**Capítulo 14: Hyperledger**

Hyperledger não é uma blockchain, mas um projeto que foi iniciado pela Linux Foundation em dezembro de 2015 para avançar a tecnologia de blockchain. Este projeto é um esforço colaborativo de seus membros para construir uma estrutura de razão distribuída de código aberto que pode ser usada para desenvolver e implementar aplicações e sistemas blockchain entre setores. O foco principal é criar e executar plataformas que suportem transações comerciais globais. O projeto também se concentra em melhorar a confiabilidade e o desempenho dos sistemas de blockchain.

Este capítulo, em sua maior parte, discutirá os projetos Hyperledger, e suas várias funcionalidades e componentes. Como o Hyperledger é tão vasto, não é possível cobrir todos os projetos em detalhes neste curto capítulo. Assim, vamos apresentar uma variedade de projetos disponíveis, e então fornecer discussões detalhadas sobre o Hyperledger Fabric. Os projetos sob o guarda-chuva do Hyperledger passam por múltiplos estágios de desenvolvimento, indo de proposta para incubação, e eventualmente graduando-se para um estado ativo. Projetos também podem ser descontinuados ou colocados em estado de fim de vida, onde não são mais ativamente desenvolvidos. Para que um projeto possa passar para o estágio de incubação, ele deve ter uma base de código totalmente funcional juntamente com uma comunidade ativa de desenvolvedores. O projeto Hyperledger atualmente tem mais de 300 organizações membros e é muito ativo, com muitos colaboradores, encontros e palestras organizadas ao redor do mundo.

Primeiramente, vamos cobrir alguns dos vários projetos sob o Hyperledger. Em seguida, passaremos a examinar o design e a arquitetura do Hyperledger Fabric com mais detalhes, cobrindo os seguintes tópicos ao longo do caminho:

* Projetos sob o Hyperledger
* Arquitetura de referência do Hyperledger
* Hyperledger Fabric
* Fabric 2.0

Os projetos sob o Hyperledger se enquadram em diferentes categorias, e começaremos com a introdução dessas categorias na próxima seção.

**Projetos sob o Hyperledger**

Há quatro categorias de projetos sob o Hyperledger. Em cada categoria, há múltiplos projetos. As categorias são:

* Razões distribuídas
* Bibliotecas
* Ferramentas
* Específicos de domínio

Vamos examinar cinco projetos de razão distribuída sob o guarda-chuva do Hyperledger: Fabric, Sawtooth, Iroha, Indy e Besu. Sob bibliotecas, exploraremos os projetos Aries, Transact, Ursa e AnonCreds. Na categoria de ferramentas, veremos projetos como Cello, Caliper, junto com um projeto específico de domínio chamado Hyperledger Grid. Existem vários outros projetos em andamento (eles podem ser revisados em <https://www.hyperledger.org>), mas para brevidade, incluiremos apenas os mais relevantes e interessantes aqui.

Observe também que esse panorama continua mudando devido a desenvolvimentos e melhorias, que você pode acompanhar aqui: <https://landscape.hyperledger.org/>

Segue uma breve introdução a todos esses projetos.

**Razões distribuídas**

Razões distribuídas, como vimos no Capítulo 1, Blockchain 101, são uma categoria ampla de bancos de dados distribuídos que são descentralizados, compartilhados entre múltiplos participantes, e atualizáveis apenas por meio de consenso.

Frequentemente, os termos blockchain e razão distribuída são usados de forma intercambiável. No entanto, uma diferença fundamental entre razões distribuídas e blockchains é que se espera que as blockchains tenham uma arquitetura em que as transações sejam tratadas em lotes de blocos, enquanto razões distribuídas não possuem tais exigências. Todos os outros atributos das razões distribuídas e blockchains são amplamente os mesmos. No Hyperledger, razão distribuída é um termo genérico usado para denotar tanto blockchains quanto razões distribuídas.

**Fabric**

Hyperledger Fabric é um projeto blockchain que foi proposto pela IBM e Digital Asset Holdings (DAH). É uma estrutura de razão distribuída permissionada de nível empresarial para o desenvolvimento de soluções e aplicações blockchain. O Fabric é baseado em uma arquitetura modular e plugável. Isso significa que vários componentes, como o mecanismo de consenso e os serviços de membros, podem ser conectados ao sistema conforme necessário. Atualmente, seu status é ativo e é o primeiro projeto a se graduar da incubação para o estado ativo. Iremos abordar o Fabric em mais detalhes posteriormente neste capítulo.

O código-fonte está disponível em: <https://github.com/hyperledger/fabric>

**Sawtooth**

Hyperledger Sawtooth é um projeto blockchain proposto pela Intel em abril de 2016. Foi doado para a Linux Foundation em 2016. Sawtooth introduziu algumas inovações inéditas, com foco na separação entre razão e transações, uso flexível em múltiplas áreas de negócio por meio de famílias de transações, e consenso plugável. É uma blockchain de nível empresarial com foco em privacidade, segurança e escalabilidade.

A separação da razão pode ser explicada mais precisamente ao se dizer que as transações são desacopladas da camada de consenso, utilizando um novo conceito chamado famílias de transações. Em vez de as transações serem acopladas individualmente com a razão, famílias de transações são usadas, o que permite mais flexibilidade, semântica rica e design aberto da lógica de negócio. As transações seguem os padrões e estruturas definidos nas famílias de transações.

Uma das características inovadoras que a Intel introduziu com o Hyperledger Sawtooth é um novo algoritmo de consenso chamado Proof of Elapsed Time (PoET). Ele faz uso do Trusted Execution Environment (TEE) fornecido pelas Intel Software Guard Extensions (Intel SGX) para fornecer um processo seguro e aleatório de eleição de líder. Ele suporta configurações tanto permissionadas quanto não permissionadas.

**Iroha**

Iroha foi contribuída pela Soramitsu, Hitachi, NTT Data e Colu em setembro de 2016. O objetivo da Iroha é construir uma biblioteca de componentes reutilizáveis que os usuários possam escolher para rodar em suas próprias razões distribuídas baseadas no Hyperledger.

O objetivo principal da Iroha é complementar outros projetos Hyperledger fornecendo componentes reutilizáveis escritos em C++ com ênfase em desenvolvimento móvel. Este projeto também propôs um novo algoritmo de consenso chamado Sumeragi, que é um algoritmo de consenso tolerante a falhas bizantinas baseado em cadeia.

Diversas bibliotecas foram propostas e estão sendo desenvolvidas pela Iroha, incluindo, mas não se limitando a, uma biblioteca de assinatura digital (ed25519), uma biblioteca de hash SHA-3, uma biblioteca de serialização de transações, uma biblioteca P2P, uma biblioteca de servidor de API, uma biblioteca para iOS, uma biblioteca para Android e uma biblioteca JavaScript.

Este projeto está disponível em: <https://github.com/hyperledger/iroha>

**Indy**

Este projeto tem status graduado sob o Hyperledger. Indy é uma razão distribuída desenvolvida para construir identidades descentralizadas. Ele fornece ferramentas, bibliotecas utilitárias e módulos, que podem ser usados para construir identidades digitais baseadas em blockchain. Essas identidades podem ser usadas em múltiplas blockchains, domínios e aplicações. O Indy possui sua própria razão distribuída e usa o Redundant Byzantine Fault Tolerance (RBFT) para consenso.

**Besu**

Besu é um cliente Ethereum baseado em Java. É o primeiro projeto sob o Hyperledger que pode operar em uma rede Ethereum pública.

Com isso, concluímos uma introdução aos projetos de razão distribuída sob o Hyperledger. Agora vejamos a próxima categoria, bibliotecas.

**Bibliotecas**

Várias bibliotecas estão atualmente disponíveis sob o projeto Hyperledger. Esta categoria inclui projetos que visam apoiar o ecossistema blockchain introduzindo plataformas para interoperabilidade, identidade, criptografia e ferramentas para desenvolvedores. Agora descreveremos brevemente cada uma delas.

**Aries**

Aries não é uma blockchain; na verdade, é uma infraestrutura para interações ponto a ponto (P2P) com raízes em blockchain. O objetivo deste projeto é fornecer código para interações P2P, gerenciamento de segredos, troca de informações verificáveis (como credenciais verificáveis), interoperabilidade e mensagens seguras para sistemas descentralizados. O objetivo final deste projeto é fornecer um conjunto dinâmico de capacidades para armazenar e trocar dados relacionados à identidade baseada em blockchain.

O código-fonte está disponível em: <https://github.com/hyperledger/aries>

**Transact**

Transact fornece uma biblioteca compartilhada que lida com a execução de contratos inteligentes. Esta biblioteca facilita o desenvolvimento de software de razão distribuída permitindo o gerenciamento de agendamento, despacho de transações e gerenciamento de estado por meio de uma biblioteca de software compartilhada. Ela fornece uma abordagem para implementar novas linguagens de desenvolvimento de contratos inteligentes chamadas engines de contrato inteligente. As engines de contrato inteligente implementam máquinas virtuais ou interpretadores para o código do contrato inteligente. Dois exemplos principais de tais engines são Seth (para contratos inteligentes EVM) e Sabre (para contratos inteligentes baseados em WebAssembly).

Transact está disponível em: <https://crates.io/crates/transact>

**Ursa**

Ursa é uma biblioteca de criptografia compartilhada que pode ser usada por qualquer projeto, seja Hyperledger ou não, para fornecer funcionalidade criptográfica. Ursa fornece uma interface de biblioteca invocável em C e um crate (biblioteca) em Rust. Há duas sub-bibliotecas disponíveis em Ursa: LibUrsa e LibZmix. LibUrsa é projetada para primitivas criptográficas, como assinaturas digitais, esquemas de criptografia padrão e esquemas de troca de chaves. LibZmix fornece um método genérico para produzir provas de conhecimento nulo (zero-knowledge proofs). Ela suporta provas de conhecimento de assinatura (PoK), bullet proofs, range proofs e pertencimento a conjuntos.

Ursa está disponível em: <https://github.com/hyperledger/ursa>

**AnonCreds**

AnonCreds significa Credenciais Anônimas. Este é outro projeto muito interessante em incubação no Hyperledger. Trata-se de um tipo de credencial verificável que permite a proteção da privacidade com base em provas de conhecimento nulo.

Agora cobrimos três projetos de biblioteca atualmente disponíveis sob o Hyperledger. Na próxima seção, exploraremos diferentes ferramentas que podem ser usadas em projetos Hyperledger.

**Ferramentas**

Há vários projetos na categoria de ferramentas no Hyperledger. As ferramentas sob o Hyperledger concentram-se em fornecer utilitários e ferramentas de software que ajudam a melhorar a experiência do usuário. Por exemplo, ferramentas de visualização como exploradores de blockchain, ferramentas de implantação e ferramentas de benchmarking se enquadram nessa categoria. Descreveremos algumas dessas brevemente.

**Cello**

O objetivo por trás do Cello é permitir a implantação fácil de blockchains. Isso fornecerá uma capacidade de implantação de blockchain como serviço. Atualmente, este projeto está em estágio de incubação.

O código-fonte do Cello está disponível em: <https://github.com/hyperledger/cello>

**Caliper**

Caliper é uma estrutura de benchmarking para blockchains. Pode ser usada para medir o desempenho de qualquer blockchain. Há diferentes indicadores de desempenho suportados na estrutura. Estes incluem taxa de sucesso, taxa de leitura de transações, taxa de transferência (throughput), latência e consumo de recursos de hardware, como CPU, memória e E/S. Ethereum, Fabric, Sawtooth, Burrow e Iroha são atualmente os blockchains suportados no Caliper.

Caliper está disponível em: <https://github.com/hyperledger/caliper>

Com isso, concluímos a introdução à categoria de ferramentas do projeto Hyperledger. A seguir, apresentaremos projetos específicos de domínio.

**Específicos de domínio**

Sob o Hyperledger, também existem alguns projetos específicos de domínio que são criados para atender a requisitos específicos. O projeto que exploraremos aqui é chamado Grid.

**Grid**

Grid é um projeto Hyperledger que fornece uma implementação de referência padrão de tipos de dados relacionados à cadeia de suprimentos, modelos de dados e lógica de negócios relevante abrangendo contratos inteligentes. Atualmente, está em estágio de incubação.

O código do Grid está disponível em: <https://github.com/hyperledger/grid>

Cada um dos projetos está em vários estágios de desenvolvimento. Espera-se que essa lista cresça à medida que mais e mais membros se juntem ao projeto Hyperledger e contribuam para o desenvolvimento da tecnologia blockchain.

Na próxima seção, examinaremos a arquitetura de referência do Hyperledger, que fornece princípios gerais e filosofias de design que podem ser seguidas para construir novos projetos Hyperledger.

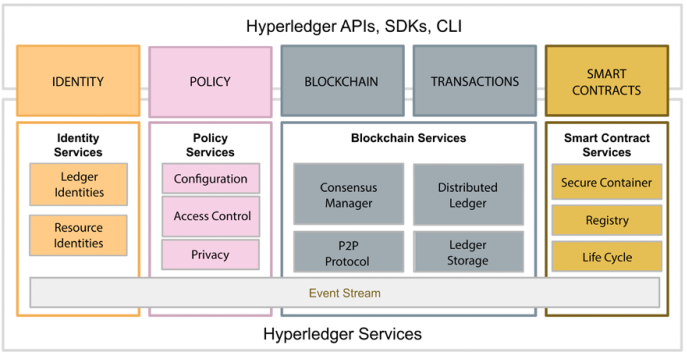
**Arquitetura de referência do Hyperledger**

O Hyperledger tem como objetivo construir novas plataformas de blockchain que sejam impulsionadas por casos de uso da indústria. Como houve muitas contribuições feitas ao projeto Hyperledger pela comunidade, a plataforma blockchain Hyperledger está evoluindo para um protocolo para transações comerciais. O Hyperledger também está evoluindo para uma especificação que pode ser usada como referência para construir plataformas blockchain, em comparação com soluções de blockchain anteriores que abordavam apenas um tipo específico de indústria ou requisito.

Nesta seção, é apresentado um modelo de arquitetura de referência que foi publicado pelo projeto Hyperledger. Essa arquitetura pode ser usada por um desenvolvedor blockchain para construir uma blockchain que esteja em conformidade com as especificações da arquitetura Hyperledger.

O Hyperledger publicou um whitepaper que apresenta um modelo de arquitetura de referência que pode servir como diretriz para construir razões distribuídas permissionadas. A arquitetura de referência consiste em vários componentes que formam uma blockchain empresarial.

Esses componentes de alto nível são mostrados no diagrama de arquitetura de referência abaixo, que foi extraído do whitepaper:

****

*Figura 14.1: Arquitetura de referência*

No diagrama anterior, começando da esquerda, vemos que temos cinco componentes de nível superior que fornecem vários serviços. O primeiro é identidade, que fornece serviços de autorização, identificação e autenticação sob os serviços de associação (membership services). Em seguida, temos o componente de política, que fornece serviços de política.

Vários whitepapers relacionados à criação e design do Hyperledger estão disponíveis aqui:  
<https://www.hyperledger.org/learn/white-papers>

Após isso, vemos blockchain e transações, que consistem no Gerenciador de Consenso, Razão Distribuída, o Protocolo P2P de rede e serviços de Armazenamento da Razão. O gerenciador de consenso garante que a razão só seja atualizável por meio de consenso entre os participantes da rede blockchain.

Finalmente, temos a camada de contratos inteligentes, que fornece serviços de chaincode no Hyperledger e faz uso de tecnologia de Contêiner Seguro para hospedar contratos inteligentes. Chaincode pode ser considerado o equivalente ao contrato inteligente no Hyperledger.

Veremos todos esses com mais detalhes na seção sobre Hyperledger Fabric em breve.

**Do ponto de vista dos componentes, o Hyperledger contém vários elementos, conforme descrito aqui:**

* **Consenso**: Esses serviços são responsáveis por facilitar o processo de acordo entre os participantes da rede blockchain. O consenso é necessário para garantir que a ordem e o estado das transações sejam validados e acordados na rede blockchain.
* **Contratos inteligentes**: Esses serviços são responsáveis por implementar a lógica de negócios de acordo com os requisitos dos usuários. As transações são processadas com base na lógica definida nos contratos inteligentes que residem na blockchain.
* **Comunicação**: Este componente é responsável pela transmissão e troca de mensagens entre os nós da rede blockchain.
* **Segurança e criptografia**: Esses serviços são responsáveis por fornecer a capacidade de permitir que vários algoritmos ou módulos criptográficos forneçam serviços de privacidade, confidencialidade e não-repúdio.
* **Armazenamento de dados**: Este componente fornece a capacidade de usar diferentes armazenamentos de dados para armazenar o estado da razão. Isso significa que os armazenamentos de dados também são plugáveis, permitindo o uso de qualquer backend de banco de dados, como couchdb ou goleveldb.
* **Serviços de política**: Este conjunto de serviços fornece a capacidade de gerenciar as diferentes políticas exigidas para a rede blockchain. Isso inclui política de endosso e política de consenso.
* **APIs e SDKs**: Este componente permite que clientes e aplicações interajam com a blockchain. Um SDK é usado para fornecer mecanismos para implantar e executar chaincode, consultar blocos e monitorar eventos na blockchain.

Na próxima seção, discutiremos os objetivos de design do Hyperledger.

*Chaincode é um programa que implementa uma interface específica. Ele geralmente é escrito em Java, Node.js ou Go. Embora os termos contrato inteligente e chaincode sejam usados de forma intercambiável no Hyperledger Fabric, há uma diferença sutil entre eles. Um contrato inteligente pode ser definido como um trecho de código que define a lógica da transação, que controla o ciclo de vida da transação e pode resultar em atualização do estado global (world state). Chaincode é um conceito relevante mas ligeiramente diferente — é um objeto implantável que contém contratos inteligentes empacotados dentro dele. Um único chaincode pode conter múltiplos contratos inteligentes, e após a implantação, todos os contratos inteligentes contidos no chaincode estão disponíveis para uso. Geralmente, no entanto, os termos são usados de forma intercambiável.*

**Princípios de design do Hyperledger**

Há certos requisitos para um serviço de blockchain. A arquitetura de referência é impulsionada pelas necessidades e exigências levantadas pelos participantes do projeto Hyperledger após o estudo dos casos de uso da indústria. Existem várias categorias de requisitos que foram deduzidas do estudo de casos de uso industriais e são vistas como princípios da filosofia de design. Descreveremos esses princípios nas seções seguintes.

* **Estrutura modular**: O principal requisito do Hyperledger é uma estrutura modular. Espera-se que uma blockchain entre setores seja usada em muitos cenários de negócios, e, como tal, funções relacionadas a armazenamento, política, chaincode, controle de acesso, consenso e muitos outros serviços blockchain devem ser modulares e plugáveis. A especificação fornecida na arquitetura de referência do Hyperledger sugere que os módulos devem ser “plug and play”, e os usuários devem ser capazes de remover e adicionar facilmente um módulo diferente que atenda aos requisitos do negócio.
* **Privacidade e confidencialidade**: Este é um dos fatores mais críticos. Como blockchains tradicionais são não permissionadas, em um modelo permissionado é de extrema importância que as transações na rede sejam visíveis apenas para aqueles que têm permissão para visualizá-las. A privacidade e a confidencialidade das transações e contratos são de absoluta importância em uma blockchain empresarial. Como tal, a visão do Hyperledger é fornecer suporte para uma gama completa de protocolos e algoritmos criptográficos. Discutimos criptografia no Capítulo 3, Criptografia Simétrica, e Capítulo 4, Criptografia Assimétrica.

Ainda sobre **Privacidade e Confidencialidade**:

Espera-se que os usuários possam escolher módulos apropriados de acordo com seus requisitos de negócios. Por exemplo, se uma blockchain empresarial precisa ser executada apenas entre partes já confiáveis e realizar operações comerciais muito básicas, talvez não seja necessário ter suporte criptográfico avançado para confidencialidade e privacidade. Portanto, os usuários devem poder remover essa funcionalidade (módulo) ou substituí-la por um módulo mais apropriado que atenda às suas necessidades.

Da mesma forma, se os usuários precisarem executar uma blockchain entre setores, então a confidencialidade e a privacidade podem ser de importância fundamental. Nesse caso, os usuários devem poder conectar um mecanismo criptográfico avançado e de controle de acesso (módulo) na blockchain, que pode até permitir o uso de módulos de segurança de hardware (HSMs). A blockchain também deve ser capaz de lidar com algoritmos criptográficos sofisticados sem comprometer o desempenho. Além dos cenários mencionados anteriormente, devido a exigências regulatórias nos negócios, também deve haver uma provisão para permitir a implementação de políticas de privacidade e confidencialidade em conformidade com requisitos regulatórios e de conformidade.

* **Identidade**: Para fornecer serviços de privacidade e confidencialidade, também é necessário um modelo de Infraestrutura de Chave Pública (PKI) flexível que possa ser usado para lidar com a funcionalidade de controle de acesso. A força e o tipo dos mecanismos criptográficos também devem variar de acordo com as necessidades e requisitos dos usuários. Em certos cenários, pode ser necessário que um usuário oculte sua identidade, e, como tal, espera-se que o Hyperledger forneça essa funcionalidade.
* **Escalabilidade**: Este é outro requisito importante que, uma vez atendido, permitirá uma taxa razoável de processamento de transações, suficiente para todas as necessidades comerciais e também para um grande número de usuários.
* **Transações determinísticas**: Este é um requisito central em qualquer blockchain. Se as transações não produzirem o mesmo resultado todas as vezes que forem executadas, independentemente de quem e onde a transação for executada, atingir o consenso é impossível. Portanto, transações determinísticas tornam-se um requisito fundamental em qualquer rede blockchain. Discutimos esses conceitos no Capítulo 8, Contratos Inteligentes.
* **Auditabilidade**: Auditabilidade é outro requisito dos negócios. Espera-se que um registro imutável de todas as identidades, operações relacionadas e quaisquer alterações seja mantido.
* **Interoperabilidade**: Atualmente, existem muitas plataformas blockchain disponíveis, mas elas não conseguem se comunicar facilmente entre si. Isso pode ser um fator limitante no crescimento de um ecossistema global de negócios baseado em blockchain. É previsto que muitas redes blockchain operarão no mundo dos negócios para necessidades específicas, mas é importante que elas sejam capazes de se comunicar entre si. Deve haver um conjunto comum de padrões que todas as blockchains possam seguir para permitir a comunicação entre diferentes razões. Existem diferentes esforços em andamento para alcançar isso, não apenas sob o guarda-chuva do Hyperledger, como o Hyperledger Quilt, mas também outros projetos como Cosmos e Polkadot.
* **Portabilidade**: O requisito de portabilidade está relacionado à capacidade de operar em múltiplas plataformas e ambientes sem a necessidade de alterar nada no nível do código. O Hyperledger Fabric, por exemplo, é concebido para ser portátil, não apenas no nível da infraestrutura, mas também nos níveis de código, biblioteca e API, de modo que possa suportar um desenvolvimento uniforme em várias implementações do Hyperledger.
* **Consultas ricas de dados**: A rede blockchain deve permitir que consultas ricas sejam executadas na rede. Isso pode ser usado para consultar o estado atual da razão usando linguagens de consulta tradicionais, como SQL, e com suporte a instruções como SELECT, o que permitirá uma adoção mais ampla e facilidade de uso.

Todos os pontos acima descrevem os princípios orientadores gerais que permitem o desenvolvimento de soluções blockchain que estejam alinhadas com a filosofia de design do Hyperledger.

Na próxima seção, veremos o Hyperledger Fabric em detalhes, que é o primeiro projeto a se graduar para o status ativo sob o Hyperledger.

**Hyperledger Fabric**

Hyperledger Fabric, ou simplesmente Fabric, é a contribuição feita inicialmente pela IBM e pela Digital Asset ao projeto Hyperledger. Essa contribuição visa permitir uma abordagem modular, aberta e flexível para a construção de redes blockchain.

**Conceitos-chave**

Várias funções no Fabric são plugáveis, e ele também permite o uso de qualquer linguagem para desenvolver contratos inteligentes. Essa funcionalidade é possível porque é baseada em tecnologia de contêiner (Docker), que pode hospedar qualquer linguagem.

O chaincode é isolado em um contêiner seguro, que inclui um sistema operacional seguro, a linguagem do chaincode, o ambiente de tempo de execução e os SDKs para Go, Java e Node.js. Outras linguagens também poderiam ser suportadas no futuro, se necessário, mas isso exigiria algum trabalho de desenvolvimento.

Essa capacidade é um recurso atraente em comparação com linguagens específicas de domínio no Ethereum ou a linguagem de script limitada no Bitcoin. É uma rede permissionada que visa resolver questões como escalabilidade, privacidade e confidencialidade. A ideia fundamental por trás disso é a modularização, que permitiria flexibilidade no design e na implementação da blockchain empresarial. Isso pode resultar na obtenção de escalabilidade, privacidade e outros atributos desejados, e ajustá-los de acordo com os requisitos.

As transações no Fabric são privadas, confidenciais e anônimas para os usuários em geral, mas ainda podem ser rastreadas e vinculadas aos usuários por auditores autorizados. Como uma rede permissionada, todos os participantes precisam estar registrados com os serviços de associação para acessar a rede blockchain. Esta razão também fornece uma funcionalidade de auditabilidade para atender às necessidades regulatórias e de conformidade exigidas pelo usuário. Existem seis capacidades principais da blockchain Hyperledger Fabric:

* **Arquitetura modular**, que torna o Hyperledger Fabric resiliente a mudanças e permite que qualquer setor o adote conforme necessário.
* **Privacidade e confidencialidade**, que permitem o compartilhamento de dados apenas entre subconjuntos específicos de membros da rede usando canais privados.
* **Eficiência e escalabilidade** na rede Hyperledger Fabric são fornecidas devido à capacidade do Fabric de atribuir funções aos nós, o que resulta em concorrência e paralelismo. As transações podem ser processadas em threads para um processamento mais rápido no Hyperledger Fabric. Além disso, como a execução pode ser exigida apenas em um subconjunto específico de nós, alguns recursos da rede podem permanecer livres para realizar outras tarefas, resultando assim em uma rede mais eficiente.
* **Lógica de negócios no Fabric** é fornecida por contratos inteligentes chamados chaincode.
* **Gerenciamento de identidade** é fornecido por meio de um provedor de serviços de associação (MSP), que gerencia IDs de usuários e fornece serviços de autenticação para os usuários na rede.
* **Governança** é uma capacidade importante que permite a criação e aplicação de políticas para garantir a segurança de controle de acesso de um sistema. Por exemplo, uma política é necessária para identificar os participantes que têm permissão para implantar um chaincode ou fazer alterações de configuração.

**Serviço de associação (Membership service)**

Este serviço é usado para fornecer capacidade de controle de acesso para os usuários da rede Fabric. Ele fornece capacidades de gerenciamento de identidade e permite políticas de controle de acesso personalizadas e complexas. Os serviços de associação executam as seguintes funções:

* Verificação da identidade do usuário
* Registro de usuários
* Atribuição de permissões apropriadas aos usuários, dependendo de seus papéis

Os serviços de associação fazem uso de uma autoridade certificadora (CA) para oferecer suporte às operações de gerenciamento de identidade e autorização. Esta CA pode ser interna (como a Fabric CA, que é a interface padrão no Hyperledger Fabric), ou uma organização pode optar por usar uma autoridade certificadora externa. A Fabric CA emite certificados de inscrição (E-Certs), que são produzidos por uma autoridade certificadora de inscrição (E-CA). Uma vez que os pares (peers) recebam uma identidade, eles têm permissão para ingressar na rede blockchain. Também são emitidos certificados temporários chamados T-Certs, que são usados para transações únicas.

Todos os peers e aplicações são identificados usando uma autoridade certificadora. Um serviço de autenticação é fornecido pela autoridade certificadora. Os MSPs também podem se conectar a serviços de identidade existentes como o Lightweight Directory Access Protocol (LDAP). A implementação padrão do MSP no Hyperledger Fabric é baseada em um padrão amplamente utilizado da internet que especifica o uso e a estrutura de certificados de chave pública, o padrão X.509. Isso significa que o Hyperledger Fabric pode suportar qualquer PKI em que os certificados sejam emitidos por autoridades certificadoras padrão da internet.

O Hyperledger Fabric também é fortalecido com um protocolo criptográfico chamado **identity mixer**, que fornece autenticação forte enquanto oferece privacidade de identidade.

Um conceito intimamente associado é o de **políticas**. Políticas controlam quem tem permissão para fazer o quê e em qual componente da rede Hyperledger. Enquanto os MSPs reconhecem identidades válidas, as políticas implementam permissões para essas identidades.

Esta seção cobriu os serviços de associação implementados no Hyperledger Fabric. A seguir, vamos introduzir alguns dos serviços de blockchain do Fabric.

**Serviços blockchain**

Os serviços blockchain estão no núcleo do Hyperledger Fabric. Os componentes dentro desta categoria são os seguintes.

**Serviços de consenso**

Um serviço de consenso é responsável por fornecer a interface para o mecanismo de consenso. Ele serve como um módulo plugável que recebe a transação de outras entidades do Hyperledger e as executa sob critérios, de acordo com o tipo de mecanismo escolhido.

O consenso no Hyperledger Fabric é implementado como um peer chamado **orderer**, que é responsável por ordenar as transações em sequência dentro de um bloco. Um orderer não contém contratos inteligentes nem ledgers. O consenso é plugável e, atualmente, os serviços de ordenação no Hyperledger Fabric são baseados em um mecanismo de consenso chamado **RAFT**, que é um protocolo tolerante a falhas por falha (crash fault-tolerant) baseado em líder-seguidor para alcançar consenso distribuído.

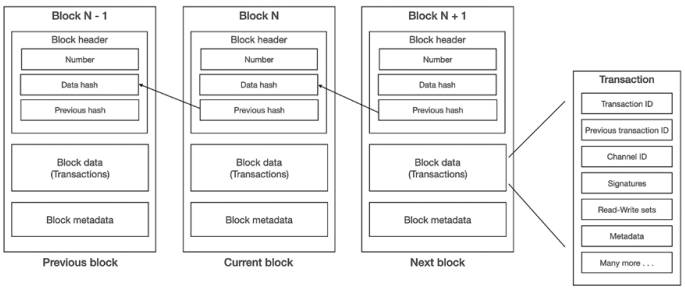
Discutimos mecanismos de consenso em detalhes apropriados no Capítulo 5, Algoritmos de Consenso. Os leitores podem consultar esse capítulo para revisar os mecanismos RAFT e PBFT.

Além desses mecanismos, outros mecanismos podem se tornar disponíveis no futuro e serem conectados ao Hyperledger Fabric.

**Razão distribuída (Distributed ledger)**

Blockchain e estado global (world state) são dois elementos principais da razão distribuída. Blockchain é simplesmente uma lista criptograficamente vinculada de blocos (como introduzido no Capítulo 1, Blockchain 101), e o estado global é um banco de dados chave-valor que armazena os valores atuais de um conjunto de estados da razão. O estado global facilita a obtenção do estado mais recente, em vez de os programas precisarem percorrer todo o banco de dados da blockchain. Este banco de dados é usado por contratos inteligentes para armazenar estados relevantes durante a execução das transações. Uma blockchain consiste em blocos que contêm transações. Essas transações contêm chaincode, que executa transações que podem resultar na atualização do estado global. Cada nó salva o estado global em disco usando LevelDB ou CouchDB, dependendo da implementação. Como o Fabric permite armazenamentos de dados plugáveis, você pode escolher qualquer armazenamento de dados para persistência.

Um bloco consiste em três componentes principais chamados **cabeçalho do bloco**, **dados do bloco (transações)** e **metadados do bloco**. O diagrama a seguir mostra uma representação da blockchain com a estrutura do bloco e das transações no Hyperledger Fabric, com os campos relevantes:

*****Figura 14.2: Blockchain e estrutura da transação*

O cabeçalho do bloco consiste em três campos, a saber: **Número**, **Hash dos dados** e **Hash anterior**.

Os dados do bloco contêm uma lista ordenada de transações. Uma transação é composta por múltiplos campos, como: **ID da transação**, **ID da transação anterior**, **ID do canal**, **Assinaturas**, **conjuntos de leitura-escrita (Read-Write sets)**, **Metadados**, e muitos outros.

Os metadados do bloco consistem principalmente na identidade do criador, o horário de criação e assinaturas relevantes.

**Protocolo ponto a ponto (P2P)**

O protocolo P2P no Hyperledger Fabric é construído usando o Google RPC (gRPC). Ele usa buffers de protocolo para definir a estrutura das mensagens.

As mensagens são passadas entre nós para executar várias funções. Existem quatro tipos principais de mensagens no Hyperledger Fabric: **descoberta**, **transação**, **sincronização** e **consenso**. Mensagens de descoberta são trocadas entre nós ao iniciar, para descobrir outros peers na rede. Mensagens de transação são usadas para implantar, invocar e consultar transações, e mensagens de consenso são trocadas durante o processo de consenso. Mensagens de sincronização são passadas entre nós para sincronizar e manter a blockchain atualizada em todos os nós.

**Armazenamento da razão (Ledger storage)**

Para salvar o estado da razão, por padrão, o LevelDB é usado, que está disponível em cada peer. Uma alternativa é usar o CouchDB, que fornece a capacidade de executar consultas ricas.

**Serviços de contrato inteligente (Smart contract services)**

Esses serviços permitem a criação de contêineres seguros que são usados para executar o chaincode. Os componentes desta categoria são os seguintes:

* **Contêiner seguro (Secure container)**: O chaincode é implantado em contêineres Docker que fornecem um ambiente isolado (sandboxed) para execução de contratos inteligentes. Atualmente, o Golang é suportado como a principal linguagem de contratos inteligentes, mas quaisquer outras linguagens mainstream podem ser adicionadas e habilitadas, se necessário.
* **Registro seguro (Secure registry)**: Este componente fornece um registro de todas as imagens contendo contratos inteligentes.
* **Eventos (Events)**: Eventos na blockchain podem ser acionados por endossadores e contratos inteligentes. Aplicações externas podem ouvir esses eventos e reagir a eles, se necessário, por meio de adaptadores de eventos. Eles são semelhantes ao conceito de eventos introduzido em Solidity no Capítulo 11, Ferramentas, Linguagens e Frameworks para Desenvolvedores Ethereum.

**APIs e CLIs**

Uma interface de programação de aplicações (API) fornece uma interface para o Fabric expondo várias APIs REST. Adicionalmente, interfaces de linha de comando (CLIs) que fornecem um subconjunto das APIs REST e permitem testes rápidos e interação limitada com a blockchain também estão disponíveis.

Nas seções anteriores, cobrimos os principais serviços do Hyperledger Fabric. Agora, vamos considerar os componentes do Hyperledger Fabric a partir de uma perspectiva de rede.

**Componentes**

Há vários componentes que podem fazer parte da rede blockchain do Hyperledger Fabric. Esses componentes incluem, mas não se limitam a: peers, clientes, canais, banco de dados do estado global, transações, provedores de serviço de associação, contratos inteligentes e componentes do provedor de serviços criptográficos.

**Peers/nós**

Os peers participam da manutenção do estado da razão distribuída. Eles também mantêm uma cópia local da razão distribuída. Os peers se comunicam por meio do protocolo gossip. Existem três tipos de peers na rede Hyperledger Fabric:

* **Peers endossadores (endorsing peers)** ou **endossadores**, que simulam a execução da transação e geram um conjunto de leitura-escrita (read-write set). A leitura é uma simulação da leitura de dados da razão por parte da transação e a escrita é o conjunto de atualizações que seriam feitas na razão caso e quando a transação for executada e registrada na razão. Endossadores executam e endossam transações. Deve-se observar que um endossador também é um validador (committer). As políticas de endosso são implementadas com o chaincode e especificam as regras para endosso de transações.
* **Peers validadores (committing peers)** ou **validadores**, que recebem transações endossadas pelos endossadores, as verificam e então atualizam a razão com o conjunto de leitura-escrita. Um validador verifica o conjunto de leitura-escrita gerado pelos endossadores junto com a validação da transação.
* **Nós ordenadores (orderer nodes)** recebem transações dos endossadores juntamente com os conjuntos de leitura-escrita, organizam-nas em uma sequência e as enviam para os peers validadores. Os validadores então realizam a validação e o registro na razão.

Todos os peers fazem uso de certificados emitidos pelos serviços de associação. Um peer pode assumir dois papéis. Um peer pode ser um **peer líder** ou um **peer âncora**. Em um canal onde uma organização tem múltiplos peers, um peer líder é responsável por distribuir transações do nó ordenador para os demais peers validadores na organização. Um peer âncora é usado quando um peer precisa se comunicar com um peer em outra organização. Um peer pode assumir os papéis de validador, endossador, líder e âncora ao mesmo tempo.

**Clientes**

Clientes são softwares que fazem uso de APIs para interagir com o Hyperledger Fabric e propor transações.

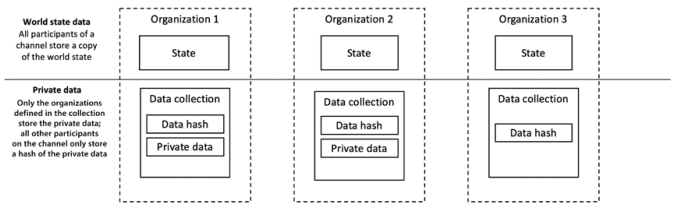
**Canais (Channels)**

Canais permitem o fluxo de transações confidenciais entre diferentes partes na rede. Eles permitem o uso da mesma rede blockchain, mas com blockchains sobrepostas separadas. Canais permitem que apenas os membros do canal vejam as transações relacionadas a eles; todos os outros membros da rede não poderão ver essas transações. Podemos pensar em um canal como uma sub-rede privada para comunicação entre dois ou mais membros da rede.

**Banco de dados de estado global (World state database)**

O estado global reflete todas as transações registradas na blockchain. Isso é essencialmente um armazenamento chave-valor que é atualizado como resultado de transações e execução de chaincode. Para esse fim, usa-se LevelDB ou CouchDB. O LevelDB é um armazenamento chave-valor, enquanto o CouchDB armazena dados como objetos JSON, o que permite executar consultas ricas no banco de dados. Um conceito relacionado é o de **dados privados**, que habilita a privacidade na blockchain do Hyperledger.

Podemos ver esse conceito visualmente no diagrama abaixo:

****

*Figura 14.3: Estado global e dados privados*

O diagrama anterior mostra um canal compartilhado entre várias organizações. Note que, embora o "estado" seja compartilhado entre todas as organizações, os dados privados estão disponíveis apenas para aqueles peers e organizações que fazem parte da transação. Os peers que não precisam conhecer a transação possuem apenas um hash dos dados privados. Esse conceito nos leva às **coleções de dados privados (PDCs)**.

**Coleções de dados privados (Private Data Collections – PDCs)**

As PDCs estão associadas a organizações. Elas definem como os dados privados devem ser disseminados e endossados, e fazem parte da definição do chaincode. Elas são compostas por uma seção privada e uma seção pública. A seção privada consiste em dados que são disseminados por meio de um protocolo P2P entre os peers das organizações que pertencem à PDC. A seção pública contém hashes dos dados privados, que são endossados e registrados na razão. As PDCs são úteis em cenários em que o serviço de ordenação não pode ser confiável para confidencialidade e quando os dados em uma única razão ou canal do Fabric devem ser visíveis apenas para partes específicas, ou seja, transações privadas restritas.

**Transações**

As mensagens de transação podem ser divididas em dois tipos: **transações de implantação** e **transações de invocação**. A primeira é usada para implantar um novo chaincode na razão, e a segunda é usada para chamar funções do contrato inteligente. As transações podem ser públicas ou confidenciais. Transações públicas são abertas e disponíveis para todos os participantes, enquanto transações confidenciais são visíveis apenas em um canal aberto para seus participantes.

**Provedor de Serviços de Associação (Membership Service Provider – MSP)**

O MSP é um componente modular usado para gerenciar identidades na rede blockchain. Este provedor é utilizado para autenticar clientes que desejam ingressar na rede blockchain. Uma autoridade certificadora (CA) é usada no MSP para fornecer verificação de identidade e serviços de vinculação.

**Contratos inteligentes**

Discutimos contratos inteligentes em detalhes no Capítulo 8, Contratos Inteligentes. No Hyperledger Fabric, o mesmo conceito de contratos inteligentes é implementado, mas são chamados de **chaincode** em vez de contratos inteligentes. Eles contêm condições e parâmetros para executar transações e atualizar a razão. O chaincode é geralmente escrito em **Golang** ou **Java**.

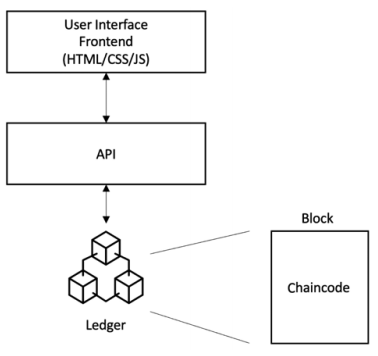
**Provedor de serviços criptográficos (Crypto service provider)**

Como o nome sugere, este é um serviço que fornece algoritmos e padrões criptográficos para uso na rede blockchain. Este serviço fornece gerenciamento de chaves, operações de assinatura e verificação, e mecanismos de criptografia e descriptografia. Este serviço é usado com o serviço de associação para fornecer suporte para operações criptográficas para elementos da blockchain, como endossadores, clientes e outros nós e peers.

Após esta introdução a alguns dos componentes do Hyperledger Fabric, veremos a seguir como se parece uma aplicação em uma rede Hyperledger.

**Aplicações**

Uma aplicação típica no Fabric é composta simplesmente por uma interface de usuário, geralmente escrita em JavaScript/HTML, que interage com o chaincode (contrato inteligente) armazenado na razão por meio de uma camada de API:

****  
*Figura 14.4: Uma aplicação típica do Fabric*

O Hyperledger fornece várias APIs e interfaces de linha de comando para permitir a interação com a razão. Essas APIs incluem interfaces para identidade, transações, chaincode, razão, rede, armazenamento e eventos.

Na próxima seção, veremos como o chaincode é implementado.

**Implementação de Chaincode**

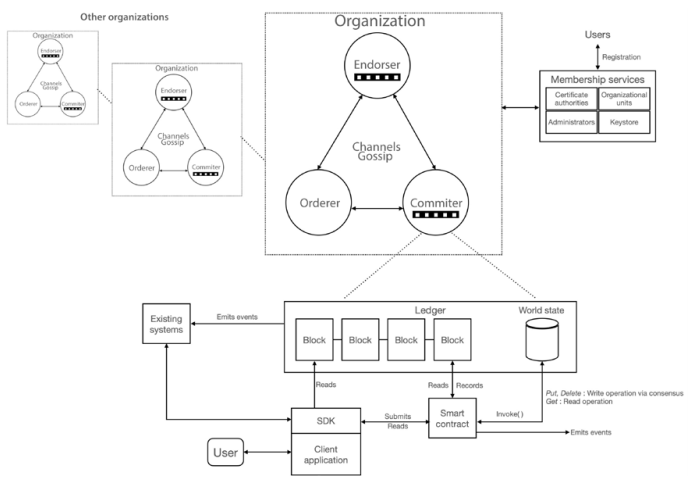
O chaincode geralmente é escrito em Golang ou Java. O chaincode pode ser público (visível para todos na rede), confidencial ou controlado por acesso. Esses arquivos de código funcionam como contratos inteligentes com os quais os usuários podem interagir por meio de APIs. Os usuários podem chamar funções no chaincode que resultam em alteração de estado e, consequentemente, atualizam a razão.

Também existem funções que são usadas apenas para consultar a razão e não resultam em nenhuma alteração de estado. A implementação do chaincode é realizada primeiro criando a interface *shim* do chaincode no código. Os pacotes shim fornecem APIs para acessar variáveis de estado, o contexto da transação do chaincode e chamar outros chaincodes. Eles podem estar no código Java ou Golang.

As quatro funções a seguir são exigidas para implementar o chaincode:

* **Init()**: Esta função é invocada quando o chaincode é implantado na razão. Ela inicializa o chaincode e resulta em uma alteração de estado, que atualiza a razão adequadamente.
* **Invoke()**: Esta função é usada quando contratos são executados. Ela recebe um nome de função como parâmetro junto com um array de argumentos. Esta função resulta em uma alteração de estado e escreve na razão.
* **Query()**: Esta função é usada para consultar o estado atual de um chaincode implantado. Esta função não faz nenhuma alteração na razão.
* **Main()**: Esta função é executada quando um peer implanta sua própria cópia do chaincode. O chaincode é registrado com o peer usando esta função.

O diagrama a seguir ilustra a visão geral do Hyperledger Fabric. Note que os clusters de peers no topo incluem todos os tipos de nós: endossadores, validadores e ordenadores:

****  
*Figura 14.5: Uma visão geral de alto nível de uma rede Hyperledger Fabric*

O diagrama anterior mostra que os peers (exibidos no centro superior) se comunicam entre si, e cada nó possui uma cópia da blockchain. No canto superior direito, os serviços de associação são mostrados, os quais validam e autenticam os peers na rede usando uma CA. Na parte inferior do diagrama, é mostrada uma visão ampliada da blockchain, pela qual sistemas existentes podem produzir eventos para a blockchain e também escutar os eventos da blockchain, os quais podem, opcionalmente, disparar uma ação. No canto inferior direito, é mostrada a interação de um usuário com a aplicação. A aplicação, por sua vez, interage com o contrato inteligente por meio do método Invoke(), e os contratos inteligentes podem consultar ou atualizar o estado da blockchain.

O chaincode no Hyperledger Fabric não precisa ser determinístico e pode ser desenvolvido em qualquer linguagem mainstream, como JavaScript, Java ou Golang.

**Modelo de aplicação (The application model)**

Qualquer aplicação blockchain para Hyperledger Fabric segue a arquitetura **MVC-B**. Ela é baseada no popular padrão de design MVC. Os componentes neste modelo são: **View**, **Control**, **Model** (o modelo de dados) e **Blockchain**:

* **Lógica de visualização (View logic)**: Está relacionada à interface do usuário. Pode ser um aplicativo desktop, web ou uma interface móvel.
* **Lógica de controle (Control logic)**: É o orquestrador entre a interface do usuário, o modelo de dados e as APIs.
* **Modelo de dados (Data model)**: Este modelo é usado para gerenciar os dados fora da cadeia (*off-chain*).
* **Lógica blockchain (Blockchain logic)**: É usada para gerenciar a blockchain por meio do controlador e do modelo de dados via transações.

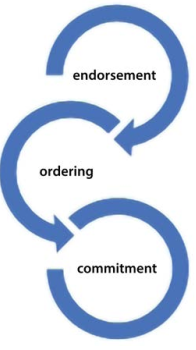
Após esta breve introdução ao modelo de aplicação, vamos passar para outro tópico importante, o consenso, que é importante não apenas no Hyperledger Fabric, mas também central no design e arquitetura de blockchains.

**Mecanismo de consenso (Consensus mechanism)**

O mecanismo de consenso no Hyperledger Fabric consiste em três etapas:

* **Endosso da transação**: Este processo endossa as transações simulando a execução da transação.
* **Ordenação**: Este é um serviço fornecido pelo cluster de ordenadores que executa o algoritmo de consenso, o qual recebe transações endossadas e decide uma sequência na qual as transações serão escritas na razão.
* **Validação e confirmação (commitment)**: Este processo é executado pelos peers validadores, que primeiro validam as transações recebidas dos ordenadores e, em seguida, registram essa transação na razão.

O serviço IBM Cloud oferece aplicações de exemplo para blockchain em sua oferta de blockchain como serviço. Ele está disponível em:  
<https://www.ibm.com/blockchain/platform/>

****  
*Figura 14.6: O fluxo de consenso*

A etapa de ordenação também é plugável. Existem vários serviços de ordenação disponíveis para o mecanismo de consenso do Fabric. Inicialmente, em algumas versões do Fabric 1.0, apenas serviços de ordenação simples como Apache Kafka e SOLO estavam disponíveis. No entanto, agora outros algoritmos de consenso avançados como o RAFT também estão disponíveis.

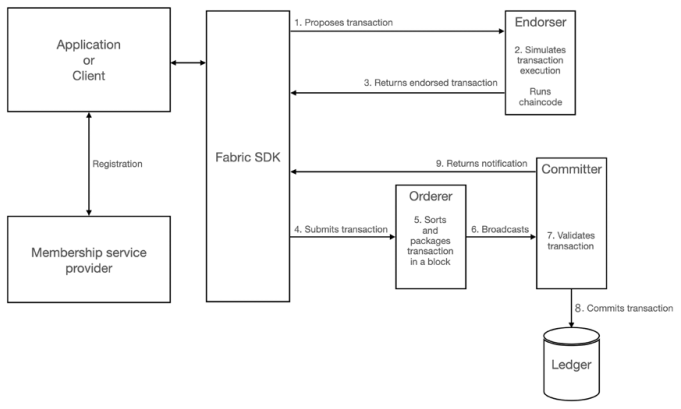
Na próxima seção, descreveremos como uma transação flui por várias etapas no Hyperledger Fabric antes de finalmente atualizar a razão.

**Ciclo de vida da transação (Transaction lifecycle)**

Há várias etapas envolvidas no fluxo de uma transação no Hyperledger Fabric. As etapas são descritas em detalhes a seguir:

1. **Proposta de transação pelos clientes**: Esta é a primeira etapa, onde uma transação é proposta pelos clientes e enviada aos peers endossadores na rede de razão distribuída. Todos os clientes precisam estar registrados pelos serviços de associação antes de poderem propor transações.
2. **A transação é simulada pelos endossadores e gera um conjunto leitura-escrita (RW set)**: Isso é realizado executando o chaincode, mas, em vez de atualizar a razão, apenas um conjunto RW (read-write) representando quaisquer leituras ou atualizações à razão é criado.
3. **A transação endossada é enviada de volta à aplicação**.
4. **As transações endossadas e os conjuntos RW são enviados ao serviço de ordenação pela aplicação**.
5. **O serviço de ordenação monta todas as transações endossadas e conjuntos RW em ordem dentro de um bloco e as organiza por ID de canal**.
6. **O serviço de ordenação transmite o bloco montado a todos os peers validadores. Os peers validadores realizam as seguintes ações**:
   1. Validam a política de endosso.
   2. Validam as versões dos conjuntos de leitura no banco de dados de estado.
   3. Registram o bloco na blockchain.
   4. Registram a transação válida no banco de dados de estado.
7. **A validação assegura que duas condições sejam atendidas**: primeiro, que as transações sejam executadas conforme a lógica da transação fornecida; segundo, que não existam conflitos de estado nas transações submetidas, por exemplo, um cenário onde duas transações estejam tentando atualizar o mesmo estado.
8. **Por fim, uma notificação de sucesso ou falha da transação pelos peers validadores é enviada de volta aos clientes/aplicações**.

O diagrama a seguir representa as etapas e a arquitetura do Fabric a partir de uma perspectiva de fluxo de transação:

****  
*Figura 14.7: O fluxo da transação*

Como se pode ver no diagrama anterior, a primeira etapa é propor transações, que um cliente faz por meio de um SDK. Antes disso, presume-se que todos os clientes e peers estejam registrados com o provedor de serviços de associação.

O Hyperledger Fabric utiliza um padrão de **executar-ordenar-validar (execute-order-validate)** que permite que as transações sejam executadas antes da blockchain alcançar o consenso, o que possibilita maior eficiência. No Fabric, cada transação é executada e endossada apenas por um subconjunto de peers, o que permite execução paralela. Além disso, o estado final de uma transação só é escrito na blockchain após a validação.

**Fabric 2.0**

Esta é uma atualização importante do protocolo. Os fundamentos permanecem os mesmos, mas alguns recursos e melhorias muito interessantes foram introduzidos, os quais serão apresentados nas seções a seguir.

**Novo gerenciamento do ciclo de vida do chaincode (New chaincode lifecycle management)**

No Hyperledger Fabric 2.0, é implementado um novo gerenciamento de ciclo de vida de chaincode descentralizado, que é a principal diferença notável entre o Fabric 1.x e o Fabric 2.x. No primeiro, um administrador de organização podia instalar o chaincode em sua própria organização, e os administradores das outras organizações tinham que concordar incondicionalmente em instanciar (implantar ou atualizar) o chaincode. Em outras palavras, não havia um processo de acordo entre as organizações para concordar com a instalação de um chaincode. Esse problema é resolvido pela introdução de verificações que permitem que os peers participem de um chaincode somente se os administradores de suas organizações tiverem concordado com ele.

Primeiro, vejamos o ciclo de vida do chaincode no Fabric 1.0:

1. Cada administrador de organização instala o chaincode em seus peers.
2. Um administrador instancia o chaincode por meio de implantação ou atualização.
3. Após os resultados serem recebidos, a transação é enviada aos ordenadores.
4. Após a ordenação, ela é enviada a todos os peers e registrada.

Agora, vejamos o ciclo de vida no Fabric 2.0, que destaca as diferenças em relação ao Fabric 1.0:

1. **Instalar o chaincode colocando o pacote no sistema de arquivos**. Cada administrador de organização instala o chaincode em seus peers.
2. **Fornecer aprovação** executando a função approveformyorg. Isso verifica se a organização concorda com a definição do chaincode.
3. **Enviar o serviço de ordenação ao orderer**.
4. **Registrar na coleção da organização**. A coleção é uma coleção de dados privados que fornece um mecanismo para permitir o compartilhamento de dados privados apenas entre aquelas organizações que fazem parte da transação, mesmo que estejam dentro do mesmo canal. Em outras palavras, coleções permitem compartilhamento de dados privados entre um subconjunto de organizações em um canal. Mesmo que outras organizações sejam membros do mesmo canal, elas não podem ver os dados compartilhados entre o subconjunto de organizações que estão usando a coleção, também chamada de **coleção de dados privados**.
5. **Os administradores das outras organizações realizam as mesmas etapas de 1 a 4**.
6. Após essas etapas serem executadas, existe um registro em cada organização indicando o acordo sobre a definição do chaincode.
7. **Definir o chaincode conforme o acordo na etapa 6**.
8. **Submeter à ordenação**.
9. **Registrar a definição em todos os peers**.
10. O administrador da organização que registrou a definição do chaincode agora **invoca a função init** em seus próprios peers e nos peers das outras organizações.
11. **Submeter à ordenação**.
12. **Registrar em todos os peers**.

Além das mudanças no gerenciamento do ciclo de vida do chaincode no Fabric 2.0, também há novos padrões de aplicação para o chaincode. Vamos apresentá-los a seguir.

**Novos padrões de aplicação de chaincode (New chaincode application patterns)**

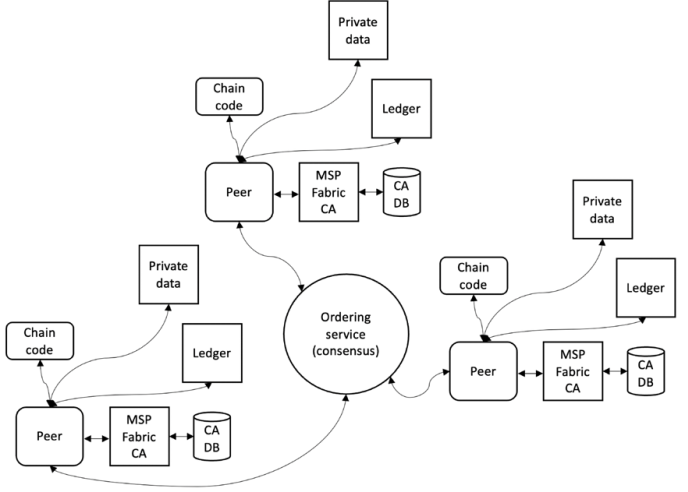
Também foram introduzidos no Fabric 2.0 novos padrões de aplicação de chaincode para colaboração e consenso. Esses padrões consistem em verificações automatizadas que podem ser adicionadas por uma organização para habilitar validação adicional antes que uma transação seja endossada. Além disso, um acordo descentralizado também é suportado, onde múltiplas organizações podem propor condições para um acordo, e quando essas condições são atendidas, um acordo coletivo pode ser estabelecido sobre uma transação.

* **Privacidade de dados aprimorada (Enhanced data privacy)**: O Fabric 2.0 aprimora a privacidade de dados ao introduzir políticas de endosso no nível da coleção, coleções por organização e compartilhamento e verificação flexíveis de dados privados.
* **Launcher de chaincode externo (External chaincode launcher)**: No Hyperledger Fabric 2.0, por padrão, APIs do Docker são usadas para construir e executar o chaincode. Além disso, builders e launchers externos podem ser usados, se necessário. Nas versões anteriores, havia um requisito de ter acesso ao daemon Docker para construir e executar o chaincode, mas no Fabric 2.0 esse requisito não existe mais. Também não há mais necessidade de executar o chaincode em contêineres Docker; o chaincode pode ser executado em qualquer ambiente considerado apropriado pelo operador da rede.

Observe que o Fabric 2.0 oferece suporte tanto aos padrões de ciclo de vida legados quanto aos novos, mas eventualmente o ciclo de vida legado será descontinuado.

* **Consenso RAFT**: O RAFT é um serviço de ordenação tolerante a falhas por falha (crash fault-tolerant – CFT) para alcançar consenso. Ele é baseado na biblioteca etcd e em um modelo de líder-seguidor. Nesse modelo, um líder é eleito cuja decisão é então replicada por seus seguidores.
* **Melhor desempenho**: No Fabric 2.0, foi introduzido cache no nível do banco de dados de estado CouchDB para melhorar o desempenho, reduzindo a latência de leitura, que era introduzida devido a operações de consulta dispendiosas em versões anteriores.

Com tudo o que aprendemos até agora, a arquitetura de aplicação do Hyperledger pode ser representada da seguinte maneira:

****  
*Figura 14.8: Arquitetura de aplicação do Hyperledger Fabric*

Com esta seção, nossa introdução ao Hyperledger Fabric está completa. O Hyperledger Fabric é o projeto principal do Hyperledger e é usado em uma grande variedade de aplicações. O Hyperledger Fabric está evoluindo continuamente, e mais recursos e mudanças são esperados em lançamentos futuros. No entanto, nenhuma mudança de design importante é esperada desde a introdução da versão 2.0.

Uma lista de projetos entre setores construídos com o Hyperledger Fabric e outros projetos Hyperledger está sendo mantida aqui:  
<https://www.hyperledger.org/learn/blockchain-showcase>  
Os leitores podem explorar esse link para entender como o Hyperledger Fabric é usado em diferentes casos de uso.

**Resumo**

Neste capítulo, passamos por uma introdução ao projeto Hyperledger. Primeiramente, discutimos as ideias centrais por trás do projeto Hyperledger, e foi fornecida uma breve introdução a todos os projetos sob o Hyperledger. Um projeto-chave do Hyperledger foi discutido em detalhes, nomeadamente, o **Hyperledger Fabric**. Todos esses projetos estão em constante evolução, e mudanças são esperadas nos próximos lançamentos. No entanto, espera-se que os conceitos centrais de todos os projetos mencionados anteriormente permaneçam inalterados ou mudem apenas ligeiramente. Os leitores são encorajados a visitar os links relevantes fornecidos no capítulo para ver as atualizações mais recentes.

É evidente que há muito acontecendo neste espaço, e projetos como o Hyperledger, da Linux Foundation, estão desempenhando um papel fundamental no avanço da tecnologia blockchain. Cada um dos projetos introduzidos neste capítulo apresenta abordagens inovadoras para resolver os problemas enfrentados em várias indústrias, e quaisquer limitações atuais dentro da tecnologia blockchain também estão sendo abordadas, como escalabilidade e privacidade. Espera-se que mais projetos sejam propostos em breve para o projeto Hyperledger, e prevê-se que, com este esforço colaborativo e aberto, a tecnologia blockchain avançará tremendamente e beneficiará coletivamente a comunidade.

**No próximo capítulo, discutiremos a tokenização**, que é uma das aplicações mais proeminentes da tecnologia blockchain e está revolucionando muitas indústrias, especialmente a indústria financeira.

Junte-se a nós no Discord!  
Para participar da comunidade Discord deste livro – onde você pode compartilhar feedback, fazer perguntas ao autor e saber sobre novos lançamentos – siga o QR code abaixo:

<https://packt.link/ips2H>