FACI

wyden

Introdução a Cripto Ativos

Profº Msc. Jeffson Celeiro Sousa
Doutorando em Ciência da Computação - UFPa

Belém, 05 de Fevereiro de 2026

Contato

FACI wyden

Resumo

- **O que vimos**
  - Funções Hash
  - Criptografia Assimétrica
  - Assinatura Digital
  - Assinatura de Transações no Bitcoin
  - Estrutura de blocos
  - Importância da Ledger

O Protocolo Ethereum

Contribuições principais:

- Introdução de contratos inteligentes (smart contracts).
- Sandbox chamada Ethereum Virtual Machine (EVM) para execução de contratos inteligentes.
- Expansão do blockchain para além de transferências de criptomoedas.

O Protocolo Ethereum

Bitcoin:

- Focado em transferências de criptomoedas.
- Aplicações de carteira para iniciar transações.

Ethereum:

- Suporte a contratos inteligentes.
- Base para operações descentralizadas de aplicações (Dapps).

A Importância da Confiança

Transações necessitam de confiança:

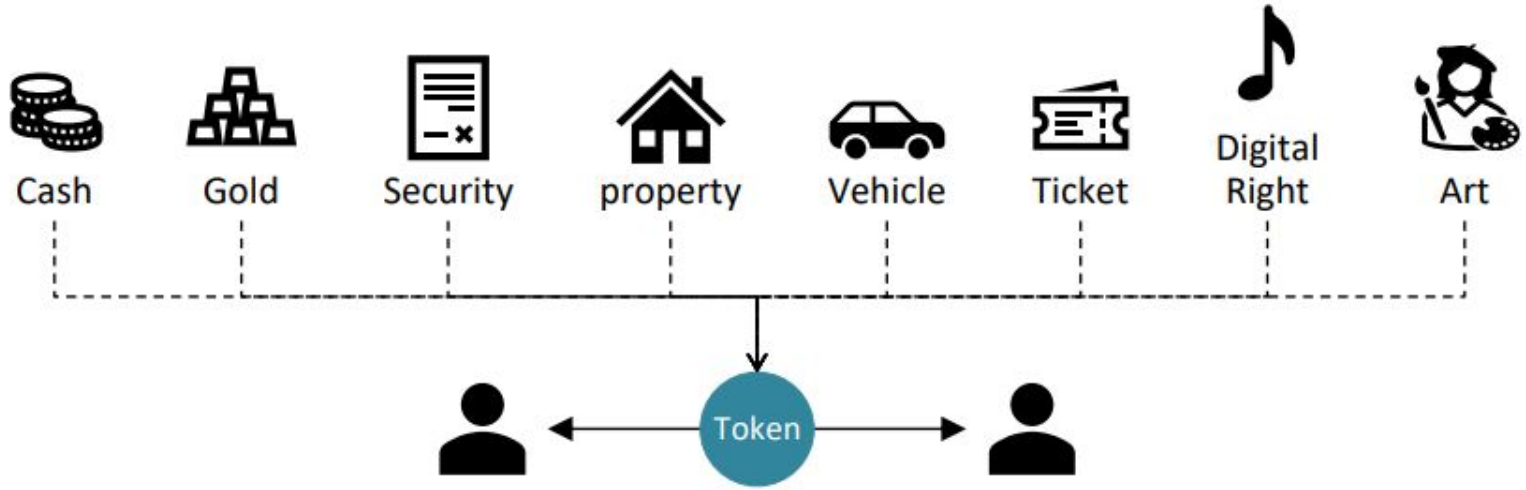
- Exemplos: Comércio, finanças, contratos legais, saúde, e relacionamentos.
- **Problema:** Como confiar em transações sem intermediários humanos ou institucionais?

Solução:

- Blockchain adiciona uma camada de confiança ao internet, chamada de ***trust layer***.

O que é um token ?

- Tokens são objetos semelhantes a moedas, embora não tenham valor por si só. Isto porque os tokens são emitidos por uma entidade privada para um uso específico e normalmente são utilizados materiais de baixo valor na sua preparação.



Origem dos tokens

- Primeiro vieram as moedas. A história começa com o **Bitcoin** – a primeira rede blockchain. O **Bitcoin** foi projetado para permitir que as pessoas façam pagamentos peer-to-peer globais, usando sua moeda nativa também chamada **Bitcoin** ou BTC. Depois vieram blockchains similares como a **Litecoin** (LTC) e a **Dogecoin** (DOGE) que oferecia um caso de uso similar.
- Depois veio a **Ethereum** e introduziu o conceito de aplicativos e protocolos descentralizados baseados em blockchain. E, embora o Ethereum também tenha sua própria "**moeda**" chamada ETH (que pode ser usada da mesma forma que o BTC), seu principal caso de uso é servir como gás para alimentar transações e operações em aplicativos e protocolos construídos na rede.

Tokens de criptografia

- Uma peculiaridade desses tokens criptográficos é que eles são todos baseados no blockchain de terceiros. Os primeiros tokens criptográficos foram desenvolvidos na blockchain Bitcoin e foram chamados Colored Coins. Sendo uma das plataformas de criação mais conhecidas Counterparty.
- Atualmente existem muitas blockchains que permitem a criação de tokens de forma simples. Por exemplo, **Waves, TRON, NEM ou Omni**.

Tokens de criptografia

- Embora Ethereum seja a plataforma por excelência, pois permite o desenvolvimento de um Smart Contract para criar um token de forma rápida e fácil. Ethereum é talvez a blockchain mais utilizada para o desenvolvimento de novos tokens.
- Essa situação foi impulsionada pela criação de padrões **ERC-20 e ERC-721**. Ambos permitem a criação rápida de tokens na blockchain Ethereum associando contratos inteligentes para ela.
- Ethereum tem mais de 450 mil **Tokens ERC-20** diferentes e mais de 1.300 Tokens **ERC-721**.

Tipos e padrões de tokens criptográficos

- **Security Token.** Estes são semelhantes a qualquer outro token conhecido, mas vinculados à segurança tradicional (valores) e suas características.
- **Utility Token**. Eles são "tokens" de aplicativos ou usuários. Eles permitem o acesso futuro aos produtos ou serviços oferecidos por uma empresa. Portanto, eles não são criados para serem um investimento.
- **Equity Token ou Security Token**. Eles são um tipo muito especial de token intimamente relacionado à segurança. Eles funcionam como um ativo de estoque tradicional. Eles representam a propriedade de algum ativo ou empresa de terceiros. Além disso, o seu valor está associado ao sucesso ou insucesso da referida propriedade.

Padrões de tokens

- **ERC-20:** é o padrão mais popular. São tokens fungíveis (ou seja, podem ser trocados entre si) e implementam uma API para seu desenvolvimento por meio de contratos inteligentes. Esse padrão permite a interação de contratos inteligentes e DApps, facilitando a criação de um ecossistema interoperável.
- **ERC-721:** É o padrão mais utilizado para criar NFT. Esta norma permite implementar características únicas a cada unidade, tornando-as únicas e não podendo ser alteradas por outra.
- **ERC-1155**: É um tipo de token que pode misturar as características dos tokens ERC-721 e ERC-20.

Padrões de tokens

- **SLP**: o padrão nativo do ecossistema Solana. Isso tem uma característica única e é que eles geralmente são compatíveis com os padrões ERC-20 e ERC-721.
- **BEP-20**: este é o padrão Binance Smart Chain (BSC) e oferece uma estrutura flexível para o desenvolvimento de utility tokens, stablecoins e até mesmo NFTs.
- **TRC-10 e TRC-20**: Este é o padrão nativo da rede Tron, com capacidade de multi-smart contract e alguns recursos exclusivos, como baixas taxas de gas ou interoperabilidade.
- **ERC-20 Waves**: É uma versão do padrão Ethereum desenvolvido pela Waves para o desenvolvimento de aplicativos descentralizados de nível empresarial. Ele oferece recursos de interoperabilidade com diferentes protocolos e trocas.
- **NEP-5**: este é o padrão dos tokens nativos da blockchain NEO. Esse padrão tem uma particularidade e é que todos os tokens contribuem de alguma forma para o desenvolvimento e expansão do ecossistema NEO.

Mas o que são tokens ERC?

- O ecossistema Ethereum é descentralizado, mas ainda precisa de alguém que estabeleça as regras, faça chamadas para atualizações e estabeleça padrões que definam o que é possível na blockchain. Para conseguir isso, os próprios usuários da Ethereum são obrigados a criar **Propostas de Melhoria da Ethereum** (EIP), discutir seus detalhes, votar sobre elas e rejeitar ou iniciar sua implementação.

- Esta subcategoria de uma EIP foi chamada de **Ethereum Request for Comments**, também conhecida como ERC. E até agora, houve muitos pedidos de comentários da Ethereum para estabelecer padrões para os tokens que podem ser criados na Ethereum. Portanto, todas os tokens que são criados na Ethereum devem seguir os padrões estabelecidos por esses ERCs. Assim, eles são chamados tokens ERC.

Mas o que são tokens ERC?

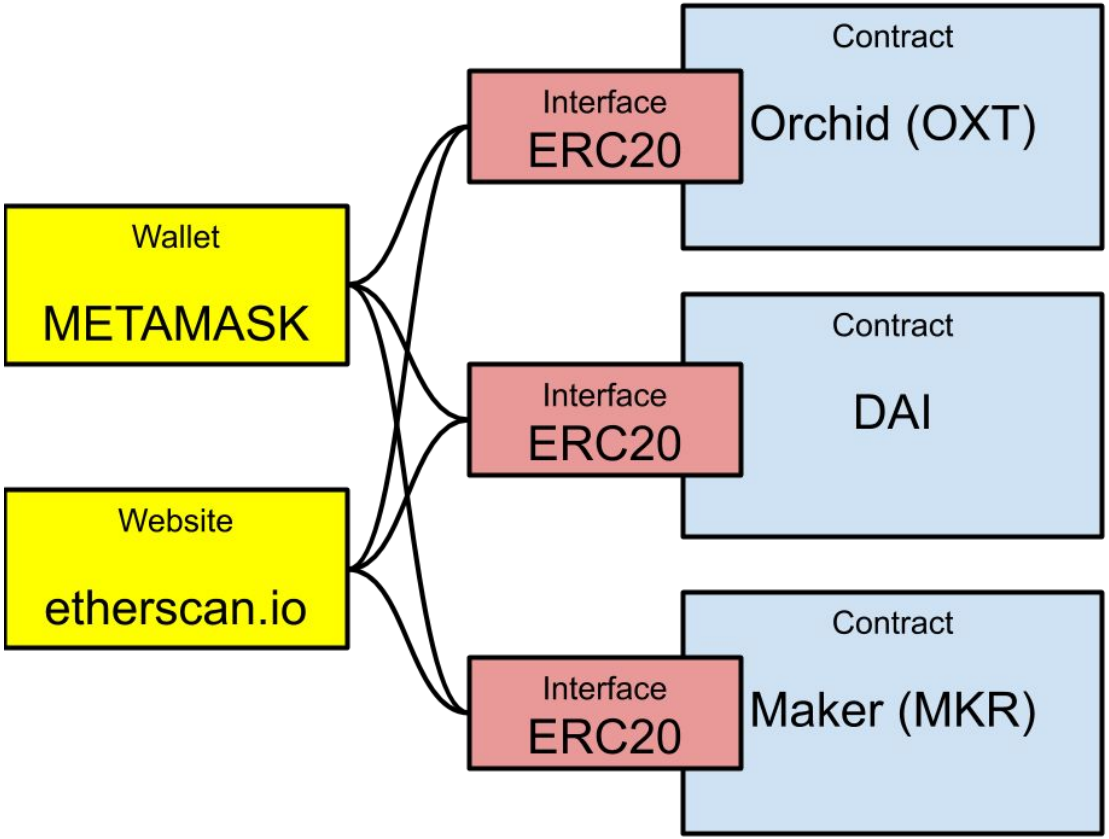
- O padrão ERC foi oficialmente proposto pelo desenvolvedor **Fabian Vogelsteller** em 2015, e formalizado na Proposta de Melhoria do Ethereum 20 (EIP-20) em 2017. Mas por que foi proposto em primeiro lugar?
- Antes do ERC20, havia problemas com a criação, uso e troca de diferentes tokens na blockchain do Ethereum devido à falta de padronização. O ERC20 foi projetado como o padrão técnico para tokens fungíveis na rede Ethereum, tornando cada token dentro de um conjunto idêntico aos outros.

FACI wyden

ERC20: o token fungível padrão

- Desde que se consolidou como o padrão para a criação de tokens fungíveis, o ERC-20 se tornou uma pedra angular do ecossistema Ethereum, permitindo que os desenvolvedores criem soluções inovadoras e impulsionem o crescimento na plataforma.
- Assim como com os tokens tradicionais Ether, todas as transações envolvendo tokens ERC20 são registradas na blockchain Ethereum, proporcionando rastreabilidade de todas as transferências de tokens e operações na rede.

Como funciona o ERC-20



FACI wyden

Funções ERC-20

- O padrão **ERC20** requer que os tokens implementem seis funções obrigatórias:
  - **TotalSupply**: retorna o fornecimento total de tokens.
  - **BalanceOf**: retorna o saldo de um endereço.
  - **Transfer**: transfere tokens de um endereço para outro.
  - **TransferFrom**: também é usada para transferência, porém para permitir que um terceiro mova fundos do seu endereço. Ou seja, Alice pode autorizar Bob a transferir fundos que pertencem a ela.
  - **Approve**: limita a quantidade de tokens que um contrato pode retirar de um endereço.
  - **Allowance**: verifica se o endereço tem saldo suficiente para enviar tokens para outro endereço.
- Funções opcionais
  - **Name**: adiciona um nome legível ao token. Por exemplo: Tether, Shiba Inu, Uniswap.
  - **Symbol**: associa um símbolo ao token. Por exemplo: USDT, SHIB, UNI.
  - **Decimal**: define em até quantas casas decimais o token pode ser dividido.

Benefícios dos Tokens ERC-20

- **Interoperabilidade**: tokens ERC-20 podem ser facilmente negociados, trocados e integrados em várias aplicações baseadas em Ethereum, aprimorando a usabilidade e a interoperabilidade entre dApps.

- **Padronização**: a interface padronizada simplifica o desenvolvimento, auditoria e integração de tokens baseados em Ethereum. Isso também ajuda a reduzir a barreira de entrada para desenvolvedores e a promover a inovação.

- **Acessibilidade**: tokens ERC-20 podem ser armazenados e gerenciados usando uma ampla gama de Ethereum carteiras, garantindo fácil acesso para os usuários em diferentes plataformas e tipos de dispositivos.

Benefícios dos Tokens ERC-20

- **Liquidez**: A proliferação de tokens ERC20 contribuiu com a liquidez necessária para o ecossistema Ethereum, impulsionando o crescimento de aplicações como trocas descentralizadas (DEXs) e pools de liquidez.

- **Escalabilidade**: Ao aproveitar a infraestrutura robusta do Ethereum, um token ERC20 herda os recursos de escalabilidade e segurança da blockchain do Ethereum, facilitando transações eficientes e execução de "contratos inteligentes". Exemplos de tokens ERC-20.

Exemplos de tokens

- USDT
- USD Coin (USDC)
- Chainlink (LINK)
- Uniswap (UNI)
- Wrapped Bitcoin (WBTC)
- Shiba Inu (SHIB)

Desafios e limitações dos tokens ERC-20

- **Vulnerabilidades de segurança**: Atores maliciosos podem explorar vulnerabilidades em contratos inteligentes ERC20, levando a violações de segurança e roubo de tokens.

- **Problemas de escalabilidade**: A demanda crescente por transações em Ethereum levou a congestionamento da rede e altas taxas de gás, dificultando a escalabilidade e o uso potencial dos tokens ERC20.

- **Incerteza regulatória**: O cenário regulatório em torno dos tokens ERC-20 continua incerto, com alguns órgãos reguladores ainda por definir sua classificação e supervisão.

- **Riscos de contrato inteligente:** Erros ou vulnerabilidades no código do contrato inteligente podem resultar em perdas irreversíveis de fundos ou comportamento inesperado, destacando a importância de auditorias e testes rigorosos.

FACI wyden

O que é o ERC-721 ?

- ERC-721 é um padrão de token não fungível (NFT) no blockchain Ethereum. Ele permite a criação e gerenciamento de ativos digitais únicos e indivisíveis, como obras de arte digitais, colecionáveis e imóveis virtuais.

- Ao contrário dos tokens fungíveis, cada token ERC-721 é único e possui propriedades exclusivas, tornando-os ideais para representar itens digitais únicos e autênticos na economia digital.

Como funciona o ERC-721?

- O ERC-721 é um **padrão de token não fungível** (NFT) no blockchain Ethereum, projetado para representar ativos únicos e indivisíveis. Ao contrário dos tokens fungíveis, como o ERC-20, cada token ERC-721 possui um identificador único e pode ser único em termos de propriedade, atributos e valor.

- Isso possibilita a criação e gerenciamento de ativos digitais exclusivos, como obras de arte digitais, colecionáveis virtuais, tokens de jogos e imóveis virtuais.

Como funciona o ERC-721?

- Cada token ERC-721 é rastreável e autenticável, permitindo a prova de propriedade e garantindo a integridade dos ativos no blockchain. Isso é alcançado através de métodos padrões como - balanceOf, ownerOf, approve, transferFrom e safeTransferFrom - que permitem transferências seguras e verificáveis de tokens entre usuários.

- Essa funcionalidade não só viabiliza novos modelos de negócios e economias digitais, mas também abre caminho para a tokenização de ativos reais e virtuais, proporcionando maior liquidez e transparência no mercado global.

ERC-721: onde se aplica no blockchain?

- O padrão ERC-721 é amplamente utilizado em diversas aplicações dentro do setor de blockchain. Especialmente em:

  - **Jogos descentralizados (DApps)**: permite a criação e troca de itens digitais únicos, como personagens e itens colecionáveis;

  - **Arte digital**: facilita a autenticação e o comércio de arte digital exclusiva, garantindo a propriedade e a proveniência;

  - **Mercados de tokens não fungíveis (NFTs)**: utilizado para a criação e gestão de tokens únicos, que representam ativos digitais exclusivos, como obras de arte, colecionáveis e imóveis virtuais.

ERC-721 vs. tokens fungíveis

- Ao explorar o blockchain, é essencial distinguir entre tokens ERC-721 e tokens fungíveis. Enquanto os tokens fungíveis são acumuláveis e podem ser trocados uns pelos outros como dinheiro, os tokens ERC-721 são únicos.

- Ou seja, enquanto você pode juntar duas notas de R$ 100 e ter então R$ 200, podendo utilizar esses R$ 200 para fazer compras no supermercado, o mesmo não é possível com os NFTs. Isso porque eles estão mais próximos de obras de arte, por exemplo, em que cada uma tem o seu valor estimado por uma série de fatores que a tornam única.

- Cada token ERC-721 possui um valor exclusivo, representando ativos físicos ou digitais como propriedades, obras de arte e colecionáveis.

O que é o ERC-1155?

- O problema com as normas **ERC20** e **ERC721** foi que elas permitem que um contrato inteligente suporte apenas um único tipo de token fungível ou não fungível. Portanto, toda vez que você quiser implantar um novo token, você tem que escrever um novo contrato inteligente. Além disso, as duas normas acima não oferecem uma maneira de criar tokens semi-fungíveis.

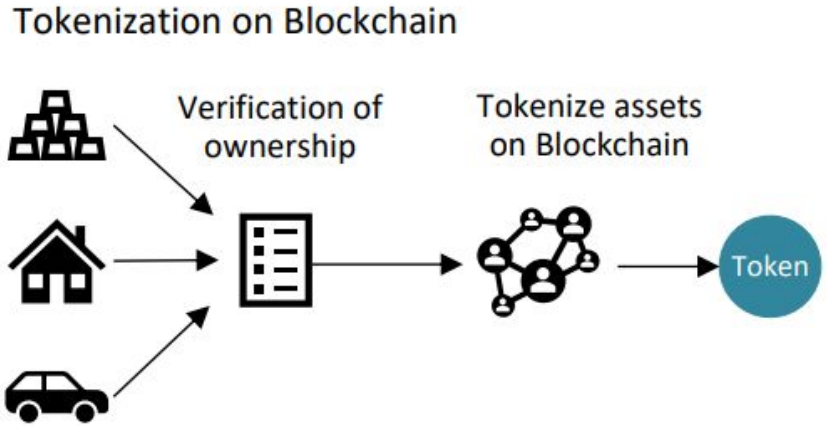
- Isso levou ao estabelecimento do padrão de tokens **ERC1155**, permitindo aos desenvolvedores do Ethereum criar tokens fungíveis, semi-fungíveis e não-fungíveis usando o mesmo padrão. Além disso, usando **ERC1155**, você pode criar um único contrato para suportar vários tipos de tokens configuráveis individualmente, reduzindo assim a complexidade do processo de criação dos tokens.

ERC1155: o padrão de tokens múltiplos

- Suponha que um desenvolvedor queira desenvolver um jogo **NFT** e planeje criar um token fungível para ser usado como moeda no jogo e vários tokens não fungíveis para ativos únicos no jogo, tais como peles, armas, mercadorias etc.
- Se eles usarem os padrões **ERC20** e **ERC721**, teriam que escrever novos contratos inteligentes para apoiar cada novo tipo de ativo que criassem. Entretanto, o uso do **ERC1155** lhes permitiria escrever apenas um contrato para apoiar todos os tipos de tokens que eles querem ter no jogo.

Benefícios da Tokenização na Blockchain

- Tokenização é o processo de transferir os direitos de um ativo existente para um token digital.
- Benefícios:
  - Menos intermediários
  - Compartilhamento de dados
  - Imutabilidade
  - Transparência
  - Procedimentos automatizados
  - Divisão de partes virtuais



Requisitos para Desenvolver um Contrato Inteligente

Plataforma Blockchain:

- Utilizaremos Besu Ethereum client.

Linguagem de Programação:

- **Solidity**: Especialmente projetada para contratos inteligentes.

Ambiente de Desenvolvimento (IDE):

- **Remix**: Para desenvolver, compilar, implantar e testar contratos.

Tecnologias Utilizadas

Ethereum: Plataforma blockchain.

- Site:besu.hyperledger.org.

Solidity: Linguagem para codificação.

- Documentação: solidity.readthedocs.io.

Remix IDE: Ambiente para testes.

- Documentação: remix-ide.readthedocs.io.

Casos de Uso

Moeda Digital:

- Simples demonstração do design de contratos inteligentes.

O Que é um Contrato Inteligente?

Definição:

- Um pedaço de código que expande a camada de confiança do blockchain.
- Adiciona programabilidade para verificar e validar transações além de criptomoedas.

Objetivo:

- Fornecer verificação e validação específicas para aplicações blockchain.
- Abrir a camada de confiança do blockchain para aplicações de propósito geral.

Estrutura de um Contrato Inteligente

Ambiente de execução:

- Implantado em uma sandbox no nó do blockchain.
- Identificado por um endereço de conta de **160 bits**.

Execução:

- Funciona em uma máquina virtual (VM) na blockchain.
- Atribuído a um número de conta único.

Expansão da Aplicabilidade com Contratos Inteligentes

Bitcoin:

- Limitado a transações financeiras.

Contratos Inteligentes:

- Permitem lógica personalizada e funcionalidade específica para aplicações descentralizadas.
- Habilitam casos de uso em votação, rastreamento de ativos, gerenciamento de contratos, e mais.

Exemplo de Caso de Uso: Moeda Digital

Por que um token ERC-20?

- Padrão amplamente utilizado em criptomoedas e tokens digitais.
- Compatibilidade com carteiras e exchanges, tornando-o útil no ecossistema Web3.

Exemplos das principais Moedas:

1. **Ethereum (ETH)**
2. **Tether (USDT)**
3. **XRP (XRP)**
4. **BNB (BNB)**
5. **Solana (SOL)**

Princípio de Design 1: Contratos são Imutáveis

Uma vez implantado na blockchain:

- Contrato é permanente, irreversível e inalterável.
- Parte da cadeia de blocos.

Recomendações:

- Projetar e testar cuidadosamente antes de implantar em uma blockchain de produção.
- Usar blockchain de teste para validações.

Objetivos do Processo de Design

Conteúdo do Contrato Inteligente:

- **Dados**: Variáveis armazenadas no contrato.
- **Funções**: Operações realizadas sobre os dados.
- **Regras**: Políticas que definem como as funções interagem com os dados.

Funções do Contrato Inteligente

Funções principais do contrato da moeda:

1. **constructor(uint256 initialSupply):** Inicializa o contrato com um supply inicial de tokens
2. **balanceOf(address account)**: Retorna o saldo de tokens de um endereço específico.
3. **transfer(address recipient, uint256 amount)**: Transfere tokens de um endereço para outro.
4. **approve(address spender, uint256 amount)**: Autoriza outro endereço a gastar uma quantidade de tokens em nome do dono do contrato.
5. **transferFrom(address sender, address recipient, uint256 amount)**: Transfere tokens de um endereço autorizado para outro.

Exemplo: Diagrama de Contrato para a Moeda

Nome do Contrato

- `Meutoken`

Ativos de Dados

- **name (string):** Nome do token. Exemplo: `"Meutoken"`
- **symbol (string):** Símbolo do token. Exemplo: `"MMC"`
- **decimals (uint8):** Decimais de precisão do token. Exemplo: `18`
- **totalSupply (uint256):** Quantidade total de tokens emitidos. Exemplo: `1000000`

Regras

- **Total Supply:** O token possui um supply fixo definido no momento do deploy.
- **Controle de Transferência:** Apenas o criador pode transferir os tokens de acordo com o total supply.
- **Funções `approve` e `transferFrom`:** Permitem que um endereço autorize outro a gastar tokens em seu nome.
- **Eventos:** O contrato emite os eventos `Transfer` e `Approval` para rastrear as transações.

Desenvolvimento de Código de Contrato Inteligente

Linguagem de Programação para Blockchain

- **Necessidade de linguagens personalizadas**:
  - Linguagens gerais (Java, Python, Go) são ricas em sintaxe e bibliotecas.
  - Smart contracts requerem instruções precisas para operações específicas de blockchain.

Por Que Usar Solidity?

Introdução:

- Criada pela Fundação Ethereum.
- Suportada por outras plataformas, como **Besu**.

Razões para escolher Solidity:

- Recursos orientados para blockchain:
  - Manipulação de endereços de contas.
  - Especificação de regras.
  - Tratamento de reversão de transações.
- Execução em sandbox para garantir consistência entre os nós.

Características de Solidity

Customização para Blockchain:

- **Sintaxe simples e direta**: Sem complexidades desnecessárias.
- **Compatibilidade blockchain**:
  - Gerenciamento de transações.
  - Imutabilidade de registros.

Execução restrita:

- Código funciona em um ambiente controlado para manter segurança e consistência.

Implementação do Contrato da Moeda

Explorar os recursos do Solidity:

- Variáveis.
- Funções.
- Modificadores de acesso.

Codificar o contrato da moeda:

- Usando os princípios e o design desenvolvido anteriormente.

Testar e depurar o código:

- Garantir conformidade com as funcionalidades planejadas.

Introdução à Linguagem Solidity

Características da Solidity

- **Estilo de Programação**:
  - Linguagem **orientada a objetos** de alto nível.
- **Influências**:
  - Inspirada em **C++**, **Python**, e **JavaScript**.
- **Principais Recursos**:
  - Tipagem estática.
  - Suporte a herança, bibliotecas e tipos definidos pelo usuário.
  - Recursos específicos para aplicações blockchain.

Por Que Escolher Solidity?

Desenvolvimento focado em blockchain:

- Projetada para contratos inteligentes e Dapps.

Familiaridade:

- Sintaxe e semântica semelhantes a linguagens populares.

Funcionalidades Avançadas:

- Gerenciamento de endereços, transações e regras específicas.

Ambiente de Desenvolvimento: Remix IDE

Remix IDE:

- Ferramenta para escrever, editar, compilar, implantar e testar código Solidity.

Por que usar Remix?

- Simulação de blockchain para desenvolvimento local.
- Interface amigável para novos desenvolvedores.

Fluxo de Desenvolvimento com Solidity no Remix

Escrita e edição de código:

- Criar contratos inteligentes utilizando a sintaxe do Solidity.

Compilação:

- Identificar erros e gerar bytecode para a blockchain.

Implantação:

- Testar o contrato em um blockchain simulado.

Testes:

- Verificar a funcionalidade e realizar ajustes no código.

Código do Contrato Inteligente: Moeda Digital

Resumo do Processo

1. **Configuração do Ambiente**:
   - Use o **Remix IDE** no navegador.
   - Selecione a linguagem **Solidity** (suporta arquivos `.sol`).
2. **Criação do Arquivo**:
   - Clique no ícone **+** no painel esquerdo.
   - Nomeie o arquivo como **Meutoken.sol**.
3. **Escrita do Código**:
   - Digite ou copie o código no editor.

Código Completo

- https://github.com/jeffsonsousa/introducao-ERC20

Instalação da rede blockchain

1. Pré-Requisitos
   - Docker
   - Java
   - Node

2. Subir a rede
   - docker compose up -d

Instalação do explorador de blocos rede blockchain

1. Setup Chainlens-Free
   - git clone https://github.com/web3labs/chainlens-free.git
   - cd chainlens-free/docker-compose
2. Run Chainlens
   - NODE_ENDPOINT=http://IP-LOCAL-BESU:8545 PORT=26000 docker-compose -f docker-compose.yml -f chainlens-extensions/docker-compose-besu.yml up

OBS.: Para rodar em segundo plano:
NODE_ENDPOINT=http://IP-LOCAL-BESU:8545 PORT=26000 docker-compose -f docker-compose.yml -f chainlens-extensions/docker-compose-besu.yml up -d

FACI wyden

Passos Finais no Remix IDE

1. Abra o Remix web IDE no seu navegador.
2. Escolha o idioma Solidity.
3. (Vyper é o outro idioma suportado.)
4. No IDE, clique no ícone + na parte superior do painel esquerdo para criar um novo arquivo.
5. Na janela pop-up que se abre, nomeie o arquivo MeuToken.sol.
6. (.sol é o tipo de arquivo para programas escritos em Solidity.)
7. Digite (enter) ou copie o código na listagem 2.1 na janela do editor.

O Remix IDE

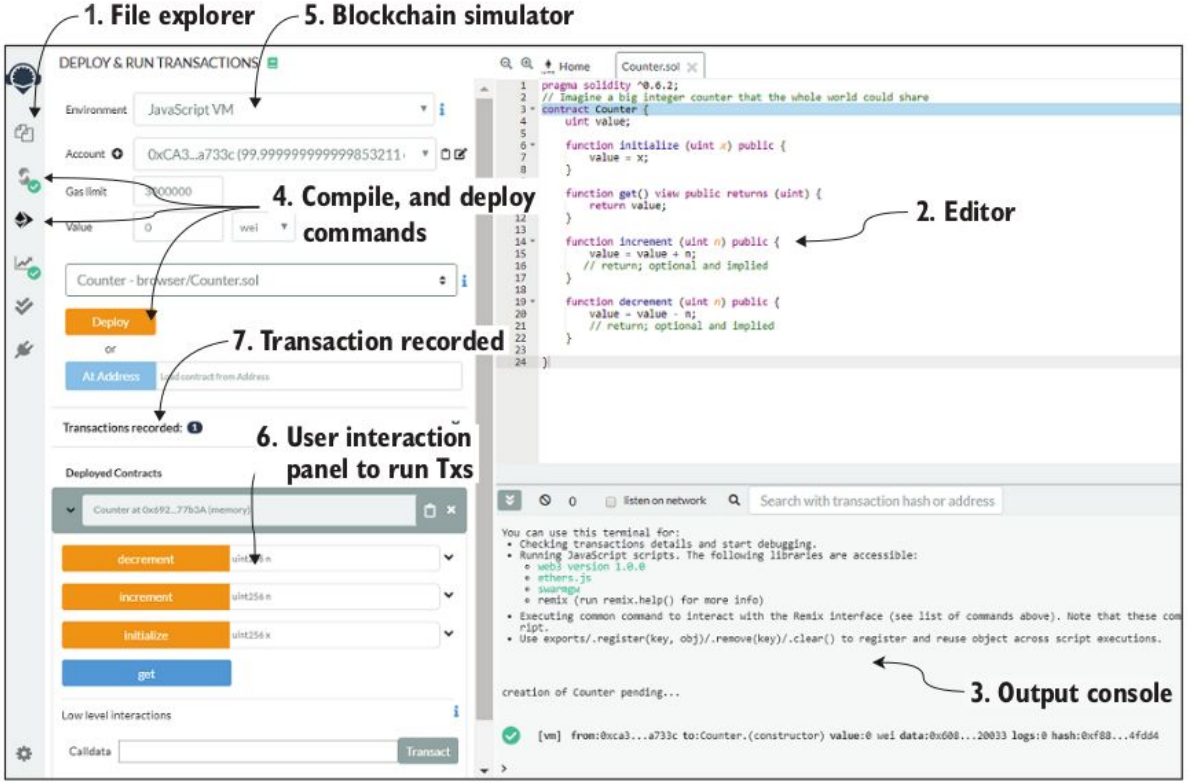
Acesso e Finalidade

- Acesse o Remix IDE em remix.ethereum.org.
- Ferramenta para desenvolvimento, teste e implantação de contratos inteligentes em Solidity.
- Suporte a simulação de blockchain com ambiente seguro e versátil.

Recursos Principais do Remix IDE

1. Explorador de Arquivos

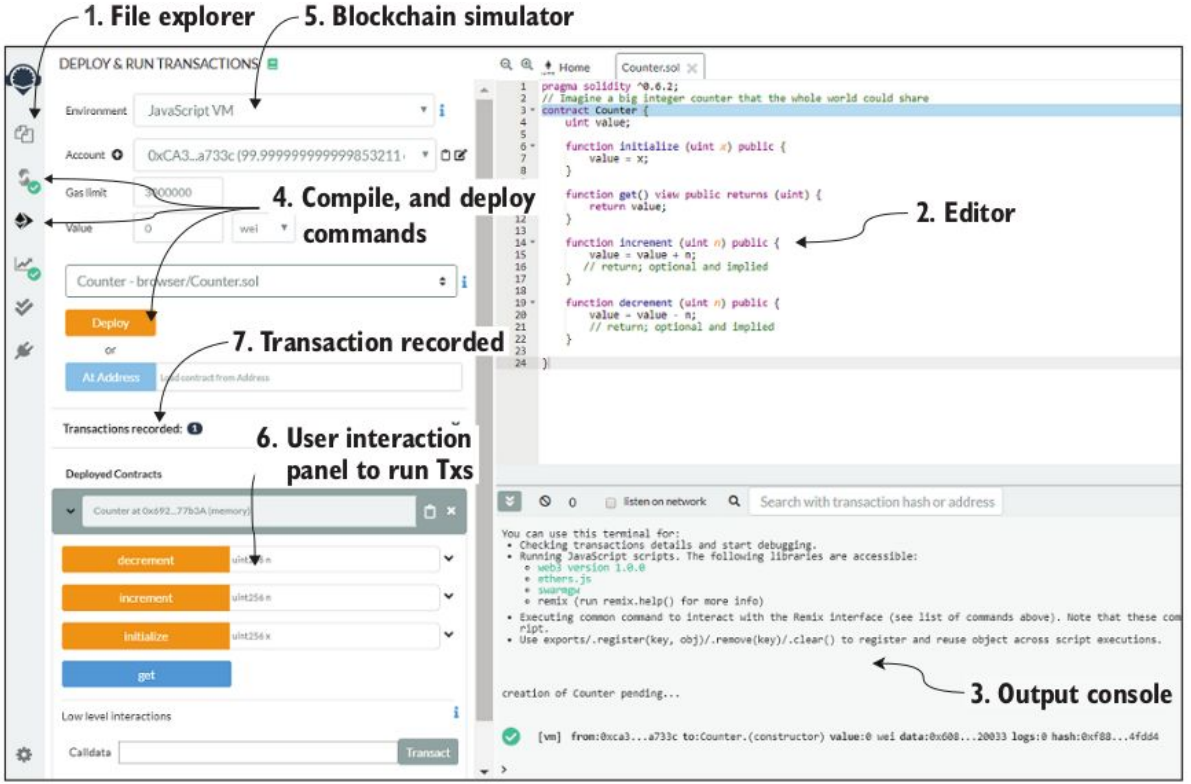
- Localizado à esquerda.
- Funções:
  - Criar, abrir, fechar, e excluir arquivos.
  - Salvar arquivos automaticamente no servidor Remix.
  - Sincronizar com o armazenamento local.



Recursos Principais do Remix IDE

2. Espaço do Editor

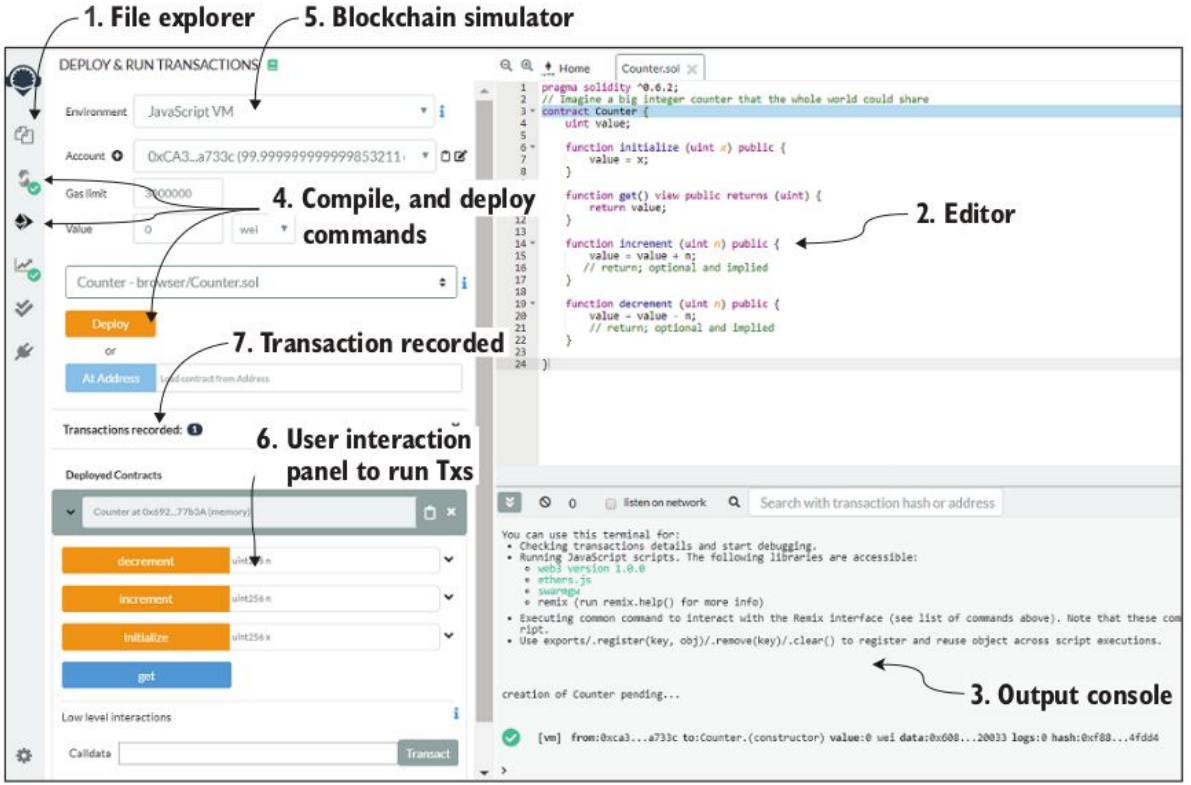
- Área central para:
  - Inserir e editar código, como arquivos `.sol` e `.json`.
  - Recursos adicionais:
    - **Compilador em tempo real**: Indica erros enquanto o código é digitado.



Recursos Principais do Remix IDE

3. Console de Saída

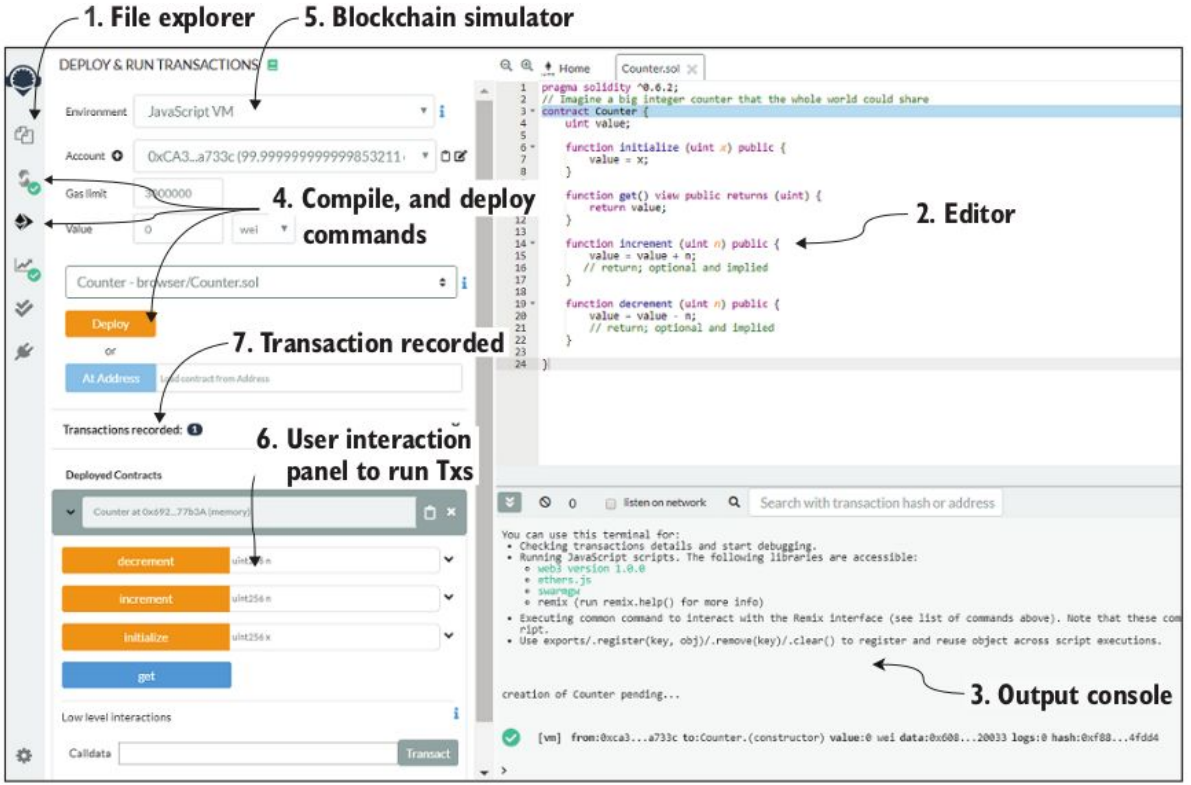
- Localizado abaixo do editor.
- Exibe:
  - Transações confirmadas e gravadas.
  - Erros e detalhes de depuração.



Recursos Principais do Remix IDE

4. Cadeia de Ferramentas

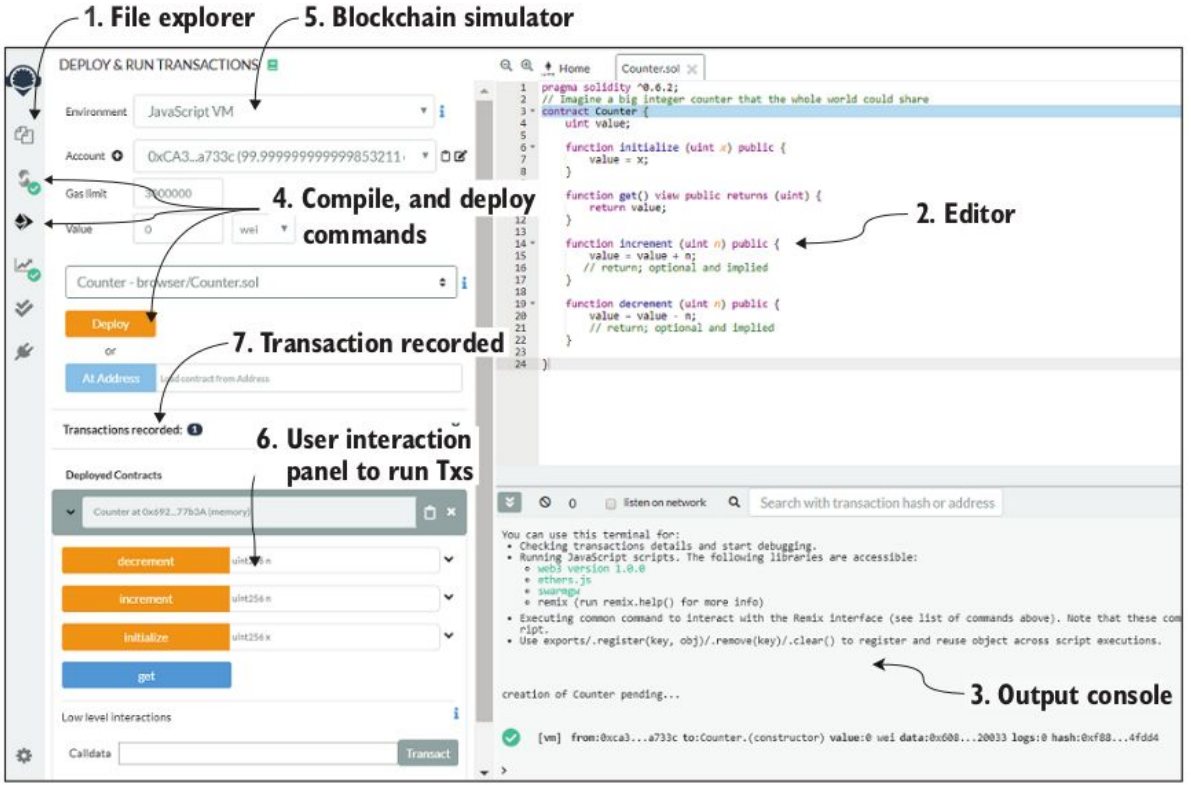
- Painel à esquerda com ícones para comandos:
  - **Compilar**: Gera bytecode do contrato.
  - **Implantar**: Envia o contrato para o blockchain.



Recursos Principais do Remix IDE

5. Simulador de Blockchain

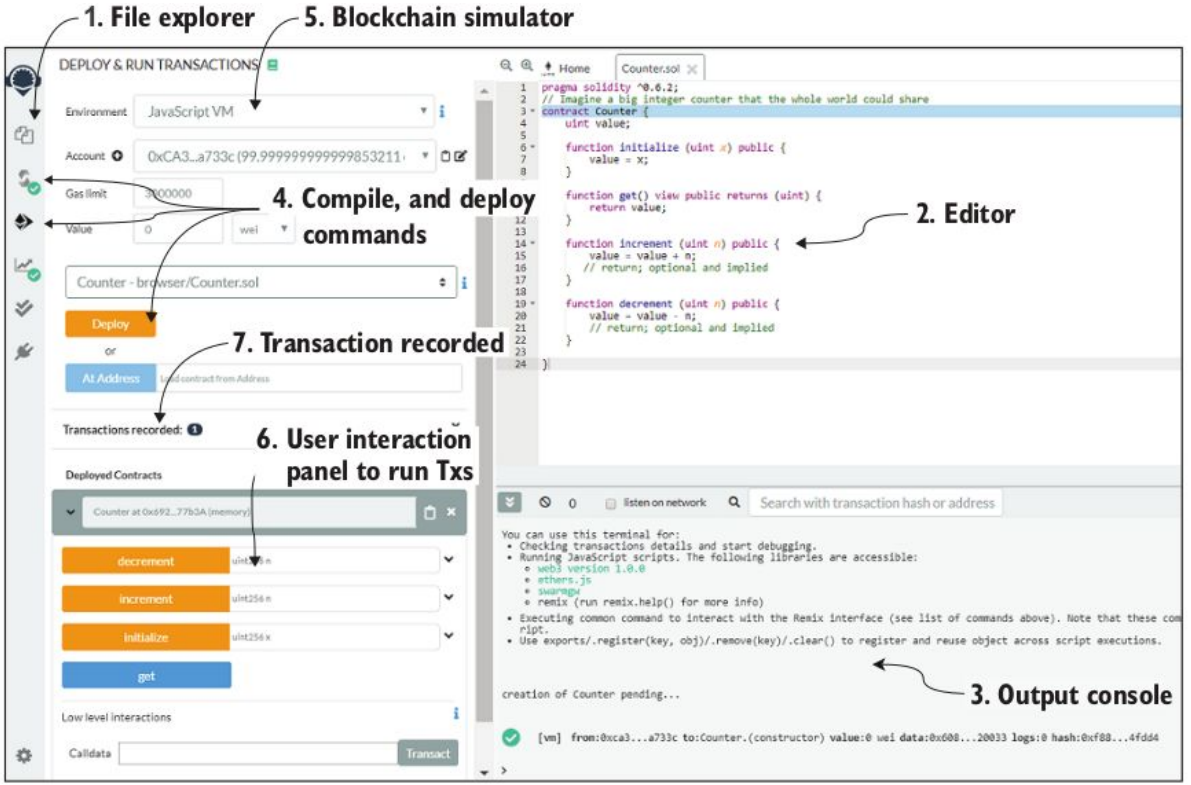
- Oferece ambiente de execução:
  - **JavaScript VM**: Simula blockchain local.
  - Conexão com redes reais de blockchain.
- Contas de teste fornecidas com identidades e saldo fictício.



Recursos Principais do Remix IDE

6. Painel de Interação do Usuário

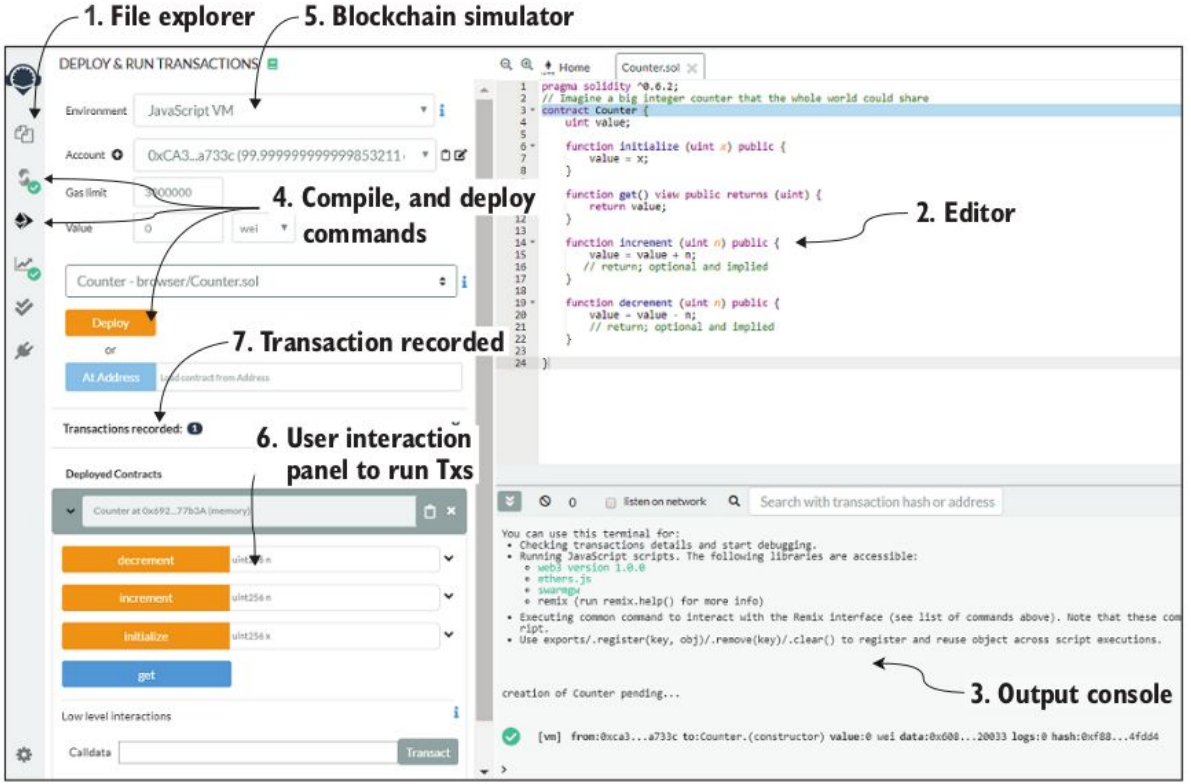
- Localizado no canto inferior esquerdo.
- Expõe:
  - Funções públicas do contrato implantado.
  - Botões para invocar funções e campos para parâmetros de entrada.
  - Resultados exibidos abaixo dos botões de função.



Recursos Principais do Remix IDE

7. Registro de Transações

- Botão de **Transações Gravadas** no painel esquerdo.
- Armazena registros em arquivos `.json` para revisão.



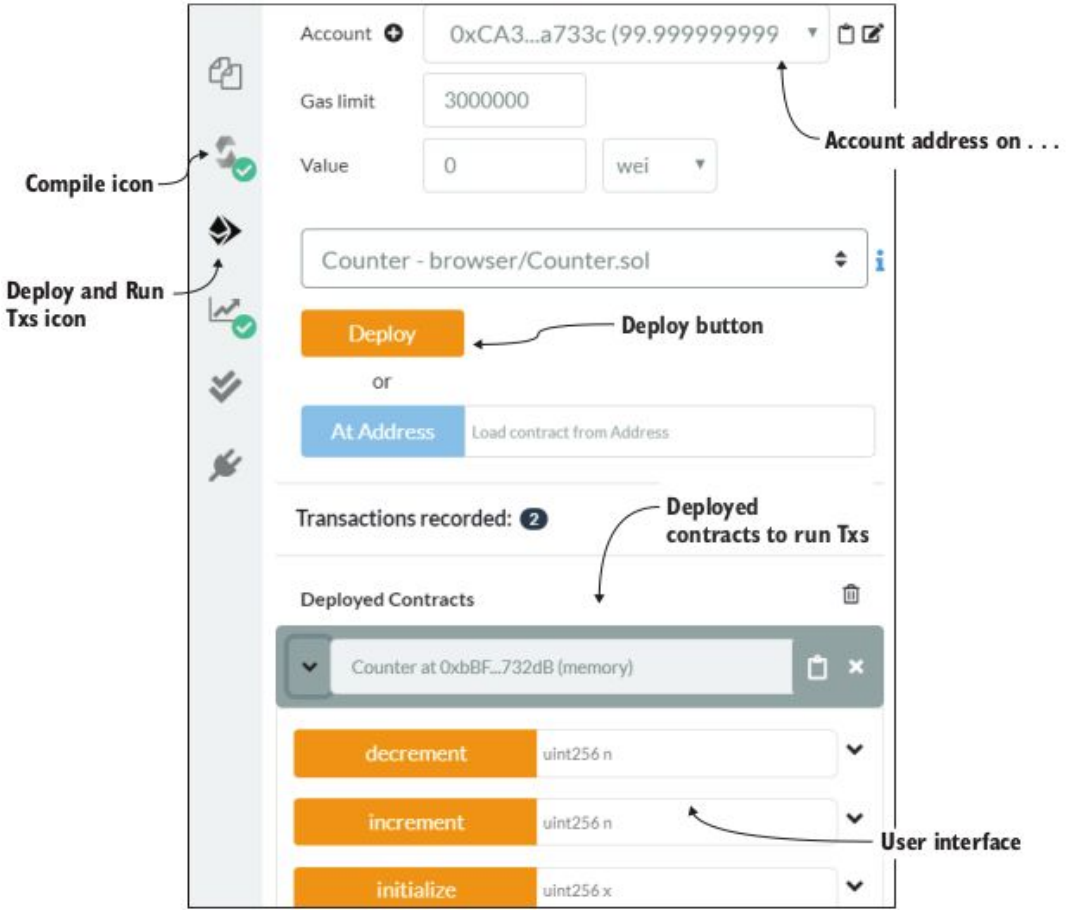
Passos para Testar o Contrato

1. Configuração Inicial

- Abra o Remix IDE e crie o arquivo **Meutoken.sol**.
- Insira o código do contrato no editor.

2. Compilação

- Navegue para a aba **Solidity Compiler**.
- Compile o código e corrija quaisquer erros detectados.



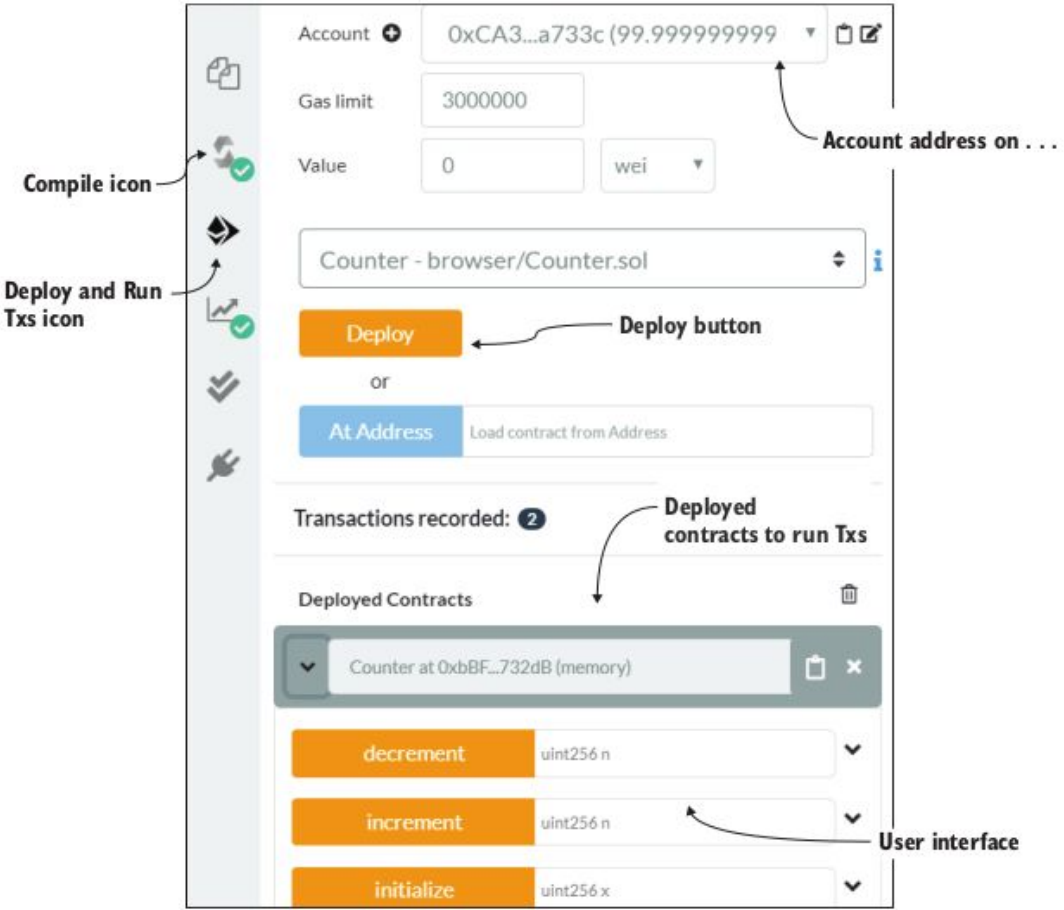
Passos para Testar o Contrato

3. Implantação

- Use a aba **Deploy & Run Transactions**.
- Selecione **JavaScript VM** como ambiente de execução.
- Clique em **Deploy** para implantar o contrato.

4. Interação

- Use o painel de interação:
  - Chame as funções públicas (**initialize, increment, decrement, get**) com diferentes parâmetros.
  - Verifique os resultados exibidos.



FACI

wyden