Capítulo 1

Construindo aplicações baseadas em *blockchain* com o *Hyperledger FireFly*

Maria do Rosário de Fátima Martins Ferreira, Bernardo Ferreira de Moura Ribeiro, Alcemir Rodrigues Santos, Ricardo de Andrade Lira Rabêlo

Abstract

Blockchain technology is no longer just about cryptocurrencies. Companies are increasingly exploring the use cases available to increase the added value of their products and services, as well as improving the security and strengthening the trust of their systems. However, developing and maintaining applications based on this technology is still a major challenge, especially when it comes to integrating them with various web applications, what are now known as Web3 applications. Hyperledger FireFly has emerged as a solution with the potential to ease new developers onto the path of developing such applications. Considering the scarcity of teaching material, as well as learning opportunities for beginners in this technology, this chapter proposes an introductory experience for this target audience to get started in the development of Web3 applications.

Resumo

Blockchain não é uma tecnologia apenas sobre criptomoedas. As empresas estão explorando cada vez mais os casos de uso disponíveis para aumentar o valor agregado de seus produtos e serviços, além de melhorar a segurança e fortalecer a confiança de seus sistemas. No entanto, o desenvolvimento e a manutenção de aplicativos baseados nessa tecnologia ainda são um grande desafio, especialmente quando se trata de integrá-los a vários aplicativos da Web, o que hoje é conhecido como Web3. O Hyperledger FireFly surgiu como uma solução com potencial para facilitar a entrada de novos desenvolvedores nesse caminho de desenvolvimento. Considerando a escassez de material didático, bem como de oportunidades de aprendizado para iniciantes nessa tecnologia, este capítulo propõe uma experiência introdutória para que esse público-alvo inicie o desenvolvimento de aplicativos Web3.

1.1. Introdução

Em 2008, o mundo foi testemunha do nascimento da primeira e ainda mais relevante e valorizada criptomoeda até o momento: o *Bitcoin* (BTC) [Nakamoto 2008]. Sob o título de "Um sistema de dinheiro eletrônico ponto a ponto", a invenção de Satoshi Nakamoto deu início a um mercado de criptomoedas que viria a ser uma das grandes revoluções financeiras e culturais do século. Até a data de publicação deste capítulo, calcula-se que existam mais de 9 mil criptomoedas no mundo. As principais possuem valor de mercado que variam de algumas centenas de milhões de dólares, como *Zcash*, a 1.2 trilhões[1] do próprio *Bitcoin*. Cada uma delas com seu próprio modelo de negócio, as razões para o sucesso das criptomoedas são diversas, mas dependem da confiança na tecnologia que baseia a implementação e operação destes ativos digitais: a *Blockchain*.

Embora tentativas anteriores como *Ecash* [Chaum 1983] e *Bit Gold* [Szabo 2008] tenham falhado, elas exemplificam como a ideia de usar moedas exclusivamente digitais começou muito antes do surgimento do BTC. O mesmo também pode ser dito quanto à outros conceitos por trás da tecnologia *Blockchain*. O texto *"Bitcoin's Academic Pedigree"*, [Narayanan and Clark 2017], destaca como o trabalho de Nakamoto é na verdade construído a partir de ideias "esquecidas" na literatura. Por exemplo, a assinatura digital e a marcação temporal da criação e última modificação de um documento foi proposta em 1991 por [Haber and Stornetta 1991]. Além disso, a noção de redes distribuídas, como a *peer-to-peer* (P2P), remete aos anos 60 [Baran 1964] e o algoritmo *Proof-of-Work* (PoW), aos anos 90 [Dwork and Naor 1993]. Essa observação não pretende minimizar a conquista de Nakamoto, mas reconhecer que sua verdadeira inovação foi combinar esses componentes subjacentes para criar sua criptomoeda.

Nakamoto uniu a descentralização característica do modelo P2P com uma robusta estratégia de registro criptografado de transações. Este arranjo, integrado por meio de um protocolo de consenso — PoW —, assegurou a imutabilidade das informações inseridas no sistema, estabelecendo uma base sólida para a realização de transações monetárias seguras sem a necessidade de uma terceira parte confiável ou autoridade central. Assim, ele conseguiu adicionar uma forte capacidade de confiança para seu sistema de dinheiro eletrônico, construindo as bases do que seria, cerca de oito anos depois, chamado de "Máquina de Confiança" [The Economist 2015]. Graças a essa inovação, o BTC e sua *blockchain* ganharam impulso nos anos seguintes, representando uma revolução não apenas na maneira como o dinheiro é tratado digitalmente, mas também na transferência de dados no que viria a ser uma nova era da Internet.

Com essa popularização, a tecnologia *Blockchain* evoluiu e o interesse de organizações privadas, governos e empresas de diversos portes pelo uso corporativo dessa tecnologia aumentou substancialmente, expandindo-se os casos de uso para muito além das criptomoedas. Essas instituições vislumbraram na tecnologia uma oportunidade de reduzir a dependência de intermediários em transações online, o que poderia diminuir atritos, atrasos e custos operacionais, além de potencialmente mitigar os riscos de ataques cibernéticos. Atualmente, já existem aplicações em áreas como finanças, saúde, imobiliária e gestão de dados, e a adoção segue em crescimento, acompanhada pela criação de ferramentas que facilitam o desenvolvimento de novas soluções. Nesse cenário,

[1] Fonte: `https://investnews.com.br/cotacao-criptomoeda/`. Acesso em 13/08/2024.

o presente capítulo tem como objetivo introduzir os principais conceitos de *Blockchain* e demonstrar como a ferramenta *Hyperledger FireFly* pode ser uma porta de a entrada de novos desenvolvedores ao universo da *Web3*.

As próximas seções estão organizadas como segue:

Seção 1.2: nesta seção, serão introduzidos os conceitos básicos associados à tecnologia *blockchain*, com o objetivo de fundamentar o conhecimento.

Seção 1.3: nesta seção, abordamos brevemente como identificar se a *blockchain* é adequada para as necessidades do seu projeto. Esta é uma consideração importante antes da construção de aplicações *Web3*.

Seção 1.4: esta seção introduz a ferramenta *Hyperledger FireFly*, seu ecossistema "guarda-chuva" e outros projetos relacionados. Além disso, são descritos sua arquitetura, funcionamento e componentes.

Seção 1.5: nesta seção, apresentar-se-á o desenvolvimento de uma aplicação exemplo com o *Hyperledger FireFly*, buscando explorar ao máximo as possibilidades.

Seção 1.6: esta seção apresenta os próximos passos para o(a) leitor(a) na construção de suas próprias aplicações *Web3*;

Seção 1.7: nesta seção de encerramento do capítulo, concluiremos apontando algumas possíveis referências de leitura futura.

1.2. A tecnologia *blockchain*

Em referente a nomenclatura, é interessante apontar que, embora a rede em que o BTC opera seja reconhecida como a primeira rede *blockchain* a ser implementada na história, o termo surgiu somente com a posterior popularização da tecnologia. No *white-paper* original, Nakamoto não utiliza-o sequer uma vez. Na verdade, ele se refere a estrutura de dados como "cadeia de blocos" (*"chain of blocks"*), que apesar de simples resume relativamente bem a essência da tecnologia. Além disso, por ser uma combinação de diversos paradigmas, a definição de *Blockchain* pode ser abordada a partir de diferentes perspectivas [Kakavand et al. 2017], o que pode gerar confusões no entendimento. Aqui, buscamos demonstrar como esses conceitos distintos, na realidade, convergem para definir a rica e multifacetada composição que é a *Blockchain*.

A norma ISO 22739:2024 [ISO Central Secretary 2024] define formalmente um padrão internacional para o uso de vocabulário relativo a tecnologias de livro-razão (*ledger*)[2] distribuído (DLTs), incluindo a *Blockchain*, com o objetivo de definir e esclarecer conceitos e termos técnicos. No documento, o termo *"blockchain"* é definido como sendo uma estrutura de dados, enquanto *"sistema blockchain"* é definido como um sistema DLT que implementa uma *blockchain*, e finalmente, *"tecnologia blockchain"* é definido como

[2]Na contabilidade, um livro-razão, uma espécie de registro de escrituração que funciona como uma ferramenta de controle e análise de finanças em empresas, permitindo o registro e a conferência de transações financeiras.

o conjunto de todas as tecnologias necessárias para operação e uso de um sistema *blockchain*. Neste capítulo, adotaremos as definições desta norma ISO, com a ressalva de que quando nos referirmos à tecnologia - *Blockchain*-, usaremos letra maiúscula para diferenciá-la da estrutura de dados e das redes -*blockchain*.

Em geral, uma *blockchain* é uma plataforma digital que armazena e verifica todo o histórico de transações entre usuários da rede de forma inviolável e à prova de edições criminosas. Ela também pode ser vista como uma estrutura de banco de dados subjacente para transações digitais seguras [Kakavand et al. 2017]. Greve *et al.* [Greve et al. 2018] apresentou os elementos fundamentais da *blockchain*: **o livro-razão distribuído, o mecanismo de consenso e as técnicas de criptografia**. A Tabela 1.1 resume as características que permitem que a tecnologia tenha o potencial de substituir transações baseadas em pura confiança por transações baseadas em regras definidas matematicamente e aplicadas mecanicamente. Está fora do escopo desta capítulo detalhar cada um desses atributos, porém mais detalhes podem ser encontrados no trabalho referenciado.

Tabela 1.1. Elementos fundamentais e propriedades. Fonte: [Greve et al. 2018]

Livro-Razão	Transparência, Auditabilidade, Imutabilidade e Irrefutabilidade
Distribuição	Descentralização, Replicação e Disponibilidade
Consenso	Desintermediação, Atualidade e Integridade
Criptografia	Identidade, Assinaturas Digitais, Segurança e Privacidade

1.2.1. Afinal, o que é uma *blockchain*?

Uma *blockchain* é um tipo de sistema descentralizado, o que significa que, diferente da maior parte das aplicações modernas, ela não é controlada por um servidor central, ou seja, não possui uma autoridade como uma grande empresa ou governo controlando-a. Essa plataforma se destaca pela capacidade de, mesmo sem uma autoridade central, verificar a validade desses envios e ainda armazenar todo o histórico de transações em um livro-razão digital distribuído e inviolável, garantindo segurança e auditabilidade para esse registro. Nesta sub-seção buscamos definir com mais precisão técnica o que de fato é uma *blockchain*, explorando algumas de suas definições e seus principais componentes.

No coração de uma *blockchain*, o livro-razão é um elemento chave na definição e no funcionamento da tecnologia. Este elemento é composto por dois componentes interligados: uma base de dados que contém o estado atual da rede, conhecida como estado mundial (*World State*), e um registro de todas as transações que levaram a esse estado, organizadas em uma estrutura de dados baseada em uma cadeia de blocos, a que se nomeia *blockchain*. A cadeia registra as mudanças de estado e "alimenta" o estado mundial, que por sua vez reflete os dados atuais do sistema, sendo implementado como um banco de dados. Isso é vantajoso pois assim ele é capaz de oferecer um conjunto eficiente de operações de armazenamento e recuperação de estados, evitando que um programa precise percorrer toda a cadeia de blocos para calcular um valor específico; em vez disso, ele pode ser obtido diretamente a partir do estado mundial. Enquanto isso, a *blockchain* mantém seguras as informações de evolução deste banco de dados, possibilitando transparência para conferências futuras.

Uma *blockchain*, portanto, é uma estrutura de dados que por razões didáticas pode ser entendida como uma lista ligada de blocos de transações (T_x), ligados entre si em forma de cadeia (Figura 1.1) e que registra todas as transações que ocorrerem entre os nós na forma de arquivo. Neste arranjo, cada bloco é criptograficamente encadeado ao bloco anterior através de um código *hash*[3], que resume as informações do antecessor, também chamado bloco-pai. Dessa forma, cada novo bloco criado possui uma marcação de tempo de criação (*Timestamp*) e um número escalar arbitrário chamado *nonce*, que ajuda a validar o bloco e, em seguida, adicioná-lo à *blockchain*. Além de ajudar a preservar a integridade do histórico, esses elementos também conferem à estrutura a característica de ser imutável e apenas anexável, ou seja, é praticamente impossível alterar ou excluir um bloco previamente inserido sem danificar toda a cadeia.

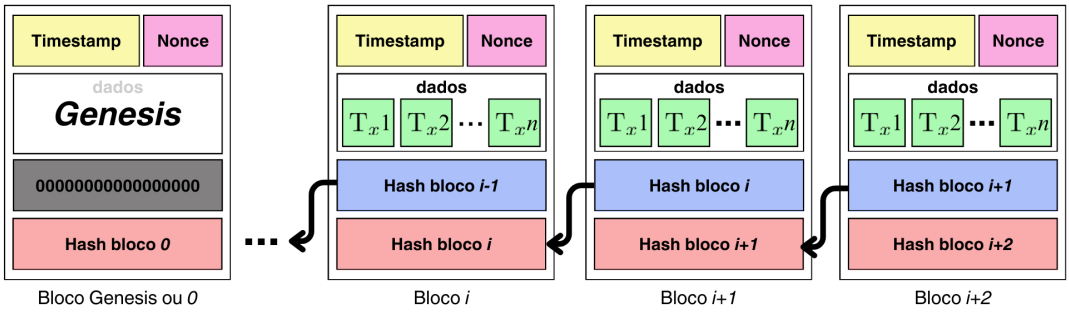

Figura 1.1. Representação de uma *blockchain*. *(Fonte: autor)*

O livro-razão da *blockchain* também tem a característica de ser distribuído através de uma rede. Com isso, surge o conceito de *blockchain* como sendo também uma rede distribuída do tipo peer-to-peer (P2P), ou seja, par-a-par. Cada par, também chamado nó (*Node*), é um *software* sendo executado em uma máquina diferente, que mantém uma cópia do livro-razão e que pode submeter e receber transações a ele através de uma conexão. O nó é um elemento fundamental da rede, enquanto essa é responsável por conectar as máquinas uma a uma, disseminar informações e sincronizar os livros-razão. Um determinado tipo de nó é capaz de sugerir novos blocos e a cada intervalo de tempo um nó é escolhido para anexar seu bloco proposto ao registro compartilhado. Para evitar que essa escolha acabe sendo partidária ou centralizada, a mesma é tomada em conjunto pelos nós, a partir de um processo chamado consenso. Uma vez escolhido, o bloco "vencedor" é transmitido através da rede por protocolos P2P.

O processo de consenso é crucial para garantir que todas as transações sejam registradas de forma precisa e segura na *blockchain*. Na rede *Bitcoin*, por exemplo, esse consenso é alcançado por meio de um mecanismo chamado *Proof-of-Work* (PoW). Nele, os nós mineradores competem para resolver complexos problemas matemáticos e o primeiro a encontrar uma solução válida adiciona um novo bloco à *blockchain*. Este processo é custoso e exige que a solução seja verificada por outros mineradores, garantindo que todas as transações sejam legítimas e que o novo bloco esteja em conformidade com as regras da rede. Por exemplo, tal como uma poupança, uma carteira de BTC não pode gastar mais

[3]Resumo das informações do bloco criptografadas por algum algoritmo como SHA-256

do que o disponível como saldo de suas transações anteriores. Uma vez que a maioria dos mineradores aceita um novo bloco como válido, ele é adicionado à *blockchain* e registrado em todas as cópias do livro-razão, assegurando que todos os participantes tenham uma visão consistente e confiável do histórico de transações, ou seja, do livro-razão.

Apesar dessas definições, ressaltamos que, conforme [Kakavand et al. 2017] em "A Revolução *Blockchain* ", essa tecnologia não tem uma única definição universalmente aceita, por se tratar de uma composição complexa de várias dimensões, incluindo tecnológica, operacional e regulatória. Ainda segundo os autores, a tecnologia já foi descrita como sendo, ao mesmo tempo, uma rede distribuída e um banco de dados equipado com segurança integrada. Entretanto, a chegada da rede Ethereum [Buterin 2013] e a concomitante formalização da tecnologia [Wood 2019] endossaram a visão de *Blockchain* também como uma máquina de estados. Por esta razão, e levando em conta a importância desse conceito para o desenvolvimento da *Web3*, consideramos essencial adicionar mais essa definição, o que será feito na seção a seguir.

1.2.2. O surgimento da *Ethereum*, Contratos Inteligentes e *Web3*

No mesmo ano em que o jornal britânico *The Economist* publicou a matéria que chamava a tecnologia *Blockchain* de "Máquina de confiança" [The Economist 2015], uma nova rede pública foi lançada: a Ethereum [Buterin 2013]. Essa revolucionária rede, apesar de funcionar de forma semelhante à Bitcoin, foi concebida com a proposta de fornecer uma *blockchain* equipada com um computador *bult-in*, além de uma linguagem de programação completa e integrada, capaz de criar programas executáveis complexos. Conhecidos como contratos inteligentes (*smart contracts*), esses programas são acordos autoexecutáveis baseados em código, projetados para automatizar processos e permitir a criação das mais diversas aplicações descentralizadas (*DApps*), com suas próprias regras de negócio.

Isso é possível pois na rede distribuída da Ethereum, há um único computador canônico, chamado de Máquina Virtual Ethereum (EVM), um computador virtual global cujo estado é aceito e copiado por todos os nós da rede. Qualquer participante pode transmitir uma solicitação, também chamada de transação, para que a EVM execute um cálculo arbitrário. Sempre que uma solicitação desse tipo é transmitida, outros nós da rede verificam, validam e executam o cálculo. Essa execução causa uma alteração de estado na EVM, que é confirmada e propagada por toda a rede. O registro de todas as transações e o estado atual da EVM são então armazenados na *blockchain*.

Por esse motivo, autores como Greve *et al.*, definem que a *blockchain* implementa uma "máquina de estados replicada para a manutenção consistente de um estado global compartilhado por um conjunto de pares distribuídos numa rede P2P" [Greve et al. 2018]. Como reconhece [Buterin 2013], tanto o modelo de *blockchain* da Bitcoin quanto da Ethereum podem ser vistos como máquinas de estado. Entretanto, enquanto o estado na primeira se refere basicamente a contas e saldos, na segunda ele é uma grande estrutura de dados que contém também um estado de máquina, que pode mudar de acordo com um conjunto predefinido de regras definidas pela EVM. É a definição de estados mais complexos, através dos contratos inteligentes, que possibilita a criação de interações mais complexas com a *blockchain*, o que forma a base para execução das *DApps*.

As *DApps* são aplicações que rodam na Internet e permitem execução de contra-

tos inteligentes em redes *blockchain*. Qualquer desenvolvedor pode criar um contrato inteligente personalizado para seu negócio e torná-lo público na rede, usando a *blockchain* como sua camada de dados. Assim, qualquer usuário pode então usar uma *DApp* para chamar o contrato e executar seu código. Essas aplicações são componentes-chave na construção da *Web3*, a emergente nova geração da internet, onde aplicações são executadas de forma descentralizada, visando mais segurança, privacidade e controle dos usuários sobre seus dados e identidades digitais. Essa revolução popularizou ainda mais a tecnologia, levando empresas e governos a adotarem-na para fins além das criptomoedas.

1.2.3. *Blockchain* pública vs privada

A norma ISO 22739, entre outras definições, categoriza as tecnologias DLT como permissionadas e não-permissionadas, bem como públicas e privadas. Na prática, contudo, os termos frequentemente são utilizados como sinônimos, sendo uma *blockchain* não-permissionada compreendida como pública, onde não são necessárias permissões e, portanto, qualquer usuário pode acessar e inserir informações no livro-razão. Em contrapartida, uma *blockchain* permissionada é frequentemente entendida como privada, pois apenas usuários com permissão podem acessá-la e modificá-la.

As redes *Bitcoin* e *Ethereum* são exemplos de redes públicas, ou seja, qualquer usuário pode executar funções de leitura e escrita. Em uma *blockchain* não-permissionada, não há uma autoridade central que controle o acesso ou que possa banir usuários ilegítimos. Essa característica de forte abertura proporciona um alto nível de transparência, permitindo que todas as transações sejam visíveis para todos os participantes da rede. Além disso, os usuários desse modelo não precisam confiar uns nos outros para realizarem transações seguras entre si. Apesar disso, algumas desvantagens das redes públicas é a baixa taxa de transações por segundo (TPS), que pode aumentar a latência e diminuir o *throughput* das transações. De fato, a rede *Bitcoin* leva, em média, por volta de 10 minutos para validar uma transação [Zheng et al. 2018]. Ademais, os elevados gastos com taxas de transação, juntamente com os altos custos computacionais e energéticos destacam-se como desvantagens do modelo não-permissionado.

Por outro lado, redes privadas oferecem a capacidade de restringir o acesso a um grupo específico de participantes, garantindo maior privacidade e controle sobre o livro-razão. Nesse modelo, uma entidade conhecida como Âncora de Confiança é responsável por conceder e revogar permissões de leitura e gravação no sistema. Algumas redes *blockchain* permissionadas limitam o acesso a usuários pré-verificados que já comprovaram sua identidade. Outras permitem a visualização para qualquer pessoa, mas apenas identidades confiáveis podem verificar transações. Embora esse modelo possa parecer contradizer o conceito original de *blockchain*, por depender da confiança em uma entidade específica, é preciso considerar que em ambientes empresariais isso é frequentemente aceitável devido ao nível pré-existente de confiança entre os participantes. Dessa forma, corporações podem aproveitar os benefícios da *Blockchain* enquanto ainda mantém um nível de controle sobre seus negócios. As redes *Hyperledger Fabric* e *R3 Corda* são exemplos notáveis de blockchains permissionadas.

1.3. Quando usar (e quando não usar) *Blockchain*?

Com a popularização da *Blockchain*, a tecnologia se tornou amplamente difundida em diversos setores, sendo vista como uma solução revolucionária para inúmeras aplicações além de seu uso inicial em finanças. Sua ascensão ao destaque levou a implementações bem-sucedidas em áreas como saúde, transporte, indústria e gestão da cadeia de suprimentos, exemplos que serão abordados mais adiante neste capítulo. No entanto, como observa [Zīle and Strazdiņa 2018], o "hype" em torno da tecnologia cria a ilusão de que a *Blockchain* é a solução para qualquer problema, o que leva algumas organizações a adotarem a tecnologia sem avaliar sua adequação às necessidades específicas de projetos, resultando em sistemas complexos, ineficientes ou de difícil manutenção.

Este capítulo já discutiu as definições e características da tecnologia *Blockchain*, incluindo sua promessa de segurança, privacidade e operações descentralizadas. Entretanto, essas características, por si só, nem sempre justificam a adoção da tecnologia. Segundo [Wust and Gervais 2018], uma análise das propriedades da *Blockchain*, considerando os requisitos e o contexto específicos de cada aplicação é necessária para avaliar se sua aplicação adequada. Esta seção pretende explorar esses fatores, ajudando a identificar quando o uso da tecnologia é realmente justificado.

Primeiro, alguns autores como [Wust and Gervais 2018], [Hassija et al. 2021] e [Enoch et al. 2023] que discutem o uso adequado de um livro-razão distribuído descrevem que a análise deve considerar: a necessidade de armazenamento de dados, o envolvimento de múltiplas partes e as dinâmicas de confiança entre elas. A *Blockchain* é mais vantajosa quando diversas entidades que não confiam totalmente umas nas outras precisam interagir e modificar o estado de um sistema sem depender de uma terceira parte confiável (TTP). Nesses casos, a *Blockchain* pode viabilizar transações e trocas de dados seguras, assegurando que nenhuma parte tenha controle unilateral sobre o sistema. No entanto, quando uma TTP está disponível e é confiável, um banco de dados tradicional pode ser uma solução mais eficiente, oferecendo melhor desempenho em termos de capacidade de processamento e latência.

Além disso, é importante discutir as diferenças entre redes *blockchain* permissionadas e não-permissionadas, e quando a aplicação de cada uma delas é mais adequada. Como mencionado anteriormente neste capítulo, redes não-permissionadas, como o *Bitcoin*, permitem que qualquer usuário ingresse na rede sem a necessidade de aprovação de uma autoridade central. Em contraste, redes permissionadas, como o *Hyperledger Fabric*, limitam o acesso a um grupo restrito de leitores e escritores, com uma autoridade central responsável por gerenciar a adesão e conceder ou revogar acessos.

Assim, uma rede permissionada é ideal para cenários onde há necessidade de interação entre um número restrito de entidades que são conhecidas entre si e há a necessidade de manter a privacidade, enquanto ainda se beneficiam de um livro-razão descentralizado. Por outro lado, as redes não-permissionadas são projetadas para situações em que o conjunto de escritores não é fixo, os participantes são desconhecidos uns dos outros, e o sistema precisa permanecer aberto a qualquer pessoa que deseje participar. Ele é particularmente vantajoso em ambientes onde a descentralização é fundamental e a verificabilidade pública deve ser garantida sem depender de uma entidade central confiável.

Em conclusão, embora a *Blockchain* tenha o potencial de revolucionar certos negócios, ela não é uma solução universal. Sua necessidade depende de fatores específicos, como a necessidade de armazenar dados, o envolvimento de várias partes e o nível de confiança entre elas. Logo, em situações onde a confiança não é uma questão ou onde sistemas centralizados podem operar com mais eficiência, o uso da tecnologia pode ser desnecessário e até contraproducente. Portanto, é fundamental realizar uma avaliação cuidadosa desses fatores antes de decidir implementar a tecnologia *Blockchain*.

1.4. O *Hyperledger FireFly*

Reconhecendo as vantagens da tecnologia *Blockchain*, várias empresas interessadas no uso corporativo dela perceberam que poderiam obter resultados mais significativos ao colaborarem entre si, unindo esforços para desenvolver soluções de código aberto acessíveis a qualquer usuário. Nesta seção, pretendemos demonstrar como um desses projetos colaborativos evoluiu para se tornar uma referência no desenvolvimento de redes *blockchain*, abrangendo não apenas o uso corporativo, mas também acadêmico e científico. Analisaremos o Ecossistema Hyperledger, sua visão, os membros que o compõem e os projetos em andamento, especialmente o *Hyperledger FireFly*, que constitui o tema deste capítulo.

1.4.1. Surgimento: a Fundação Hyperledger

No contexto de contribuição coletiva, em 2015, sob a guarda da Fundação Linux, a Fundação Hyperledger teve início. Originalmente conhecida como Projeto Hyperledger, hoje fundação, tinha como objetivo ser um esforço colaborativo para construir uma plataforma de livros-razão distribuídos e código aberto, para uso em demandas industriais específicas [Foundation 2016]. Na atualidade, a Fundação Hyperledger é um ecossistema global aberto com foco no uso empresarial da tecnologia. A fundação não se considera uma rede de blockchains, mas como uma rede de consórcios onde empresas e outras organizações podem implementar suas soluções de DLT [Foundation 2024].

A Hyperledger propõe nutrir uma comunidade que combina as possibilidades empresariais do uso da *blockchain* com as vantagens do desenvolvimento de código aberto, criando uma plataforma de criação e uso coletivo de DLTs. Todo o código do portifólio da Fundação Hyperledger é desenvolvido publicamente e está disponível sob a licença Apache. As organizações se associam à Fundação Hyperledger para demonstrar liderança técnica, colaborar e interagir com outras empresas, além de aumentar a conscientização sobre seus esforços na comunidade de *blockchain* empresarial. Os membros da Hyperledger incluem organizações líderes em diversos setores: finanças e bancos, como American Express, Visa e BNDES; saúde; cadeias de suprimentos, como Walmart; e tecnologia, com IBM, Oracle e Huawei, entre outros. Em seguida, apresentamos sobre os projetos desenvolvidos no ecossistema Hyperledger.

1.4.2. Ecossistema: Projetos Hyperledger

A lista de projetos mantidos pela fundação é dinâmica[4], com a inclusão ou remoção de projetos à medida que novos projetos embrionários vão ganhando maturidade ou proje-

[4]É possível encontrar detalhes dos projetos mantidos atualmente na página Web principal da Fundação: `https://www.hyperledger.org/`

tos maduros vão perdendo espaço na comunidade. Atualmente[5], existem três categorias de projetos orientados à *blockchain* dentro do ecossistema Hyperledger. A saber, *(i)* os DLTs, que podem ter propósito geral ou específico de identidade digital; *(ii)* as bibliotecas; e *(iii)* as ferramentas. A Tabela 1.2 enumera projetos de cada uma dessas categorias. Embora muitos dos projetos tenham aplicações no contexto de *Web3*, não é do escopo deste capítulo detalhar e/ou apresentar todos os projetos disponíveis para este propósito.

Tabela 1.2. Projetos Hyperledger.

DLTs	Besu, Fabric, Iroha, Inddy, Identus
Bibliotecas	Aries, Web3J
Ferramentas	AnonCreds, Bevel, Cacti, Caliper, Cello, FireFly, Solang

Dentre os projetos graduados, destacam-se as DLTs *Fabric* e *Besu*. O *Hyperledger Fabric* é uma plataforma para criar soluções *blockchain* com uma arquitetura modular e que oferece um alto grau de confidencialidade, flexibilidade, resiliência e escalabilidade. Isso permite que soluções desenvolvidas com a Fabric sejam aplicáveis a qualquer setor. Na rede *Fabric* ainda não existe um token nativo (como o *Bitcoin* ou ETH), porém tokens podem ser criados a nível de chaincode de aplicação, emulando tokens fungíveis (ERC-20) ou não fungíveis (*NFT*) (ERC-721). Por outro lado, o *Hyperledger Besu*, diferente do *Fabric* que implementa uma rede própria, é um cliente *Ethereum* de fácil utilização empresarial. Ele foi projetado para uso em redes públicas e privadas, além de redes de teste (*e.g.* Sepolia, Görli). O *Besu* tem suporte nativo para tokens fungíveis, *NFTs* e outros padrões, assim como uma gama bastante variada de opções de consenso (*Proof of Stake*, *Proof of Work* e *Proof of Authority*).

Dentre as ferramentas, o *Hyperledger FireFly*, objeto deste capítulo, e o *Hyperledger Cacti* se destacam. O projeto do *Hyperledger FireFly* será melhor detalhado na sub-seção seguinte (1.4.3). Enquanto isso, o *Hyperledger Cacti* é uma plataforma de integração de *blockchain* que permite aos usuários realizar transferências de ativos entre *blockchains* direfentes (*cross-chain*). Em outras palavras, ele permite trocar, por exemplo, um carro representado por um *token* em uma rede Fabric por ETH na rede principal Ethereum. Por fim, dentre as bibliotecas, destacamos o *Hyperledger Aries*, um kit de ferramentas completo para soluções de identidade descentralizada e confiança digital. Ele permite emitir, armazenar e apresentar credenciais verificáveis com o máximo de preservação da privacidade e estabelecer canais de comunicação confidenciais e contínuos.

1.4.3. Conhecendo melhor o FireFly

O *Hyperledger FireFly* é o projeto *Hyperleger* que tem por objetivo principal encapsular diversas camadas de configuração para construção de aplicações *Web3* escaláveis e seguras. A documentação do projeto[6] o define como um *supernó*, *i.e.* uma pilha de aplicações completa. De fato, é possível entender o *FireFly* como uma caixa de ferramentas, na qual os desenvolvedores podem tanto acoplar aplicações já existentes ao mundo *Web3*, quanto

[5]Refere-se ao tempo de escrita do capítulo, Julho de 2024.

[6]Disponível em: `https://hyperledger.github.io/FireFly`

construir aplicações *Web3* do zero. A partir de agora, iremos dar detalhes sobre como o projeto foi concebido e como utilizá-lo para alcançar o seu propósito.

Em outras palavras, o *Hyperledger FireFly* pode ser visto como um *gateway* para um mundo *Web3*, o que inclui todo o ecossistema que sua aplicação deverá participar, *i.e.* múltiplas redes *blockchain*, múltiplas aplicações de finanças descentralizadas, ou mesmo múltiplas redes de negócio. O *Hyperledger FireFly* não é uma nova implementação de uma *blockchain*, na verdade, ele é uma forma de orquestrar todas estas conexões mencionadas, bem como uma camada de dados. De fato, redes *blockchain* públicas, aplicações descentralizadas de armazenamento, padrões e ecossistemas de *tokens*, contratos inteligentes e *frameworks* de identidade são todos exemplos de tecnologias descentralizadas que o *Hyperledger FireFly* permite a integração.

Sem o *Hyperledger FireFly*, o time de desenvolvimento de aplicações teria que construir além da aplicação do negócio, todo o controle de *tokens*, identidade, mensagens, privacidade, gerenciamento de documentos e dados, além da orquestração de dados, do nó da *blockchain* e as conexões com a *Web3*. A proposta do *Hyperledger FireFly* é encapsular tudo isso, para permitir que o time de desenvolvimento se concentre na camada de negócio, deixando que a comunidade possa desenvolver o *Hyperledger FireFly* por meio de um esforço coletivo. De fato, este conceito de *supernó* evoluiu ao longo da última década, com os desenvolvedores percebendo a necessidade de muito mais que a criação de um nó de *blockchain* para que seus projetos fossem bem sucedidos, o que consumiria uma parcela enorme do orçamento disponível para o projeto.

Arquitetura do Hyperledger FireFly

De acordo com os idealizadores do projeto, o supernó *FireFly* foi projetado para ser extensível, com *runtimes* de execução modulares e separadas, orquestradas em uma API intuitiva para desenvolvedores. Uma das principais características do *FireFly* é a modularidade e alta configurabilidade de seus componentes. Para isso, adota-se uma abordagem baseada em microsserviços, que combina pontos de conexão (*plug-points*) no *runtime* principal com extensibilidade de API para *runtimes* remotos, implementados em diversas linguagens de programação.

Figura 1.2 mostra a arquitetura central de um nó FireFly. Ela é composta por três áreas principais: os diferentes *runtimes* que encapsulam o nó, as responsabilidades do *runtime* principal e seus elementos plugáveis, e o código em execução dentro do nó. Um nó é, essencialmente, uma coleção de múltiplos *runtimes* unificados por uma única API HTTPS/WebSocket, exposta pelo *Core*, com um banco de dados privado para armazenar dados locais e conectividade para interagir com outros nós da rede por meio de suas respectivas *runtimes*.

Funcionamento do *Hyperledger FireFly*

O *Hyperledger FireFly* opera em dois modos principais: *Web3 Gateway* e *Multiparty*. Um único runtime pode funcionar em ambos os modos por meio do uso de *namespa-*

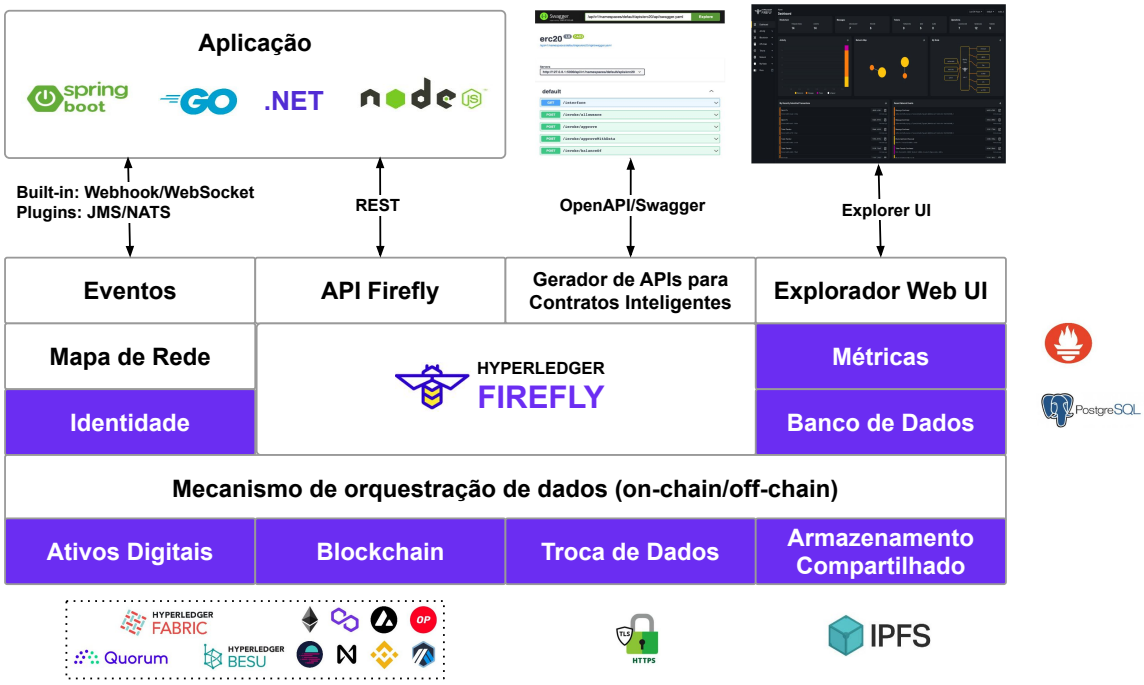

Figura 1.2. Arquitetura do *Hyperledger FireFly*. *(Fonte: Adaptado da documentação)*

ces distintos. O modo *Web3 Gateway* facilita a interação com qualquer aplicação *Web3*, independentemente de outros membros da rede de negócios utilizarem o *FireFly*. As principais funcionalidades incluem transferências de valores *tokenizados*, invocação de contratos inteligentes, indexação de dados da *blockchain*, disparo de eventos para aplicações e sistemas centrais e gerenciamento de dados descentralizados (como *NFTs*).

O modo *Multiparty*, por outro lado, é projetado para a construção de sistemas com várias partes (organizações) onde um *runtime* de aplicação comum é implantado por cada participante empresarial. Esse modo possibilita o desenvolvimento de aplicações sofisticadas que utilizam as APIs modulares do *Hyperledger FireFly* para oferecer soluções empresariais completas. Além de todas as funcionalidades disponíveis no modo *Web3 Gateway*, esse modo também permite o compartilhamento de formatos de dados comuns, troca privada de dados por um barramento criptografado, coordenação de dados *on-chain* e *off-chain*, além de mascaramento de atividades *on-chain* via *hashes* e uso de um livro de endereços compartilhado para gerenciar identidades

Principais componentes principais do *Hyperledger FireFly*

A Figura 1.3 apresenta um esquema das funcionalidades disponibilizadas pelo *Hyperledger FireFly*. Está fora do escopo deste capítulo explorar cada um dos componentes do *Hyperledger FireFly*. Em vez disso, focaremos nos componentes que serão essenciais para o tutorial deste trabalho: as ferramentas que integram o ambiente de desenvolvimento. Mais especificamente, estudaremos o que são e como utilizar: a Interface de Linha de Comando (*FireFly CLI*), o Explorador (*Web UI*), a OpenAPI (*Swagger API UI*) e a *Sandbox UI*. Além dessas ferramentas, a própria API do *FireFly* e o kit de desenvolvimento (SDK)

serão essenciais para a criação de aplicações *Web3*.

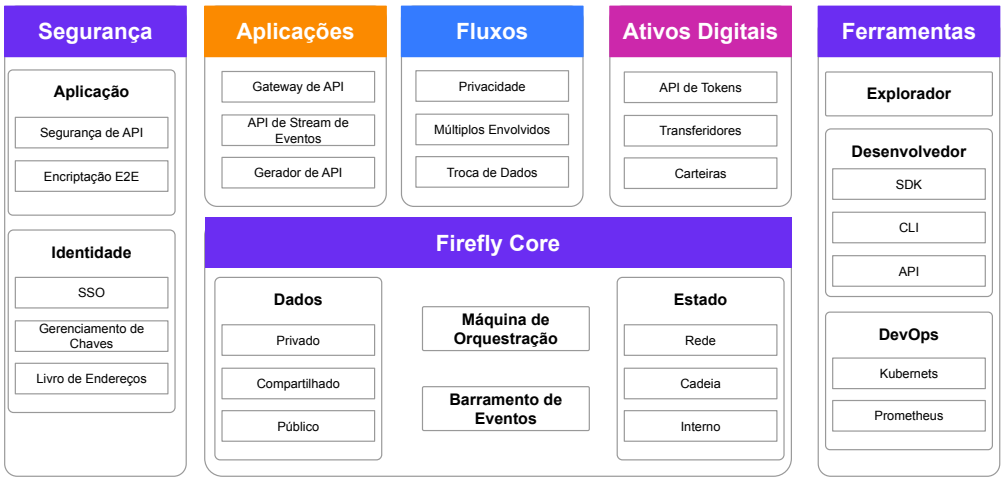

Figura 1.3. Funcionalidades do *Hyperledger FireFly*. *(Fonte: Adaptado da documentação)*

Dado que a maioria das plataformas de *blockchain* oferece a capacidade de executar contratos inteligentes, o *Hyperledger FireFly* suporta a interação com esses contratos por meio de APIs *REST*, além de permitir a escuta de eventos via *WebSocket*. A API garante uma experiência de uso uniforme, independentemente da implementação específica da *blockchain* subjacente. O *Hyperledger FireFly* também possui uma *SDK* cliente para *Node.js*, que permite que aplicações sejam criadas com facilidade.

A CLI do *FireFly* é a ferramenta de desenvolvimento fundamental, a partir da qual as demais serão acessadas. Com a CLI, nós locais do *FireFly* podem ser criados e executados para facilitar o desenvolvimento *offline* de aplicações baseadas em *blockchain*. Isso permite que os desenvolvedores iterem rapidamente em suas ideias sem a necessidade de configurar uma grande infraestrutura antes de escrever a primeira linha de código. Já o Explorador faz parte do próprio *FireFly* Core e oferece uma visão do sistema que permite monitorar o estado atual da rede e investigar transações, mensagens e eventos específicos. Além disso, é uma ferramenta útil para que desenvolvedores visualizem os resultados de suas implementações ao utilizar a API do *FireFly*.

Enquanto isso, o *Swagger* é uma interface interativa gerada pela especificação OpenAPI do *FireFly*, que serve como uma documentação visual e também como ferramenta de teste para a API. Ela permite que desenvolvedores explorem e façam chamadas diretamente na API do FireFly de forma simples e intuitiva. A *Sandbox* do *Hyperledger FireFly* não faz parte do super-nó. Na verdade, é um aplicativo Web de usuário final que usa a API do FireFly. Assim como o *Swagger*, a *Sandbox* também serve ao propósito de orientar os desenvolvedores a usar mais facilmente a API, mas diferente dele, ela tem um backend e um frontend projetados para tal. A *Sandbox* também fornece trechos de código como exemplos de como criar esses recursos em seu próprio aplicativo.

1.5. Desenvolvendo soluções com Hyperledger FireFly

1.5.1. Instalação da FireFly CLI

Para começar a desenvolver aplicações utilizando o *Hyperledger FireFly*, o primeiro passo é a instalação das dependências e da interface de linha de comando (CLI) do FireFly. Há várias maneiras de instalar a CLI do FireFly, porém destacamos apenas a maneira mais fácil, que consiste em fazer o download de um binário pré-compilado (no formato compactado `gzip`) da versão mais recente, cujo download pode ser feito a partir do repositório oficial[7]. Até o momento, estão disponíveis somente versões do binário para os sistemas operacionais Linux e MacOS, mas usuários do sistema Windows podem executá-lo através do módulo Subsistema Windows para Linux (WSL)[8] ou através de máquinas virtuais. A Tabela 1.3 destaca os pré-requisitos necessários para o funcionamento da CLI.

Tabela 1.3. Pré-requisitos FireFly CLI.

Docker	Tecnologia de conteinerização e virtualização.
Docker Compose	A pilha do FireFly será executada em um projeto `docker-compose`.
OpenSSL	Para funções básicas de criptografia e autenticação

Após a instalação dos pré-requisitos e o download do pacote apropriado para seu sistema operacional e arquitetura de CPU, o passo seguinte compreende a extração do binário. Supondo que o pacote do *GitHub* tenha sido baixado em seu diretório *Downloads*, execute o comando presente no Código 1.1 via terminal *bash* para extrair o binário e movê-lo para `/usr/bin/local`. Caso o pacote baixado se encontre em um diretório diferente, será necessário alterar o comando abaixo para incluir o local correto onde o arquivo `FireFly-cli_*.tar.gz` estiver localizado.

```
1 sudo tar -zxf ~/Downloads/FireFly-cli_*.tar.gz -C /usr/local/bin
      ff && rm ~/Downloads/FireFly-cli_*.tar.gz
```

Código 1.1. Extração do binário da FireFly CLI

Um método de instalação alternativo é instalar a CLI via *Go*, caso a máquina possua um ambiente de desenvolvimento local apropriado e a variável `PATH` esteja configurada corretamente. Para ambos os métodos, um tutorial mais detalhado pode ser encontrado na documentação oficial do *Hyperledger FireFly*[9].

1.5.2. Inicialização do ambiente

Uma vez configurada a CLI, a próxima etapa é criar e iniciar uma pilha (*stack*) do *FireFly* para executar um ou vários super-nós. Uma pilha é um conjunto de super-nós com rede e configuração projetados para trabalhar juntos em uma única máquina de desenvolvimento. Dessa forma, no modo *multiparty*, uma pilha tem vários membros (também chamados de

[7] `https://github.com/hyperledger/FireFly-cli/releases/tag/v1.3.0`
[8] `https://learn.microsoft.com/pt-br/windows/wsl/`
[9] `https://hyperledger.github.io/FireFly/latest/gettingstarted/FireFly_cli`

organizações) e cada membro tem seu próprio super-nó dentro da pilha. Isso permite que os desenvolvedores criem e testem fluxos de dados com uma combinação de dados públicos e privados entre várias partes, tudo em um único ambiente de desenvolvimento. Cada membro da pilha possui uma instância de cada ferramenta do *FireFly* (*e.g. Sandbox*, Explorador, *Swagger*), através das quais a API pode ser chamada.

Como mencionado anteriormente, um super-nó é uma estrutura complexa que integra diversos módulos e controladores. Uma das principais vantagens do *Hyperledger FireFly* é trazer facilidade e rapidez na criação desses elementos complexos utilizando apenas uma linha de código. O comando presente no Código 1.2 permite a configuração básica de uma nova pilha, sendo possível ainda, durante este processo, selecionar a *blockchain* que servirá de base (e.g. Ethereum, Fabric ou Tezos), garantindo a integração adequada com a infraestrutura desejada. Ao executar esse trecho, serão solicitados o nome da nova pilha e o número de membros desejados.

```
ff init < ethereum | fabric | tezos >
```

Código 1.2. Criando uma stack FireFly

```
ff init ethereum dev 3 --node-name=pessoa1_node --org-name=
   pessoa1_org --node-name=pessoa2_node --org-name=pessoa2_org
   --node-name=pessoa3_node --org-name=pessoa3_org
```

Código 1.3. Criando uma stack baseada em ethereum com 3 membros

A escolha da rede base depende no geral dos objetivos do projeto. É importante destacar, porém, que a rede Fabric não possui suporte nativo para criação, gerenciamento e transação de *tokens*, embora seja possível anexar módulos específicos para isso, como o Fabric Token SDK[10]. Por razões de praticidade, neste tutorial criaremos, via Docker Compose, uma pilha baseada em Ethereum com 3 membros. Para isso, é crucial garantir a disponibilidade de pelo menos 3 GB de memória *RAM* na máquina. Para sistemas que executam contêineres Docker dentro de uma máquina virtual, como o macOS, é necessário certificar-se de ter alocado memória suficiente.

Ao inicializar uma pilha no FireFly, há diversas opções de customização disponíveis, que permitem ajustar as configurações de acordo com as necessidades do projeto. Por enquanto, utilizaremos apenas argumentos para especificar os nomes dos nós e organizações, para evitar a geração randômica e facilitar a visualização. Nessa inicialização, definimos também automaticamente o nome da pilha como {*dev*} e a quantidade de membros como três (3). O Código 1.3 exibe o comando com os argumentos apropriados. No mais, manteremos os padrões. Para ver a lista completa de opções do *FireFly*, basta executar `ff-help` na interface de linha de comando.

Após a criação da pilha, um arquivo Docker Compose correspondente será gerado, mas ainda será necessário executar mais um comando para iniciar a pilha. Para isso, basta rodar o comando mostrado no Código 1.4. Na primeira vez que o comando é executado, esse processo pode levar alguns minutos, durante os quais a CLI realiza várias etapas automaticamente: faz o *download* das imagens Docker dos componentes do

[10] `https://github.com/hyperledger-labs/fabric-token-sdk`

super-nó; inicializa e configura uma nova *blockchain* dentro de um contêiner; estabelece a comunicação entre os componentes; implanta os contratos inteligentes necessários (*e.g.* *BatchPin* e *token* ERC-1155); e registra as identidades dos membros e nós da rede.

```
ff start dev
```

Código 1.4. Executando a stack *dev*

Depois que sua pilha terminar de iniciar, ela imprimirá os *links* de acesso para o Explorador Web (Web UI), a interface de interação com a API (Swagger API UI) e a Sandbox de cada membro. Caso tudo funcione corretamente, deve ser retornado algo semelhante a 1.5. Atenção: Para os usuários de macOS, a porta padrão (5000) já está em uso pelo serviço *ControlCe (AirPlay Receiver)*. É possível desativar esse serviço ou, alternativamente, criar a pilha usando uma porta diferente, adicionando o argumento {-p 8000} ao comando de criação (1.2 ou 1.3).

```
Web UI for member '0': http://127.0.0.1:5000/ui
Swagger API UI for member '0': http://127.0.0.1:5000/api
Sandbox UI for member '0': http://127.0.0.1:5109

Web UI for member '1': http://127.0.0.1:5001/ui
Swagger API UI for member '1': http://127.0.0.1:5001/api
Sandbox UI for member '1': http://127.0.0.1:5209

Web UI for member '2': http://127.0.0.1:5002/ui
Swagger API UI for member '2': http://127.0.0.1:5002/api
Sandbox UI for member '2': http://127.0.0.1:5309
```

Código 1.5. Links e portas correspondentes a execução local da stack *dev*

1.5.3. Usando a Sandbox

A Sandbox (Figura 1.4) é dividida em três colunas, cada uma com uma função. Na esquerda da página, destacado em vermelho, há três guias que alternam entre os três conjuntos principais de funcionalidade do Sandbox. Nessas guias, você pode criar mensagens, *tokens* e contratos simplesmente preenchendo campos de formulário.

Por razões práticas, esta seção explorará como exemplo apenas a função de enviar mensagens, ou seja, a guia MENSAGENS. É nela que podemos enviar transmissões *broadcasts* e mensagens privadas através da *Blockchain*. Algumas ações a serem exploradas incluem: enviar uma mensagem de *broadcast* e visualizar o conteúdo em todos os membros no Explorador; enviar uma mensagem privada a um único membro e verificar que o conteúdo não está visível para o terceiro membro; ou enviar um arquivo de imagem e realizar o *download* a partir do Explorador de outro membro.

Na coluna do meio, em azul, você poderá visualizar o código que será enviado ao servidor para executar aquela tarefa e, abaixo, ver a resposta. Ao digitar os dados no formulário no lado esquerdo da página, você poderá notar que o código-fonte na parte superior central da página é atualizado automaticamente. Se você estiver criando uma aplicação de *backend*, esse é um exemplo de código que seu aplicativo poderia usar para

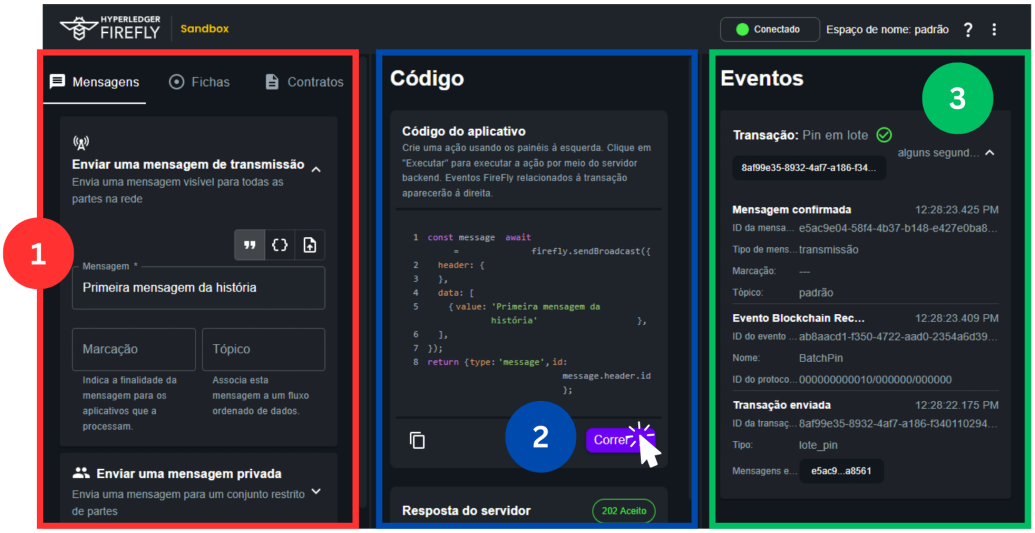

Figura 1.4. Sandbox *Hyperledger FireFly* correspondente ao membro 1 *(Fonte: autor)*

chamar o FireFly SDK. A coluna do meio também contém um botão *RUN* (na imagem, correr) para realmente enviar a solicitação de transação. Na imagem acima, é possível notar que a mensagem em questão é enviada através da função como um JSON no campo data. Ao construir sua própria aplicação, esse recurso pode ajudar na essencial tarefa de se atentar aos dados e formatos adequados a cada função da API.

Finalmente, no lado direito da página, é possível acompanhar o fluxo de eventos recebidos pela conexão *WebSocket* que o *backend* estabeleceu com o *FireFly*. Por exemplo, ao realizar solicitações para envio de mensagens, é possível visualizar a confirmação dessas mensagens em tempo real. Esse mesmo histórico de eventos pode ser visualizado integralmente a partir do Explorador Web. Outros exemplos de uso como a criação e mineração de *tokens* podem ser econtrados na documentação oficial do *FireFly* [11].

1.5.4. Desenvolvimento de um contrato inteligente simples

Nesta sub-seção, descreveremos em etapas como implementar e fazer *deploy* de um contrato inteligente simples em uma *blockchain* Ethereum. Em seguida, exploraremos como o *FireFly* pode ser usado para interagir com o contrato a fim de enviar transações, consultar estados e ouvir eventos. Para esta parte do guia, é requerido que esteja em execução na máquina tanto o Docker quanto uma pilha *FireFly* local com pelo menos dois membros. Detalhamos como uma pilha pode ser executada na sub-seção 1.5.2, Código 1.4.

Criação, compilação e *deploy*

O contrato inteligente que usaremos como exemplo (Código 1.6) é chamado SimpleStorage e foi extraído da documentação oficial do *Hyperledger FireFly*. Como o nome indica, é um contrato muito simples que armazena um número inteiro sem sinal (*unsigned*) de

[11] `https://hyperledger.github.io/FireFly/latest/gettingstarted/sandbox/\#things-to-try-out`

256 *bits* e que emite um evento na *Blockchain* sempre que o valor armazenado é atualizado, permitindo que o valor atual seja recuperado. Este contrato foi escrito na linguagem Solidity e deve ser compilado usando o compilador *solc* ou alguma outra ferramenta. Recomendamos que a compilação seja feita com ajuda de uma IDE online chamada *Remix*, que permite uma rápida configuração de versão do compilador e dispensa mais instalações na máquina, facilitando o processo. Para ambos os casos, após compilar o contrato, um arquivo JSON será retornado e o utilizaremos para solicitar o *deploy*.

```
// SPDX-License-Identifier: Apache-2.0
pragma solidity ^0.8.10;

// Declara o contrato
contract SimpleStorage {
    // Define a variavel que armazenara um valor inteiro sem
       sinal
    uint256 x;

    // Funcao que altera o valor da variavel
    function set(uint256 newValue) public {
        x = newValue;
        emit Changed(msg.sender, newValue);
    }

    // Retorna o valor armazenado na variavel x
    function get() public view returns (uint256) {
        return x;
    }

    event Changed(address indexed from, uint256 value);}
```

Código 1.6. Exemplo de contrato inteligente em Solidity *(Fonte: adaptado da documentação)*

É crucial destacar que as versões mais recentes do compilador `solc` estão apresentando problemas de incompatibilidade com algumas versões da EVM, impossibilitando o *deploy* de alguns contratos por razões que fogem do escopo deste capítulo. Entretanto, por segurança, recomendamos o uso da versão `0.8.19` do compilador e, se possível, da versão paris da EVM. Essas configurações podem ser facilmente ajustadas através do *Remix*, na aba de configurações avançadas.

Uma vez compilado, o próximo passo é fazer *deploy* do contrato utilizando a API de implantação de contrato integrada do *Hyperledger FireFly*. Existem duas formas de fazer isso, a primeira delas utilizando o comando apresentado no Código 1.7 diretamente na CLI do *FireFly*. Assim, caso tudo ocorra corretamente, o endereço de *blockchain* do seu contrato será impresso. Esse endereço será essencial em seguida para registrar a API do contrato e é ele que nos referiremos nas sub-seções seguintes.

```
ff deploy ethereum dev simple_storage.json
```

Código 1.7. Deploy de contrato inteligente através da CLI

A segunda abordagem, que consideramos mais simples, envolve o uso da interface *Swagger*, especificamente a rota `/postContractDeploy` (Figura 1.5). Nesse método, enviamos ao servidor uma requisição com base em campos específicos do JSON, que incluem: *i)* o *bytecode* compilado do contrato inteligente, inserido no campo *"contract"* como uma cadeia de caracteres codificada em hexadecimal ou Base64; *ii)* o *array* ABI completo, que deve ser inteiramente copiado para o campo *"definition"*. Além disso, se aplicável, é necessário fornecer uma lista ordenada dos argumentos do contato. Devido a simplicidade do exemplo, basta definir o corpo da solicitação substituindo as seções apropriadas pela saída resultante da compilação do *Remix*, de acordo com o Código 1.8.

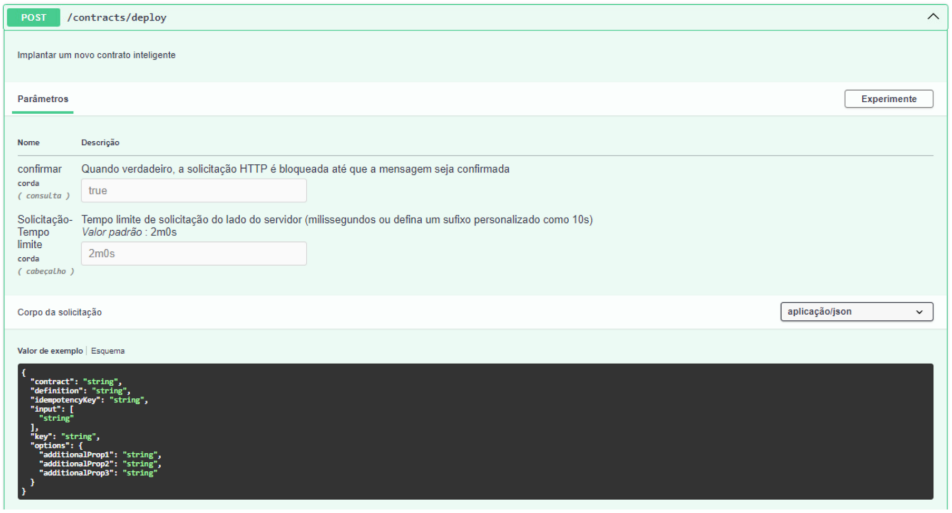

Figura 1.5. Rota para *deploy* de contrato com o *Swagger* do *FireFly*. *(Fonte: autor)*

```
{
"contract": "INSIRA_BYTECODE_AQUI",
  "definition": INSIRA_ABI_AQUI,
  "input": []
}
```

Código 1.8. Esquema de requisição para deploy de contrato

Se a requisição for bem-sucedida, estará confirmada a transação e uma resposta detalhada será gerada. Ao rolar para baixo, na parte inferior, o endereço do contrato implantado estará disponível no campo *"address"*. Um evento correspondente ao *deploy* também será registrado na *blockchain* e será possível examiná-lo através da interface do *Explorer*. Para obter informações detalhadas sobre o bloco, que serão necessárias em uma etapa posterior deste guia, basta acessar o evento e verificá-las.

Gerando uma interface e uma API HTTP para o contrato inteligente

Fazer *deploy* de um contrato armazena-o na *blockchain*, mas não disponibiliza nenhuma interação com ele aos demais membros da rede. Assim, para que esses possam referenciá-lo, é preciso criar e compartilhar uma descrição do contrato, de suas funções, tipos de

entrada e retornos, ou seja, uma Interface de Contrato (FFI). A API do *Hyperledger FireFly* possui uma rota específica para isso, cujo uso é preferível à escrita manual da FFI. A partir deste passo, todos os passos seguintes que configuram chamadas à API podem ser executados tanto através do Swagger quanto da Sandbox. Novamente, por questões de praticidade, descreveremos apenas a segunda maneira, deixando como recomendação a leitura da documentação para demais instruções.

Na Sandbox, acesse a guia "contratos" à esqueda da página e navegue até o formulário "Definir Interface de Contrato", selecione a opção Solidity ABI, dê um nome e uma versão à sua interface preenchendo os campos apropriados. Cole no campo Esquema (*schema*) a mesma ABI do contrato que usamos no passo anterior e, por fim, garanta que a *checkbox "Publish"* esteja ativada, para que a interface seja automaticamente publicada à rede. Execute o código e visualize a confirmação do evento na aba à direita. De volta à guia de contratos, navegue até o formulário logo a baixo do anterior para enfim gerar uma API HTTP para interagir com o contrato. Neste passo, será necessário selecionar a interface desejada, nomear o contrato e indicar o endereço dele na *blockchain* (vide subseção anterior). Por fim, indique que a API deve ser publicada automaticamente na rede, execute o comando e verifique o registro do evento.

Agora, o contrato deve estar acessível a todos os membros da pilha e portanto é possível verificar a API e obter sua interface OpenAPI interativa por meio do *Explorador*. Utilizando as rotas `post/invoke`, já é possível interagir com o contrato.

1.5.5. Integração do contrato inteligente na aplicação *Web3*

Finalmente, poderemos criar nossas próprias aplicações *Web3* utilizando a API que criamos para interagir com o contrato desenvolvido. Para isso, o *Hyperledger FireFly* disponibiliza um Kit de Desenvolvimento de Software (SDK) em `Node.js` para criação de projetos utilizando a linguagem *typescript*. A Tabela 1.4 resume algumas das principais funções disponíveis para manipulação da API do FireFly, que permitem que o desenvolvedor realize todas as instruções anteriores deste guia diretamente de sua própria aplicação *backend*, com maior grau de controle e personalização.

Tabela 1.4. Principais funções da FireFly Node.js SDK. Fonte: `https://github.com/hyperledger/FireFly-sdk-nodejs/tree/main`

Função	Uso
`sendBroadcast`	Envia mensagens em transmissão *Broadcast*
`createTokenPool`	Cria um pool de *token* personalizado
`deployContract`	Possibilita o *deploy* de contrato inteligente
`invokeContractAPI`	Invoca a API de um contrato inteligente

Além disso, a função `invokeContractAPI` possibilita a interação com contratos inteligentes customizados. A partir daqui, o leitor já tem as ferramentas para começar a construir sua primeira aplicação *Web3* completa, desde o *setup* da infraestrutura, o deploy de contratos inteligentes e a criação de APIs para interação com seu *backend*, tudo isso facilitado pelo *Hyperledger FireFly* e suas ferramentas. A próxima seção indicará os

possíveis próximos passos para escolha ou contrução de um estudo de caso que inspire novos desenvolvedores a construir soluções baseadas em *blockchain* utilizando o FireFly.

1.6. Próximos passos

1.6.1. Construindo o seu caso de uso

Antes de tudo, a equipe de desenvolvimento precisa *(Passo 0)* decidir se o uso de *blockchain* é mesmo necessário. Como apresentado, decidir usar *blockchain* sem a real necessidade pode implicar no aumento desnecessário de complexidade e custo para o seu projeto. Em caso afirmativo, o próximo passo seria *(Passo 1)* definir o tipo de rede *blockchain* a ser utilizada, seja permissionada ou não, pública ou privada. Após esta decisão, é necessário *(Passo 2)* selecionar entre as opções que estiverem disponíveis a rede *blockchain* a ser utilizada. No momento de escrita deste capítulo, a rede privada permissionada mais utilizada é sem dúvidas a *Hyperledger Fabric*, enquanto a pública não-permissionada é *Ethereum*. É posssível usar ambas na construção de *aplicações descentralizadas* com o *Hyperledger FireFly*, como visto anteriormente. Por fim, o time deve *(Passo 3)* projetar os casos de uso de sua aplicação que terão dados persistidos em *blockchain – on-chain –* e aqueles que não – *off-chain*.

1.6.2. Casos de uso existentes para inspiração

A expansão da tecnologia *Blockchain* para além de suas aplicações iniciais no setor financeiro criou uma ampla gama de oportunidades em diversos setores. Embora aproximadamente 30% das soluções que utilizam *blockchain* atualmente no mercado continuem focadas em finanças [Zīle and Strazdiņa 2018], o potencial da tecnologia vai muito além desse setor. Zile e Strazdina, em um survey de 2018, exploram esse cenário mais amplo, apresentando uma variedade de soluções propostas e avaliando sua viabilidade. Isso inclui o uso da *blockchain* para aumentar a transparência e a segurança de ativos, produtos agroalimentares, dados e outros, demonstrando como a *Blockchain* está influenciando os setores ao melhorar a segurança, a eficiência e a confiança. Nesta seção, exploraremos alguns desses casos de uso da tecnologia

Transferência e rastreabilidade de propriedade

Por suas características de segurança e auditabilidade, a tecnologia *Blockchain* tem encontrado bom uso na transferência e rastreabilidade de propriedades de artigos de luxo. Nesse contexto, a Everledger teve inicio com a proposta de rastrear diamantes para garantir que não sejam provenientes de zonas de conflito. Construída sobre o *Hyperledger Fabric*, a solução da empresa combina *blockchain*, IA, IoT e nanotecnologia para criar um gêmeo digital seguro de cada diamante, detalhando sua procedência, caracterização e quem o possui. Além disso, a Everledger busca eliminar conflitos na produção dos diamantes e melhorar os direitos dos trabalhadores, promovendo a sustentabilidade, melhorando a rastreabilidade e a transparência, e a aumentando a confiança no setor de pedras preciosas por meio de uma avançada tecnologia de rastreamento de ativos.

Enquanto isso, Santos [Santos 2021] apresentou alternativas voltadas à compensação justa de pesquisadores por seus trabalhos, além de propostas para ajudar na adaptação

do processo de avaliação de artigos científicos por meio dos princípios de ciência aberta. Por outro lado, a Orvium está integrando tecnologia *Blockchain*, contratos inteligentes, computação em nuvem, análises de *big data* e aprendizado de máquina na busca por transformar o cenário da publicação científica e acadêmica. O uso do registro imutável da *blockchain* garante que todas as transações, como a publicação ou compra de uma publicação, sejam registradas de forma segura e não possam ser alteradas, proporcionando proteção de direitos autorais e reduzindo problemas de pirataria digital. À medida que a publicação digital evolui, a Orvium é um exemplo de como a *blockchain* pode aprimorar a segurança, a verificação de propriedade e a colaboração dentro da indústria, abrindo caminho para novos padrões na área.

Transferência e gerenciamento de dados

Já quanto a transferência e armazenamento de dados, tanto a Storj quanto a Filecoin são plataformas que utilizam a tecnologia *Blockchain* para melhorar o armazenamento de dados, oferecendo alternativas descentralizadas, seguras e eficientes aos serviços tradicionais de nuvem. Ambas as plataformas permitem que os usuários monetizem o espaço de armazenamento não utilizado em seus computadores focando em garantir a integridade e disponibilidade dos dados por meio de criptografia. Ao oferecer maior privacidade e redução de custos, essas plataformas propõem ampliar as possibilidades de armazenamento de dados e também democratizar o acesso e o controle sobre a informação.

Além disso, na área de gestão de dados de identidade, o *IBM Digital Credentials* é um sistema de identidade descentralizado e projetado para melhorar a troca segura de informações e dar a indivíduos e organizações controle sobre sua identidade e dados. Por meio da colaboração com líderes da indústria, como a *Decentralized Identity Foundation* e as *W3C Verifiable Credentials*, a IBM propõe garantir um alto nível de confiança e privacidade nas interações digitais. A plataforma, baseada em tecnologia Hyperledger, é construída em código aberto com foco em escalabilidade, segurança e interoperabilidade, e permite que as organizações adotem rapidamente processos de identidade e credenciamento descentralizados.

Rastreabilidade de cadeia de suprimentos agrícolas

O Walmart, gigante do varejo, se deparou com o desafio de rastrear rapidamente a origem de alimentos durante surtos de doenças transmitidas por alimentos, um processo que poderia levar semanas. Em busca de uma solução, a empresa identificou o potencial da tecnologia *blockchain* para melhorar a rastreabilidade no complexo ecossistema da cadeia de fornecimento de alimentos. Em parceria com a IBM, o Walmart desenvolveu um sistema de rastreabilidade baseado em Hyperledger Fabric e testou sua eficácia com dois projetos-piloto: um para rastrear mangas nos EUA e outro para rastrear carne suína na China. O sistema provou ser eficiente, reduzindo o tempo de rastreamento das mangas de 7 dias para apenas 2,2 segundos e possibilitando o registro de certificados de autenticidade para carne suína, aumentando a confiança no processo. Atualmente, o Walmart rastreia mais de 25 produtos de diferentes fornecedores e planeja expandir a aplicação da solução

para outras categorias, exigindo que fornecedores de verduras frescas também adotem o sistema.

Outros

No contexto de sistemas de saúde, a *Blockchain* pode ajudar a contornar o desafio de manter históricos médicos confiáveis e, ao mesmo tempo, proteger dados sensíveis dos pacientes e prestadores de serviço. Nesse âmbito, o *Synaptic Health Alliance* é uma coalizão de líderes de empresas fornecedoras de planos de saúde dos EUA (incluindo importantes representantes da Humana, Multiplan, UnitedHealth, entre outros), que utiliza o *Hyperledger FireFly* integrado ao Quorum para manter de forma segura uma base de dados descentralizada baseada em *blockchain*, visando abordar o desafio de manter dados precisos com segurança, privacidade e eficiência.

Além disso, a tecnologia *Blockchain* oferece uma série de benefícios para a Internet das Coisas (IoT), principalmente em termos de segurança, transparência e descentralização. [Chowdhury et al. 2020] revisaram alguns dos benefícios da tecnologia à área e apresentaram alguns casos de uso em que a adoção tecnologia é uma solução promissora. Segundo eles, essa união permitiria comunicação segura e direta entre dispositivos (M2M). Além disso, com contratos inteligentes, há mais eficiência e redução de custos. Casos de uso incluem a gestão de cadeias de suprimentos, onde a rastreabilidade dos produtos é melhorada; a automação em casas inteligentes; e a segurança em cidades inteligentes, com monitoramento descentralizado e em tempo real.

No âmbito industrial, a *TradeGo* é um consorcio internacional de transporte de mercadorias que através de um produto chamado *Digital Presentation* permite o compartilhamento eficiente e transparente de documentos comerciais originais e confidenciais em formato digital, agilizando operações comerciais. A adoção de *Blockchain* no intermédio dessas transações reduz e pode até eliminar o tempo gasto na transferência de documentos em papel, além de eliminar custos e desburocratizar todo o processo.

1.7. Considerações finais

Neste capítulo buscamos levar o leitor em uma jornada desde os conceitos fundamentais da tecnologia *blockchain* até a construção de suas primeiras aplicações *Web3* com o *Hyperledger FireFly*. Entendemos que a tecnologia é complexa e que existe um caminho longo até a compreensão de muitos dos detalhes aqui apresentados. Além disso, consideramos que a escassez de materiais didáticos escritos na língua portuguesa constitui uma barreira para o ingresso de desenvolvedores e pesquisadores brasileiros à área. Por esses motivos, a intenção deste capítulo foi diminuir tanto a barreira linguística, quanto da dispersão de conteúdo na Internet, que também pode afastar muitos dos interessados. Esperamos que este conteúdo ajude e motive estudantes nesta caminhada em direção a uma carreira de desenvolvedor de aplicações *Blockchain*.

O(a) leitor(a) pode dar continuidade aos estudos utilizando-se dos trabalhos aqui referenciados e, claro, da documentação do *Hyperledger FireFly* e das redes *Blockchain* escolhidas para sua aplicação. Entendemos que a documentação do *Hyperledger FireFly* é uma fonte de informações bastante rica, que contém detalhes sobre o seu funcionamento,

os quais não foram abordados aqui por limitação de espaço. Nesta mesma documentação, também existe bastante conteúdo sobre as redes *blockchain* mencionadas neste capítulo (*Hyperledger Fabric* e *Ethereum*), assim como exercícios interessantes. Além disso, acreditamos que existem diversas oportunidades de aprendizado com outras redes não trabalhadas aqui, tais como *Internet Computer*, *Cardano*, *Polygon*, entre outras. Independente do caminho que você escolher traçar, desejamos sucesso.

Referências

[Baran 1964] Baran, P. (1964). On distributed communications networks. *IEEE Transactions on Communications Systems*, 12(1):1–9.

[Buterin 2013] Buterin, V. (2013). Ethereum white paper: A next generation smart contract & decentralized application platform.

[Chaum 1983] Chaum, D. (1983). Blind signatures for untraceable payments. In Chaum, D., Rivest, R. L., e Sherman, A. T., editors, *Advances in Cryptology*, pages 199–203, Boston, MA. Springer US.

[Chowdhury et al. 2020] Chowdhury, M. J. M., Ferdous, M. S., Biswas, K., Chowdhury, N., e Muthukkumarasamy, V. (2020). A survey on blockchain-based platforms for iot use-cases. *The Knowledge Engineering Review*, 35:e19.

[Dwork and Naor 1993] Dwork, C. e Naor, M. (1993). Pricing via processing or combatting junk mail. In Brickell, E. F., editor, *Advances in Cryptology — CRYPTO' 92*, pages 139–147, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Enoch et al. 2023] Enoch, J. D., Omijeh, B. O., e Okeke, R. O. (2023). Comparative analysis of blockchain and database. *European Journal of Science, Innovation and Technology*, 3(3):454–463.

[Foundation 2016] Foundation, H. (2016). Open source blockchain effort for the enterprise elects leadership positions and gains new investments. `https://www.hyperledger.org/announcements/2016/03/29/open-source-blockchain-effort-for-the-enterprise-elects-leadership-positions-and-gains-new-investments`. Accessed June 2024.

[Foundation 2024] Foundation, H. (2024). An overview of hyperledger foundation. `https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Offers/HL_Whitepaper_IntroductiontoHyperledger.pdf`. Accessed: 18 May 2024.

[Greve et al. 2018] Greve, F., Sampaio, L., Abijaude, J., Coutinho, A. A., Brito, I., e Queiroz, S. (2018). *Blockchain e a Revolução do Consenso sob Demanda*, page v. 30. SBC.

[Haber and Stornetta 1991] Haber, S. e Stornetta, W. S. (1991). How to time-stamp a digital document. *J. Cryptology*, 3:99–111.

[Hassija et al. 2021] Hassija, V., Zeadally, S., Jain, I., Tahiliani, A., Chamola, V., e Gupta, S. (2021). Framework for determining the suitability of blockchain: Criteria and issues to consider. *Transactions on Emerging Telecommunications Technologies*, 32(10):e4334.

[ISO Central Secretary 2024] ISO Central Secretary (2024). Blockchain and distributed ledger technologies — vocabulary. Standard ISO 22739:2024, International Organization for Standardization, Geneva, CH.

[Kakavand et al. 2017] Kakavand, H., Sevres, N. K. D., e Chilton, B. (2017). The blockchain revolution: An analysis of regulation and technology related to distributed ledger technologies. *SSRN*. Disponível em SSRN: https://ssrn.com/abstract=2849251 ou http://dx.doi.org/10.2139/ssrn.2849251.

[Nakamoto 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`. Accessed: 2021-Jun-09.

[Narayanan and Clark 2017] Narayanan, A. e Clark, J. (2017). Bitcoin's academic pedigree: The concept of cryptocurrencies is built from forgotten ideas in research literature. *Queue*, 15(4):20–49.

[Santos 2021] Santos, A. R. (2021). Open scientist in the wonderland: advocating for blockchain-based decentralized applications for science. In *Proceedings of the 1st Workshop on Open Science Practices for Software Engineering*, pages 1–3, Porto Alegre, RS, BRA. SBC OpenLib.

[Szabo 2008] Szabo, N. (2008). Bit gold. `https://unenumerated.blogspot.com/2005/12/bit-gold.html`. Accesso em: 02 jul. 2024.

[The Economist 2015] The Economist (2015). The trust machine. `https://www.economist.com/leaders/2015/10/31/the-trust-machine`. Accessed: 2024-05-14.

[Wood 2019] Wood, G. (2019). Ethereum: A secure decentralised generalised transaction ledger.

[Wust and Gervais 2018] Wust, K. e Gervais, A. (2018). Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE.

[Zheng et al. 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.

[Zīle and Strazdiņa 2018] Zīle, K. e Strazdiņa, R. (2018). Blockchain use cases and their feasibility. *Applied Computer Systems*, 23(1):12–20.