas propriedades que contribuem de forma inovadora para o desenvolvimento de novas soluções tecnológicas, entre elas as aplicações descentralizadas (DApps) como por exemplo [Greve e outros 2018]:

- Descentralização: Sistemas e aplicações que usam a BC não precisam de uma entidade central para coordenar as ações, as tarefas são executas de forma distribuída;
- Disponibilidade e integridade: Os dados e as transações são replicados para todos os participantes da BC, mantendo o sistema seguro e consistente;

- Transparência e auditabilidade: A cadeia de blocos que registra as transações é pública e pode ser auditada e verificada;
- Imutabilidade e Irrefutabilidade: os registros são imutáveis e a correção só pode ser feita a partir de novos registros. O uso de recursos criptográficos garante que os lançamentos não podem ser refutados;
- Privacidade e Anonimidade: As transações são anônimas, com base nos endereços dos usuários. Os servidores armazenam apenas fragmentos criptografados dos dados do usuário;
- Desintermediação: A BC consegue eliminar terceiros em suas transações, atuando como um conector de sistemas de forma confiável e segura;
- Cooperação e incentivos: Uso do modelo de teoria dos jogos como forma de incentivo.

Em 2013, surge a plataforma *Ethereum*, que evolui para além das transações de uma criptmoeda. Implementada sob um modelo de máquina de *turing* completa, com uma nova criptomoeda e ancorada sob alguns conceitos de seu antecessor, esta nova plataforma inova ao permitir que programas de computador possam ser armazenados e executados nas cadeias de blocos. Tais programas, conhecidos como contratos inteligentes.

A blockchain, segundo [Wu e outros 2019], tem uma arquitetura divida em quatro camadas: (i) Dados, onde se encontram os blocos, o armazenamento de dados e a estrutura de árvore utilizada; (ii) Rede, onde se encontra a rede P2P e os mecanismos de comunicação; (iii) Consenso, onde naturalmente estão os protocolos de consenso; e, (iv) Aplicação, onde estão os contratos inteligentes, as criptomoedas e as *sidechains*. A Figura 2.3 ilustra esta divisão.

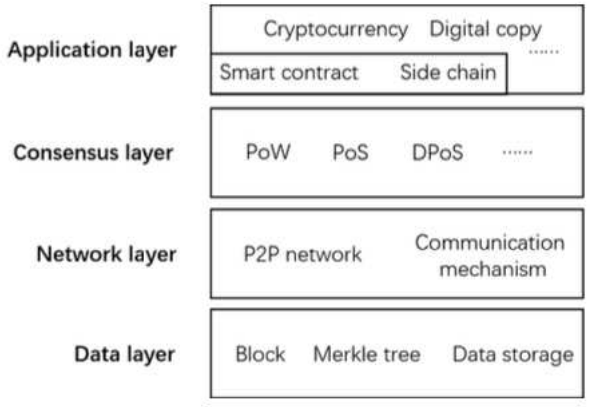

Figura 2.3. Arquitetura blockchain em quatro camadas. *Fonte* [Wu e outros 2019]

Na camada de dados estão a estrutura, organização e armazenamento de dados. Fatores como desempenho e acesso são cruciais para a rede Blockchain. Cada uma destas redes utiliza estruturas diferentes. De um modo geral, o banco de dados para armazenamento é o Google LevelDb. A rede Bitcoin usa a árvore de *Merkle* como forma de organizar e armazenar as informações, enquanto a *Ethereum* usa a *Merkle Patricia Tree*.

A Camada de rede da blockchain é autonomamente mantida e gerenciada por uma rede P2P composta por mineradores e usuários. É uma estrutura descentralizada e sem a necessidade de controle de entrada e saída, com tolerância a falhas A comunicação e a autenticação devem ser protegidas em caso de ataques.

A camada de consenso é fundamental em uma rede blockchain. Chegar a um consenso não é uma tarefa não trivial e muitos algoritmos de consenso foram propostos para atingir esse objetivo[Greve 2005]. Esses algoritmos ou mecanismos podem ser classificados em: PoW (do inglês *Proof-of-Work*), PoS (do inglês *Proof-os-Stake*) e suas variantes; BFT (do inglês *Bizantine Fault Tolerance*) e suas variantes.

Por fim, a camada de Aplicação estende a capacidade do blockchain e torna mais fácil para os desenvolvedores construir aplicativos blockchain através dos contratos inteligentes, explicados na sequência; das *sidechains* e do emprego de BaaS (do inglês *Blockchain as a Service*) [Samaniego e outros 2016], por exemplo. Nesta camada também estão as criptomoedas e uma série de aplicações incluindo *Fintechs*, seguros, pagamentos, governo, etc. Existe inclusive a possibilidade de ser combinada com inteligência artificial, big data, computação quântica, IoT etc.

2.1.3. Contratos Inteligentes

Os Contratos Inteligentes (CIs), definidos pela primeira vez por Nick Szabo [Szabo 1997], representam "um conjunto de promessas, especificado em formato digital, incluindo protocolos nos quais as partes cumprem estas promessas". Este conceito evoluiu, especialmente após a introdução de plataformas blockchain descentralizadas.

Com o surgimento da *Ethereum* [Buterin e outros. 2014], tivemos o arcabouço tecnológico capaz de implementar a definição de Szabo através de programas de computador imutáveis, que são executados de forma determinística, no contexto de uma Máquina Virtual Ethereum (EVM, do inglês *Ethereum Virtual Machine*), como parte do protocolo de rede *Ethereum*.

Tais contratos, então, são sistemas que movem ativos digitais automaticamente de acordo com regras pré-especificadas. A palavra **contrato** não tem significado legal neste contexto. Com a implantação generalizada da blockchain, o contrato inteligente recebeu grande atenção das empresas e academia.

Os CIs são imutáveis, por que uma vez implementado em uma rede *Ethereum*, o código não pode ser alterado e nem substituído. A única forma de se modificar o seu conteúdo é implementando um novo contrato, o qual terá um novo endereço.

Assim como os softwares, os contratos são determinísticos, pois o resultado de sua execução é sempre o mesmo para todos os que o executam, conservando-se o contexto no momento da execução. Os CIs estão em constante evolução e operam com um contexto muito limitado, por enquanto. No caso dos CIs para a rede *Ethereum*, existem diversas versões do compilador (*solc*), com mudanças significativas entre elas. De modo geral, os CIs acessam seu próprio estado, o contexto da transação que os chamou e algumas informações sobre os blocos mais recentes.

2.1.4. Blockchain e IoT

Blockchains para IoT são sistemas de blockchain personalizados e otimizados para aplicações de IoT. Estas aplicações são desenvolvidas em muitos campos. No entanto, a maioria desses aplicativos possui problemas como vazamento e confiabilidade de dados. Para mitigar esses efeitos problemáticos, a blockchain pode ser usada para fornecer maior segurança e estabilidade aos aplicativos IoT tradicionais.

A tecnologia Blockchain envolve muitos elementos além de simplesmente conectar blocos em uma cadeia. Ao adicionar elementos de IoT, esta complexidade ganha novos contornos que precisam ser desvendados. A arquitetura de aplicações para blockchain-IoT é composta de 5 camadas: Física, Rede, Blockchain, Middleware e Aplicação [Lao e outros 2020]. A Figura 2.4 ilustra estas camadas.

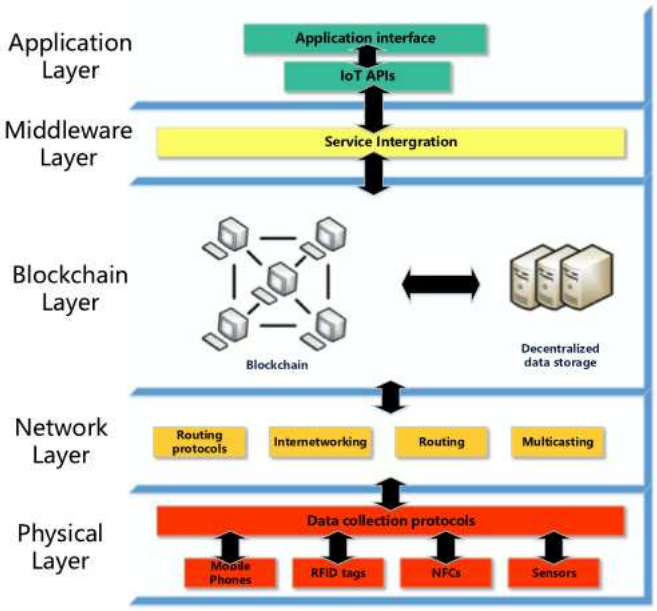

Figura 2.4. Arquitetura Blockchain-IoT em 5 camadas. Fonte [Lao e outros 2020]

A camada física da arquitetura blockchain-IoT é a mesma que a camada física da IoT [Lee e Lee 2015]. Inclui os sensores, atuadores, dispositivos inteligentes, etiquetas RFID, telefones celulares, câmeras de monitoramento e quaisquer outros dispositivos de IoT relacionados à aplicações dentro deste perfil. Os protocolos de coleta de dados empregados também são os mesmos usados em IoT.

A camada de rede é responsável por funções de roteamento, interconexão de redes e *multicasting* [Jiang e outros 2016]. Esta camada é muito similar à camada de rede tradicional da blockchain. A camada de blockchain é composta por funções de consenso, armazenamento de dados e compartilhamento de dados. Pode ser uma plataforma de blockchain específica ou pública[Nakamoto 2008, Ethereum 2014].

As ultimas duas camadas, middleware e aplicação, são responsáveis, respectivamente, por gerenciar a integração de IoT com blockchain, fornecendo serviços de segurança adicionais [Alphand e outros 2018] e fornecer abstrações por intermédio de API's e aplicações, de forma semelhante ao sistema IoT e às arquiteturas blockchain

tradicionais[Dorri e outros 2017].

2.1.5. Aplicações Distribuídas - DApps

Um aplicativo convencional, basicamente é composto de *front-end* e *back-end*. O primeiro elemento representa o uso de linguagens como HTML, CSS e JavaScript, dentre outras, para desenvolver uma interface gráfica a ser apresentada ao usuário. O segundo, refere-se ao desenvolvimento do lado do servidor. Ele é composto de bancos de dados, scripts, arquitetura de sites, lógica do negócio, etc. Esta parte da aplicação contém atividades ocorrem durante a execução de qualquer ação no *front-end*.

A Figura 2.5 ilustra uma possível arquitetura de um sistema convencional. Temos um fomulário apresentado ao usuário, que interage com o *Web Server*, e este, eventualmente com com outro servidor.

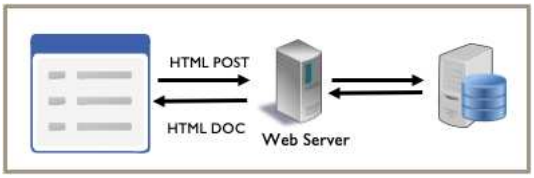

Figura 2.5. Arquitetura web.

Uma DApp é um aplicativo que é majoritariamente ou totalmente descentralizado. Em uma DApp, os contratos inteligentes são usados para armazenar a lógica de negócios (código do programa) e o estado relacionado de seu aplicativo.

A Figura 2.6 exemplifica uma transação em uma DApp que envolve dispositivos de IoT. Em (1), dados provenientes de dispositivos de IoT são enviados para um middleware. O middleware encaminha estes dados para um contrato inteligente hospedado na plataforma *Ethereum* (2), que após o processo de mineração, envia uma confirmação(3) de volta ao middleware. O usuário, tempos depois, através do *front-end* envia um pedido de informações sobre o sensor ao servidor web (4). Para atender a esta requisição, o servidor interage com a blockchain, solicitando ao respectivo contrato, através de uma função específica, o envio dos dados pedidos pelo cliente(5). Ao receber os dados, o servidor os envia para o *front-end* (6).

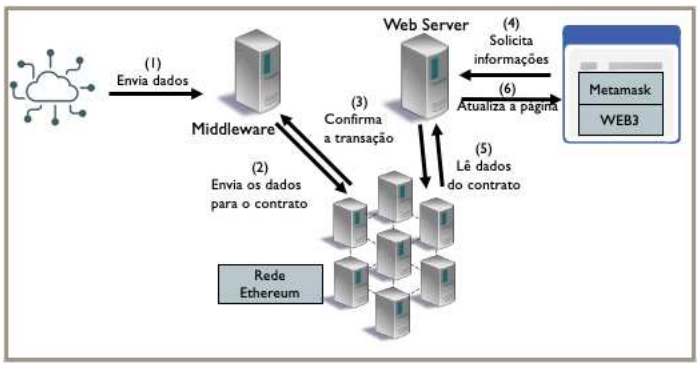

Figura 2.6. Arquitetura que exemplifica uma das formas de uma DApp trocar informações com a IoT.

Existem muitas vantagens na criação de um DApp que uma arquitetura centralizada típica não pode fornecer, entre elas a resiliência, a transparência e a resistência [Antonopoulos 2017].

Uma DApp é resiliente e não perde a sua conexão com a rede e seus clientes. Ela continua disponível enquanto a plataforma blockchain permanecer operando. Ao contrário de um aplicativo implantado em um servidor centralizado.

Como a cadeia de blocos é pública e imutável, as operações e o código da DApp podem ser inspecionados. Com isto temos um grau de confiabilidade sobre sua função e garante-se a transparência.

A rede blockchain dispensa um elemento certificador ou um nó centralizador, portanto, se um usuário consegue acesso a um nó da rede, ele acessa também a DApp, demonstrando a resitência.

Por fim, os dados do sensor ou informações sobre a lógica do negócio estão disponíveis na blockchain, independente do *Web Server*. Tais dados, herdam por conseguinte, as propriedades da blockchain descritas em 2.1.2.

2.2. Plataforma Ethereum e Contratos Inteligentes

Esta seção é divida em duas partes, A primeira descreve a plataforma *Ethereum* e a segunda os contratos inteligentes, preparando o leitor para a prática deste capítulo.

Serão abordados os componentes da *Ethereum*, como a EVM, os conceitos de *ether* e *gas*, as transações, os clientes *Ethereum* , as carteiras eletrônicas, as redes principal e de testes e outros detalhes.

Na segunda parte, apresenta-se os CIs e suas características, o ciclo de vida de um contrato, a linguagem Solidity, o processo de compilação e implementação de um contrato na plataforma *Ethereum* e um contrato inteligente com fins didáticos, exemplificando algumas particularidades inerentes à plataforma *Ethereum*. Estes conceitos são abordados conforme [Buterin e outros. 2014]

2.2.1. Plataforma Ethereum

O *Ethereum* é uma máquina de estado determinística acessível globalmente, composta por uma máquina virtual que aplica alterações a esse estado. Esta plataforma de blockchain transaciona ativos financeiros e é capaz de armazenar e executar códigos de computador em sua estrutura.

A plataforma *Ethereum* é composta por:

- Rede P2P: uma rede *peer-to-peer* capaz de suportar a rede *ethereum* principal e redes de teste. A rede principal é endereçada na porta TCP 30303 e executa o protocolo DEVp2p.
- Regras de Consenso: As regras para o consenso estão definidas no *yellow paper* [Buterin e outros. 2014].
- Transações: são mensagens capazes de transacionar ativos digitais, programas de

computador e informações. Estas transações possuem campos como remetente, destino, valor e área de dados.

- Máquina de estados:A EVM é uma máquina de estados executa os contratos inteligentes escritos em linguagens de alto nível, como o solidity. Os contratos são submetidos a um processo de compilação, resultando nos bytecodes, os quais são enviados para a rede e executados pela EVM.

- Estrutura de dados: os dados são armazenados em uma estrutura binária chamada *Merkle Patrícia Tree*. O banco de dados utilizado para isto é geralmente o LevelDB.

- Algoritmos de Consenso: A plataforma *Ethereum* utiliza o mesmo tipo de protocolo de consenso do Bitcoin. O protocolo PoW implementado na rede *Ethereum* é o *Etash*.A próxima geração, conhecida como Ethereum 2.0 utilizará o PoS, uma vez que o uso de PoW tem-se mostrado ineficiente sob o ponto de vista energético.

- Clientes: São implementações de software, interoperáveis. Estes clientes podem ser completos ou remotos.

Sob uma perspectiva mais prática, a *Ethereum* é uma infraestrutura de computação globalmente descentralizada e de código aberto que executa programas chamados contratos inteligentes. Ela usa a blockchain para sincronizar e armazenar as mudanças de estado do sistema, e incorpora a criptomoeda *ether* para medir e restringir os custos dos recursos de execução [Antonopoulos 2017].

2.2.2. EVM

A Máquina Virtual *Ethereum* é um dos principais componentes da desta plataforma uma vez que ela executa os contratos inteligentes e auxilia na manutenção do estado global da rede.

O termo máquina virtual é bastante conhecido na computação. Ele pode ser empregado para designar tecnologias que emulam um computador virtual através de reserva de recurso em uma máquina real. Também é conhecido como uma abstração que permite executar *bytecodes* gerados após o processo de compilação nas linguagens de alto nível como .Net e Java.

No âmbito da plataforma Ethereum, a EVM opera em um ambiente limitado, fornecendo um mecanismo computacional e executando seus próprios *bytecodes*. Pode-se afirmar ela implementa uma máquina de turing quase completa [Antonopoulos 2017].

A máquina de turing prevê que programas de computador podem ser executados em *loops* infinitos. Isto muitas vezes é útil e simples de ser implementado. No entanto, por descuido, ao executar pequisas ou correlações complexas, ou então de forma intencional ao implementando um ataque de Negação de Serviço (DoS, do inglês *Denial Of Service*), por exemplo, estes *loops* infinitos podem acontecer.

Isto, particularmente, na rede *Ethereum*, é um problemas de proporções sérias. Imagine um contrato inteligente, que ao ser executado em um nó, entre em *loop* infinito, consumindo recursos, energia, poder de processamento e monopolizando o nó durante um

longo tempo. A EVM não pode prever quando um programa vai ser encerrado, portanto, *a priori*, não é capaz de identificar tal situação. Este desperdício de recursos tem impacto global, uma vez que a blockchain possui o mesmo alcance.

Para evitar situações como esta, a plataforma *Ethereum* possui uma abstração monetária - o *gas*. À medida que o EVM executa um contrato inteligente, um algoritomo contabiliza as instruções (computação, acesso a dados, etc.). Cada instrução tem um custo predeterminado em unidades de *gas*.

Quando uma transação aciona a execução de um contrato inteligente, ela possui uma quantidade de *gas* que define o limite superior do que pode ser consumido ao ser executado o contrato inteligente. A EVM encerrará a execução se a quantidade de *gas* consumido pelo cálculo exceder o *gas* disponível na transação. O *gas* é o mecanismo que a *Ethereum* usa para permitir a implementação da máquina completa de *Turing*, enquanto limita os recursos que qualquer programa pode consumir.

2.2.3. Conceitos de *ether* e *gas*

O *ether* é a criptomoeda do *Ethereum*. Para adquirir *ethers* é necessário comprá-los com dólares em uma casas de câmbio virtual existentes. Os *ethers* possuem frações conforme ilustrado na Tabela 2.1.

Tabela 2.1. Divisão de unidades do ether

Valor (em wei)	Potencia	Nome
1	1	wei
1.000	10^3	babage ou Kwei
1.000.000	10^6	lovelace ou Mwei
1.000.000.000	10^9	shanon ou Gwei
1.000.000.000.000	10^{12}	szabo ou Microether
1.000.000.000.000.000	10^{15}	finney ou Miliether
1.000.000.000.000.000.000	10^{18}	ether
1.000.000.000.000.000.000.000	10^{21}	grand ou Kiloether

O *gas* é como se fosse o combustível do *Ethereum*. O *gas* não é *ether*, é uma moeda virtual separada com sua própria taxa de câmbio em relação ao *ether*. O *Ethereum* utiliza o *gas* separado do *ether* como forma de isolar a cotação da moeda *ether* no mundo real do valor das transações na rede pelos quais o *gas* paga (computação, memória e armazenamento).

Existem dois conceitos importantes relativos ao *gas*: O *gasLimit* e o *gasPrice*.

O *gasLimit* indica qual o limite de *gas* a ser consumido pela transação, ou seja, o número máximo de unidades de *gas* que o emissor da transação está disposto a comprar para concluir a transação. O *gasPrice*, em uma transação, permite que o emissor da transação defina o preço que está disposto a pagar para adquirir o *gas*. O preço é medido em *wei* por unidades de *gas*.

Por exemplo, uma transação simples, de transferência de *ether* de uma conta para outra, custa 21.000 unidades de *gas*. O valor a ser pago em *ether* é encontrado multiplicando-se 21.000 x *gasPrice*. As carteiras geralmente possuem um valor médio cobrado na rede para que uma transação seja confirmada dentro de um tempo aceitável

(no momento da escrita deste texto estava em torno de 12,3 *gwei*). Neste caso, uma transferência de *ether* custaria 21.000 x 12,3 = 266.700 *gwei*,o que equivale a 0.0002667 *ether*.

As carteiras podem ajustar o *gasPrice* nas transações originadas para obter uma confirmação mais rápida das transações. Quanto maior o *gasPrice*, mais rápido a transação provavelmente será confirmada. Por outro lado, as transações de baixa prioridade podem ter um preço reduzido, resultando em uma confirmação mais lenta. O valor mínimo que *gasPrice* pode ser definido é zero, o que significa uma transação sem taxas. Durante os períodos de demanda por espaço em um bloco, essas transações podem ser preteridas.

2.2.4. Clientes *Ethereum*

O acesso a plataforma *Ethereum* pode ser classificado sob dois aspectos: (a) Acesso para os desenvolvedores e (b) Acesso para os usuários. Em cada um deles há ferramentas e métodos diferentes.

Em (a), os desenvolvedores, comumente, usam a `web3`. Ela é uma coleção de bibliotecas que permitem a interação com um nó *Ethereum*, local ou remoto, usando HTTP, IPC ou *WebSocket* [Web3js 2016], com APIs(*Application Programming Interface*) disponíveis para Java, Javascript, .Net e outras linguagens de programação.

Em (b), os usuários conectam-se à rede *Ethereum* usando um cliente remoto (um aplicativo de software que implementa a especificação *Ethereum* e se comunica pela rede ponto a ponto com outros clientes *Ethereum*). Estes clientes remotos oferecem um subconjunto da funcionalidade de um cliente completo. Eles não armazenam a blockchain *Ethereum* completa, são mais rápidos de configurar e requerem menos armazenamento de dados.

Geralmente, os clientes remotos permitem: (1) Gerenciar chaves privadas e endereços *Ethereum* em uma carteira; (2) Criar, assinar e transmitir transações; (3) Interagir com contratos inteligentes, usando a carga útil de dados; (4) Navegar e interagir com DApps; (5) Oferecer links para serviços externos, como exploradores de blocos; (6) Converter unidades de *ether* e recuperar taxas de câmbio de fontes externas; (7) Injetar uma instância `web3` no navegador web como um objeto JavaScript; (8) Usar uma instância `web3` fornecida/injetada no navegador por outro cliente; e/ou (9) Acessar os serviços RPC em um nó *Ethereum* local ou remoto.

As carteiras móveis (*wallets*) são clientes remotos, já que os smartphones não têm recursos adequados para executar um cliente *Ethereum* completo. Os mais populares são *Jaxx* [Jaxx 2018], *Status* [Status 2019] , *Trust Wallet* [Trust 2019] e *Coinbase* [Coinbase 2018].

Os usuários podem também usar navegadores, onde as carteiras estão disponíveis como plugins ou extensões dos principais navegadores, como Chrome ou Firefox, por exemplo. Estes clientes remotos são executados no navegador. Os mais populares são *Metamask* [Metamask 2018], *Jaxx*, *MyEtherWallet* [Myetherwallet 2019], *Nifty* e MyCrypto [MyCrypto 2019].

O *Metamask* é um gateway para aplicações da plataforma *Ethereum*. Ele fornece acesso a todas as redes da plataforma com uma única conta, permitindo que se gerencie

as carteiras de todas as redes *Ethereum*. O *Metamask* pode ser instalado nos principais navegadores sob forma de extensão. Ao instalar, você receberá 12 palavras mnemônicas, e deve guardá-las sob o maior sigilo, pois são a única forma de recuperar a sua conta ou fazer transações usando a `web3`. Estas palavras devem ser informadas na ordem em que são apresentadas, na criação da conta, quando for necessário. O procedimento para instalação do *Metamask* pode ser acessado na página do curso [Abijaude e outros 2021].

O *Metamask* pode criar outras contas de acesso às redes *Ethereum*. Isto quer dizer que com as mesmas palavras mnemônicas e com a mesma instância instalada no navegador, o usuário pode ter vários endereços de contas. Isto é muito importante, principalmente quando formos usar o CI e a DApp, descritas no decorrer do texto.

2.2.5. Redes *Ethereum*

A plataforma Ethereum possui mais de uma rede disponível para os usuários. A Figura 2.7 ilustra isto. Na rede principal acontecem as transações reais, com impactos financeiros. Os *ethers* aqui precisam ser comprados com dólares.

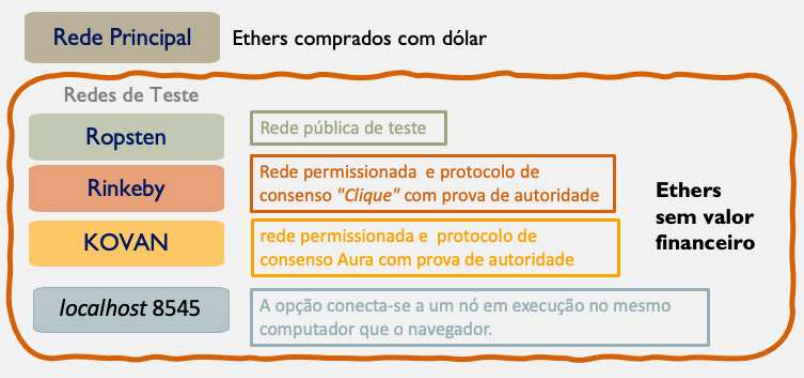

Figura 2.7. Exemplos de redes que compõem a plataforma Ethereum.

As redes de teste (Ropsten, Rinkeby, Kovan) trabalham com *ethers* que não possuem valor real e que podem ser adquiridos em geradores de *ethers* na Internet, sem custos financeiros. A opção *localhost* 8545 conecta-se a um nó em execução no mesmo computador que o navegador.

Além destas redes, há a opção via RPC que permite conexão a qualquer nó com uma interface de Chamada de Procedimento Remoto (RPC) compatível com *Geth*.

Este capítulo usa a rede de testes *Rinkeby* e o plugin *Metamask*. Para abastecer a carteira do *Metamask* com *ethers* sem valor comercial na rede *Rinkeby*, usa-se, por exemplo, o site `https://faucet.rinkeby.io/`. Ao criar uma conta no *Metamask*, automaticamente temos acesso a todas as redes *Ethereum*.

2.2.6. Transações na Blockchain Ethereum

As transações na rede *Ethereum* são originadas por um proprietário de uma conta e enviadas para execução pela EVM. A rede *Ethereum* não é autônoma, os contratos não podem sozinhos dispararem uma ação, eles precisam ser provocados por uma conta para que possam realizar uma tarefa. A Tabela 2.2 lista os campos que compõem as transações.

Basicamente há dois tipos de transação: A transação de criação de contrato e a transação convencional. A transação de criação de contrato, como o próprio nome sugere, adiciona um novo contrato na blockchain *Ethereum*. Isto é feito enviando a transação para um endereço especial conhecido como *zero address* `0x0`. Este endereço não representa uma conta e nem um contrato, ele é exclusivo a criação de novos contratos.

A transação convencional é aquela utilizada para transferência de *ethers* ou informações entre duas contas *Ethereum*, entre dois contratos inteligentes ou entre contas e contratos inteligentes. Estas transações não apontam para o endereço `0x0`. Elas apontam para endereços válidos que representam suas respectivas entidades.

A Tabela 2.2 lista os campos que compõem uma transação.

Tabela 2.2. Campos disponíveis na transação enviadas em uma rede *Ethereum*.

Campo	Descrição
nonce	Número de sequência da transação
gasPrice	Preço do *gas* em *wei* que o usuário está disposto a pagar
gasLimit	Limite de unidades de *gas* que o usuário aceita pagar pela transação
recipient	Endereço da conta *Ethereum* do cliente
value	Quantidade de *ether* que será enviada na transação
data	Campo que contém informações como chamada para funções do contrato ou *bytecodes*
v,r,s	Elementos criptográficos para a assinatura da transação

Entre estes, três campos merecem destaque: *nonce*, *data* e *value*. O primeiro é um número escalar que representa a quantidade de transações realizadas e confirmadas por uma conta. É um método que garante o ordenamento cronológico das transações e evita gastos duplicados.

Se uma fonte de dados encaminhar uma sequência de transações, elas serão processadas em ordem crescente do *nonce*. Uma situação hipotética que ilustra isto é um dispositivo de IoT que encaminha, por exemplo, 10 leituras de batimentos cardíacos para um contrato inteligente. Estas informações, ao serem recepcionadas pela rede *Ethereum* são direcionadas para uma piscina de transações e ao serem escolhidas para um bloco, a rede verifica o campo *nonce*. Se a última transação tiver o valor de *nonce* 6 e dois nós distintos da blockchain elegem transações com *nonce* 7 e 8 na mesma rodada, a transação de *nonce* 8 retornará para as piscina até que a transação de *nonce* 7 seja validada.

Por outro lado, se o *nonce* não existisse um sensor de batimentos cardíacos coletaria um valor e enviaria para um determinado contrato inteligente na blockchain. Minutos depois, na próxima coleta, o mesmo sensor poderia coletar e enviar o mesmo valor, já em outra transação. A rede teria então de processar duas transações idênticas. Caso isto fosse possível, seria possível então que qualquer sensor ou usuário criasse outras transações, com estes valores, enviando-as para o contrato inteligente com intuito malicioso. Com o *nonce* incremental implementado, fica impossível duplicar as transações.

Os campos *data* e *value* podem ser preenchidos ou enviados em branco alternadamente. Cada uma destas combinações tem consequências diferentes, conforme exibido na Tabela 2.3:

Quando envia-se para a blockchain *Ethereum* uma transação com os campos *value*

Tabela 2.3. Significado das transações de acordo como o preenchimento dos campos *data* e *value*.

value	*data*	**Significado**
Vazio	Vazio	Não faz ação nenhuma
Preenchido	Vazio	Operação de pagamento
Vazio	Preenchido	Chamado de função
Preenchido	Preenchido	Pagamento e chamado de função

e *data* vazios, como resultado tem-se apenas o gasto de *gas* e consequentemente de *ethers*. Se a transação contiver apenas o campo *value* preenchido, isto representa um pagamento, portanto, a transferência de *ethers* entre as contas envolvidas. Já uma transação que tenha apenas o campo *data* preenchido representa um chamado, uma invocação a uma função de um contrato, por exemplo. Por fim, caso os dois campos estejam preenchidos, tem-se então um pagamento e uma chamada de função em um contrato inteligente.

2.2.7. Contratos Inteligentes

Conforme definido na Seção 2.1.3, os contratos são programas de computador imutáveis e hospedados na blockchain. As linguagens de programação para a escrita de tais contratos são de alto nível, como *LLL*, *Serpent*, *Vyper*, *Bambu*, *Python* e *Solidity*. Esta última é a mais popular, suportada pela plataforma *Ethereum* e utilizada neste capítulo.

Evidentemente que este não é um capítulo de *Solidity*, pois esta é uma linguagem poderosa e em constante evolução. No entanto, serão apresentados pontos da linguagem, como tipos de variáveis, métodos e funções com o objetivo de fornecer uma base suficiente para que os alunos possam explorar sozinhos novos conhecimentos, entender os contratos apresentados e realizar a prática ao proposta. A Tabela 2.4 lista os principais tipos de dados da linguagem.

O *Solidity* também oferece alguns recursos adicionais que facilitam a construção dos contratos. Quando uma transação é criada e enviada para a rede, algumas informações são encapsuladas na mensagem de forma automática e estão prontas para auxiliar o desenvolvedor. Estas facilidades são agrupadas em 3 categorias: `msg, block e tx`.

`msg` - O objeto `msg` é uma chamada de transação originada de um cliente *Ethereum* ou de um contrato. Ela contém uma série de atributos úteis:

- `msg.sender`: Representa o endereço que iniciou a chamada de contrato
- `msg.value`: O valor de *ether* enviado com esta chamada (em *wei*).
- `msg.gas`: A quantidade de *gas* restante no suprimento de *gas* desse ambiente de execução. Isso foi descontinuado no *Solidity* v0.4.21 e substituído pela função `gasleft()`.
- `msg.data`: A carga útil de dados desta chamada no contrato.
- `msg.sig`: Os primeiros quatro bytes da carga de dados, que é o seletor de função.

`block` - O objeto de bloco contém informações sobre o bloco atual:

Tabela 2.4. Tabela com os tipos de dados disponíveis no *Solidity*

Tipo	Descrição
`int`	Inteiros positivos ou negativos (`int`) declarados em incrementos de 8 bits (`int8`, `int16`, ... `int256`).
`uint`	Inteiros positivos declarados em incrementos de 8 bits (`uint8`, `uint16`... `uint256`).
`bool`	Valor lógico, verdadeiro ou falso, com operadores lógicos ! (não), && (e), || (ou), == (igual) e! = (diferente).
`fixed/ ufixed`	Números de ponto fixo, declarados com `(u) fixedMxN` em que M é o tamanho em bits (incrementos de 8 até 256) e N é o número de decimais após o ponto (até 18); por exemplo, `ufixed32x2`.
`address`	Usado para armazenar endereços *Ethereum* de 20 bytes. O objeto de endereço tem muitas funções membro úteis, como `balance` (retorna o saldo da conta) e `transfer` (transfere *ether* para uma conta).
`byte array (fixed)`	Matrizes de bytes de tamanho fixo, declaradas com bytes.
`byte array (dynamic)`	Matrizes de bytes de tamanho variável, declaradas com bytes ou `string`.
`enum`	Tipo definido pelo usuário para enumerar valores discretos `enum name {rotulo1, rotulo2...}`.
`array`	Um `array` de qualquer tipo, fixo ou dinâmico.
`struc`	`Conteiners` de dados definidos pelo usuário para agrupar variáveis `struct Car { String year; int color;}`.
`Mapping`	Tabelas de pesquisa de *hash* para pares chave => `mapping (key_type=>value_type)`.

- `block.blockhash(blockNumber)`: O *hash* de um bloco específico. Em desuso e substituído pela função `blockhash()` no *Solidity* v0.4.22.
- `block.coinbase`: O endereço do destinatário das taxas do bloco atual e da recompensa do bloco.
- `block.difficulty`: A dificuldade (prova de trabalho) do bloco atual.
- `block.gaslimit`: A quantidade máxima de *gas* que pode ser gasta em todas as transações incluídas no bloco atual.
- `block.number`: O número do bloco atual.
- `block.timestamp`: O carimbo de data/hora colocado no bloco atual pelo minerador.

`tx` - O objeto *tx* fornece um meio de acessar informações relacionadas à transação:

- `tx.gasprice`: o preço do *gas* na transação de chamada.
- `tx.origin`: O endereço da conta *Ethereum* de origem para esta transação. Esta é uma operação considerada insegura!

A manipulação de dados relativos aos endereços de contas *Ethereum* passados como entrada possuem também alguns métodos e atributos que auxiliam a escrita dos contratos. Os principais estão listados abaixo:

- `address.balance`: O saldo do endereço, em *wei*. Por exemplo, o saldo do contrato atual é `address(this).balance`.
- `address.transfer(quantidade)`: Transfere o valor (em *wei*) para este endereço, lançando uma exceção para qualquer erro.
- `address.send(quantidade)`: Semelhante ao `transfer`. Ao invés de lançar uma exceção, ele retorna falso em caso de erro.
- `address.call(payload)`: pode construir uma chamada de mensagem arbitrária com uma carga de dados. Retorna falso em caso de erro. Mas o destinatário pode (acidentalmente ou maliciosamente) esgotar todo o seu *gas*, fazendo com que seu contrato seja interrompido com uma exceção.

Além disto, os usuários podem criar suas próprias funções na escrita dos contratos. Elas podem ser chamadas por uma transação originada em uma carteira *Ethereum* ou em outro contrato. A sintaxe usada para declarar estas funções é a seguinte:

`function FunctionName ([parâmetros]) public|private|internal|external [pure|constant|view|payable] [modifier] [return (tipos de retorno)]`, onde:

`FunctionName` é o nome usado para chamar a função em uma transação de uma carteira *Ethereum*, de outro contrato ou de dentro do mesmo contrato. Uma função pode ser definida sem um nome. Neste caso, é a função de *fallback*, que é chamada quando nenhuma outra função é nomeada. A função de *fallback* não pode ter argumentos ou retornos.

Os parâmetros vêm após o nome, especificado os argumentos que devem ser passados para a função, com seus nomes e tipos.

O próximo atributo especifica a visibilidade da função. O padrão são funções públicas que podem ser chamadas por outros contratos, transações de carteiras *Ethereum*, ou de dentro do contrato. As funções com atributo `external` são como funções públicas, exceto que não podem ser chamadas de dentro do contrato, a menos que explicitamente prefixadas com a palavra-chave this.

As funções com atributo `internal` são acessíveis apenas de dentro do contrato ou por contratos derivados de outro contrato. Elas não podem ser chamadas por outro contrato ou transações de carteiras *Ethereum*. As funções com o atributo `private` são como funções `internal`, mas não podem ser chamadas por contratos derivados.

Lembre-se de que os termos `internal` e `private` são um tanto enganosos. Qualquer função ou dado dentro de um contrato está sempre visível na blockchain pública, o que significa que qualquer pessoa pode ver o código ou os dados. As palavras-chave descritas aqui afetam apenas como e quando uma função pode ser chamada.

O segundo conjunto de palavras-chave (`pure, constant, view, payable`) afetam o comportamento da função:

Uma função `constant` ou `view` promete não modificar nenhum estado. Os termos possuem o mesmo objetivo e o primeiro será descontinuado em uma versão futura.

Uma função `pure` é aquela que não lê nem grava nenhuma variável no armazenamento. Ele só pode operar em argumentos e retornar dados, sem referência a nenhum dado armazenado.

Uma função `payable` é aquela que pode aceitar pagamentos recebidos. Funções não declaradas como `payable` rejeitarão pagamentos recebidos.

Existem 3 tipos especiais de funções que deve-se ficar atento: construtoras, auto-destruição e modificadoras.

As funções construtoras são executadas apenas uma vez, durante a criação do contrato e possuem a palavra-chave `constructor()`. As funções de auto-destruição possuem a palavra-chave `destroy()` e são utilizadas, como o próprio nome diz, para destruir o contrato implementado. As funções modificadoras são aplicadas adicionando-se o nome do `modifier` na declaração da função. São usados para criar condições que se aplicam a muitas situações em um contrato, e para isto, basta acrescentar o seu nome na declaração de uma função.

Após escritos, os contratos precisam ser compilados para depois serem implementados em uma rede *Ethereum*. Como resultado, o processo de compilação cria os *bytecodes* e a *Application Binary Interface* (ABI), conforme ilustrado na Figura 2.8.

Figura 2.8. O contrato, após compilado, gera duas saídas - Os *bytecodes* e a ABI.

Os *bytecodes*, de baixo nível, são implementados na plataforma *Ethereum* usando uma transação de criação de contrato enviada para um endereço especial de criação de contratos. Cada contrato, portanto, possui um endereço *Ethereum*, que é derivado da transação de criação do contrato em função da conta e do *nonce* de origem. Este endereço pode ser usado, em uma transação, para receber *ethers*, por exemplo, de uma outra conta contrato ou de uma conta *Ethereum* cliente.

A ABI possui informações de como acessar as funções do contrato. Somente através dela é que as DApps podem enviar *ethers* ou dados para o contrato. Esta interface, obrigatoriamente, precisa ser importada pela DApp.

É importante acrescentar que os contratos somente executam funções se forem chamados por uma transação. Os contratos nunca podem chamar a si próprios ou atuarem em *background*, mas podem chamar outros contratos em cadeia. As transações são atômicas, e caso a execução ocorra sem erros até o final, toda a transação é registrada.

As contas que representam CI possuem diferenças em relação às contas que representam apenas uma carteira eletrônica. Enquanto nestas, uma única conta pode acessar as

diversas redes Ethereum, nas contas de CI isto não é possível. Um conta que representa um CI só pode acessar a rede na qual ela foi implementada. Para que este contrato possa ser implementado em outra rede *Ethereum* é necessário implementar uma nova instância do contrato nesta nova rede, com outro pagamento das taxas da transação.

Os CIs podem ser gerados basicamente de duas formas. Usando editores online como *Studio Ethereum* [StudioEthereum 2019], *Ethfidlle* [Ethfiddle 2017] e o *Remix* [Remix 2015], ou através de qualquer editor de texto, após configurar adequadamente um ambiente local para desenvolvimento, o qual será discutido na Seção 2.5.

O *Remix* é um ambiente online configurado para programar, compilar e implementar CIs. Além de um editor de texto integrado, ele possui diversas versões de compiladores prontos para usar. Nele, existem 3 modos de se implementar os contratos: (a) Através de uma máquina virtual JavaScript, implementada no navegador; (b) usando a `web3` injetada pelo *Metamask*; ou (c) fornecendo um endereço para conexão de um provedor `web3`.

A Figura 2.9 ilustra o ambiente de desenvolvimento do *Remix*. O contrato em tela é um exemplo didático e pode ser encontrado na página do capítulo. Este contrato é compilado na versão 7.4 do `solc` (compilador do *solidity*). Na linha 1 é informado o tipo de licença para o contrato. Em seguida, na linha 2 informa-se qual a versão do `solc` será usada para compilar o contrato.

Figura 2.9. Exemplo didático de CI utilizando o editor on-line Remix.

Entre as linhas 4 e 18 está o CI propriamente dito. A linha 4 define o nome do contrato e a linha 5 declara a variável que será utilizada. As linhas 7 e 8 declaram o construtor do contrato, que será executado uma única vez, recebendo como parâmetro uma mensagem inicial.

Por fim, temos duas funções definidas no contrato. A função `setMessage`, na linha 11, quando invocada, recebe como parâmetro uma mensagem nova e atualiza a variável do contrato. A função `getMessage`, na linha 15, retorna o valor armazenado na variável `message`. Observe que, enquanto a primeira mensagem modifica o valor de uma variável do contrato, a segunda apenas lê o seu conteúdo.

Isto implica que a função `setMessage`, quando for invocada, vai gerar custos para executar a transação e será necessário desembolsar *ethers* da carteira para que a transação seja completada. Já a função `getMessage` não tem custo nenhum para ser executada.

A função `getMessage` está presente neste contrato apenas como exemplo didático. Todas as variáveis declaradas no contrato, automaticamente, terão uma função `get` associada no momento da compilação.

2.3. Desafios da Blockchain e IoT na área de Saúde

O setor de saúde tem particularidades associadas à segurança e privacidade devido aos requisitos legais para proteger as informações médicas dos pacientes. A Internet compartilha registros e dados armazenando-os na nuvem. Com a adoção de dispositivos móveis na saúde, o risco de ataques maliciosos e de informações privadas serem comprometidas à medida que são compartilhadas tornam-se, portanto, evidentes.

As informações ficam mais fáceis de serem obtidas, principalmente por meio do uso de componentes de IoT acoplados a pacientes ou a equipamentos médicos, portanto o compartilhamento e a privacidade destas informações são uma preocupação adicional.

É neste cenário que a blockchain e IoT precisam se engajar, proporcionando um ambiente adequado para celebrar esta união. Para explorar melhor estes conceitos, divide-se esta seção em três partes: Principais requisitos, Desafios técnicos e Protocolos de Consenso.

2.3.1. Principais Requisitos

A literatura classifica os dados de saúde em dois grupos básicos: Registros Pessoais de Saúde (PHR, do inglês Personal Health Record) e Registros Eletrônicos de Saúde (EHR, do inglês Electronic Health Records), que são controlados por hospitais, e não por pacientes. Estas informações possuem requisitos exclusivos da área de saúde que são evidenciados aqui. Segundo [McGhin e outros 2019], são: Controle de acesso, autenticação e não repudiação; Interoperabilidade; Compartilhamento de dados; e Mobilidade.

Controle de Acesso, Autenticação e não repudiação

Os dados médicos podem ser obtidos por intermédio de sensores corporais ou registros médicos com informações sobre o paciente. Estes registros, quando em formato digital, precisam garantir itens de segurança como integridade, não repudiação, confidencialidade e disponibilidade. Isto permite, por exemplo, que os pacientes armazenem e compartilhem com segurança seus EHRs em um servidor na nuvem para que médicos ou cuidadores acessem. Os médicos podem encaminhar o prontuário dos pacientes a outros especialistas para diagnósticos e pesquisas, sempre que necessário, garantindo que as informações dos pacientes permaneçam privadas [Yüksel e outros 2017, Au e outros 2017].

Interoperabilidade

Segundo [Azaria e outros 2016], o processo de compartilhamento e transferência de dados entre diferentes fontes é a definição para interoperabilidade. Entre as principais limitações que dificultam este processo está o emprego de armazenamento centralizado dos dados médicas em bancos de dados. Outras fontes que atuam neste sentido são diferenças de padrões entre sistemas e a legislação de proteção de dados entre diferentes nações.

Compartilhamento de dados

O PHR/EHR estão dispersos em clínica, hospitais e laboratórios. Isto impede que informações abrangentes e atualizadas sobre os pacientes possam ser compartilhadas [Roehrs e outros 2017]. Paralelamente, a ausência de um identificador comum entre as bases é outro fator que dificulta o compartilhamento dos dados.

Mobilidade

A questão da mobilidade está relacionada com aplicações de saúde móveis, dispositivos de IoT e redes sem fio. Existem muitas soluções que aplicam o conceito de aplicações móveis em saúde, usando massivamente as redes sem fio e os dispositivos de IoT, mas não possuem conhecimentos adequados para isto [Kotz e outros 2016]. Entre os principais requisitos estão a disponibilidade da rede, autenticação de dados, confiabilidade e localização. Os dispositivos inteligentes e sensores que registram e enviam dados vitais de saúde ao médico para visualização e avaliação remotas das condições, como por exemplo relógios inteligentes, lentes de contato, pulseiras de fitness, microchips sob a pele e sensores sem fio, às vezes, não se preocupam tanto com a segurança [Zhang e outros 2017].

2.3.2. Desafios Técnicos

A seguir, listamos alguns desafios técnicos da tecnologia blockchain, quando empregada na área de saúde [De Aguiar e outros 2020, Hoy 2017, Yli-Huumo e outros 2016, McGhin e outros 2019]:

Latência O processo para validar um bloco na plataforma Ethereum leva cerca de 15 segundos. Este tempo de espera pode ser prejudicial uma vez que os sistemas de saúde são dinâmicos e devem ser acessados o tempo todo. Uma alternativa seria o uso de blockchains permissionadas como a Hyperledger, cujo tempo necessário para geração de blocos é determinístico, com base na latência da rede, que deve operar na casa de milissegundos [Cachin e outros 2016].

Vazão Os sistemas de saúde, em alguns casos, necessitam de alto rendimento com um tempo de resposta muito curto, pois isto pode afetar negativamente um diagnóstico que e pode envolver vidas. Com o incremento do número de transações e o tempo de bloqueio para plataformas que usam algum tipo de prova, em especial o PoW descrito na Seção 2.3.3), a vazão pode ser um fator proibitivo para algumas aplicações.

Consumo de energia: Para blockchains que empregam o protocolo PoW, é preciso ponderar o alto consumo energético durante o processo de mineração dos blocos. A tendência é a adoção, por parte de algumas plataformas, de protocolos de consenso mais eficientes sob este ponto de vista ou de então de alternativas a protocolos que não usem provas para alcançar o consenso, como a plataforma IOTA [Silvano e Marcelino 2020, Popov e outros 2020].

Centralização: Alguns protocolos de consenso, apesar dos esforços e mecanismos para manter a justiça, tendem a centralizar os mineradores, e como resultado, isso reduz o nível de confiabilidade da rede. [Zheng e outros 2018, De Aguiar e outros 2020].

Privacidade: As blockchains públicas fornecem um certo grau de anonimidade e privacidade, mas todos os seus registros são públicos e auditáveis. Devido às leis e regulamentações de privacidade, os sistemas baseados em blockchain devem estar em conformidade com o Regulamento Geral de Proteção de Dados (GDPR). Uma possível alternativa seria o emprego de redes permissionadas, onde os dados armazenados na blockchain não são públicos.

2.3.3. Protocolos de Consenso

O consenso é um problema fundamental em computação distribuída e permite com que um conjunto de participantes (ou nós) numa rede chegue a um acordo sobre um conjunto de transações, ou sobre um determinado estado do sistema, apesar da ocorrência de falhas ou da presença de nós maliciosos, que podem subverter o sistema [Greve e outros 2018].

O consenso portanto mantém o estado consistente das réplicas e a disponibilidade do sistema. No contexto da saúde, o consenso da blockchain precisa ser bem elaborado para atender aos requisitos acima enumerados. Desta forma, as aplicações que envolvem dados de saúde e que contemplam dispositivos de IoT podem ser resolvidas com a introdução de um mecanismo de consenso distribuído adequado.

Esta seção apresenta os protocolos de consenso PoW, PoS, Variantes do PoW e PoS, BFT e Grafo Direcionado Acíclico (DAG, do inglês (*Directed Acyclic Graph*), ilustrados na figura 2.10. Onde for possível, correlaciona-se estes protocolos com aplicações na área de saúde [Al Omar e outros 2017, Ramachandran e outros 2020, Azaria e outros 2016, Patel 2019]. O termo PoX (*Proof of Somethings*) é uma forma de referir-se genericamente aos protocolos que necessitam de alguma prova para alcançar o consenso [Lao e outros 2020].

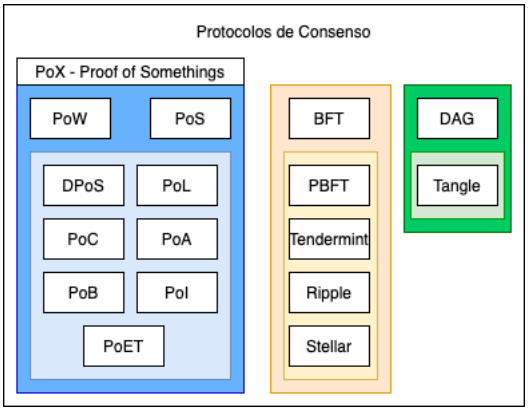

Figura 2.10. Diagrama com os pricipais protocolos de consenso.

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

O protocolo de consenso *Proof of Work* (PoW) surge com a blockchain do Bitcoin, no famoso artigo [Nakamoto 2008]. Desde então, diversas variações apareceram. No geral, o PoW adota a seguinte estratégia: cada nó da rede precisa resolver um desafio computacional para poder propor à rede um bloco de transações. Assim, através de um mecanismo de competição, em um processo exaustivo, o nó que resolver o quebra-cabeça matemático obterá uma recompensa na forma de criptomoeda, o bitcoin. Após a formação do bloco, o nó irá encaminhá-lo à rede, e todos os nós irão agregá-lo a uma "blockchain", estrutura de dados contendo toda a cadeia de blocos até então acordada pelos nós, de tal forma que o bloco recentemente transmitido aponta para o anterior.

Desta forma, observa-se dois princípios básicos que fazem o consenso PoW funcionar [Lao e outros 2020]:

(i) *A regra da cadeia mais longa*: o nó considera a cadeia mais longa como a cadeia certa. Isso porque, como mais de um nó pode resolver o puzzle ao mesmo tempo, mais de um bloco é transmitido na rede para estender a cadeia. Por princípio, os nós irão sempre estender a cadeia mais longa, e portanto, após um tempo, todos estarão com a mesma estrutura de blockchain, obtendo-se assim o acordo.

(ii) *A regra de incentivo*: um nó será recompensado ao encontrar um bloco adequado. Desta forma, os nós estarão motivados a despender recursos computacionais participando da competição (ou mineração de blocos).

Estas premissas são a base de funcionamento da rede Bitcoin. Elas garantem a exatidão e exclusividade da cadeia de blocos, evitando o duplo gasto e a manipulação da cadeia de blocos (ou livro razão) por um nó malicioso. Evidentemente, há outros desafios a serem tratados em um sistema complexo que movimenta ativos digitais tão valiosos. Para uma melhor base sobre esses elementos de segurança, recomendamos o livro [Narayanan e outros 2016].

O MedRec oferece aos pacientes registros imutáveis e de fácil acesso em locais de tratamento, gerenciando autenticação, confidencialidade, responsabilidade e compartilhamento de dados. Utiliza o fornecimento de dados agregados e anônimos como recompensas para pesquisadores, autoridades de saúde públicas, hospitais e clinicas que aceitem ser mineradores de uma rede baseada em PoW [Azaria e outros 2016].

O MedBChain é um sistema de gerenciamento de dados de saúde centrado no paciente usando Blockchain com base em PoW como armazenamento para obter privacidade. O pseudo anonimato é garantido pelo uso de funções criptográficas para proteger os dados do paciente [Al Omar e outros 2017].

A validade dos EHRs encapsulados na blockchain, com uso de PoW, empregando um esquema de assinatura baseada em atributo com várias autoridades, no qual um paciente endossa uma mensagem de acordo com um atributo, sem divulgar nenhuma informação, além da evidência que ele atestou. é descrito em [Guo e outros 2018].

A implementação de uma arquitetura de informação em larga escala para acessar (EHRs) com base em Contratos Inteligentes como mediadores de informação é explicado em [da Conceição e outros 2018], baseado em uma arquitetura de blockchain que também emprega o protocolo PoW.

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

Prova de Participação (ou Prova de Posse) - *Proof of Stake* (PoS) [Kiayias e outros 2017] é um dos algoritmos em ascensão para muitas aplicações de blockchain. Após anos de uso do PoW, algumas desvantagens ficaram evidentes, como segurança (ataques de duplo gasto possíveis), alto consumo energético e desperdício de recursos (no processo de competição/mineração) ou baixa vazão - *throughput* (pouca quantidade de transações acordadas no tempo).

De forma simplista, o PoS baseia-se na hipótese de que os usuários com a posse de mais moedas (ou recursos computacionais) são mais propensos a garantir a confiabilidade

do sistema e têm menos probabilidade de se comportar como nós maliciosos. Assim, o algoritmo PoS considera a porcentagem do número total de moedas (ou recursos) que um nó envolvido na competição detém, e eventualmente considera o tempo que o nó leva com o montante de moedas (ou recursos) para estabelecer uma porcentagem de direito de participação no consenso. Quanto mais moedas (ou recursos), mais probabilidade o nó terá de participar do consenso para decidir sobre os blocos.

A fim de permitir que cada bloco seja gerado mais rapidamente, o mecanismo PoS elimina o processo exaustivo de resolução do quebra-cabeça criptográfico. Mas, há problemas que também tonaram-se ou podem se tornar evidentes, como aconteceu com o PoW. Por exemplo, os usuários com mais moedas por um longo período têm maior possibilidade de serem selecionados pelo sistema para gerar o próximo bloco, ocasionando um elitismo e centralização das decisões.

As referência [Patel 2019] apresenta um *framework* para compartilhamento de imagens entre domínios que usa um blockchain, com base em PoS, como um armazenamento de dados distribuído, para estabelecer um livro-razão de estudos radiológicos e permissões de acesso definidas pelo paciente.

2.3.3.3. Variantes do PoW e PoS

Existe uma coletânea de protocolos que alcançam o consenso exigindo recursos computacionais de cada nó participante da rede, empregando mecanismos probabilísticos específicos e sem obrigação de informações completas sobre as operações do nó no sistema. A premissa é que há mais nós benignos, os quais podem ter mais recursos. Dentre estes protocolos, estão:

Prova de Participação Delegada - DPoS (*Delegated Proof of Stake*)

O DPoS [Larimer 2014] resolve o problema de centralização através da introdução do mecanismo de delegação. Os nós da rede elegem nós especiais, os super nós ou delegados, que passam a gerar e assinar os blocos. Com esta estratégia, o tempo de se confirmar uma transação melhora, pois o protocolo DPoS elimina a necessidade de se aguardar a confirmação de nós não confiáveis. Parece um paradoxo tentar centralizar decisões em um sistema descentralizado, no entanto qualquer nó pode ser alçado à condição de delegado. Quando um deles viola quaisquer regras do protocolo, seus direitos são negados e outro delegado será eleito.

Como exemplo de uso do DPoS temos as plataformas *BitShares* [1], *Steem* [2] e EoS [3], que apresentam 101, 21 e 21 delegados, respectivamente.

Comparado com o PoW, o DPoS é mais rápido e eficiente. Além disso, é mais democrático e flexível do que o PoS.

Prova de Sorte - PoL (*Proof of Luck*)

Este protocolo emprega funções TEE(*Trustworthy Execution Environment*) para

[1] https://wallet.bitshares.org/#/

[2] https://steem.com/

[3] https://eos.io/

fornecer justiça de mineração, segurança de tempo e certeza da identidade do nó. Surge como uma alternativa ao PoW, que exige cada vez mais poder de processamento e consumo de energia. Através da geração de um número aleatório executado em um TEE, estas funções bloqueiam as plataformas que podem ser usadas para processamento das operações como forma de limitar o poder desigual da computação. Após a geração do número, o PoL escolhe um líder para o consenso, e, com isto, obtém-se economia no consumo de energia, baixa latência para confirmação de transações e equidade na mineração [Milutinovic e outros 2016].

Prova de Capacidade ou Prova de Espaço - PoC (*Proof of Capacity* ou *Proof of Space*)

O PoC [Dziembowski e outros 2015] usa o espaço disponível no disco rígido para definir privilégios ao invés do poder computacional dos nós concorrentes. A probabilidade de propor um bloco é proporcional ao espaço de armazenamento cedido à rede por um nó minerador. Quanto maior a capacidade de armazenamento em disco, maior o domínio sobre o consenso.

Prova de Atividade - PoA (*Proof of Activity)*

Os algoritmos de PoA contam com um conjunto de N nós confiáveis chamados de autoridades. Cada autoridade é identificada por um único id e a maioria delas é considerada honesta, ou seja, pelo menos N / 2 + 1. As autoridades chegam a um consenso para ordenar as transações emitidas pelos clientes. O consenso em algoritmos de PoA depende de um esquema de rotação de mineração, uma abordagem amplamente usada para distribuir de forma justa a responsabilidade da criação de blocos entre as autoridades. O tempo é dividido em etapas, cada uma das quais tem uma autoridade eleita como líder de mineração [De Angelis e outros 2018].

As principais implementações de PoA são o *Clique* e o *Aura*. Ambas têm um primeiro turno onde o novo bloco é proposto pelo líder atual (proposta de bloco); então o *Aura* requer uma nova rodada (aceitação do bloco), enquanto o *Clique* não.

Prova de Queima - PoB (*Proof of Burn*)

Neste protocolo de consenso, os mineradores devem comprovar que queimaram algumas moedas, enviando-as para alguns endereços onde não podem ser gastos. A quantidade dessas moedas destruídas determina a probabilidade de um minerador emitir um novo bloco. O PoB funciona como uma espécie de mineração virtual, queimando moedas virtuais [Frankenfield 2018].

Prova de Importância - PoI (*Proof of Importance*)

Empregando o conceito de importância, este protocolo consegue medir a capacidade de uma conta de minerar um bloco. Para isto, a quantidade de moedas que possui e o número de transações realizadas são considerados. Há uma certa similaridade com o PoS, sob o ponto de vista do saldo em criptomoedas quando se observa o uso do saldo como critério decisivo para eleger nó, no entanto há de se observar que no PoI também se considera o volume de transação realizada. Para minerar um blocoos nós devem realizar transações ativamente. Ao aplicar PoI, a blockchain ganha vantagens de eficiência energética e alta taxa de transação [Bach e outros 2018].

Prova de Tempo Decorrido - PoET (*Proof of Elapsed Time*)

O algoritmo de consenso PoET [PoET 2018] faz com que os nós eleitos estocasticamente aguardem um tempo de espera aleatório criado pelo sistema. O nó que primeiro esgotar o tempo será eleito o líder para a criação do novo bloco. O PoET é um algoritmo semelhante a uma loteria que atende à justiça, ao investimento e à verificação. Para evitar trapaças, dois requisitos precisam ser verificados: O primeiro é que o líder realmente espera por um tempo aleatório em vez de um curto período de tempo para vencer. O segundo é que o líder realmente espera pelo tempo de espera determinado pelo protocolo.

2.3.3.4. Tolerância a Falhas Bizantinas - BFT(*Byzantine Fault Tolerance*)

Os protocolos baseados em BFT pertencem a uma classe que conseguem obter um acordo em um sistema, onde os processadores podem falhar de forma arbitrária, denominado Problema Geral Bizantino [Lamport e outros 1982]. A seguir, define-se os seguintes protocolos baseados em BFT: PBFT, *Tandermint*, *Ripple* e *Stellar*.

PBFT (*Practical Byzantine Fault Tolerance*)

O PBFT [Castro e outros 1999] foi o primeiro algoritmo prático a tolerar falhas bizantinas e adaptou-se para ser usado em ambientes assíncronos. O *BFT-Smart* é um outro projeto promissor em bom estágio de maturidade [Bessani e outros 2014]. O PBFT é oferecido pelo Hyperledger Fabric como camada de acordo (ordenação de transações). Além disso, o *BFT-Smart* [Sousa e outros 2018] também foi recentemente incorporado ao projeto [Greve e outros 2018].

A referencia [Dubovitskaya e outros 2017] propõe uma estrutura para gerenciar e compartilhar dados para atendimento a pacientes com câncer, apresentando um protótipo que garante privacidade, segurança, disponibilidade e controle de acesso refinado sobre os dados em uma blockchain com protocolo de consenso PBFT.

O AuditChain é um protótipo que aproveita a tecnologia de blockchain do Hyperleger Fabric para resolver problemas de interoperabilidade, conteúdo, estrutura e consolidação de log de auditoria. Especificamente, usa o livro razão e contratos inteligentes para padronizar o conteúdo, simplificar o acesso e garantir que os logs de auditoria contenham todas as informações necessárias e úteis [Anderson 2018].

O MedChain oferece uma solução de blockchain, com base no PBFT, e armazenamento distribuído para EMRs e informações de saúde protegidas através de uma arquitetura extensível [Sandgaard e Wishstar 2018].

O Medicalchain permite que o paciente forneça aos profissionais de saúde acesso aos registros e exames médicos de forma auditável, transparente e segura empregando tokens chamadas MedTokens [Albeyatti 2018].

Tendermint

O *Tendermint* [Kwon 2014] é um protocolo de consenso BFT quase assíncrono, baseado em validadores, propondo blocos de transações e votando neles. Ele requer apenas duas rodadas de votação para chegar a um consenso. Em cada rodada, há três etapas (ou seja, propor, prevenir, pré-comprometer). Quando mais de 2/3 dos votos pré-comprometidos forem recebidos para alcançar o consenso em uma rodada, o consenso

para a próxima rodada começará.

Ripple

O *Ripple Protocol Consensus Algorithm* (RPCA) [Todd 2015] utiliza sub-redes confiáveis coletivamente dentro da rede maior para chegar a um consenso para o Problema Geral Bizantino. No Ripple, a Unique Node List (UNL) é um conjunto de outros servidores mantidos por cada servidor, que desempenha um papel importante quando um servidor faz consultas para determinar o consenso. Apenas os votos dos servidores na UNL são considerados na determinação do consenso. Esta é uma diferença óbvia de muitos algoritmos de consenso. A UNL representa um subconjunto da rede que exige sabedoria coletiva para chegar a um consenso. A premissa da RPCA é que cada servidor confia nos outros servidores da UNL e acredita que eles não entrarão em conluio. A RPCA procede em várias rodadas para chegar a um consenso. Em cada rodada, cada servidor primeiro coleta o máximo de transações para se preparar para o consenso e torná-las públicas na forma de "conjunto de candidatos". Em seguida, cada servidor faz uma união dos conjuntos candidatos dos servidores em seu UNL e vota em cada transação. De acordo com o resultado da votação, as transações que obtiverem votos abaixo de um percentual mínimo serão descartadas ou colocadas em candidatos definidos no próximo consenso para o próximo bloco do livro-razão, enquanto aqueles que obtiverem votos suficientes irão para o próximo turno [Wu e outros 2019]

Stellar

O *Stellar Consensus Protocol* (SCP) [Mazieres 2015] é um protocolo do acordo bizantino federado (FBA). Ele é considerado o primeiro mecanismo de consenso comprovadamente seguro a desfrutar simultaneamente de quatro propriedades principais: controle descentralizado, baixa latência, confiança flexível e segurança assintótica. Segurança assintótica significa que a segurança do SCP depende de assinaturas digitais e famílias de *hash* cujos parâmetros podem ser ajustados de forma realista para proteger contra adversários com um poder de computação inimaginavelmente vasto.

2.3.3.5. Grafo Direcionado Acíclico - DAG (*Directed Acyclic Graph*)

Existe uma categoria de protocolos, a exemplo do Dagcoin [Lerner 2015] e do Tangle [Popov 2018], que apresentam uma estratégia visando explorar o paralelismo do sistema tradicional de blockchain de cadeia única, e empregam uma estrutura de dados de grafo direcionado acíclico para conectar blocos. O mecanismo de consenso, distinto dos demais abordados anteriormente, consiste em que cada transação fique vinculada aos dois registros de transações anteriores através do grafo. Desta forma, a conformidade da transação atual pode ser comprovada referenciando-se às transações anteriores. Se comparado com outros protocolos de consenso que estabelecem algum tipo de prova, observa-se que o DAG preocupa-se apenas com as transações vinculadas, constituindo um modo bem mais simples do que as diferentes provas a que se submetem os nós. O IOTA é uma plataforma de blockchain que através do Tangle utiliza este conceito.

2.3.3.6. Características de protocolos de consenso para IoT

Os dispositivos de IoT possuem limitações computacionais, energéticas e restrições de armazenamento de dados. Os protocolos de consenso precisam, além destas características, de um ambiente distribuído para garantir validade e consistência. Atingir este equilíbrio é o desafio para tornar este casamento duradouro e estável. A alta eficiência energética e um processo de consenso leve podem atenuar estes problemas.

Aplicações que envolvem blockchain-IoT precisam então driblar tais limitações e herdar os benefícios da blockchain. Se as restrições dos dispositivos de IoT forem premissas, nem os clientes leves nem os mineradores são indicados.

A blockchain IOTA é um exemplo de plataforma desenvolvida para IoT. Ela adota o consenso Tangle, baseado em DAG. Esta rede não possui mineradores, portanto o consumo energético é mais eficiente. Cada nó participante desta blockchain que necessita criar/enviar transações, primeiramente deve participar ativamente do processo de consenso aprovando duas transações anteriores.

A rede no grafo Tangle é composta por nós que são entidades que emitem e validam transações, e cada nó também representa uma transação [Popov e outros 2020]. Para que um nó adicione uma transação à rede, primeiro ele deve escolher duas transações para que possa aprová-las, conforme o algoritmo *Markov Chain Monte Carlo* (MCMC); Em seguida, o nó verifica se as transações escolhidas estão em conflito. Se isto ocorre, o nó deve desaprovar as transações conflitantes, e assim prevenir o gasto duplo; o próximo passo é resolver uma espécie de prova de trabalho (PoW), encontrando um valor *nonce*, tal que, seu *hash* seja concatenado com alguns dados da transação aprovado. É necessário destacar que este esforço computacional é uma versão muito mais leve do que a realizada pelos protocolos PoW tradicionais; Após conseguir este valor, o usuário envia sua transação para a rede, e ela se torna um *tip* (transação não aprovada); Por fim, o *tip* aguarda a confirmação por meio de aprovação direta ou indireta até que seu peso acumulado atinja o limite predefinido.

2.4. Pesquisas e Aplicações Recentes

O emprego de blockchain, contratos inteligentes e IoT oferecem novas possibilidades tecnológicas na saúde. A literatura apresenta muitas pesquisas e sistemas que exploram esta convergência. Esta seção apresenta dez classificações de pesquisas e aplicações recentes nesta área, exemplificando-as. Em seguida, apresenta-se a RNDS, uma das maiores redes de saúde com blockchain em operação no planeta.

2.4.1. Aplicações de blockchain na área de saúde

Segundo Ahmad [Ahmad e outros 2021], as aplicações e pesquisas em blockchain podem ser classificadas de acordo com a Figura 2.11. A seguir, comenta-se sobre cada uma delas.

Gerenciamento de dados clínicos do paciente

A eficácia do atendimento e do monitoramento da saúde de pacientes à distância depende da integridade e da manutenabilidade de seus registros digitais de acompanhamento médico, que incluem, entre outros: imagens, prescrições de medicação, análi-

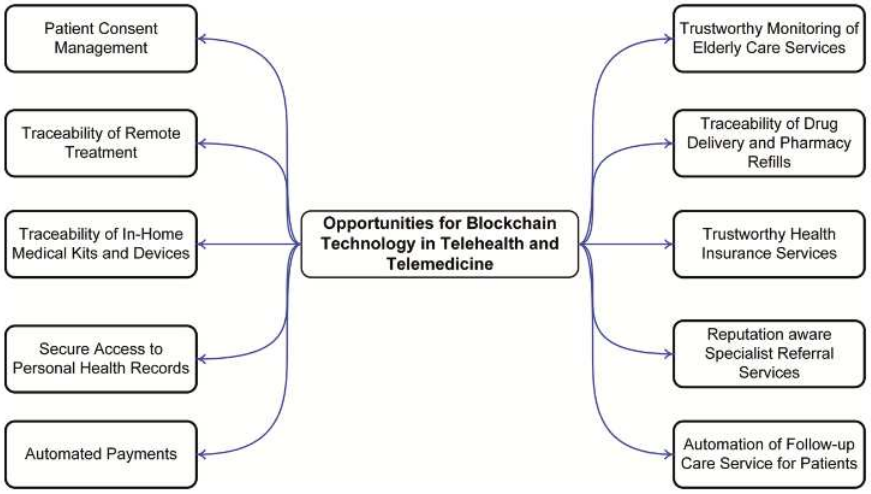

Figura 2.11. Diagrama com as principais oportunidades de aplicação da tecnlogia blockchain na área de saúde. Fonte: [Ahmad e outros 2021]

ses e históricos médicos, planos de tratamento e resultados. Esses registros são informações altamente confidenciais que precisam ser compartilhadas com segurança entre profissionais da área da saúde (médicos, farmacêuticos, hospitais, etc.) [Albeyatti 2018, Saweros e Song 2019]. Os sistemas que gerenciam tais registros enfrentam vários desafios, como por exemplo: na limitação em conduzir testes confiáveis de auditoria e na confiabilidade dos servidores (de terceiros) que armazenam tais dados. As características de imutabilidade, rastreabilidade e transparência oferecidas pela blockchain podem ajudar a conduzir testes de auditoria que verifiquem a conformidade dos dados clínicos dos pacientes. Igualmente, a tecnologia de blockchain pode ajudar a reforçar o nível de confiabilidade na manipulação e proteção de tais informações, uma vez que a terceira parte de confiança deste processo pode ser eliminada.

Rastreabilidade de tratamento remoto

O acompanhamento remoto de pacientes utiliza plataformas computacionais que permitem a troca de dados digitalizados que ajudam especialistas na área de saúde a diagnosticar o estado clínico de seus pacientes, auxiliando-os em suas consultas e procedimentos cirúrgicos [Mannaro e outros 2018, Hussien e outros 2021]. Nos sistemas de telemedicina existentes, clínicas, hospitais e especialistas não são capazes de compartilhar as bases de dados de seus pacientes. Para superar esse problemática, a tecnologia blockchain pode fornecer uma visão única e coerente das informações clínicas de pacientes em todos os nós participantes da cadeia de dados. A visibilidade e a transparência dos registros de saúde permitem que os participantes da blockchain rastreiem o histórico médico de cada paciente em análise, de forma a se produzir um tratamento adequado, com uma visão unificada dos especialistas envolvidos. A blockchain também pode permitir que auditorias sejam realizadas para descobrir quem acessou certos registros clínicos, e quais as transações foram realizadas nestas consultas.

Rastreabilidade de kits e dispositivos médicos residenciais

A adoção de kits e dispositivos de teste para uso domiciliar auxiliam no tratamento precoce de doenças, reduzindo os custos associados a exames médicos que antes

eram realizados apenas em laboratórios e hospitais [Weissman e outros 2018]. Tais dispositivos ajudam o paciente, em sua casa, a realizar auto-exames que podem o ajudar a precocemente detectar níveis indesejados de certas substâncias em seu sangue, como por exemplo, na avaliação do nível de glicose diretamente relacionado à variação de corrente elétrica resultante de reações eletroquímicas em dispositivos de sensoriamento digital (glicosímetros) [Sobreira 2005]. Entretanto, em sistemas de telessaúde centralizados tradicionais, a falta de transparência, visibilidade e proveniência de alguns fabricantes de tais kits, leva pacientes e médicos a adotarem apenas dispositivos de empresas farmacêuticas mundialmente renomadas. Esta problemática poderia ser atenuada pelo uso da blockchain, uma vez que esta tecnologia permite registrar, de forma imutável e transparente, as transações relacionadas à propriedade e ao desempenho de tais kits no livro-razão distribuído da cadeia. Contratos inteligentes poderiam ser usados para registrar pontuações de reputação para todos os dispositivos médicos residenciais, com base em suas avaliações de desempenho. Estas informações poderiam ajudar pacientes e médicos na escolha de dispositivos precisos e confiáveis (não necessariamente construídos por fabricantes renomados), segundo as informações presentes na blockchain.

Acesso seguro a registros pessoais de saúde

Em sistemas usados para oferecer serviços virtuais de armazenamento de informações de saúde, um paciente geralmente possui: um registro de acompanhamento médico, que contém informações clínicas mais detalhadas, devendo ser criado e gerenciado por laboratórios e/ou hospitais (anteriormente discutido), e um registro de saúde pessoal, que armazena seu histórico clínico, devendo ser criado e mantido pelo próprio paciente [of Health e outros 2008]. Tais sistemas são geralmente baseados em plataformas em nuvem, que são menos confiáveis devido ao fato de serem gerenciadas por uma única entidade certificadora. A natureza descentralizada da tecnologia blockchain permite que o proprietário dos dados médicos mantenha a privacidade destes. Contratos inteligentes podem registrar e autorizar usuários a acessar tais informações, em conformidade com a política de consentimento de seus proprietários [Shahnaz e outros 2019, Guo e outros 2019].

Pagamentos automatizados

Planos de saúde geralmente empregam serviços (centralizados) terceirizados para liquidar os pagamentos e dividendos de seus clientes (clínicas, médicos e pacientes). Entretanto, estes procedimentos normalmente são não-transparentes, relativamente lentos, vulneráveis a ataques cibernéticos, e geralmente caros, não sendo viáveis à pagamentos com baixos valores financeiros. A blockchain pode ser utilizada como plataforma para a realização de micro-pagamentos no setor da telessaúde, através da transferência direta de tokens de criptomoeda de forma rápida, segura, transparente, auditável e sem a necessidade de serviços de mediação central [Albeyatti 2018, Halamka e outros 2019]. Além disso, na blockchain as transações financeiras são assinadas digitalmente, o que garante às partes envolvidas a não-repudiação futura das transações realizadas. A tecnologia blockchain também pode ser usada para reduzir as chances de fraude em processos de "pagamento na entrega", como por exemplo, na operacionalização do serviço de entrega remota de medicamentos por tele-farmácias, onde contratos inteligentes podem ser programados para manter e transferir os tokens de criptomoeda para a carteira da farmácia apenas quando os produtos são recebidos com sucesso pelo paciente, em sua residência.

Monitoramento confiável de serviços de atendimento a idosos

Graças aos avanços tecnológicos proporcionados por arquiteturas computacionais embarcadas e pelo conceito de IoT, pacientes idosos podem permanecer em suas residências e serem monitorados remotamente por biossensores acoplados em seus corpos [Kazmi e outros 2019, Salah e outros 2020]. Tais dispositivos podem continuamente armazenar, processar e enviar dados de pacientes (temperatura corporal, indicadores de pressão arterial, etc.), ajudando profissionais da saúde a analisar e tomar decisões em relação aos seus estados clínicos. De forma a otimizar este serviço de atendimento domiciliar, a tecnologia blockchain e contratos inteligentes poderiam ser os responsáveis por proativamente notificar médicos e farmácias para a compra de medicamentos (quando necessário), e por disparar alertas a hospitais/centros de saúde (em situações de emergência).

Rastreabilidade da entrega de medicamentos

Os mecanismos de distribuição e renovação de receitas médicas entre profissionais da saúde, pacientes e farmácias podem ser realizados através do uso da blockchain auxiliada por contratos inteligentes. Farmacêuticos cadastrados podem acessar a prescrição de medicamentos armazenada na blockchain para verificar, preparar e enviar a encomenda aos respectivos pacientes. Contratos inteligentes podem periodicamente disparar a renovação de receitas e novos pedidos de envio de medicamentos às farmácias parceiras, segundo a especificação e os critérios pré-definidos na prescrição receitada pelo médico responsável pelo paciente. Em resposta, a farmácia pode autenticar e validar a atualização da receita, enviar os medicamentos e atualizar os registros de saúde do paciente com tais informações. Os medicamentos em trânsito podem ser rastreados pela farmácia e pelo paciente, que poderão acompanhar o trajeto realizado pela encomenda. Por fim, médicos e pacientes podem verificar a legitimidade de um medicamento por meio da análise da proveniência de seus dados [Thatcher e Acharya 2018].

Serviços de plano de saúde confiáveis

Devido a políticas rígidas de preservação da privacidade, pacientes não possuem o hábito de informar os dados de seus registros médicos às seguradoras de seus planos de saúde. Como consequência, podemos observar cotidianamente vários tipos de fraudes realizadas por pacientes mal-intencionados, que não informam as suas doenças pré-existentes, ou ainda, que realizam pedidos de indenização médica de exames e especialidades não previstas em suas apólices de seguro. A tecnologia blockchain pode ajudar os planos de saúde a minimizar tais fraudes através do acesso aos registros médicos digitais de seus clientes (com base em consentimento). Como incentivo a esta prática, várias seguradoras solicitam acesso a estas informações e oferecem em contrapartida tokens de criptomoeda a clientes que mantenham um seu estilo de vida saudável, através, por exemplo, de visitas a academias de ginástica (dispositivos inteligentes presentes na academia, ou conectados ao paciente, podem realizar transações na blockchain para a persistência de tais informações) [Raikwar e outros 2018, Albeyatti 2018].

Serviços especialistas de recomendação baseados em reputação

A telemedicina permite que dados remotos de pacientes possam ser acessados e analisados à distância por especialistas multidisciplinares [Lee 2019]. Em uma solução baseada em blockchain, o provedor de assistência médica pode armazenar os documentos

de referência dos pacientes em um servidor distribuído, que retorna o *hash* (dos registros) a serem armazenados na blockchain. Por meio do *hash* armazenado na cadeia, é possível se identificar se o documento armazenado no servidor distribuído foi alterado. O médico pode examinar o relatório de saúde do paciente para, em seguida, armazená-lo no livro-razão da blockchain.

Automação do serviço de acompanhamento de pacientes

O serviço de acompanhamento virtual permite que médicos monitorem remotamente a saúde de seus pacientes. Em certos casos, este serviço exige que o paciente compartilhe os relatórios de seus exames de sangue e de urina antes da realização da consulta virtual. A tecnologia blockchain pode automatizar este serviço por meio de contratos inteligentes que poderiam, por exemplo, disparar lembretes ao paciente para a submissão destes exames no sistema, ou ainda, para que ele não se esqueça do dia do encontro remoto com o seu médico [Siyal e outros 2019]. Além disso, ao usar servidores distribuídos que hospedam relatórios de exames clínicos, o paciente pode usar um contrato inteligente para registrar e compartilhar o *hash* de seu registro digital com o seu médico, que poderá então acessar os relatórios de saúde de seu paciente de uma forma transparente e segura.

2.4.2. Rede Nacional de Pesquisa

A Rede Nacional de Dados em Saúde (RNDS) é a plataforma nacional de interoperabilidade de dados em saúde. Um projeto estruturante do Conecte SUS, programa do Ministério da Saúde (MS) do Governo Federal, cujo objetivo é promover a troca de informações entre os pontos da Rede de Atenção à Saúde, permitindo a transição e continuidade do cuidado nos setores público e privado [da Saúde 2020].

Para que isto seja possível, existe integração das seguintes informações: resumo de atendimento; sumário de alta; imunização; medicamentos dispensados e exames realizados. Essa integração é possível devido à padronização de interoperabilidade, por meio de padrões médicos (SNOMED-CT, TISS, DICOM, LOINC, ISBT, 128, CID, CIAP-2, TUSS, CBHPM); padrões arquiteturais (FHIR - RES, REST, JSON); padrões de interoperabilidade (HL7 FHIR) e padrões dados dos pacientes (IHE PIX).

Isso promove a quebra dos silos de saúde, uma vez que os documentos clínicos federados estão disponíveis juntamente com a linha do tempo do paciente, a qual está distribuída por meio da tecnologia blockchain. Assim, torna-se possível trabalhar com os documentos clínicos em cada um dos estabelecimentos de origem, porém com uma visão de linha do tempo unificada para o cidadão. A ideia é ter informações otimizadas por meio do *Fast Healthcare Interoperability Resources* (FHIR), que é um padrão para troca de dados de saúde [Bender e Sartipi 2013].

Isto potencializa análises clínicas, integração de dados, atendimento otimizado para o cidadão (sumário inteligente do paciente e interoperabilidade nativa com a nuvem do MS), bem como o *e-patient*, por intermédio do qual o paciente gerencia a sua própria saúde. Em função disso, será possível falar de medicina preventiva com dispositivos inteligentes e sensores de IoT.

Atualmente, a RNDS é o maior case de blockchain em saúde do mundo com mais de 1 bilhão de transações e apoia-se em três pilares: confiabilidade; distribuição

e rastreabilidade. Totalmente enquadrada na Lei nº 13.709 - Lei Geral de Proteção a Dados (LGPD), utiliza blockchain para garantir segurança, privacidade e consentimento dos pacientes. A estratégia de Saúde Digital para o Brasil trata-se do uso de recursos de TIC para produzir e disponibilizar informações confiáveis, sobre o estado de saúde para quem precisa no momento que precisa.

2.5. Integrando sistemas Web, IoT e blockchain

Esta seção contempla a parte prática proposta neste capítulo. Os participantes podem fazer esta prática junto com os apresentadores ou fazer os tutoriais posteriormente acessando um material complementar em [Abijaude e outros 2021].

A prática proposta é criar uma DApp que ilustre o rastreamento do envio de vacinas entre a origem e o destino, coletando a temperatura durante o percurso. Este exemplo didático abrange o desenvolvimento de um contrato inteligente e de uma DApp que interagem com dispositivos de IoT e um middleware.

A Figura 2.12 ilustra a tela principal da DApp. Estes dados são enviados para o contrato no momento de sua criação através do uso de funções construtoras.

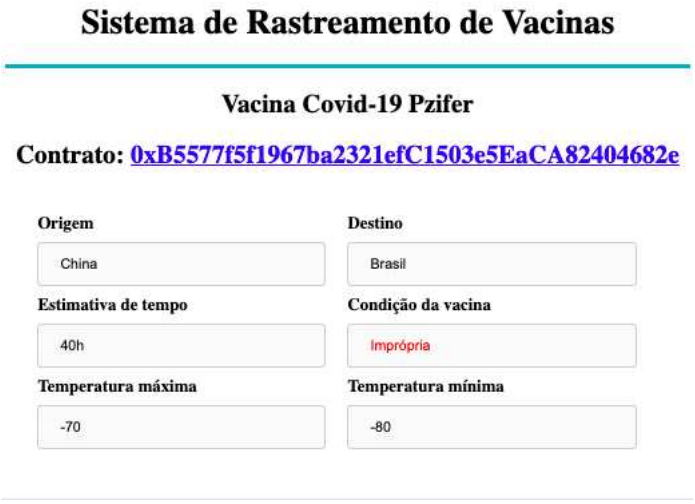

Figura 2.12. Tela da DApp que faz o rastreamento da vacina com base no contrato inteligente.

Os sensores captam o valor de temperatura durante o transporte da vacina e enviam os dados para um middleware, que então encaminha para um CI hospedado na plataforma Ethereum. Caso a temperatura esteja fora das especificações, o contrato vai considerar a vacina imprópria para consumo humano. Quando isto ocorrer, o contrato envia para a DApp uma informação e o registro do local, data, hora e temperatura coletados, conforme ilustrado na Figura 2.13.

Como se trata de um exemplo didático que deve ser replicado em outros ambientes, substituímos a geração dos dados dos sensores por mensagens REST geradas manualmente em programas como Insominia ou Postman. Estas mensagens serão enviadas para um endereço que emula um middleware capaz de recepcionar os dados e enviá-los diretamente para a blockchain.

Figura 2.13. Informação enviada a DApp sobre quando e onde ocorreu uma violação

No entanto, considerando uma melhor técnica para explicar o fluxo de informações, o middleware vai encaminhar as mensagens temporariamente para a aplicação, onde o usuário poderá verificar estes dados e então liberar o envio para o contrato inteligente, conforme mostrado na Figura 2.14. Ressalta-se que esta técnica é meramente didática, pois em um sistema real, o envio dos dados coletados pelos sensores são enviados para o middleware e deste diretamente para os contratos.

Figura 2.14. Informação enviada a DApp sobre quando e onde ocorreu uma violação

Todos os detalhes da construção da aplicação, do contrato inteligente, do middleware, do hardware e dos detalhes de compilação e implementação dos contratos estão na página web com o material complementar.

2.5.1. Ambiente de desenvolvimento

Ainda não há um Ambiente de Desenvolvimento Integrado (IDE, do inglês *Integrated Development Environment*) ou um ambiente amigável para o desenvolvimento dos contratos e de DApps. Pode-se usar editores on-line, como por exemplo o Remix ou preparar um ambiente de desenvolvimento local.

O uso de um ambiente local para desenvolvimento permite mais liberdade ao desenvolvedor. Esta solução é a utilizada pelos autores e já testada em cursos na Universidade Estadual de Santa Cruz (UESC), Universidade Estadual do Sudoeste da Bahia (UESB) e Universidade Federal da Bahia (UFBA).

Este ambiente local é composto por um editor de código de sua preferencia, o Node.js e os seguintes pacotes adicionais: a) `solc`: compilador *Solidity*; b) `mocha`: *framework* para testar os contratos antes de implementá-los em uma rede blockchain; c) `web3`: coleção de bibliotecas que permite interagir com um nó *Ethereum* local ou remoto usando HTTP; d) `ganache-cli`: é uma blockchain pessoal para desenvolvimento rápido de aplicativos distribuídos *Ethereum* e *Corda* em um ambiente seguro e determinístico; e) `truffle-hdwallet-provider`: para realizar as assinaturas usando as palavras mnemônicas. Estas palavras são informadas ao usuário no momento da instalação do metamask e devem ser guardadas, pois através delas conseguiremos autorizar as transações.

Os projetos que estão disponíveis para a prática estão com todas as dependências configuradas e com um tutorial que permite instalar todas elas automaticamente.

Ao realizar o download e descompactar o arquivo, será criado um diretório chamado `Vacina`, com as subpastas `api`, `deploy` e `frontend`. Estas três pastas representam os projetos que serão detalhados adiante.

Antes porém, o usuário deverá abrir um terminal, ir até a cada uma destas pastas e e executar o comando `npm install` para baixar as dependências e configurar o ambiente do projeto. Isto deverá ser feito individualmente em cada uma das pastas.

A pasta `deploy` contém o projeto relacionado à escrita, compilação e implementação dos CIs. Dentro dela está a pasta `contracts` com os arquivos `Vacina.sol` e `vacina.abi.json`. O primeiro, mostrado na Figura 2.15 é o contrato inteligente escrito em Solidity. O segundo é a ABI já recuperada, e que será utilizada no projeto do *front-end*. A seguir, explica-se a lógica do contrato.

As primeiras linhas do contrato, que não aparecem na figura, declaram as variáveis e definem a função construtora, que recebe como parâmetros nome da vacina, local de origem, local de destino, duração prevista para o transporte e temperaturas máxima e mínima. A parte do código entre as linhas 40 e 62 representam a lógica do negócio. Na linha 41 está a função *insertRegistro()* que, ao ser invocada pela DApp, recebe os parâmetros digitados na tela da DApp e executa duas verificações: a primeira é para garantir que o valor da temperatura está dentro do intervalo esperada; a segunda, verifica se em algum registro adicionado anteriormente, este intervalo foi já violado. Caso estas condições tenham sido violadas, registra-se então os valores que tornaram a vacina imprópria por ter sido armazenada fora dos padrões exigidos. Em seguida os dados são armazenados em uma matriz de endereços.

A função `getRegistro()`, na linha 51, retorna todas as etapas da viagem enviando a matriz `registros`. A função `close()`, na linha 55, destrói o contrato, desde que seja invocada pelo endereço que o implementou. Observe que na declaração desta função, ela recebe um indicativo que terá de consultar a função verificadora antes de sua execução, através da palavra `verificaOwner()`. Isto desvia o fluxo de execução do


```
40      // Cadastra uma etapa do transporte
41      function insertRegistro(int16 _value, uint64 _time, string memory _local)
42          // Verifica se a temperatura esta entre a maxima e a mininma entre
43          // se nao teve perda antes dessa inserção então execute o if
44          if((_value > tempMax || _value < tempMin) && perdaVacina.value == 0){
45              perdaVacina = Registro(_value, _time, _local);
46              condicao = false;
47          }
48          registros.push(Registro(_value, _time, _local));
49      }
50      // Retorna etapas do transporte
51      function getRegistro() public view returns(Registro[] memory _registro) {
52          return registros;
53      }
54      // Destroi contrato
55      function close() public verificaOwner {
56          address payable addr = payable(msg.sender);
57          selfdestruct(addr);
58      }
59      modifier verificaOwner(){
60          require(msg.sender == owner);
61          _;
62      }
```

Figura 2.15. Parte do contrato inteligente `Vacina.sol`.

contrato para a linha 59, executando a função modificadora até encontrar o sinal de "_", quando então retorna para execução da função `close()`, a partir da linha 56.

Ainda na pasta do projeto `deploy`, existem três arquivos importantes localizados na raiz da pasta. São eles o `.env`, e os *scripts* `compile.js` e `deploy.js`. O primeiro é um arquivo de configuração que possui apenas duas linhas. A primeira linha deve conter as palavras mnemônicas e a segunda linha o endereço do nó que permite a conexão com a blockchain. Este endereço adquire-se ao acessar o site `www.infura.io`, seguindo o tutorial que está na página web complementar.

O segundo arquivo, `compile.js`, é responsável pela compilação do projeto, e consequentemente pela produção da ABI e dos *bytecodes*. Este *script* pode ser reaproveitado para outros contratos, bastando apenas atualizar as variáveis `contractName` e `contractFileName`.

O terceiro arquivo é o *script deploy.js*. Ele invoca o *script* de compilação e manipula os resultados de modo a enviar para a o endereço especificado no `.env` os bytecodes, e assim consequentemente, implementar o contrato na rede blockchain. Para executá-lo, digite no terminal o comando `node deploy.js` e aguarde. No término de sua execução, o terminal recebe como resultado a ABI gerada pelo compilador, que neste caso, foi copiada para o arquivo `vacina.abi.json`. O endereço da conta utilizada para implementação e o endereço que identifica o contrato na rede blockchain. Guarde o endereço do contrato, pois vocÊ irá precisar dele para informar à DApp.

Este *script* também pode ser utilizado por outros contratos. Para tanto deve ser ajustada a variável `contract`. Esta variável configura uma parte da transação a ser enviada, e, em particular, neste caso, envia os parâmetros esperados pela função construtora.

A pasta `frontend` possui a DApp que será executada. Na pasta `src` há a pasta `contracts` e os arquivos `index.js` e `App.js`.

A pasta `frontend/src/contracts` possui dois arquivos. O arquivo `web3.js` configura a `web3` para ser usado pela DApp. O arquivo `vacina.contracts.js` possui as variáveis `address`, cujo valor deve ser substituído pelo endereço do contrato implementado quando você executou o *script* `deploy.js` e a variável `abi` cujo valor deve ser a ABI, também impressa na tela no processo de implementação. Para facilitar, o arquivo `/deploy/contracts/vacina.abi.js` já possui este conteúdo.

O arquivo `index.js` é o arquivo padrão que será lido pelo servidor web. Este arquivo importa o `App.js` que contém de fato a DApp.

Este arquivo é divido basicamente em duas partes: Uma que declara variáveis e funções e outra que prepara a tela a ser exibida para o usuário. A Figura 2.16 ilustra, entre as linhas 92 e 98, invocações às funções do contrato. Para que isto ocorra, a DApp utiliza as informações na ABI, fornecida ao projeto pelo arquivo `vacina.contracts.js`, para enviar uma solicitação à rede blockchain. Cada uma dessas funções descritas no contrato tem a finalidade de retornar um valor.

```
89  const pegaInfoContrato = async () => {
90    try {
91      // Pega informações do contrato
92      const _nome = await vacina.methods.nome().call();
93      const _origem = await vacina.methods.origem().call();
94      const _destino = await vacina.methods.destino().call();
95      const _duracao = await vacina.methods.duracao().call();
96      const _tempMax = await vacina.methods.tempMax().call();
97      const _tempMin = await vacina.methods.tempMin().call();
98      const _condicao = await vacina.methods.condicao().call();
```

Figura 2.16. Parte do arquivo `App.js` que recupera dados do contrato.

A Figura 2.17, entre as linhas 147 e 152, mostra a variável `responseTrx` que recebe o resultado do envio de dados para o CI. Para que isto seja possível, novamente a ABI é consultada pela aplicação para saber como encaminhar para a função `insertRegistro()` os valores da temperatura (`_value`), data/hora (`_time`) e local (`_local`). As taxas deste envio serão debitadas no endereço representado pela variável `contas[0]`.

```
147     const responseTrx = await vacina.methods
148       .insertRegistro(_value, _time, _local)
149       .send({
150         // Diz a carteira que está enviando os dados
151         from: contas[0],
152       });
```

Figura 2.17. Parte do arquivo `App.js` que recupera dados do contrato.

Para deixar o servidor web com o *front-end* ativo, digite na pasta do projeto `npm start` e aguarde. O navegador padrão será aberto e a tela da DApp exibida.

A pasta `api` contém os arquivos para emular um middleware, na porta TCP 3333, para recepcionar os dados enviados por um sensor ou por um gerador de mensagens HTTP e reencaminhá-los para a aplicação. O endereço utilizado para enviar as mensagens é `localhost:3333/insert-data`. Estas mensagens possuem o formato JSON `{"value":-75,"local":"Brasil"}`. O terceiro parâmetro, referente a data/hora é recuperado automaticamente pelo sistema. Para ativar este serviço, vá até o terminal e digite na pasta do projeto o comando `npm start`. Para enviar mensagens use na linha de comando o aplicativo `curl` ou um programa como o Postman ou Insominia. Nós recomendamos fortemente que seja utilizado um destes dois programas, em especial o insomnia, cujo tutorial está publicado na página web deste capítulo.

Ao acessar a página do curso, teremos um tutorial completo para a execução da prática. Esta prática pode ser feita de duas formas: usando um dispositivo de IoT com interface de rede ou através de um programa que envie mensagens como o Postman ou Insomnia. Todos os códigos estão comentados e explicados de forma bem didática.

2.6. Como montar um curso de Blockchain e IoT aplicado à saúde

A programação de aplicações para blockchain e IoT não é uma tarefa trivial. Envolve conceitos, propriedades e conhecimentos que vão além da blockchain, dos contratos inteligentes e da linguagem de programação *Solidity* .

Existe uma escassez de material teórico e prático para o ensino de tecnologias emergentes como blockchain, CIs e desenvolvimento de DApps. Uma alternativa para isto são as plataformas MOOC (*Massive Open Online Course*), como Udemy, Coursera, edX. Algumas Universidades promovem cursos de extensão ou treinamentos para seus alunos [Rao e Dave 2019, Dettling 2018, Araujo e outros 2019].

Os autores deste capítulo elaboraram e ministraram um curso em três universidades na Bahia: A Universidade Estadual de Santa Cruz (UESC), a Universidade Estadual do Sudoeste da Bahia (UESB) e a Universidade Federal da Bahia (UFBA).

Esta seção objetiva esclarecer e auxiliar a criação de cursos de extensão ou disciplinas de graduação/pós-graduação que pretendem explorar este tema através de três subseções: Infraestrutura, conteúdo programático e sugestão de práticas.

2.6.1. Infraestrutura

O hardware empregado para o desenvolvimento dos laboratórios é bastante simples, uma vez que não há plataformas que necessitem de muitos recursos computacionais. Nos 3 treinamentos ministrados havia computadores equipados com processadores que vão desde o Core 2 Duo com 4 Gb de RAM, até o Core i7 com 32 Gb de RAM. O espaço em disco também não é um fator limitante, uma vez que a maioria das máquinas possui espaço de armazenamento suficiente para os experimentos realizados.

O conjunto de softwares necessários à realização de todas as atividades práticas do treinamento é composto por navegadores, extensões para navegadores, ferramentas, pacotes e aplicativos hospedados em sites. Os softwares são compatíveis com praticamente todos os sistemas operacionais, como por exemplo Windows, Linux, Unix e MacOs.

2.6.2. Conteúdo a ser abordado

Sugere-se três módulos para serem ministrados aos alunos. O primeiro, básico, aborda conceitos de BC, CIs e DApps, criando um contrato simples e uma DApp. O curso básico pode ser ministrado com carga horária de 16h. Seriam 2 dias de aula, com 4 horas no período da manhã e 4 horas no período da tarde. No primeiro dia, durante o período da manhã foi abordada toda a parte teórica e conceitual da blockchain com ênfase na plataforma Ethereum, além da apresentação do editor de contratos on-line *Remix*. Durante o período da tarde, configuramos as máquinas locais e realiza-se rotinas de testes.

O segundo, explora mais profundamente as funções da linguagem *Solidity*, construindo um contrato bem mais complexo, utilizando técnicas de engenharia de software, e criando uma DApp multi página. Neste nível intermediário, durante o período da manhã apresenta-se mais um pouco de teoria com foco na interação com as redes *Ethereum*. Aprende-se mais um pouco sobre a linguagem de programação *Solidity* e escreve-se, compila-se, testa-se e implementa-se um contrato na rede *Rinkeby*. No período da tarde, desenvolve-se uma DApp que interage com o contrato implementado.

O terceiro módulo serviu de base para a criação deste capítulo e prevê a integração com a Internet das Coisas, coletando dados dos sensores, armazenando-os em CIs e disparando ações quando determinadas situações forem alcançadas.

É possível encontrar mais recursos na internet, como por exemplo:

a) *CryptoZombies*: uma plataforma online onde o intuito é ensinar sobre contratos inteligentes de forma interativa. O usuário desenvolve um jogo com foco em zumbis onde a logística é administrada por um contrato e com interação visual através de html, css e javascript. Disponível em `https://cryptozombies.io/pt/`.

b) *Ethernaut*: uma plataforma online que apresenta diversos tutoriais voltados para jogos. Ela foca puramente na criação de contratos, sem implementação visual. Alguns exemplos possibilitam a interação através da ferramenta do desenvolvedor do navegador. Disponível em `https://ethernaut.openzeppelin.com/`.

c) *Vyper Tutorials*: semelhante à ideia do CryptoZombies, esta plataforma propõe a criação de um jogo de *pokémon*. Atualmente está em fase de desenvolvimento, mas já é possível aprender como funciona a criação de contratos. Futuramente serão adicionadas interações através de interface assim como *CryptoZombies*. Disponível em `https://vyper.fun/#/`.

d) *Ethereum Studio*: uma ferramenta para desenvolvedores que desejam aprender sobre como construir aplicações na rede *Ethereum*. Os modelos ensinam como escrever um contrato inteligente, implementá-lo e interagir com os CIs por meio de um aplicativo baseado na web. Disponível em `https://studio.ethereum.org/`.

2.6.3. Práticas Propostas

O primeiro conjunto de práticas propostas teria a finalidade de apresentar o Remix e desenvolvimento do primeiro contrato. Sugere-se um exemplo didático que sirva para o aluno se familiarizar com o ambiente e começar a entender os conceitos de compilação, implementação, rede de testes, etc.

A segunda prática, recomenda-se que seja a montagem de ambiente local de desenvolvimento, ressaltando as vantagens e desvantagens desta abordagem. Nessa prática, o aluno vai lidar com scripts de compilação e implementação, compreender conceitos de ABI, bytecodes, instalar uma carteira Ethereum e abastecê-la com ethers sem valor comercial.

O conceitos de testes de contratos e seus benefícios devem ser introduzidos como o terceiro momento da prática. É importante deixar claro que os contratos são imutáveis, e uma vez implementados não é possível modificá-los e nem mesmo apagá-los. Aqui o aluno aprende a configurar o ambiente de testes, montar uma rede blockchain local em sua máquina e executar testes básicos.

Na sequência, as práticas evoluem para contratos mais sofisticados, que representam exercícios mais elaborados e envolvem a transferência de moedas entre as contas. Novos testes também são propostos aqui e, ao final, a integração com um sistema Web simples, com apenas uma página, criando a primeira DApp.

Após a criação deste contrato, sugere-se criar contratos mais complicados, que envolvem conceitos de engenharia de software, como padrão *fabric*. Questões de quem vai pagar por eventuais operações nos contratos e recursos mais avançados da linguagem solidity serão tratados nesta prática. Depois de novas rotinas de teste, constrói-se uma DApp multipágina, que representa um sistema mais elaborado e com grau de dificuldade maior.

A última e mais desafiadora prática e construir um pequeno sistema que seja capaz de enviar dados de dispositivos IoT para um contrato e exibí-los em um sistema, empregando, por exemplo o estilo arquitetural REST. Caso não seja possível ter os dispositivos de IoT, pode-se optar pela geração de dados que simule tais equipamentos.

Como mecanismo de avaliação sugere-se que os alunos pesquisem e elaborem um projeto que contemple os conhecimentos adquiridos, para em seguida, apresentá-lo a todos da sala, detalhando as atividades realizadas.

2.7. Desafios, Perspectivas e Conclusão

Este capítulo abordou a IoT, blockchain e contratos inteligentes aplicados à saúde. Após uma sessão de nivelamento, onde conceitos sobre estes temas e sobre o desenvolvimento de DApps foram tratados, os autores se aprofundaram na plataforma ethereum e nos contratos inteligentes.

Na sequência, foram abordados os desafios da IoT e blockchain na área de saúde, enumerando os principais requisitos e os desafios técnicos impostos pela tecnologia. Os principais protocolos de consenso foram agrupados em PoW, PoS, BFT e suas variantes, e sempre que possível, citadas referências que aplicações na área de saúde para exemplificar o emprego de tais protocolos.

O emprego da blockchain e IoT possui várias pesquisas e aplicações em andamento, conforme descrito na Seção 1.4, no entanto ainda há desafios de pesquisa para resolver ou aprimorar questões como desafios organizacionais para adição da blockchain; segurança e vulnerabilidade dos contratos inteligentes; grande e crescente volume de da-

dos na área de saúde e Interoperabilidade e suporte para transações entre plataformas [Ahmad e outros 2021].

Os Desafios organizacionais para a adoção de blockchain esbarra nos sistemas tradicionais de telemedicina, que dependem principalmente de métodos desatualizados para armazenar, manter e proteger os dados dos pacientes, o que pode limitar as oportunidades de colaboração entre os participantes e provedores de saúde. Conforme foi abordado, a tecnologia Blockchain garante que o histórico médico completo e confiável de um paciente possa ser mantido e rastreado pelos usuários autorizados por meio de registros imutáveis. No entanto, a falta de consciência, imaturidade da tecnologia e indisponibilidade de padrões de segurança e privacidade impedem os atores de empregar plenamente os benefícios da blockchain, inclusive os relacionados aos incentivos monetários para as organizações participantes [Kolan e outros].

Os contratos inteligentes ainda possuem riscos de adulteração e segurança e isto pode afetar significativamente o histórico médico de um paciente, como um ataque de vulnerabilidade [Liu e outros 2018]; um contrato inteligente, que tem privilégios exclusivos para se comunicar com outro contrato, pode alterar o EHR de um paciente, ou pode recuperar fundos da carteira de um usuário legítimo. Há ferramentas de diagnóstico, como ZeppolinOS, SolCover e Oyente. Essas ferramentas ajudam a identificar as características vulneráveis de contratos inteligentes para ajudar os desenvolvedores a propor contramedidas contra ameaças externas. No entanto, as soluções propostas são inadequadas para identificar todos os tipos de vulnerabilidades e bugs em um contrato inteligente. Portanto, os testes rigorosos são fundamentais para detectar vulnerabilidades antes de sua implementação.

O histórico médico consistente e atualizado de um paciente é sempre crescente e isto pode gerar uma enorme quantidade de dados que requer processamento rápido e constante. No entanto, para as plataformas atuais de blockchain, a grande quantidade de dados de saúde é um problema para as taxas de transação e o tempo total de mineração. O emprego de *sidechains* ou uma camada na névoa pode ajudar a minimizar a taxa de transação [Debe e outros 2019].

Construir plataformas de blockchain interoperáveis é desafiador devido a vários problemas, como diferenças nas linguagens suportadas e protocolos de consenso das plataformas de blockchain [Herlihy 2018]. Isto é um empecilho para o suporte à interoperabilidade dos sistemas de saúde que precisam ser interoperáveis e exigem transações seguras nas plataformas de blockchain. Apesar

Por fim, espera-se que este trabalho motive e contribua positivamente para a comunidade de Computação Aplicada à Saúde. Aqui apresentou-se uma visão geral e esclarecedora de um conjunto de tecnologias, que para operarem em conjunto, não possuem métodos triviais.

Referências

[Abijaude e outros 2021] Abijaude, J., Serra, Henrique Bezerra, A., Barretto, R., Sobreira, P., e Greve, F. (2021). Internet das coisas, blockchain e contratos inteligentes aplicados à saúde. `https://github.com/lifuesc/sbcas2021`. Acessado

em 09/06/2021.

[Adibi 2015] Adibi, S. (2015). A mobile health network disaster management system. In *2015 seventh international conference on ubiquitous and future networks*, pages 424–428. IEEE.

[Ahmad e outros 2021] Ahmad, R. W., Salah, K., Jayaraman, R., Yaqoob, I., Ellahham, S., e Omar, M. (2021). The role of blockchain technology in telehealth and telemedicine. *International Journal of Medical Informatics*, page 104399.

[Al Omar e outros 2017] Al Omar, A., Rahman, M. S., Basu, A., e Kiyomoto, S. (2017). Medibchain: A blockchain based privacy preserving platform for healthcare data. In *International conference on security, privacy and anonymity in computation, communication and storage*, pages 534–543. Springer.

[Albeyatti 2018] Albeyatti, A. (2018). Meddicalchain white paper. https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf.

[Alphand e outros 2018] Alphand, O., Amoretti, M., Claeys, T., Dall'Asta, S., Duda, A., Ferrari, G., Rousseau, F., Tourancheau, B., Veltri, L., e Zanichelli, F. (2018). Iotchain: A blockchain security architecture for the internet of things. In *2018 IEEE wireless communications and networking conference (WCNC)*, pages 1–6. IEEE.

[Anderson 2018] Anderson, J. (2018). Securing, standardizing, and simplifying electronic health record audit logs through permissioned blockchain technology.

[Antonopoulos 2017] Antonopoulos, A. (2017). *Mastering Bitcoin: Programming the Open Blockchain*. O'reilly, 2nd edition.

[Araujo e outros 2019] Araujo, P., Viana, W., Veras, N., Farias, E. J., e de Castro Filho, J. A. (2019). Exploring students perceptions and performance in flipped classroom designed with adaptive learning techniques: A study in distributed systems courses. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 30, page 219.

[Au e outros 2017] Au, M. H., Yuen, T. H., Liu, J. K., Susilo, W., Huang, X., Xiang, Y., e Jiang, Z. L. (2017). A general framework for secure sharing of personal health records in cloud system. *Journal of Computer and System Sciences*, 90:46–62.

[Azaria e outros 2016] Azaria, A., Ekblaw, A., Vieira, T., e Lippman, A. (2016). Medrec: Using blockchain for medical data access and permission management. In *2016 2nd International Conference on Open and Big Data (OBD)*, pages 25–30. IEEE.

[Bach e outros 2018] Bach, L. M., Mihaljevic, B., e Zagar, M. (2018). Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. IEEE.

[Bender e Sartipi 2013] Bender, D. e Sartipi, K. (2013). Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, pages 326–331.

[Bessani e outros 2014] Bessani, A., Sousa, J., e Alchieri, E. E. (2014). State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE.

[Buterin e outros. 2014] Buterin, V. e outros. (2014). A next-generation smart contract and decentralized application platform. *white paper*.

[Cachin e outros 2016] Cachin, C., Caro, A. D., Christidis, K., e Yellick, J. (2016). Architecture of the hyperledger blockchain fabric. Technical report, IBM Research - Zurich.

[Castro e outros 1999] Castro, M., Liskov, B., e outros. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

[Coinbase 2018] Coinbase (2018). Coinbase wallet. `https://wallet.coinbase.com/`. Acessado em 03/03/2021.

[da Conceição e outros 2018] da Conceição, A. F., da Silva, F. S. C., Rocha, V., Locoro, A., e Barguil, J. M. (2018). Eletronic health records using blockchain technology. *arXiv preprint arXiv:1804.10078*.

[da Saúde 2020] da Saúde, M. (2020). Rede nacional de dados em saúde. `https://rnds.saude.gov.br/`. Acessado em 05/06/2021.

[De Aguiar e outros 2020] De Aguiar, E. J., Faiçal, B. S., Krishnamachari, B., e Ueyama, J. (2020). A survey of blockchain-based strategies for healthcare. *ACM Computing Surveys (CSUR)*, 53(2):1–27.

[De Angelis e outros 2018] De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A., e Sassone, V. (2018). Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain.

[Debe e outros 2019] Debe, M., Salah, K., Rehman, M. H. U., e Svetinovic, D. (2019). Iot public fog nodes reputation system: A decentralized solution using ethereum blockchain. *IEEE Access*, 7:178082–178093.

[Dettling 2018] Dettling, W. (2018). How to teach blockchain in a business school. In *Business Information Systems and Technology 4.0*, pages 213–225. Springer.

[Dorri e outros 2017] Dorri, A., Kanhere, S. S., Jurdak, R., e Gauravaram, P. (2017). Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE.

[Dubovitskaya e outros 2017] Dubovitskaya, A., Xu, Z., Ryu, S., Schumacher, M., e Wang, F. (2017). Secure and trustable electronic medical records sharing using blockchain. In *AMIA annual symposium proceedings*, volume 2017, page 650. American Medical Informatics Association.

[Dziembowski e outros 2015] Dziembowski, S., Faust, S., Kolmogorov, V., e Pietrzak, K. (2015). Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer.

[Ethereum 2014] Ethereum, W. (2014). A secure decentralised generalised transaction ledger [j]. *Ethereum project yellow paper*, 151:1–32.

[Ethfiddle 2017] Ethfiddle (2017). Ethfiddle editor rinkeby. `https://ethfiddle.com/`. Acessado em 03/03/2021.

[Frankenfield 2018] Frankenfield, J. (2018). Proof of burn. `https://www.investopedia.com/terms/p/proof-burn-cryptocurrency`. Acessado em 04/05/2021.

[Greve e outros 2018] Greve, F. G., Sampaio, L. S., Abijaude, J. A., Coutinho, A. C., Valcy, Í. V., e Queiroz, S. Q. (2018). Blockchain e a revolução do consenso sob demanda. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos*.

[Greve 2005] Greve, F. G. P. (2005). Protocolos fundamentais para o desenvolvimento de aplicaçoes robustas. In *Minicursos SBRC 2005: Brazilian Symposium on Computer Networks*, pages 330–398.

[Guo e outros 2018] Guo, R., Shi, H., Zhao, Q., e Zheng, D. (2018). Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. *IEEE access*, 6:11676–11686.

[Guo e outros 2019] Guo, R., Shi, H., Zheng, D., Jing, C., Zhuang, C., e Wang, Z. (2019). Flexible and efficient blockchain-based abe scheme with multi-authority for medical on demand in telemedicine system. *IEEE Access*, 7:88012–88025.

[Halamka e outros 2019] Halamka, J. D., Alterovitz, G., Buchanan, W. J., Cenaj, T., Clauson, K. A., Dhillon, V., Hudson, F. D., Mokhtari, M. M., Porto, D. A., Rutschman, A., e outros. (2019). Top 10 blockchain predictions for the (near) future of healthcare. *Blockchain in Healthcare Today*.

[Herlihy 2018] Herlihy, M. (2018). Atomic cross-chain swaps. In *Proceedings of the 2018 ACM symposium on principles of distributed computing*, pages 245–254.

[Hoy 2017] Hoy, M. B. (2017). An introduction to the blockchain and its implications for libraries and medicine. *Medical reference services quarterly*, 36(3):273–279.

[Hussien e outros 2021] Hussien, H. M., Yasin, S. M., Udzir, N. I., Ninggal, M. I. H., e Salman, S. (2021). Blockchain technology in the healthcare industry: Trends and opportunities. *Journal of Industrial Information Integration*, 22:100217.

[Jaxx 2018] Jaxx (2018). Jaxx safely manager ethereum. `https://jaxx.io/`. Acessado em 03/03/2021.

[Jiang e outros 2016] Jiang, J., Wen, S., Yu, S., Xiang, Y., e Zhou, W. (2016). Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys & Tutorials*, 19(1):465–481.

[Kazmi e outros 2019] Kazmi, H. S. Z., Nazeer, F., Mubarak, S., Hameed, S., Basharat, A., e Javaid, N. (2019). Trusted remote patient monitoring using blockchain-based smart contracts. In *International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 765–776. Springer.

[Kiayias e outros 2017] Kiayias, A., Russell, A., David, B., e Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer.

[Kolan e outros] Kolan, A., Tjoa, S., e Kieseberg, P. Medical blockchains and privacy in austria-technical and legal aspects.

[Kotz e outros 2016] Kotz, D., Gunter, C. A., Kumar, S., e Weiner, J. P. (2016). Privacy and security in mobile health: A research agenda. *Computer*, 49(6):22–30.

[Kwon 2014] Kwon, J. (2014). Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 1(11).

[Lamport e outros 1982] Lamport, L., Shostak, R., e Pease, M. (1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

[Lao e outros 2020] Lao, L., Li, Z., Hou, S., Xiao, B., Guo, S., e Yang, Y. (2020). A survey of iot applications in blockchain systems: Architecture, consensus, and traffic modeling. *ACM Computing Surveys (CSUR)*, 53(1):1–32.

[Larimer 2014] Larimer, D. (2014). Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 81:85.

[Lee 2019] Lee, C. K. (2019). Blockchain application with health token in medical & health industrials. In *2nd International Conference on Social Science, Public Health and Education (SSPHE 2018)*, pages 233–236. Atlantis Press.

[Lee e Lee 2015] Lee, I. e Lee, K. (2015). The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440.

[Lerner 2015] Lerner, S. D. (2015). Dagcoin: a cryptocurrency without blocks. *White paper*.

[Liu e outros 2018] Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., e Roscoe, B. (2018). Reguard: finding reentrancy bugs in smart contracts. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*, pages 65–68. IEEE.

[Mannaro e outros 2018] Mannaro, K., Baralla, G., Pinna, A., e Ibba, S. (2018). A blockchain approach applied to a teledermatology platform in the sardinian region (italy). *Information*, 9(2):44.

[Mazieres 2015] Mazieres, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 32.

[McGhin e outros 2019] McGhin, T., Choo, K.-K. R., Liu, C. Z., e He, D. (2019). Blockchain in healthcare applications: Research challenges and opportunities. *Journal of Network and Computer Applications*, 135:62–75.

[Metamask 2018] Metamask (2018). Metamask crypto wallet and gateway. `https://metamask.io/`. Acessado em 03/03/2021.

[Milutinovic e outros 2016] Milutinovic, M., He, W., Wu, H., e Kanwal, M. (2016). Proof of luck: An efficient blockchain consensus protocol. In *proceedings of the 1st Workshop on System Software for Trusted Execution*, pages 1–6.

[MyCrypto 2019] MyCrypto (2019). Mycrypto. `https://mycrypto.com/`. Acessado em 03/03/2021.

[Myetherwallet 2019] Myetherwallet (2019). Myetherwallet original wallet. `https://www.myetherwallet.com/`. Acessado em 03/03/2021.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Technical report, Bitcoin Org.

[Narayanan e outros 2016] Narayanan, A., Bonneau, J., Felten, E., Miller, A., e Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies*. Princeton University Press.

[of Health e outros 2008] of Health, U. D., Services, H., e outros. (2008). Personal health records and the hipaa privacy rule. *Washington, DC. URL: https://www. hhs. gov/sites/default/files/ocr/privacy/hipaa/understanding/special/healthit/phrs. pdf [accessed 2016-06-20][WebCite Cache]*.

[Patel 2019] Patel, V. (2019). A framework for secure and decentralized sharing of medical imaging data via blockchain consensus. *Health informatics journal*, 25(4):1398–1411.

[PoET 2018] PoET (2018). Poet 1.0 specification. `https://sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html`. Acessado em 04/05/2021.

[Popov 2018] Popov, S. (2018). The tangle, iota whitepaper. `https://iota.org/IOTA\_Whitepaper.pdf`. Acessado em 03/03/2021.

[Popov e outros 2020] Popov, S., Moog, H., e et al. (2020). The coordicide. *white paper Iota Foundation*.

[Raikwar e outros 2018] Raikwar, M., Mazumdar, S., Ruj, S., Gupta, S. S., Chattopadhyay, A., e Lam, K.-Y. (2018). A blockchain framework for insurance processes. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–4. IEEE.

[Ramachandran e outros 2020] Ramachandran, S., Kiruthika, O. O., Ramasamy, A., Vanaja, R., e Mukherjee, S. (2020). A review on blockchain-based strategies for management of electronic health records (ehrs). In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 341–346. IEEE.

[Rao e Dave 2019] Rao, A. R. e Dave, R. (2019). Developing hands-on laboratory exercises for teaching stem students the internet-of-things, cloud computing and blockchain applications. In *2019 IEEE Integrated STEM Education Conference (ISEC)*, pages 191–198. IEEE.

[Remix 2015] Remix (2015). Remix ide. `https://remix.ethereum.org/`. Acessado em 03/03/2021.

[Roehrs e outros 2017] Roehrs, A., Da Costa, C. A., e da Rosa Righi, R. (2017). Omniphr: A distributed architecture model to integrate personal health records. *Journal of biomedical informatics*, 71:70–81.

[Salah e outros 2020] Salah, K., Alfalasi, A., Alfalasi, M., Alharmoudi, M., Alzaabi, M., Alzyeodi, A., e Ahmad, R. W. (2020). Iot-enabled shipping container with environmental monitoring and location tracking. In *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE.

[Samaniego e outros 2016] Samaniego, M., Jamsrandorj, U., e Deters, R. (2016). Blockchain as a service for iot. In *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pages 433–436. IEEE.

[Sandgaard e Wishstar 2018] Sandgaard, J. e Wishstar, S. (2018). Medchain white paper. http://www.medchain.us/doc/Medchain%20Whitepaper%20v1.0.pdf. Acessado em 04/06/2021.

[Santos e outros 2016] Santos, B. P., Silva, L. A., Celes, C., Borges, J. B., Neto, B. S. P., Vieira, M. A. M., Vieira, L. F. M., Goussevskaia, O. N., e Loureiro, A. (2016). Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuidos*, 31.

[Saweros e Song 2019] Saweros, E. e Song, Y.-T. (2019). Connecting heterogeneous electronic health record systems using tangle. In *International Conference on Ubiquitous Information Management and Communication*, pages 858–869. Springer.

[Shahnaz e outros 2019] Shahnaz, A., Qamar, U., e Khalid, A. (2019). Using blockchain for electronic health records. *IEEE Access*, 7:147782–147795.

[Silvano e Marcelino 2020] Silvano, W. F. e Marcelino, R. (2020). Iota tangle: A cryptocurrency to communicate internet-of-things data. *Future Generation Computer Systems*, 112:307–319.

[Siyal e outros 2019] Siyal, A. A., Junejo, A. Z., Zawish, M., Ahmed, K., Khalil, A., e Soursou, G. (2019). Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. *Cryptography*, 3(1):3.

[Sobreira 2005] Sobreira, P. (2005). Desenvolvimento de uma arquitetura microcontrolada para tratamento de sinais biológicos. Master's thesis, Universidade Federal de Pernambuco.

[Sousa e outros 2018] Sousa, J., Bessani, A., e Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 51–58. IEEE.

[Status 2019] Status (2019). Status private, secure communication. `https://status.im/`. Acessado em 03/03/2021.

[StudioEthereum 2019] StudioEthereum (2019). Studio ethereum. `https://studio.ethereum.org//`. Acessado em 03/03/2021.

[Szabo 1997] Szabo, N. (1997). Formalizing and securing relationships on public networks. *First Monday*, 2(9).

[Sztajnberg e outros 2018] Sztajnberg, A., da Silva Macedo, R., e Stutzel, M. (2018). Protocolos de aplicação para a internet das coisas: conceitos e aspectos práticos. *Sociedade Brasileira de Computação*.

[Thatcher e Acharya 2018] Thatcher, C. e Acharya, S. (2018). Pharmaceutical uses of blockchain technology. In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6. IEEE.

[Todd 2015] Todd, P. (2015). Ripple protocol consensus algorithm review. *Ripple Labs Inc White Paper (May, 2015) https://raw. githubusercontent. com/petertodd/rippleconsensus-analysis-paper/master/paper. pdf*.

[Trust 2019] Trust (2019). Trust wallet - secure crypto wallet. `https://trustwallet.com/`. Acessado em 03/03/2021.

[Web3js 2016] Web3js (2016). Ethereum javascript api. `https://web3js.readthedocs.io/en/v1.3.0/index.html`. Acessado em 03/03/2021.

[Weissman e outros 2018] Weissman, S. M., Zellmer, K., Gill, N., e Wham, D. (2018). Implementing a virtual health telemedicine program in a community setting. *Journal of genetic counseling*, 27(2):323–325.

[Wu e outros 2019] Wu, M., Wang, K., Cai, X., Guo, S., Guo, M., e Rong, C. (2019). A comprehensive survey of blockchain: From theory to iot applications and beyond. *IEEE Internet of Things Journal*, 6(5):8114–8154.

[Yli-Huumo e outros 2016] Yli-Huumo, J., Ko, D., Choi, S., Park, S., e Smolander, K. (2016). Where is current research on blockchain technology?—a systematic review. *PloS one*, 11(10):e0163477.

[Yüksel e outros 2017] Yüksel, B., Küpçü, A., e Öznur Özkasap (2017). Research issues for privacy and security of electronic health services. *Future Generation Computer Systems*, 68:1–13.

[Zhang e outros 2017] Zhang, Y., Liu, T., Li, K., e Zhang, J. (2017). Improved visual correlation analysis for multidimensional data. *Journal of Visual Languages and Computing*, 41:121–132.

[Zheng e outros 2018] Zheng, Z., Xie, S., Dai, H.-N., Chen, X., e Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375.

