**Capítulo 15**

**Tokenização**

Um token é uma representação de um objeto. Usamos tokens em muitos campos diferentes, incluindo economia e computação. Na vida cotidiana, os tokens têm sido usados para representar algo de valor, como um vale-presente que pode ser resgatado em troca de itens. Em computação, diferentes tipos de tokens são usados, os quais podem ser definidos como objetos que representam elegibilidade para realizar alguma operação. Por exemplo, um token de acesso é usado para identificar um usuário e seus privilégios em um sistema computacional. Um token de segurança de hardware é usado em sistemas computacionais para fornecer um meio de autenticar um usuário (verificar sua identidade) em um sistema computacional. Tokens também são usados em mecanismos de segurança computacional para substituir dados sensíveis por equivalentes não sensíveis a fim de evitar o acesso direto às informações sensíveis. Por exemplo, em sistemas de pagamento móvel, os tokens são usados para proteger as informações do cartão de crédito, que são representadas por um token no dispositivo móvel em vez dos dados reais do cartão de crédito.

Assim como o uso geral de tokens no cotidiano, no mundo do blockchain, os tokens também representam algo de valor. No entanto, a diferença crítica é que o token existe digitalmente em um blockchain e é operado criptograficamente, o que significa que eles são gerados, protegidos e transferidos usando protocolos criptográficos. Agora que entendemos o que é um token, podemos definir tokenização como um processo que converte um ativo em um token digital em um blockchain. Especificamente, é um processo de converter os direitos de propriedade de um ativo do mundo real em um token criptográfico/digital em um blockchain.

Neste capítulo, cobriremos tokens de blockchain (ou cripto-tokens) e tokenização. Também abordaremos os seguintes tópicos:

* Tokenização em um blockchain
* Tipos de tokens
* Processo de tokenização
* Ofertas de tokens
* Padrões de tokens
* Criação de um token ERC-20
* Conceitos emergentes

**Tokenização em um blockchain**

Agora, começaremos nossa discussão sobre tokenização no contexto do blockchain, que é o principal objetivo deste capítulo.

Tokenização, no contexto do blockchain, é o processo de representar um ativo digitalmente em um blockchain. Pode ser usada para representar commodities, imóveis, propriedade de arte, moeda ou qualquer outra coisa de valor.

Após esta breve definição, vamos explorar como a tokenização pode ser benéfica.

**Vantagens da tokenização**

A tokenização oferece vários benefícios. A seguir estão alguns dos mais importantes desses benefícios:

* **Processamento mais rápido de transações**: Como as transações e todas as partes relevantes estão presentes no blockchain e prontamente disponíveis, não há necessidade de esperar por uma resposta de uma contraparte ou aguardar operações de compensação e liquidação. Todas essas operações podem ser realizadas de forma eficiente e rápida em um blockchain.
* **Flexibilidade**: Devido à adoção mundial de sistemas que usam tokens, como sistemas de pagamento, a tokenização se torna simples devido ao uso transfronteiriço facilitado.
* **Baixo custo**: Em comparação com produtos financeiros tradicionais, a tokenização requer um custo mais baixo para implementação e gera um custo menor para o usuário final devido à digitalização. Mais recentemente, a introdução de pagamentos digitais revolucionou a indústria financeira. No mesmo espírito, a tokenização usando blockchain pode ser considerada mais um passo rumo à obtenção de maior eficiência e redução de custos. De fato, a tokenização dá origem a um sistema financeiro melhor, mais eficiente e mais democrático. Por exemplo, apenas a capacidade de compartilhar dados de clientes registrados em um blockchain entre todas as instituições financeiras reduz custos. Da mesma forma, a possibilidade de converter ativos ilíquidos em ativos líquidos é outra forma de aumentar a eficiência e os lucros.
* **Descentralização**: Os tokens são apresentados em um blockchain público, aproveitando a descentralização oferecida pela tecnologia blockchain. No entanto, em alguns casos, um nível de centralização razoavelmente aceitável é introduzido devido ao controle e à supervisão exigidos por investidores, corretoras e outras partes envolvidas e interessadas.

Lembre-se, para o restante do capítulo, quando nos referirmos à tokenização, será no contexto de blockchain.

Discutimos os benefícios da descentralização em detalhes no Capítulo 2, *Descentralização*. Consulte aquele capítulo para revisar esse conceito.

**Vantagens da tokenização** (continuação)

* **Segurança**: Como os tokens são protegidos criptograficamente e produzidos usando um blockchain, eles são seguros. No entanto, observe que a implementação adequada deve seguir boas práticas e atender aos padrões da indústria de segurança estabelecidos, caso contrário, falhas de segurança podem resultar em exploração por hackers, levando à perda financeira.
* **Transparência**: Como estão em um blockchain, os tokens são mais transparentes do que os sistemas financeiros tradicionais, o que significa que qualquer atividade pode ser prontamente auditada em um blockchain e está visível para todos.
* **Confiança**: Como resultado das garantias de segurança e transparência, mais confiança é naturalmente depositada na tokenização pelos investidores.
* **Propriedade fracionada**: Imagine um cenário em que você possui uma pintura de Vincent van Gogh. É quase impossível, em cenários tradicionais, ter múltiplos proprietários da pintura sem imensos desafios legais, financeiros e operacionais. No entanto, em um blockchain usando tokens, qualquer ativo (como nossa pintura de van Gogh) pode ser fracionado de modo que possa ser possuído em parte por muitos investidores. A mesma situação é válida para uma propriedade: se eu quisesse ter a propriedade compartilhada com alguém, isso exigiria procedimentos legais complicados. No entanto, com a tokenização, a propriedade fracionada de qualquer ativo, seja arte, imóveis ou qualquer outro ativo do mundo real fora da cadeia (*off-chain*), é rápida, fácil, eficiente, segura e muito menos complicada do que os métodos tradicionais.
* **Baixa barreira de entrada**: Sistemas financeiros tradicionais exigem mecanismos tradicionais de verificação, os quais podem levar muito tempo para um cliente. Embora sejam necessários e cumpram sua finalidade nos sistemas financeiros tradicionais, esses processos levam muito tempo devido aos procedimentos de verificação exigidos e ao envolvimento de diferentes entidades. No entanto, em um blockchain, essa barreira de entrada é reduzida porque não há necessidade de passar por longas verificações. Isso ocorre porque a tokenização se baseia em garantias criptográficas fornecidas pelo blockchain, e para muitos aplicativos descentralizados (DApps) neste ecossistema, basicamente é apenas uma questão de baixar um DApp em seu dispositivo móvel, depositar alguns fundos se necessário e tornar-se parte do ecossistema.
* **Aplicações inovadoras**: A tokenização resultou em muitas aplicações inovadoras, incluindo sistemas de empréstimo baseados em blockchain, seguros e outras inúmeras aplicações financeiras, incluindo corretoras descentralizadas. Títulos agora podem ser tokenizados e apresentados em blockchain, o que resulta em maior confiança e satisfação dos clientes devido à melhor segurança e tempos de processamento mais rápidos.
* **Maior liquidez**: Isso se deve à fácil disponibilidade e acessibilidade dos tokens ao público em geral. Ao usar tokens, mesmo ativos ilíquidos como pinturas podem ser tokenizados e negociados em uma corretora com propriedade fracionada.

Com todas essas vantagens, existem, no entanto, algumas questões que precisam ser abordadas no ecossistema de tokenização. Vamos discutir essas próximas.

**Desvantagens da tokenização**

Nesta seção, apresentamos algumas das desvantagens da tokenização. No topo da lista, temos os requisitos regulatórios:

* **Questões regulatórias**: A regulamentação é um tema crucial de muito debate. É imperativo que os tokens sejam regulamentados para que os investidores possam ter o mesmo nível de confiança que têm ao investir por meio de instituições financeiras tradicionais. O grande problema com a tokenização — e, em geral, com qualquer tecnologia pública de blockchain — é que elas são, na maioria das vezes, descentralizadas, e nos casos em que nenhuma organização única está no controle, torna-se bastante difícil responsabilizar alguém se algo der errado. Em um sistema tradicional, podemos recorrer às autoridades reguladoras ou aos serviços de ouvidoria relevantes, mas quem é responsabilizado em um blockchain?

Alguma parte dessa situação mudou com os recentes padrões de tokenização de valores mobiliários (*security tokenization*) e legislação, os quais consideram tokens como valores mobiliários. Isso significa que tokens de segurança serão então tratados como valores mobiliários, e as mesmas implicações legais, financeiras e regulatórias serão aplicadas a esses tokens conforme ocorre na indústria financeira tradicional atualmente estabelecida. Consulte <https://www.sec.gov/answers/about-lawsshtml.html>, onde diferentes leis que regem a indústria de valores mobiliários são apresentadas. Isso ajuda a aumentar os níveis de confiança dos clientes no sistema; no entanto, ainda existem muitos desafios a serem enfrentados.

Um novo tipo de crime financeiro pode estar evoluindo com esse ecossistema de tokenização, em que, em vez de utilizar métodos tradicionais bem conhecidos e estudados de fraude financeira, criminosos podem optar por lançar um ataque tecnicamente sofisticado. Para um usuário comum, esse tipo de ataque é difícil de identificar e compreender, pois estão totalmente baseados em blockchain e digitalizados. Novas formas de *front running* e manipulação de mercado em plataformas de finanças descentralizadas estão se tornando cada vez mais preocupantes.

* **Legalidade dos tokens**: Esta é uma preocupação em algumas jurisdições onde tokens e criptomoedas são ilegais de possuir e negociar.
* **Barreira tecnológica**: Sistemas financeiros tradicionais têm sido a norma com bancos físicos. Estamos acostumados com esse sistema, mas as coisas mudaram e espera-se que mudem rapidamente com a tokenização. Ecossistemas financeiros baseados em tokenização são mais fáceis de usar para muitas pessoas, mas para algumas, o analfabetismo tecnológico pode se tornar um problema e representar uma barreira. Às vezes, as interfaces e os aplicativos necessários para usar plataformas de tokenização, como corretoras de negociação, são difíceis de usar para o usuário médio.
* **Questões de segurança**: A tecnologia subjacente do blockchain é considerada sólida do ponto de vista de segurança, e às vezes afirma-se que, devido ao uso da criptografia, é impossível lançar ataques e cometer fraudes em um blockchain. No entanto, isso não é inteiramente verdade, mesmo com a base de segurança robusta que os blockchains fornecem. A forma como as plataformas de tokenização e os DApps são implementados no blockchain resulta em problemas de segurança que podem ser explorados por hackers, potencialmente levando à perda financeira. Em outras palavras, mesmo que o blockchain subjacente seja relativamente seguro, o DApp de tokenização implementado por cima pode ter vulnerabilidades que poderiam ser exploradas. Essas fraquezas podem ter sido introduzidas por práticas de desenvolvimento deficientes ou limitações inerentes às linguagens de contratos inteligentes ainda em evolução.

**Tipos de tokens**

Com o rápido desenvolvimento da tecnologia blockchain e das aplicações relacionadas, houve um aumento tremendo no desenvolvimento de vários tipos de tokens e ecossistemas relevantes. Primeiro, vamos esclarecer a diferença entre uma moeda (*coin*) e um token. O Bitcoin é um token? Ou tokens e moedas são a mesma coisa?

Uma **moeda** é um token nativo de um blockchain. É a criptomoeda padrão do blockchain no qual ela opera. Exemplos comuns de tal token são o Bitcoin e o ether. Ambos os tokens ou moedas têm seus próprios blockchains nativos nos quais operam: o blockchain do Bitcoin e o blockchain do Ethereum.

Por outro lado, um **token** é uma representação de um ativo que roda sobre um blockchain. Por exemplo, o Ethereum não apenas tem sua própria criptomoeda ether como token nativo (ou moeda), mas também possui milhares de outros tokens que são construídos sobre o Ethereum para diferentes aplicações. Graças ao seu suporte a contratos inteligentes, o Ethereum se tornou uma plataforma para todos os tipos de tokens diferentes, desde tokens utilitários simples — que geralmente não são mineráveis e permitem aos usuários realizar alguns serviços específicos em produtos na rede/ecossistema blockchain — até tokens de jogos (usados para jogos) e tokens específicos de aplicações de alto valor.

Agora que entendemos a diferença entre uma moeda e um token, vamos observar o uso e a importância dos diferentes tipos de tokens. Os tokens podem ser amplamente divididos em duas categorias com base em seu uso: **tokens fungíveis** e **tokens não fungíveis**.

**Tokens fungíveis**

Do ponto de vista econômico, fungibilidade é a intercambiabilidade de um ativo com outros ativos do mesmo tipo. Por exemplo, uma nota de dez libras é intercambiável com uma nota de dez libras diferente ou duas notas de cinco libras. Não importa quais denominações específicas elas sejam, desde que sejam do mesmo tipo (libra) e tenham o mesmo valor (a soma de duas notas de cinco libras), as notas são aceitáveis.

Tokens fungíveis funcionam no mesmo princípio. Eles são:

* **Indistinguíveis**: Tokens do mesmo tipo são indistinguíveis uns dos outros. Em outras palavras, são idênticos.
* **Intercambiáveis**: Um token é totalmente intercambiável com outro token de mesmo valor.
* **Divisíveis**: Tokens são divisíveis em frações menores.

**Tokens não fungíveis**

Vamos considerar tokens não fungíveis (NFTs), também chamados de “*niftys*”, com a ajuda de um exemplo. Vimos que a fungibilidade permite que os mesmos tipos de tokens sejam intercambiáveis, mas os NFTs **não** são intercambiáveis. Por exemplo, uma pintura colecionável é um ativo não fungível, pois não pode ser substituída por outra pintura do mesmo tipo e valor. Cada pintura é única e distinguível das outras.

Os NFTs são:

* **Únicos**: NFTs são únicos e diferentes de outros tokens do mesmo tipo ou da mesma categoria.
* **Não intercambiáveis**: Como são únicos e representam atributos específicos, esses tokens não são intercambiáveis com tokens do mesmo tipo. Por exemplo, uma pintura rara é única devido aos seus atributos e não é intercambiável com outra, mesmo uma réplica exata. Podemos também pensar no certificado de autenticidade que acompanha uma pintura rara: ele também não é intercambiável com outro, devido a seus atributos únicos que representam a arte rara.
* **Indivisíveis**: Esses tokens estão disponíveis apenas como uma unidade completa. Por exemplo, um diploma universitário é um ativo único que se distingue de outros diplomas do mesmo tipo. Está associado a um indivíduo único e, portanto, não é intercambiável nem é racional dividi-lo em frações, tornando-o indivisível.

Na próxima seção, introduziremos outro tópico interessante, **tokens estáveis**, que são tokens com propriedades interessantes que os tornam mais atraentes para alguns investidores.

**Tokens estáveis**

Tokens estáveis ou *stable coins* são um tipo de token cujo valor está atrelado ao valor de outro ativo, como moedas fiduciárias ou metais preciosos. Tokens estáveis mantêm um preço estável em relação ao preço de outro ativo. Exemplos comuns de tokens estáveis são USDT, desenvolvido em 2014 pela Tether Limited, e USDC, introduzido em 2018 pela Coinbase e Circle.

O Bitcoin e outras criptomoedas semelhantes são inerentemente voláteis e experimentam flutuações dramáticas em seus preços. Essa volatilidade os torna inadequados para uso diário comum. Tokens estáveis surgiram como uma solução para essa limitação nos tokens.

A estabilidade de preço é mantida ao lastrear o token com um ativo estável. Isso é conhecido como **colateralização**. Moedas fiduciárias são estáveis porque são lastreadas por alguma reserva, como reservas cambiais ou ouro. Além disso, políticas monetárias sólidas e regulamentações pelas autoridades desempenham um papel vital na estabilidade de uma moeda. Tokens ou criptomoedas carecem desse tipo de suporte e, portanto, sofrem com a volatilidade.

Um dos principais exemplos de NFTs é o jogo **CryptoKitties** — <https://www.cryptokitties.co>. O CryptoKitties teve um grande papel na popularidade dos NFTs, pois foi o primeiro jogo (DApp) baseado em NFTs. A partir disso, a comunidade cresceu, assim como o interesse dos usuários no mecanismo de NFT subjacente. Um termo popular, “*crypto collectibles*” (colecionáveis cripto), também surgiu com esse desenvolvimento. Alguns outros projetos que usam NFT e oferecem colecionáveis cripto incluem *Gods Unchained*, *Decentraland*, *Skyweaver* e *My Crypto Heroes*.

Desde a edição anterior deste livro, a popularidade dos NFTs explodiu. Vamos discuti-los com mais detalhes no Capítulo 21, *Finanças Descentralizadas*.

**Tokens estáveis (continuação)**

Existem quatro tipos de *stable coins*:

* **Lastreadas em moeda fiduciária (fiat collateralized)**: Esses tokens estáveis são lastreados ou atrelados a uma moeda fiduciária tradicional, como dólares americanos. Moedas estáveis lastreadas em fiat são o tipo mais comum de *stable coin*. Algumas das moedas estáveis comuns disponíveis hoje são:
  + Gemini Dollar (GUSD) – <https://gemini.com/dollar>
  + Paxos (PAX) – <https://www.paxos.com/pax/>
  + USDT (Tether) – <https://tether.to/en/>
  + Libra – <https://libra.org/>
* **Lastreadas em commodities (commodity collateralized)**: Como o nome sugere, essas moedas estáveis são lastreadas por commodities fungíveis (ativos), como petróleo ou ouro. Exemplos comuns desses tokens são:
  + Digix gold tokens – <https://www.crunchbase.com/organization/digix-global>
  + Tether gold – <https://tether.to>
* **Lastreadas em criptoativos (crypto collateralized)**: Esse tipo de moeda estável é lastreado por outra criptomoeda. Por exemplo, a moeda estável Dai – <https://makerdao.com/en/> – aceita ativos baseados em Ethereum como colateral, além de manter um peg suave com o dólar americano.
* **Estáveis por algoritmo (algorithmically stable)**: Esse tipo de token estável não é lastreado por uma reserva, mas estabilizado por algoritmos que acompanham a oferta e demanda do mercado. Por exemplo, o Basecoin – <https://basecoin.cc> – pode ter seu valor atrelado ao dólar americano, a uma cesta (termo financeiro para grupo ou coleção de ativos semelhantes) de ativos ou a um índice. Ele também depende de um protocolo para controlar a oferta de tokens com base na taxa de câmbio entre ele mesmo e o peg utilizado, como o dólar americano. Esse mecanismo ajuda a manter a estabilidade do token.

**Tokens de segurança**

Tokens de segurança derivam seu valor de ativos negociáveis externos. Por exemplo, tokens de segurança podem representar ações de uma empresa. A diferença é que um valor mobiliário tradicional é mantido em um registro bancário e negociado em mercados secundários tradicionais, enquanto tokens de segurança estão disponíveis em um blockchain. Por serem valores mobiliários, são regidos por todas as leis e regulamentações tradicionais que se aplicam a valores mobiliários, mas devido à sua natureza um tanto descentralizada — na qual nenhum intermediário é necessário — os tokens de segurança são vistos como uma opção mais eficiente, escalável e transparente.

Agora que cobrimos diferentes tipos de tokens, vamos ver como um ativo pode ser tokenizado explorando o processo de tokenização.

**Processo de tokenização**

Nesta seção, apresentaremos o processo de tokenização, discutiremos o que pode ser tokenizado e forneceremos um exemplo básico da tokenização de ativos.

Praticamente qualquer ativo pode ser tokenizado e apresentado em um blockchain, como commodities, títulos, ações, imóveis, metais preciosos, empréstimos e até propriedade intelectual. Bens físicos que são tradicionalmente de natureza ilíquida, como colecionáveis, propriedade intelectual e arte, também podem ser tokenizados e transformados em ativos líquidos negociáveis.

Um processo genérico para tokenizar um ativo ou, em outras palavras, oferecer um token de segurança, é descrito aqui:

1. O primeiro passo é incorporar um investidor que esteja interessado em tokenizar seu ativo.
2. O ativo apresentado para tokenização é verificado e auditado, e a propriedade é confirmada. Essa auditoria é geralmente realizada pela organização que oferece o token de segurança. Pode ser uma corretora ou uma startup de criptomoeda.
3. O processo de segurança tokenizada começa, o que leva à oferta de token de segurança (*Security Token Offering* – STO).
4. O ativo físico é colocado sob custódia (no mundo real) para segurança.
5. O token derivado, representando o ativo, é criado pela organização que oferece o token e emitido aos investidores por meio de um blockchain.
6. Os negociadores começam a comprar e vender esses tokens usando corretoras em um mercado secundário, e essas negociações são liquidadas (o comprador faz um pagamento e recebe os bens) em um blockchain.

Plataformas comuns para tokenização incluem Ethereum, Solana e EOS. Tezos também está surgindo como outra plataforma para tokenização devido ao seu suporte a contratos inteligentes. De fato, qualquer blockchain que suporte contratos inteligentes pode ser usado para construir tokens; no entanto, os mais comuns são Ethereum e Solana.

Neste ponto, surge uma pergunta sobre como todos esses diferentes tipos de tokens chegam ao público para investimento. Na próxima seção, examinaremos como isso é alcançado, explicando primeiro o que são ofertas de tokens e analisando os diferentes tipos.

**Ofertas de tokens**

Ofertas de tokens são mecanismos para levantar fundos e obter lucro. Existem alguns tipos diferentes de ofertas de tokens. Vamos apresentar cada uma separadamente agora. Um atributo comum principal de cada um desses mecanismos é que eles são hospedados em um blockchain e utilizam tokens para facilitar diferentes atividades financeiras. Essas atividades financeiras podem incluir financiamento coletivo (*crowdfunding*) e negociação de valores mobiliários.

**Ofertas iniciais de moedas (Initial Coin Offerings – ICOs)**

Oferta inicial de moedas ou oferta inicial de moeda (ICO) é um mecanismo para levantar fundos usando criptomoedas. As ICOs têm sido um mecanismo muito bem-sucedido, embora um tanto controverso, para levantar capital para novos projetos de criptomoedas ou tokens. As ICOs são por vezes controversas devido a um projeto mal desenhado ou má governança, mas em outros casos, algumas ICOs revelaram-se fraudes completas. Uma lista de esquemas fraudulentos está disponível em:  
<https://dfpi.ca.gov/crypto-scams/>.  
Esses incidentes contribuíram para a má reputação e controvérsia que cercam as ICOs em geral. Embora existam alguns golpes, também há muitas ICOs legítimas e bem-sucedidas. O retorno sobre o investimento (ROI) de várias ICOs foi bastante elevado e contribuiu para o sucesso sem precedentes de muitos desses projetos. Alguns desses projetos são Ethereum, NXT, NEO, IOTA e QTUM.

Uma pergunta comum aqui é sobre a diferença entre ofertas públicas iniciais tradicionais (IPOs) e ICOs, já que ambos os mecanismos são fundamentalmente projetados para levantar capital. A diferença é simples: ICOs são ofertas de tokens baseadas em blockchain que geralmente são iniciadas por startups para levantar capital permitindo que investidores invistam em sua startup. Normalmente, nesse caso, as contribuições dos investidores são feitas usando criptomoedas já existentes e estabelecidas, como Bitcoin ou ether. Como retorno, quando o projeto é concluído, os investidores iniciais recebem seu retorno sobre o investimento.

Por outro lado, IPOs são mecanismos tradicionais usados por empresas para distribuir ações ao público. Isso é feito usando os serviços de *underwriters*, que geralmente são bancos de investimento. IPOs geralmente só são permitidas para empresas que já estão bem estabelecidas, enquanto as ICOs podem ser oferecidas por startups. IPOs também oferecem dividendos como retorno, enquanto as ICOs oferecem tokens que se espera que valorizem quando o projeto for lançado.

Outra comparação importante é que IPOs são mecanismos tradicionais regulados e centralizados, enquanto as ICOs são descentralizadas e não regulamentadas.

Como as ICOs têm sido não regulamentadas, o que resultou em vários golpes e pessoas perdendo seu dinheiro, outro tipo de captação de recursos conhecido como **oferta de tokens de segurança (Security Token Offerings – STO)** foi introduzido. Vamos discutir isso a seguir.

**Ofertas de tokens de segurança (Security Token Offerings – STOs)**

STOs são um tipo de oferta em que valores mobiliários tokenizados são negociados em corretoras de criptomoedas. Valores mobiliários tokenizados ou tokens de segurança podem representar qualquer ativo financeiro, incluindo commodities e valores mobiliários. Os tokens oferecidos em STOs são classificados como valores mobiliários. Como tal, são mais seguros e podem ser regulamentados, em contraste com as ICOs. Se um STO é oferecido em uma bolsa de valores tradicional, então ele é conhecido como uma **IPO tokenizada** (*Tokenized IPO*). IPO tokenizada é outro nome para STO, usado quando um STO é oferecido em uma bolsa de valores regulada, o que ajuda a diferenciar entre um STO oferecido em uma bolsa tradicional regulamentada e um oferecido em corretoras de criptomoedas. Os STOs são regulamentados sob a *Markets in Financial Instruments Directive – MiFID II* na União Europeia.

**Ofertas iniciais em corretora (Initial Exchange Offerings – IEOs)**

A IEO é outra inovação no espaço de tokenização. A principal diferença entre uma IEO e uma ICO é que, em uma ICO, os tokens são distribuídos via mecanismos de financiamento coletivo para carteiras dos investidores, mas em uma IEO, os tokens são disponibilizados por meio de uma corretora (*exchange*).

As IEOs são mais transparentes e confiáveis do que as ICOs devido ao envolvimento de uma corretora e à diligência prévia realizada por ela.

**Ofertas de tokens de participação acionária (Equity Token Offerings – ETOs)**

ETOs são outra variação de ICOs e STOs. ICOs oferecem tokens utilitários, enquanto tokens de participação acionária são oferecidos nas ETOs. Quando comparadas com STOs, as ETOs oferecem ações de uma empresa, enquanto os STOs oferecem ações em qualquer ativo, como moedas ou commodities. Sob esse ponto de vista, as ETOs podem ser consideradas um tipo específico de STO, onde apenas ações de uma empresa ou empreendimento são representadas como tokens.

Existem alguns exemplos em que meios tradicionais, como a assinatura de contratos em papel, foram usados para fornecer algum nível de proteção legal para as partes envolvidas, mas as ICOs são amplamente não regulamentadas. Isso é especialmente verdadeiro do ponto de vista de grandes órgãos reguladores financeiros e serviços de ouvidoria que fornecem proteção ao investidor e lidam com reclamações de consumidores na indústria de serviços financeiros tradicional.

**Oferta inicial de moeda autônoma descentralizada**

(**Decentralized Autonomous Initial Coin Offering – DAICO**)

A **DAICO** é uma combinação de **Organizações Autônomas Descentralizadas (DAOs)** e ICOs que permite aos investidores terem mais controle sobre seu investimento, sendo vista como uma versão mais segura, descentralizada e automatizada das ICOs.

**Outras ofertas de tokens**

Variações diferentes de ICOs e novos conceitos estão sendo introduzidos com bastante frequência, e espera-se apenas que a inovação continue crescendo nessa área.

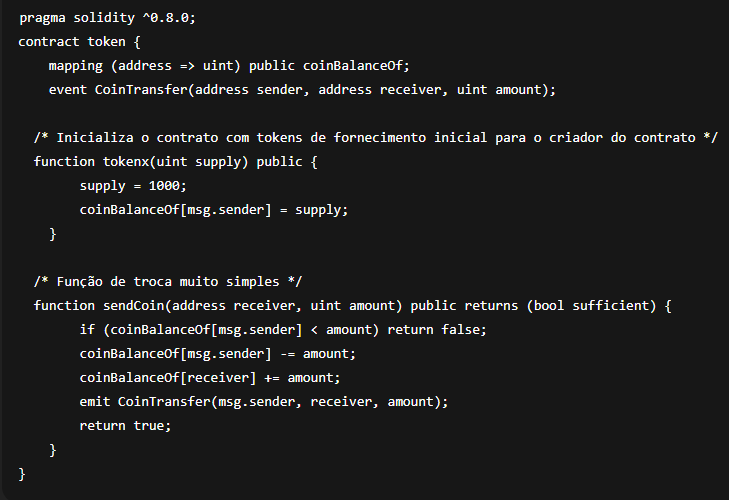
Uma **comparação entre diferentes tipos de ofertas de tokens** é apresentada na tabela a seguir:

| **Nome** | **Conceito** | **Introdução** | **Escala de descentralização** | **Como investir** | **Regulação** |
| --- | --- | --- | --- | --- | --- |
| **ICO** | Financiamento coletivo por meio de um token utilitário | Julho de 2013 com Mastercoin | Semidescentralizada | Investidores enviam criptomoeda para um contrato inteligente liberado pelo ofertante do token | Baixa regulação |
| **STO** | Título tokenizado como ações e títulos | 2017 | Um tanto centralizada | Usa a plataforma fornecida pela corretora | Regulada sob leis e diretrizes estabelecidas em muitas jurisdições |
| **IEO** | Tokens disponibilizados por meio de uma corretora | 2018 | Um tanto centralizada | Usa a plataforma fornecida pela corretora | Baixa regulação |
| **ETO** | Oferece ações de qualquer ativo | Dezembro de 2018 com a ETO da Neufund | Um tanto centralizada | Usa a plataforma fornecida pela corretora | Maior parte regulada |
| **DAICO** | Combinação de DAO e ICO | Maio de 2018 com ABYSS DAICO | Maior parte descentralizada | Investidores enviam criptomoeda para o contrato inteligente DAICO | Baixa regulação |

Com todos esses diferentes tipos de tokens e processos associados, surge naturalmente a necessidade de padronização para que mais adoção e interoperabilidade possam ser alcançadas. Para esse fim, diversos padrões de desenvolvimento foram propostos, os quais discutiremos a seguir.

**Padrões de tokens**

Com o advento de plataformas de contratos inteligentes como o Ethereum, tornou-se bastante fácil criar um token usando um contrato inteligente. Tecnicamente, um token ou moeda digital pode ser criado no Ethereum com apenas algumas linhas de código, como mostrado no exemplo a seguir:



O código acima funciona e pode ser usado para criar e emitir tokens. No entanto, o problema é que, sem um mecanismo padrão, cada pessoa implementaria contratos inteligentes de tokenização à sua maneira, com base em seus próprios requisitos. Essa falta de padronização resultará em problemas de interoperabilidade e usabilidade, o que impede a ampla adoção da tokenização.

Para resolver isso, o primeiro padrão de tokenização foi proposto no Ethereum, conhecido como **ERC-20**.

**ERC-20**

ERC-20 é o padrão de token mais famoso na plataforma Ethereum. Muitas ofertas de tokens são baseadas no ERC-20 e há carteiras disponíveis, como a MetaMask, que oferecem suporte a tokens ERC-20.

Este código é baseado em um dos primeiros códigos publicados pelo Ethereum como exemplo no ethereum.org.

O ERC-20 foi introduzido em **novembro de 2015** e, desde então, tornou-se um padrão muito popular para tokens fungíveis. Esse padrão foi usado em muitas ICOs e resultou em moedas digitais (tokens) valiosas nos últimos anos, incluindo EOS, Golem e muitos outros. Tokens famosos como **USDT**, **BNB**, **USDC** e **Matic** são todos baseados no padrão ERC-20. Há quase **1.000 projetos de tokens ERC-20 listados no Etherscan** (<https://etherscan.io/tokens>), o que é uma indicação clara da popularidade desse padrão.

Embora o ERC-20 tenha definido um padrão para tokens fungíveis e tenha sido amplamente adotado, ele tem algumas falhas que resultam em problemas de segurança e usabilidade. Por exemplo, uma falha de segurança no ERC-20 resulta em perda de fundos se os tokens forem enviados para um contrato inteligente que não tem funcionalidade para lidar com tokens. Os tokens efetivamente “queimados” resultam em perda de fundos para o usuário.

Para resolver essas deficiências, foi proposto o **ERC-223**.

**ERC-223**

ERC-223 é um padrão de token fungível. Uma grande vantagem do ERC-223 em comparação com o ERC-20 é que ele consome apenas **50% do gás** utilizado pelo ERC-20, o que o torna **menos caro** para usar na rede principal do Ethereum. O ERC-223 é compatível com versões anteriores do ERC-20 e é usado em vários projetos importantes de tokens, como **LINK** e **CNexchange (CNEX)**.

**ERC-777**

O principal objetivo do ERC-777 é abordar algumas das limitações do ERC-20 e do ERC-223. Ele é compatível com versões anteriores do ERC-20. Define diversos recursos avançados para interagir com tokens ERC-20. Ele permite o envio de tokens em nome de outro endereço (contrato ou conta). Além disso, introduz um recurso de **“ganchos” (*hooks*)**, que permite que os detentores de tokens tenham mais controle sobre seus tokens.

A proposta de melhoria do Ethereum (Ethereum Improvement Proposal – EIP) está disponível aqui:  
<https://eips.ethereum.org/EIPS/eip-777>

**ERC-721**

ERC-721 é um padrão de **token não fungível (NFT)**. O ERC-721 exige várias regras que devem ser implementadas em um contrato inteligente para que este esteja em conformidade com o ERC-721. Essas regras governam como esses tokens podem ser gerenciados e negociados. O ERC-721 foi popularizado pelo projeto **CryptoKitties**. CryptoKitties é um jogo baseado em blockchain que permite aos jogadores criar (*cruzar*) e negociar diferentes tipos de gatos virtuais no blockchain. Cada “gatinho” é **único** e **negociável por um valor** no blockchain.

**ERC-884**

Este é o padrão para tokens de ações compatíveis com o ERC-20 que estão em conformidade com a **Lei Geral de Corporações de Delaware (Delaware General Corporations Law)**.

O EIP para o padrão de token está disponível aqui:  
<https://github.com/ethereum/EIPs/blob/master/EIPS/eip-884.md>

A legislação está disponível aqui:  
https://legis.delaware.gov/json/BillDetail/GenerateHtmlDocument?legislationId=25730&legislationTypeId=1&docTypeId=2&legislationName=SB69

**ERC-1400**

Este é um padrão de **token de segurança**, que define como construir, emitir, negociar e gerenciar tokens de segurança. Sob o ERC-1400, há alguns outros padrões, que são os seguintes:

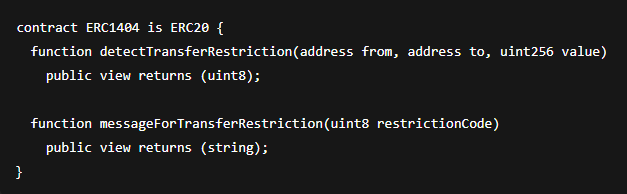
* **ERC-1410**: Um padrão de token parcialmente fungível que define uma interface padrão para agrupar os tokens de um proprietário em partições.
* **ERC-1594**: Define um padrão central de token de segurança.
* **ERC-1643**: Padrão de gerenciamento de documentos.
* **ERC-1644**: Padrão de operações de controle de tokens.
* **ERC-1066**: Especifica uma forma padrão de projetar códigos de status no Ethereum.

O objetivo do ERC-1400 e dos padrões associados é cobrir todas as atividades relacionadas à emissão, gerenciamento, controle e processamento de tokens de segurança.

**ERC-1404**

Esse padrão ERC permite a emissão de tokens com **restrições regulatórias de transferência**. Essas restrições permitem que os usuários controlem a transferência de tokens de diferentes formas. Por exemplo, os usuários podem controlar **quando**, **para quem** e **sob quais condições** os tokens podem ser transferidos. Um emissor, por exemplo, pode optar por emitir tokens apenas para um destinatário presente em uma lista autorizada (*whitelist*), ou verificar se há restrições de tempo nos tokens do remetente.

O ERC-1404 introduz duas novas funções no padrão ERC-20 para introduzir mecanismos de restrição e controle. Essas duas funções são listadas a seguir:



Mais informações sobre o ERC-1404 estão disponíveis em:  
<https://erc1404.org>

**ERC-1155**

Este é o **padrão de multi-tokens** (*multi-token standard*). Esse padrão permite o **agrupamento de transações** para economizar custos e realizar negociações de forma mais eficiente. Ele pode ser usado para criar **tokens utilitários** e **NFTs**. A ideia é criar uma interface de contrato inteligente que possa lidar e representar qualquer número de tipos fungíveis e não fungíveis.

Isso significa que o ERC-1155 pode funcionar como ERC-20 e ERC-721, **até mesmo ao mesmo tempo**. Ele também melhora a eficiência e a precisão dos padrões ERC-20 e ERC-721.

Mais informações estão disponíveis no padrão EIP-1155 – Multi-token standard:  
<https://eips.ethereum.org/EIPS/eip-1155>

**ERC-4626**

Este é o **padrão de cofres tokenizados** (*tokenized vault standard*), que visa otimizar e unificar os parâmetros técnicos dos cofres geradores de rendimento (*yield-bearing vaults*) tokenizados.

Cofres tokenizados são utilizados no Ethereum em diferentes protocolos, como **Yearn Finance**, **Tulip Garden** e **Pickle Finance**. Esses cofres permitem que um detentor de tokens **deposite seus tokens no cofre** e ganhe um **rendimento** sobre esses tokens. Quando um titular deposita tokens em um cofre, ele recebe um **token do cofre** (*vault token*). Esse token pode valorizar com o tempo. O detentor pode obter seus tokens originais de volta quando devolver os tokens do cofre.

O problema é que **não existe um padrão unificado** para implementar operações padrão como depósitos e saques. Essa diferença nas implementações cria atrito entre diferentes protocolos e resulta em tentativas de integração que podem ou não funcionar corretamente. Por exemplo, se você quiser escrever um DApp que interaja com todos esses diferentes cofres tokenizados, então será necessário escrever **adaptadores** para conversar com todos eles, o que dificulta a vida de integradores e agregadores.

O ERC-4626 visa resolver esse problema ao fornecer uma forma padronizada de interagir com esses cofres. Ele fornece uma interface que padroniza depósitos, saques, leitura de saldos e outros parâmetros. Este padrão resultará em **interoperabilidade**, **composabilidade** e melhoria de todo o ecossistema de finanças descentralizadas e blockchain.

Mais detalhes sobre o padrão estão disponíveis aqui:  
<https://eips.ethereum.org/EIPS/eip-4626>

Com isso, concluímos nossa introdução aos padrões ERC. Agora, depois de toda essa base teórica, vamos ver como um token pode ser criado no Ethereum. Vamos construir nosso próprio token compatível com ERC-20.

**Criando um token ERC-20**

Nesta seção, construiremos um token ERC-20. Em capítulos anteriores, vimos várias maneiras de desenvolver contratos inteligentes, incluindo escrever contratos inteligentes no **Visual Studio Code**, compilá-los e depois implantá-los em uma rede blockchain. Também usamos o **Remix IDE**, **Truffle** e **MetaMask** para experimentar várias formas de desenvolver e implantar contratos inteligentes.

Nesta seção, usaremos um método rápido para desenvolver, testar e implantar nosso contrato inteligente na **rede de testes Sepolia**. Não usaremos o Truffle ou o Visual Studio Code neste exemplo, pois já vimos esse método antes. Vamos explorar uma maneira diferente e mais rápida de construir e implantar nosso contrato.

Neste exemplo, veremos quão rápida e facilmente podemos construir e implantar nosso próprio token na rede blockchain do Ethereum. O objetivo deste exercício é entender como o MetaMask pode ser usado para implantar nosso novo contrato inteligente de token em uma rede Ethereum. Também veremos como podemos importar tokens ERC-20 no MetaMask e usá-los para transferir fundos de uma conta para outra.

Usaremos os seguintes componentes no nosso exemplo, que foram introduzidos no **Capítulo 10 – Ethereum na prática**:

* **Remix IDE**, disponível em <http://remix.ethereum.org>. Observe que, no exemplo a seguir, a interface do usuário e os passos exigidos podem ser ligeiramente diferentes dependendo do estágio de desenvolvimento. Isso ocorre porque o Remix IDE está em intenso desenvolvimento, e novos recursos estão sendo adicionados a um ritmo acelerado — assim, são esperadas algumas mudanças na interface. No entanto, os fundamentos não devem mudar, e nenhuma alteração importante é esperada na interface do usuário.
* **MetaMask**. Usaremos a rede que configuramos no Capítulo 10, mas você pode criar uma nova conta, se necessário.

Agora, vamos começar a escrever nosso código.

**Construindo o contrato em Solidity**

Primeiro, exploramos como é a interface do ERC-20 e, em seguida, começaremos a escrever nosso contrato inteligente passo a passo no Remix IDE.

A interface ERC-20 define várias **funções** e **eventos** que devem estar presentes em um contrato inteligente compatível com ERC-20. Algumas regras adicionais que devem estar presentes em um token compatível com ERC-20 são listadas a seguir:

* **totalSupply**: Esta função retorna o número total de tokens emitidos:

function totalSupply() public view returns (uint256)

**balanceOf**: Esta função retorna o saldo do proprietário do token:

function balanceOf(address \_owner) public view returns (uint56 balance)

**transfer**: Esta função recebe o endereço do destinatário e um valor especificado como número de tokens, e transfere esse valor para o endereço informado:

function transfer(address \_to, uint256 \_value) public returns (bool success)

**transferFrom**: Esta função recebe os parâmetros \_from (endereço do remetente), \_to (endereço do destinatário) e \_value (quantia), e retorna verdadeiro ou falso. Ela é usada para transferir fundos de uma conta para outra:

function transferFrom(address \_from, address \_to, uint256 \_value) public returns (bool success)

**approve**: Esta função recebe \_spender (destinatário) e \_value (número de tokens) como parâmetros, e retorna um valor booleano. É usada para autorizar o endereço do destinatário a realizar transferências em nome do proprietário do token até o valor aprovado:

function approve(address \_spender, uint256 \_value) public returns (bool success)

**allowance**: Esta função recebe o endereço do proprietário do token e o endereço do destinatário, e retorna o número restante de tokens que o destinatário tem permissão para sacar do proprietário do token:

function allowance(address \_owner, address \_spender) public view returns (uint256 remaining)

Há também três funções opcionais, listadas a seguir:

* **name**: Esta função retorna o nome do token como uma string:

function name() public view returns (string)

**symbol**: Retorna o símbolo do token como uma string:

function symbol() public view returns (string)

**decimals**: Esta função retorna o número de casas decimais que o token utiliza, como um número inteiro:

function decimals() public view returns (uint8)

Por fim, há dois eventos que devem estar presentes em um token compatível com ERC-20:

* **Transfer**: Este evento deve ser acionado quando os tokens forem transferidos, incluindo quaisquer transferências de valor zero:

event Transfer(address indexed \_from, address indexed \_to, uint256 \_value)

**Approval**: Este evento deve ser acionado quando uma chamada bem-sucedida for feita para a função approve:

event Approval(address indexed \_owner, address indexed \_spender, uint256 \_value)

Agora vamos ver o **código-fonte** do nosso token ERC-20. Ele é escrito em **Solidity**, que já conhecemos e exploramos em detalhes no **Capítulo 11 – Ferramentas, Linguagens e Frameworks para Desenvolvedores Ethereum**.

O código-fonte para nosso token ERC-20 é o seguinte — vamos explicá-lo passo a passo antes de escrevê-lo no Remix IDE.

Primeiro está o **identificador de licença SPDX**:

// SPDX-License-Identifier: GPL-3.0

Em seguida, temos a **versão do compilador e linguagem Solidity**, que especifica a versão do compilador para a qual nosso programa foi escrito:

pragma solidity ^0.8.0;

Depois criamos o **objeto do contrato**:

contract MyERC20Token {

Após isso, o objeto do contrato é definido com o nome MyERC20Token.

**Código do contrato (continuação)**

Depois, definimos os **mapeamentos**:

mapping (address => uint256) \_balances;

mapping (address => mapping(address => uint256)) \_allowed;

Esses são os dois mapeamentos usados no contrato inteligente ERC-20. O primeiro é para armazenar os saldos, e o outro é usado para armazenar as permissões (*allowances*).

Essas são as **variáveis de estado**:

string public name = "My ERC20 Token";

string public symbol = "MET";

uint8 public decimals = 0;

uint256 private \_totalSupply = 100;

Elas descrevem o nome, símbolo, casas decimais e o fornecimento total do nosso token.

Este é o **evento Transfer**:

event Transfer(address indexed \_from, address indexed \_to, uint256 \_value);

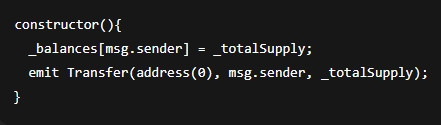
Ele possui três parâmetros: from, to e value. from representa o endereço de onde os tokens estão vindo, to é a conta para a qual os tokens estão sendo transferidos, e value é o número de tokens.

Este é o **evento Approval**:

event Approval(address indexed \_owner, address indexed \_spender, uint256 \_value);

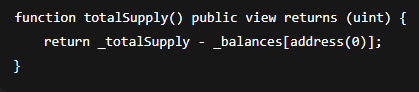
Ele possui três parâmetros: endereço do proprietário, endereço do destinatário e valor. A palavra-chave indexed permite que pesquisemos um item de log específico em vez de buscar por todos os logs. Isso habilita filtragem de logs para extrair apenas os dados necessários, em vez de retornar todos os logs.

Este é o **construtor**, que é executado quando o contrato é criado:



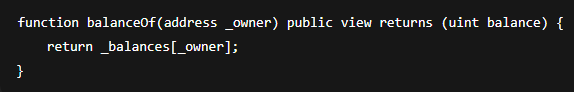
Ele é opcional em Solidity e é usado para executar código de inicialização. Em nosso exemplo, o código de inicialização contém as instruções para transferir todo o saldo (\_totalSupply) para o criador do contrato inteligente; neste caso, msg.sender. Também emite o evento Transfer, indicando que a transferência ocorreu de address(0) para msg.sender (o criador do contrato) e de \_totalSupply, que é **100** no nosso exemplo, apenas para manter as coisas simples.

Esta é a função **totalSupply**:



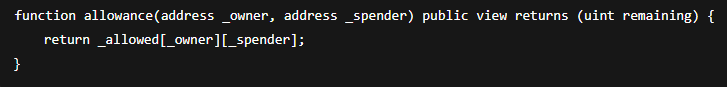
Esta função retorna a quantidade total de tokens, subtraída do saldo da conta address(0).

Esta é a função **balanceOf**:



A função balanceOf retorna o saldo do proprietário do token.

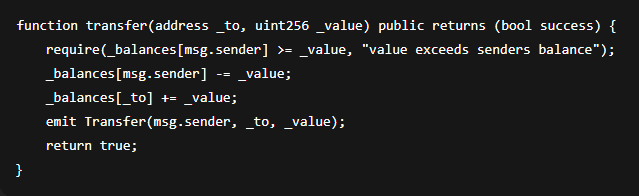
A função **allowance** é a seguinte:



A função allowance retorna o total restante de tokens permitidos.

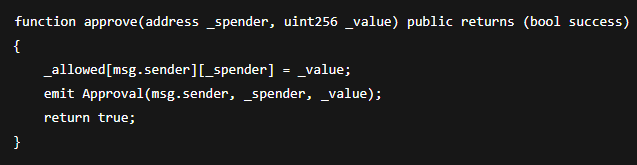
**Código do contrato (continuação)**

Esta é a função **transfer**, que retorna true ou false dependendo do resultado da execução:



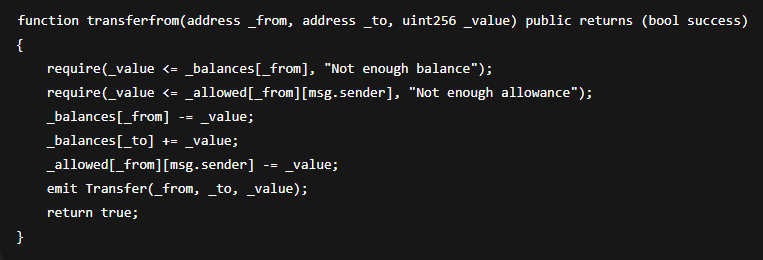
A função de conveniência require é usada para verificar determinadas condições e lançar uma exceção caso essas condições não sejam atendidas. Em nosso exemplo, require verifica se o valor excede o saldo do remetente, e se exceder, uma mensagem de erro será gerada indicando que o "valor excede o saldo do remetente". Se essa verificação for bem-sucedida, a transferência ocorre, e após emitir o evento Transfer, a função retorna true, indicando a transferência bem-sucedida dos tokens.

Esta é a função **approve**, que retorna true ou false dependendo do resultado da execução:



Esta função recebe \_spender (o destinatário) e \_value (número de tokens) como argumentos e serve como um mecanismo para fornecer autorização ao usuário para adquirir o número permitido de tokens do nosso contrato ERC-20.

Esta é a função **transferFrom**, que pode ser usada para automatizar a transferência de tokens de um endereço para outro:



Ela recebe três parâmetros: \_from, \_to e \_value. Retorna true ou false dependendo da execução da função. Primeiro, com as funções require, são verificadas as condições de saldo e permissões para garantir que haja saldo e autorização suficientes. Depois disso, a transferência ocorre, e eventualmente o evento Transfer é emitido, seguido pelo retorno de true, indicando a transferência bem-sucedida.

Agora que entendemos o que nosso código-fonte faz, o próximo passo é escrevê-lo no **Remix IDE** e implantá-lo.

**Implantando o contrato na Máquina Virtual JavaScript do Remix**

Neste passo, simplesmente pegamos o código-fonte e o escrevemos ou colamos diretamente no Remix IDE. Para isso, siga os passos a seguir:

1. Abra o **Remix IDE**.
2. Quando o Remix IDE iniciar, selecione **SOLIDITY** sob *Environments*, pois vamos escrever contratos inteligentes usando a linguagem Solidity.
3. Após selecionar o ambiente Solidity, crie um novo arquivo escolhendo a opção **FILE EXPLORERS** na lista de ícones no lado esquerdo e adicione um novo arquivo clicando no sinal de **+**.
4. Crie um novo arquivo no Remix IDE chamado erc20example.sol.
5. Depois de criar o novo arquivo, simplesmente escreva o código-fonte no IDE ou cole-o diretamente.
6. Compile o código-fonte. Para isso, clique em **Compile erc20example.sol**. Opcionalmente, **Auto compile** também pode ser selecionado em **COMPILER CONFIGURATION**, o que compilará o código automaticamente assim que for escrito no IDE.

**<IMAGEM>**  
*Figura 15.1: Compilador Solidity no Remix*

**Implantando o contrato na Máquina Virtual JavaScript do Remix (continuação)**

1. Uma vez compilado com sucesso, ele estará pronto para ser implantado. Primeiro, vamos implantá-lo na **JavaScript VM** disponível no Remix IDE para garantir que tudo funcione. Depois de testarmos e confirmarmos que ele pode ser implantado corretamente e funciona conforme o esperado, poderíamos implantá-lo na rede principal utilizando o MetaMask. Em nosso exemplo, no entanto, vamos implantá-lo na **rede de testes Sepolia** (<https://sepolia.dev>) usando o MetaMask.

Lembre-se de que temos algum ether restante do nosso exercício no Capítulo 13, *The Merge and Beyond*, e temos uma conta de teste na rede Sepolia. Podemos usar a mesma conta para este exemplo ou criar uma nova conta e financiá-la utilizando o processo descrito naquele capítulo. Como alternativa, você pode usar *faucets* disponíveis para a rede Sepolia em:  
<https://sepolia-faucet.pk910.de>

1. Após a compilação, implantamos o contrato inteligente usando a interface **Deploy & Run Transactions** disponível no Remix IDE. Certifique-se de que o ambiente **JavaScript VM** esteja selecionado, uma conta esteja escolhida para a implantação e, em seguida, selecione **Deploy**.

**<IMAGEM>**  
*Figura 15.2: Implantando e executando transações no Remix*

Uma vez implantado, o contrato estará disponível sob **Deployed Contracts** e, nos registros (*logs*), podemos ver detalhes relevantes sobre a implantação do contrato.

Para implantar na rede principal, basta escolher **main net** no MetaMask após garantir que há fundos suficientes disponíveis para implantá-lo na rede principal.

**Logs de implantação**

O mais importante é que podemos ver nos registros que, quando o contrato é criado, o primeiro evento emitido é **Transfer**. Aqui, podemos ver que todos os 100 tokens foram transferidos para a conta do proprietário, 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8:



Uma vez implantado, veremos sob **Deployed Contracts** todas as funções expostas pelo contrato no Remix IDE:

**<IMAGEM>**  
*Figura 15.3: Contratos implantados – funções expostas no Remix para nosso contrato*

Podemos testar a funcionalidade do nosso contrato usando essa interface. Por exemplo, ao chamar totalSupply, veremos o valor **100**, que indica o número de tokens, e ao chamar symbol, veremos a string **MET**, o símbolo do nosso token.

**Implantando na rede de testes Sepolia com MetaMask**

Neste ponto, nosso contrato funciona e o testamos localmente. Agora podemos implantá-lo na **rede de testes Sepolia**. Usaremos o **MetaMask** para esse propósito:

1. No **Remix IDE**, sob **DEPLOY & RUN TRANSACTIONS**, selecione a opção **Injected Web3** na lista **ENVIRONMENTS**. Este é o ambiente de execução fornecido ao ativar o Web3 no navegador por meio da extensão MetaMask.
2. Em seguida, confirme que o MetaMask está em execução e conectado à **rede de testes Sepolia**. Isso deve ser fácil de verificar, pois usamos essa mesma rede no **Capítulo 13 – The Merge and Beyond**.

Se tudo estiver funcionando no MetaMask, ele deve exibir uma tela semelhante à mostrada abaixo. Observe que talvez seja necessário fazer login novamente no MetaMask. Uma vez logado, selecione a rede Sepolia e uma conta que tenha algum ether nela.

**<IMAGEM>**  
*Figura 15.4: Rede de testes Sepolia no MetaMask*

1. No Remix IDE, escolha **Injected Provider – MetaMask** no campo **ENVIRONMENT** em **DEPLOY & RUN TRANSACTIONS**. Uma vez conectado, você notará que a rede Sepolia (Sepolia (11155111)) será exibida no Remix IDE logo abaixo do campo *Environment*.

Observe que o MetaMask também mostrará que está conectado à rede Sepolia, e exibirá também a conta que configuramos no MetaMask. Agora estamos prontos para implantar isso na rede de testes Sepolia.

1. Clique em **Deploy**, o que abrirá a janela do MetaMask para confirmar a transação.
2. Clique em **Confirm**, e o contrato será implantado. Podemos ver isso no histórico do MetaMask. Além disso, nos registros do Remix IDE, podemos verificar se ele foi implantado com sucesso.

Assim como nos testes que realizamos anteriormente ao implantar nosso contrato na JavaScript VM, podemos invocar diferentes funções expostas pelo nosso novo token ERC-20 diretamente do Remix IDE. Também podemos ver nosso token no rastreador de tokens do Etherscan, por meio do seguinte link:

<https://sepolia.etherscan.io/tx/0xfd427145a393fb6dbb08bf9f3e4c945acb1039404503380692b81b366b0f23e6>

A página do Etherscan também mostra informações detalhadas sobre o contrato do nosso token, como fornecimento total, endereço do contrato etc., acessível aqui:

<https://sepolia.etherscan.io/token/0x07c152a6ab577e8f78e3bede502d79652787a9fc>

Assim, temos nosso próprio token **MET** implantado na rede de testes Sepolia. Agora, se desejado, poderíamos implantá-lo na rede principal Ethereum, desde que tenhamos algum ether disponível. Podemos realizar os mesmos passos para implantá-lo, com a única diferença sendo que, no MetaMask, selecionaremos a **main net** como a rede.

**Adicionando tokens no MetaMask**

Uma vez que implantamos nosso contrato e nosso token ERC-20 está agora no blockchain (por assim dizer), a menos que sejamos capazes de visualizá-lo e realizar operações sobre ele, o token por si só não tem muita utilidade.

Para realizar operações sobre o token, podemos criar comandos manualmente usando o método sendRawTransaction da biblioteca **Web3.js** e usar o console de linha de comando do **geth** para interagir com ele. Como alternativa, podemos usar uma opção mais fácil e simplesmente **adicionar o token a uma carteira**. As carteiras abstraem as complexidades associadas à criação e gerenciamento de transações e fornecem uma interface fácil de usar para realizar transferências e tarefas semelhantes.

O **MetaMask** pode servir como uma carteira para tokens e fornece uma interface para adicionar tokens. Nesta seção, veremos como podemos adicionar nosso token **MET** ao MetaMask e realizar algumas operações com ele:

1. Abra o MetaMask e encontre a opção **Add Token** na interface principal, onde diz "Don’t see your token? Import tokens" (*Não vê seu token? Importe tokens*) e clique em **Import token**. Alternativamente, abra o menu da conta e ele exibirá a opção de importação de conta.
2. Clique em **Add Token** e selecione **Custom Token**.

**Adicionando tokens no MetaMask (continuação)**

1. Insira o **endereço do contrato**:  
   0x07C152A6ab577E8F78e3bedE502D79652787A9FC  
   Esse campo preencherá automaticamente o **símbolo do token** e os **pontos decimais**, como mostrado na imagem a seguir:

**<IMAGEM>**  
*Figura 18.5: Token personalizado no MetaMask*

1. Pressione **Add custom token** (*Adicionar token personalizado*) e depois pressione **Import tokens** (*Importar tokens*) na próxima tela. Agora temos **100 MET** em nossa carteira MetaMask! Vamos enviar 10 MET para outra conta na rede Sepolia.
2. Clique em **Send**, depois insira os dados da conta de destino.
3. Insira **10 MET** e clique em **Next**.
4. Em seguida, clique em **Confirm**, e a transação será processada.
5. Agora, voltando ao **Remix IDE**, podemos ver que nosso saldo foi reduzido em 10 MET. Podemos verificar isso passando o endereço do detentor do token:

0xb838042b89ebdfb1d5ed0de32323f798eb5dc96e

e clicando no botão **balanceOf**. Da mesma forma, podemos verificar o saldo da conta de destino:

0xb838042b89ebdfb1d5ed0de32323f798eb5dc96e

para a qual transferimos 10 MET, e agora ela possui 10 MET. Você pode verificar isso passando o mesmo endereço para a função balanceOf.

Podemos ver todas as transferências do nosso token ERC-20, **MET**, no **Etherscan**:

<https://sepolia.etherscan.io/token/0x07c152a6ab577e8f78e3bede502d79652787a9fc>

Com isso, cobrimos como **criar um token ERC-20 do zero** e **implantá-lo na blockchain do Ethereum**.

**Conceitos emergentes**

Com o advento do blockchain e da tokenização, diversos **novos conceitos** surgiram nos últimos anos. Apresentaremos alguns deles agora.

**Tokenomics / Economia de tokens**

**Tokenomics** ou **economia de tokens** é uma disciplina emergente preocupada com o estudo da atividade econômica, modelos econômicos e o impacto da tokenização. Ela lida com os bens e ativos que foram tokenizados e com as entidades envolvidas em todo o processo de emissão, venda, compra e investimento em tokens.

Com o uso do mecanismo de prova de trabalho (**Proof of Work – PoW**) no Bitcoin, demonstrou-se pela primeira vez que protocolos computacionais podem ser projetados de modo que atacar um sistema **não resulte** em uma vantagem injusta ou benefício financeiro. Esse conceito evoluiu até o que hoje chamamos de **criptoeconomia** (*cryptoeconomics*). Isso também pode ser entendido como uma combinação de **economia**, **teoria dos jogos** e **criptografia**.

Talvez você já tenha ouvido o termo **criptoeconomia**, que é um termo relacionado, mas um pouco diferente. Criptoeconomia trata dos mesmos tópicos, mas é um **conjunto maior** que inclui a **tokenomics**. Em outras palavras, tokenomics é um subconjunto da criptoeconomia. Tokenomics se preocupa apenas com tokens e ecossistemas de tokenização, mas não inclui as redes blockchain mais amplas, os protocolos e as criptomoedas.

**Tokenomics / Economia de tokens (continuação)**

Tokenomics, portanto, lida com:

* a modelagem econômica de um token;
* a análise de seu fornecimento e demanda;
* os incentivos fornecidos a investidores e participantes da rede;
* a emissão, distribuição, destruição (*burning*) e outras políticas relacionadas ao token;
* o impacto geral de um token sobre o sistema ou rede.

Vários projetos de blockchain e contratos inteligentes oferecem documentos de **tokenomics** que descrevem todos esses aspectos de seus tokens. Esses documentos são geralmente partes importantes dos *white papers* e ajudam investidores, desenvolvedores e usuários a entenderem como o token será tratado. A seguir estão alguns componentes típicos de um modelo de tokenomics:

* **Modelo de fornecimento**: define o fornecimento total do token, cronograma de emissão, métodos de criação e destruição do token;
* **Distribuição**: define como os tokens serão alocados entre diferentes partes interessadas;
* **Governança**: define como decisões serão tomadas em relação ao token;
* **Incentivos**: descreve como os participantes da rede serão incentivados a contribuir com o sistema;
* **Utilidade**: define o que pode ser feito com o token — ele pode ser usado para acesso, participação, governança, etc.;
* **Mecanismos de valorização**: descreve como o valor do token pode ser mantido ou aumentado;
* **Política monetária**: define como serão tratadas inflação e deflação no sistema.

**Conclusão**

Neste capítulo, aprendemos sobre **tokenização**, **tipos de tokens**, diferentes **métodos de emissão de tokens**, como **ICOs**, **STOs**, **IEOs**, **ETOs** e **DAICOs**. Discutimos **padrões de tokens**, como os vários **ERCs**, incluindo ERC-20, ERC-721, ERC-1155 e outros.

Também aprendemos como escrever um contrato inteligente **ERC-20 do zero**, compilá-lo e implantá-lo usando o **Remix IDE** e **MetaMask** na **rede de testes Sepolia**. Exploramos **tokenomics**, uma área emergente que trata da economia dos tokens.

Com isso, encerramos este capítulo. No próximo capítulo, veremos como os **contratos inteligentes podem ser auditados** para segurança e desempenho.