



Configuração e Administração.

Extensão da Troca de Ethereum.

# Guia do utilizador

versão 1.0.0 Beta

Novembro de 2020.

Blockoin.org é uma marca registada da Bankoin Inc, sob uma licença de uso livre e comercial. Termos e condições de utilização em: [www.CoinSolidation.org](http://www.CoinSolidation.org)

## Conteúdo

1.	Introdução.....	3
2.	O que é a programação Blockly? .....	5
3.	O que é uma extensão?.....	5
4.	O que é BlockoinEthereum e BlockoinFURA? .....	5
5.	Conceitos básicos aplicados na plataforma Ethereum. ....	6
6.	Instalação e configuração de extensão e ambiente de teste Ethereum. ....	13
7.	Definição e utilização de blocos (função genérica).....	20
8.	Intercâmbio de características e eventos de Extensão de Etéreo (EEE). .....	21
9.	Passos para criar um Token CryptoToken ou Cryptomoney.....	32
10.	Como colocar à venda um novo bem ou a sua cript token (Token ERC20).....	33
11.	Cálculo do custo de transacção padrão e transacção de contrato inteligente .....	56
12.	Coinsolidtion.org Taxas .....	59
13.	Crie a sua aplicação Android (Exchange) em 15 minutos. ....	60
14.	Token CoinSolidation.....	63
15.	Licenciamento e utilização de software. ....	64

## 1. Introdução.

Ethereum é uma plataforma de código aberto, descentralizada ao contrário de outras cadeias de blocos, Ethereum pode fazer muito mais. É programável, o que significa que os programadores podem utilizá-lo para criar novos tipos de aplicações. A sua moeda digital é chamada "Éter".

Estas aplicações descentralizadas (ou "dapps") obtêm os benefícios da criptomontagem e da tecnologia de cadeias de bloqueio. São fiáveis e previsíveis, o que significa que, uma vez "carregados" no Ethereum, funcionarão sempre dentro do prazo previsto. Podem controlar os activos digitais para criar novos tipos de aplicações financeiras. Podem ser descentralizados, o que significa que nenhuma entidade ou pessoa os controla.

Neste momento, milhares de criadores em todo o mundo estão a criar aplicações no Ethereum e a inventar novos tipos de aplicações, muitas das quais podem ser utilizadas hoje em dia:

- Carteiras de divisas criptográficas que lhe permitem fazer pagamentos baratos e instantâneos com ETH ou outros activos
- Aplicações financeiras que lhe permitem pedir emprestado, emprestar ou investir os seus activos digitais
- Mercados descentralizados, permitindo a troca de bens digitais, ou mesmo a troca de "previsões" sobre eventos do mundo real.
- Jogos em que tem bens no jogo e pode mesmo ganhar dinheiro real.

### Como funciona o éter?

O éter, tal como outras moedas criptográficas, utiliza um livro digital partilhado onde todas as transacções são registadas. É acessível ao público, completamente transparente e muito difícil de modificar posteriormente.

A isto chama-se uma **cadeia de bloqueio**, e é construída através do processo de **mineração**.

Os mineiros são responsáveis pela verificação de grupos de transacções de éteres para formar "blocos" e pela codificação dos mesmos através da resolução de algoritmos complexos. Estes algoritmos podem ser mais ou menos difíceis como forma de manter alguma consistência no tempo de processamento do bloco (cerca de um a cada 14 segundos).

Os novos blocos são então ligados à cadeia de blocos anterior e o mineiro em questão recebe uma **recompensa**, ou seja, um número fixo de *fichas de éter*. Normalmente são 5 unidades de éter, embora este número possa ser visto como variável se a moeda da cripta continuar a subir.

## Como é que o Ethereum funciona?

A cadeia de bloqueio Ethereum é muito semelhante à do bitcoin, mas a sua linguagem de programação permite aos programadores criar software através do qual podem gerir transacções e automatizar certos resultados. Este software é conhecido como um **contrato inteligente**.

Se um contrato tradicional descreve os termos de uma relação, um contrato inteligente assegura o cumprimento desses termos, escrevendo-os em código. Estes são programas que executam automaticamente o contrato assim que as condições predefinidas são cumpridas, eliminando o atraso e o custo que existe quando se executa um acordo manualmente.

Para dar um exemplo simples, um utilizador Ethereum poderia criar um contrato inteligente para enviar uma quantidade definida de éter a um amigo numa determinada data. Escreveriam este código na cadeia de blocos e quando o contrato for concluído (isto é, quando a data acordada for atingida) o éter será enviado automaticamente.

Esta ideia básica pode ser aplicada a configurações mais complexas, e o seu potencial é provavelmente ilimitado, com projectos que já fizeram progressos notáveis em sectores tais como seguros, imobiliário, serviços financeiros, serviços jurídicos e micro-finâncias.

Os contratos inteligentes também têm vários benefícios adicionais:

1. Eliminam a figura do intermediário, oferecendo ao utilizador o controlo total e minimizando os custos adicionais
2. São registados, codificados e duplicados na cadeia de blocos públicos, onde todos os utilizadores podem ver a actividade do mercado
3. Eliminar o tempo e o esforço necessários nos processos manuais

Claro que os contratos inteligentes são ainda um sistema muito novo com muitos detalhes para polir. O código é traduzido literalmente, pelo que quaisquer erros durante a criação do contrato podem levar a resultados indesejados que não podem ser alterados.

## DApps vs. contratos inteligentes

Os contratos inteligentes partilham semelhanças com os **DApps** (aplicações descentralizadas), mas também os separam de algumas diferenças importantes.

Tal como os contratos inteligentes, um DApp é uma interface que liga um utilizador a um serviço de um fornecedor através de uma rede de pares descentralizada. Mas, enquanto os contratos inteligentes precisam de um número fixo de participantes para serem criados, os DApps não têm limite para o número de utilizadores. Além disso, não se limitam apenas a aplicações financeiras tais como contratos inteligentes: um DApp pode servir qualquer propósito em que se possa pensar.

## 2. O que é a programação Blockly?

Blockly é uma metodologia de **programação visual** baseada na linguagem de programação JavaScript. Consiste num conjunto simples de comandos que podemos combinar como se fossem as peças de um puzzle. É uma ferramenta muito útil para aqueles que querem **aprender a programar de forma** intuitiva e simples ou para aqueles que já sabem programar e querem ver o potencial deste tipo de programação.

Blockly é uma forma de programação em que não é necessário qualquer tipo de linguagem informática, isto porque é apenas juntar blocos gráficos como se estivéssemos a jogar lego ou um puzzle, só é preciso ter alguma lógica e mais nada!

Qualquer pessoa pode criar programas para telemóveis (smartphones) sem se meter com as linguagens de programação difíceis de compreender, basta juntar blocos de forma gráfica de uma forma simples, fácil e rápida de criar.

## 3. O que é uma extensão?

Uma extensão é um módulo feito numa linguagem de programação Java dado num ficheiro com extensão .AIX

No projecto Blockoin.org, baseamo-nos na criação de extensões de fácil utilização para a área financeira, com as quais podem fazer aplicações móveis em poucos minutos sem terem um vasto conhecimento de programação.

## 4. O que é BlockoinEthereum e BlockoinFURA?

BlockoinEthereum e BlockoinFURA são software de uso livre (extensões) que inclui as seguintes soluções tecnológicas (algoritmos) para poder criar a ser utilizado na rede do Ethereum:

- CRIAÇÃO DE NOVAS CONTAS SOBRE A PLATAFORMA ETHEREUM.
- CONSULTA DO BALANÇO, TRANSFERÊNCIAS DE BENS (CRIPTOMONEDA-ÉTER E FICHAS)
- CRIAÇÃO DE NOVAS FICHAS ERC20 E CRIPTOMONEDA-TOKEN.
- COMPILAÇÃO, CRIAÇÃO, CONSULTA E PUBLICAÇÃO DE CONTRATO INTELIGENTE
- CRIAÇÃO DE TRANSACÇÕES OFFLINE E ONLINE.
- CARREGAMENTO DE CHAVES PRIMÁRIAS E PÚBLICAS.
- TAXAS DE CÂMBIO E CONSULTA DE ESTAÇÕES DE SERVIÇO.
- DESENVOLVIMENTO, TESTES E AMBIENTES DE REDE DE NÚCLEO DE ETÉREO (REDES)
- INCLUI AS 39 CARACTERÍSTICAS DA INFURA.

## 5. Conceitos básicos aplicados na plataforma Ethereum.

### O que é uma cadeia de bloqueio?

A cadeia de bloqueio é geralmente associada à moeda Bitcoin e outras moedas criptográficas, mas estas são apenas a ponta do iceberg uma vez que não é apenas utilizada para dinheiro digital, mas pode ser utilizada para qualquer informação que possa ter um valor para os utilizadores e/ou empresas. Esta tecnologia, que tem as suas origens em 1991, quando Stuart Haber e W. Scott Stornetta descreveram o primeiro trabalho sobre uma cadeia de blocos criptografados, só foi notada em 2008, quando se tornou popular com a chegada do bitcoin. Mas actualmente a sua utilização está a ser exigida noutras aplicações comerciais e prevê-se que cresça no futuro médio em vários mercados, tais como instituições financeiras ou a Internet das Coisas da Internet, entre outros sectores.

A cadeia de bloqueio, mais conhecida pelo termo cadeia de bloqueio, é um registo único, acordado e distribuído por vários nós (dispositivos electrónicos tais como PCs, smartphones, tablets, etc.) numa rede. No caso de moedas criptográficas, podemos pensar nisto como o livro de contabilidade onde cada uma das transacções é registada.

O seu funcionamento pode ser complexo de compreender se entrarmos nos detalhes internos da sua implementação, mas a ideia básica é simples de seguir.

É armazenado em cada bloco:

- 1.- uma série de regtos ou transacções válidos,
- 2.- informações relativas a esse bloco,
- 3.- a sua ligação com o bloco anterior e o bloco seguinte através do hash de cada bloco –un código único que seria como a impressão digital do bloco.

Portanto, **cada bloco** tem um **lugar específico e inamovível dentro da cadeia**, uma vez que cada bloco contém informação do hash do bloco anterior. Toda a cadeia é armazenada em cada nó da rede que compõe a cadeia de bloqueio, pelo que **uma cópia exacta da cadeia é armazenada em todos os participantes da rede**.

**O que é um endereço ou uma conta dentro da plataforma Ethereum da cadeia de bloqueio?**

É uma cadeia de 42 caracteres na plataforma Ethereum que representa um número em base hexadecimal, onde os bens definidos no Ethereum serão depositados ou enviados. Em outras plataformas de cadeia de bloqueio, o número de caracteres da conta ou endereço pode ser diferente, por exemplo:

0x5d2Acdb34c279Aa6d1e94a77F7b18aB938BFb2bB

## O que é uma kryptomoney?

É uma moeda digital ou virtual concebida para funcionar como um meio de troca. Utiliza criptografia (segurança digital) para garantir e verificar as transacções, bem como para controlar a criação de novas unidades de uma criptomoney em particular.

## O que é o Éter?

Ether é a moeda digital nativa da plataforma da cadeia de bloqueio Ethereum, uma plataforma descentralizada desenvolvida em 2013. Um Éter é uma unidade que pode ser dividida em unidades mais pequenas chamadas WEI e tem a seguinte equivalência.

1 Ether = 1.000.000.000.000.000.000 Wei. (Em expressão matemática é  $10^{18}$  ).

## Que unidades são manuseadas quando se usa um Éter?

1	ether
$10^3$	finney
$10^6$	szabo
$10^9$	Shannon ou GWEI isto é utilizado para o GasPrice.
$10^{12}$	babbage
$10^{15}$	lovelace
$10^{18}$	wei

## O que é uma ficha?

As fichas são bens digitais que podem ser utilizados dentro de um determinado ecossistema de projecto.

A principal distinção entre fichas e moedas criptográficas é que as primeiras requerem outra plataforma de cadeia de bloqueio (não a sua própria) para funcionar. O Ethereum é a plataforma mais comum para a criação de fichas, principalmente devido à sua função de contrato inteligente. As fichas criadas na cadeia de bloqueio Ethereum são geralmente conhecidas como fichas ERC-20 embora existam outros tipos de fichas mais especializadas,

tais como a ficha ERC-721 utilizada principalmente para bens colecionáveis (cartões, utilização em jogos de vídeo, obras de arte, etc.).

## O que é o Gás?

O gás é o custo de realizar uma operação ou conjunto de operações na rede Ethereum. Estas operações podem ser várias: desde fazer uma transacção até executar um contrato inteligente ou criar uma aplicação descentralizada.

Por outras palavras, mais simplesmente, o Gás é a unidade para medir o trabalho realizado no Ethereum.

Tal como no mundo físico, no Ethereum também há empregos que custam mais do que outros: se a operação que queremos realizar exige uma maior utilização de recursos pelos nós que formam a plataforma, isto fará com que o Gás também aumente e vice-versa.

O Gás serve principalmente para desempenhar três funções na plataforma Ethereum:

1.- Atribui um custo à execução de tarefas; no Ethereum, a execução de tarefas também tem um custo. Dependendo **da dificuldade da tarefa ou da velocidade a que queremos que essa tarefa seja processada, o custo computacional dessa operação será maior ou menor** em conformidade e o número de Gás aumentará ou diminuirá em proporção.

2.- Fixa o sistema; O sistema Ethereum é um sistema seguro e isto é em grande parte possível graças ao Gás.

Ao exigir o pagamento de uma comissão por cada transacção realizada, a plataforma assegura que nenhuma transacção inutilizável seja processada na rede. Isto ajuda a tornar a cadeia de bloqueio mais leve, uma vez que não irá acrescentar muitos Megabytes inúteis de informação à cadeia de bloqueio.

Além disso, com o Gás, o sistema também está protegido contra "spam" e o uso infinito de loops: instruções para executar tarefas repetitivas por código.

Por exemplo, se o Gás não existisse, nada impediria que uma tarefa se repetisse infinitamente, colapsando o sistema e tornando-o inutilizável.

3.- Recompensar os mineiros; Os mineiros são sistemas externos independentes distribuídos por todo o mundo que estão encarregados de executar as transacções dentro da plataforma Ethereum. Quando fazemos uma transacção ou executamos um contrato inteligente, "pagamos" uma certa quantidade de Gás.

**Este Gás serve para "pagar" aos mineiros pelos recursos que utilizaram** (hardware, electricidade e tempo) e **também acrescenta uma recompensa** pelo seu trabalho. Portanto, poderíamos dizer que o Gás também ajuda a manter o equilíbrio da plataforma.

Falamos de "pagar" gás - entre aspas - porque é isso que as operações custam. Por outras palavras, "paga" as despesas que custa ao Ethereum processar as transacções.

### O que é o preço do gás?

Uma unidade de Gás corresponde à execução de uma instrução, tal como um passo de computador.

Então, como é que os mineiros "descontam" o gás que adquirem como recompensa?

O gás em si não vale nada, e por isso não pode ser cobrado. Para "cobrar" por este gás usado, é necessário dar valor a estes recursos consumidos, ou seja, dar um valor monetário ao trabalho realizado pelos mineiros. A quantidade de Gás utilizada numa transacção ou num contrato inteligente tem um preço equivalente em Éter. Este preço é chamado "preço do gás", é normalmente atribuído pelos mineiros e é variável dependendo do quanto ocupada é a cadeia de bloqueio Ethereum. É normalmente tratado em unidades GWEI.

### O que é o Limite de Gás?

Estes dados indicam o valor máximo de Gás que uma transacção pode consumir para ser válida.

Normalmente, o software utilizado para fazer transacções na rede Ethereum calcula automaticamente uma estimativa da quantidade de Gás necessária para realizar a transacção e mostra-nos imediatamente.

### O que é Gas Station?

É o local onde os valores tratados pelos mineiros são referidos para decidir em consenso qual é o Preço do Gás necessário para realizar os diferentes tipos de transacções na cadeia de bloqueio Ethereum.

Dependendo de quanto de GasPrice é seleccionado, o tempo prioritário dado à execução da transacção será ponderado, normalmente são tratados três tipos de GasPrice: TRADER: ASAP, FAST < 2 minutos, STANDARD < 5 minutos. Estes são valores médios e podem variar o tempo em função da carga de trabalho (tracções) da rede principal do Ethereum.

<https://ethgasstation.info/>

### O que é uma troca?

Uma Bolsa de Kryptomaniac é o ponto de encontro onde se realizam trocas de Kryptomaniacs em troca de fiat money ou outros Kryptomaniacs. Nestas casas de câmbio online é gerado o preço de mercado que marca o valor das criptomónias com base na oferta e procura.

### O que são as taxas de câmbio?

Estas são as taxas para o valor de um Éter nas moedas de cada país. Por exemplo, no dia da criação deste manual, um Éter tem um valor em dólares americanos de \$430,94

### O que é um contrato inteligente?

No Ethereum um contrato Smart é um programa em linguagem de programação chamado Solidity que está dentro da cadeia de blocos Ethereum, que tem instruções para ser executado automaticamente com base em regras pré-estabelecidas, esta propriedade faz uma evolução em toda a cadeia de blocos existente, pois pode criar muitos contratos Smart adaptados a diferentes sectores privados e públicos fazendo operações mais eficientes de empresas ou, quando apropriado, adaptar-se a sistemas ou programas de todos os tipos.

### O que é "nonce"?

É um contador incremental de "número hexadecimal" que mantém o registo das transacções em cada direcção ou conta criada no Ethereum.

### O que é uma transacção?

É a execução ou transferência de algum tipo de activo não tangível que pode receber um valor pré-estabelecido dentro do sistema Ethereum e que pode mais tarde ser alterado para um valor tangível para uma empresa ou pessoa.

### O que é txHash?

É um número hexadecimal que ajuda a rastrear o resultado em detalhe de cada transacção.

### Que tipos de transacções existem?

Tem dois tipos, um é a transacção "offline" que isto cria sem a necessidade de ter ligação à rede principal do Ethereum pode ser armazenada até optar por se ligar à rede do Ethereum e libertar a transacção, tem a vantagem da segurança porque toda a transacção é processada offline o que evita qualquer anomalia que possa estar na ligação à rede. A outra transacção é a "online" que precisa sempre de estar ligada à Internet com as vantagens e desvantagens que a ligação traz em termos de segurança.

### O que é INFURA.io?

Infura.io é uma plataforma que fornece um conjunto de ferramentas e infra-estruturas que permitem aos programadores tirar facilmente a sua cadeia de bloqueios de aplicações dos testes, à escalada, com acesso simples e fiável ao Ethereum e IPFS.

## O que é um endereço na rede Ethereum?

Um endereço ou conta é composto por três partes, o endereço, a chave pública e a chave privada, estas duas chaves são uma cadeia de números e caracteres em formato hexadecimal que são utilizados para enviar e receber (activo) ou éter (moeda digital).

A chave primária nunca deve ser partilhada com ninguém, pois é a que autoriza a libertação do saldo (assina as transacções) mantido na conta.

A chave pública é conhecida de todo o público e é partilhada com qualquer pessoa, pois é a referência para confirmar que a transacção é correcta tanto em termos de valor como para quem a mesma é enviada.

Exemplos:

```
{  
  "private": "429a043ea6393b358d3542ff2aab9338b9c0ed928e35ec0aed630b93adb14a1c",  
  "public":  
    "049b4b7e72701a09d3ee09165bba460f2549494a9d9fd7a95aaac57c2827eac162fd9e105b  
    2461cd6594ca8ca6a8daf10fe982f918be1b0060c87db9cfbcd289a8",  
  "address": "88ab6dcecc3603c7042f4334fc06db8e8d7062d5"  
}
```

## O que é Coinsolidation.org?

É uma plataforma de consolidação de moedas criptográficas, criámos os primeiros endereços híbridos no mundo financeiro centralizado e descentralizado, utilizando tecnologia Blockly que é uma forma de programação visual sem a necessidade de conhecimentos avançados de programação baseados em extensões funcionais. Ver o nosso WhitePaper em [www.coinsolidation.org](http://www.coinsolidation.org)

## Que tipos de redes para testes e rede principal na cadeia de bloqueio Ethereum?

<https://mainnet.infura.io>

Rede principal, rede de produção onde todas as transacções são feitas em tempo real.

<https://ropsten.infura.io>

A rede de teste Ropsten permite que os desenvolvimentos da cadeia de bloqueios testem o seu trabalho num ambiente vivo, mas sem a necessidade de fichas ETH reais, com um algoritmo chamado (*Proof-of-Work*).

<https://kovan.infura.io>

Kovan test network, que ao contrário do seu predecessor ropsten utiliza um algoritmo chamado Proof of Authority.

<https://rinkeby.infura.io>

Test Network, utiliza um algoritmo chamado Proof of Authority (Prova de Autoridade). Um dos mais frequentemente utilizados para testes.

<https://goerli.infura.io>

Goerli é uma rede pública de testes para o Ethereum que o motor de consenso POA (Proof of Authority) apoia com vários clientes. À prova de spam.

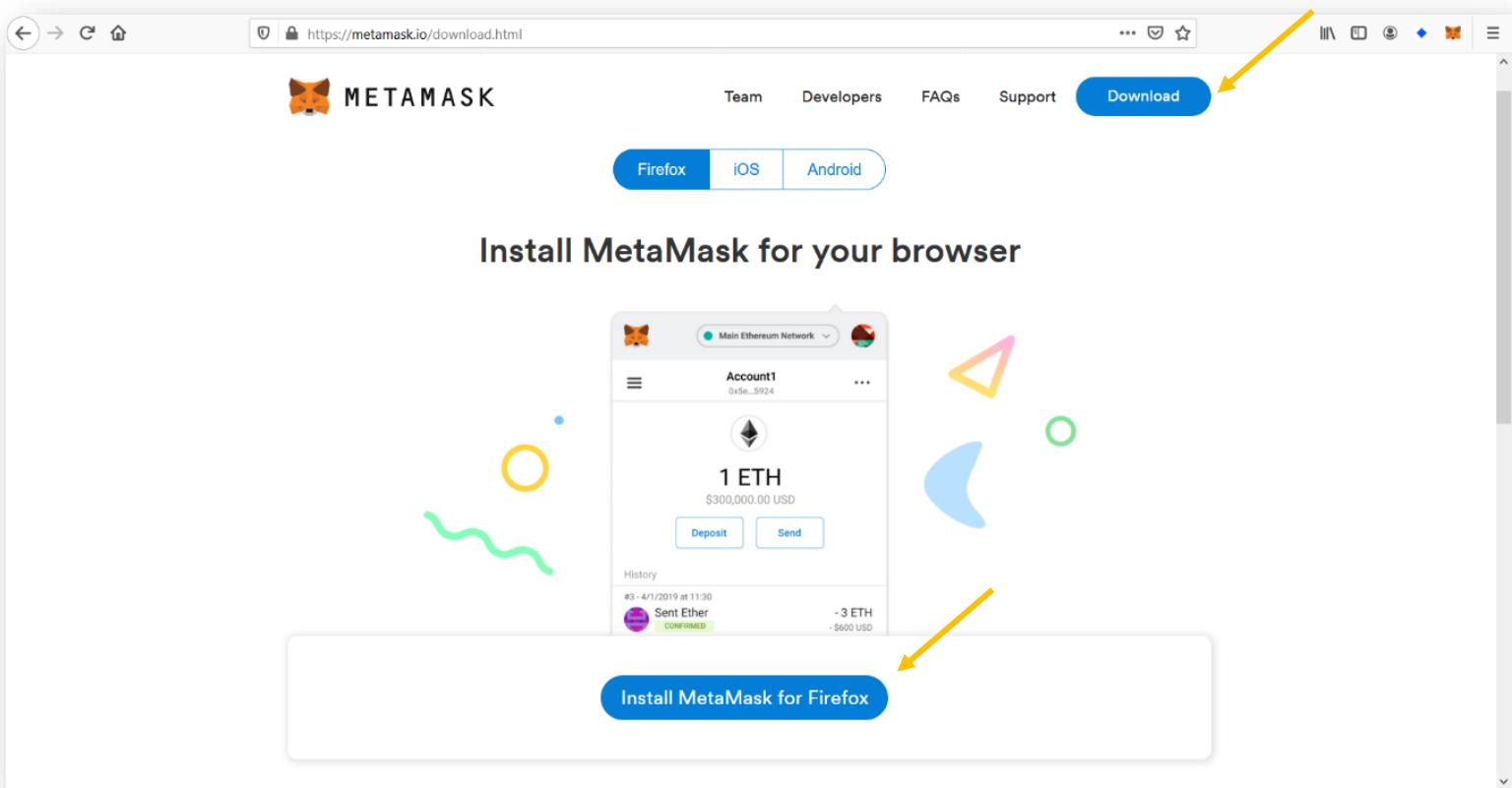
No total existem 5 redes baseadas na cadeia de bloqueio Ethereum, uma para produção ou principal e quatro para testes, então utilizaremos a rede de Rinkeby para instalar o nosso ambiente de teste.

## 6. Instalação e configuração de extensão e ambiente de teste Ethereum.

Precisamos primeiro de um ambiente de teste. Para aprender a utilizar as diferentes funcionalidades das duas extensões que serão utilizadas para criar, enviar, publicar, rever e obter dados de todas as transacções que fazemos na plataforma Ethereum.

A primeira coisa que temos de fazer é criar o nosso ambiente de teste. Teremos de instalar a aplicação **METAMASK**. Esta é uma carteira para criar, importar contas Ethereum e gerir transacções a partir do navegador no nosso navegador de Internet que utilizaremos é Mozilla e também pode ser utilizada em Chrome.

Ir para o site oficial [www.metamask.io](https://metamask.io) e clicar no botão "Down load".

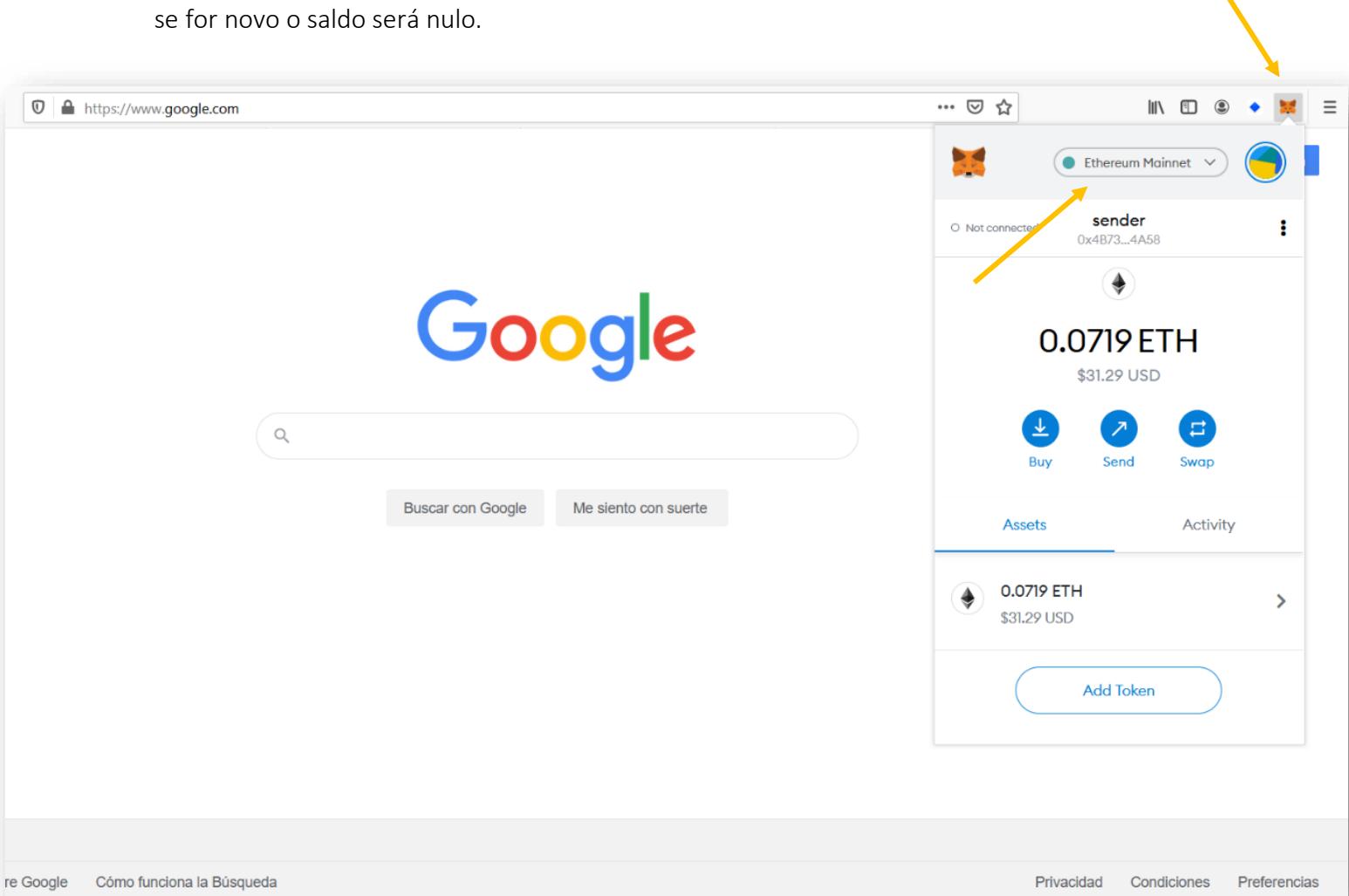


Depois clicar no botão "Install MetaMask for Firefox" e aceitar as opções padrão. Após a instalação, veremos um ícone no lado superior direito, onde veremos já instalado o software Metamask.

Clique no ícone Metamask e aparecerá a opção de importar uma conta existente ou criar uma nova conta. No nosso caso, importaremos uma conta que já temos em funcionamento utilizando o método de "restaurar através da semente". Se não tiver uma, basta criar uma

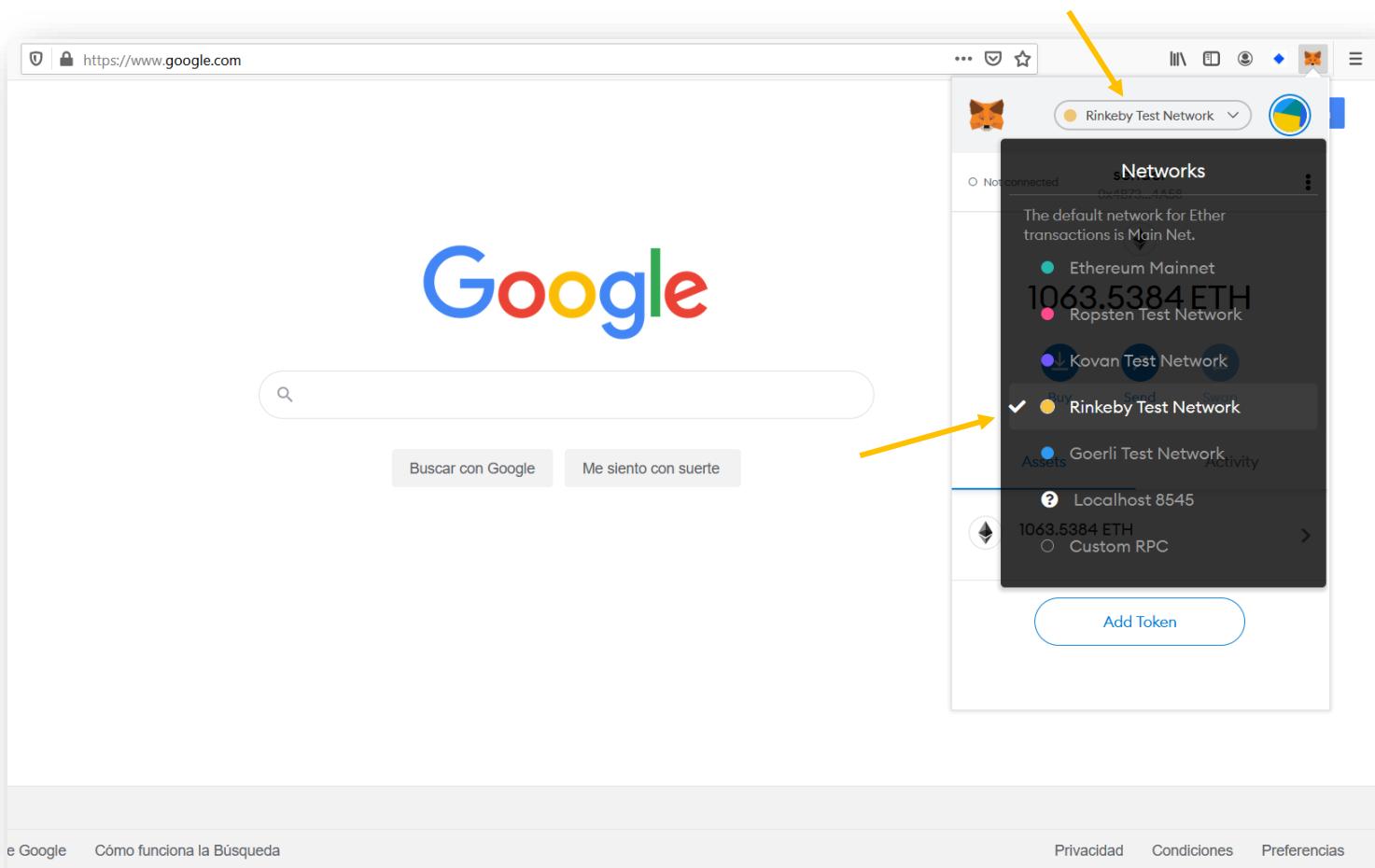
nova conta. Neste caso, ser-nos-á pedida uma palavra-passe a ser atribuída em qualquer dos casos.

Mais tarde entramos com a "password" e podemos verificar que saldo temos, logicamente se for novo o saldo será nulo.



Procedemos agora à revisão de como vamos depositar alguns éteres (moedas digitais) na nossa conta teste Rinkeby.

No topo temos a opção de escolher o tipo de rede que iremos utilizar, por defeito quando a introduzir coloca-o na "Ethereum Mainnet", contudo, quando clica podemos rever todas as redes opcionais que podemos escolher, no nosso caso seleccionamos a rede Rinkeby.



O passo seguinte é depositar éteres na nossa conta de teste referenciada da rede Rinkery.

**NOTA IMPORTANTE:** Os éteres (moeda digital) que depositamos na nossa conta referenciada à rede de teste Rinkery não têm valor no mercado de criptografia, apenas serão por nós utilizados para testes de software. Verificar sempre em que rede estamos a trabalhar para evitar erros nas transacções.

A fim de obter éter para testes, precisamos de realizar o seguinte procedimento.

Em qualquer conta no twitter que tenha teremos de entrar no twitter e criar um comentário que inclua apenas a conta e/ou endereço que temos no Metamask, então teremos de copiar o link do comentário uma vez que o temos, iremos para o próximo link para ancorar a nossa conta.

<https://faucet.rinkeby.io/>

Criámos um comentário no twitter e copiámos o link do comentário.

The screenshot shows a Twitter post from the user AGAVE (@oomissede). The tweet contains the following text:

0x88ac6dcecc3603c7042f4334fc06db8e8d7062d5

Below the tweet, there is a link to "Traducir Tweet". The timestamp is 2:22 a. m. · 6 nov. 2020 · Twitter Web App. There are four interaction icons below the tweet: a speech bubble, a retweet icon, a heart icon, and a share icon.

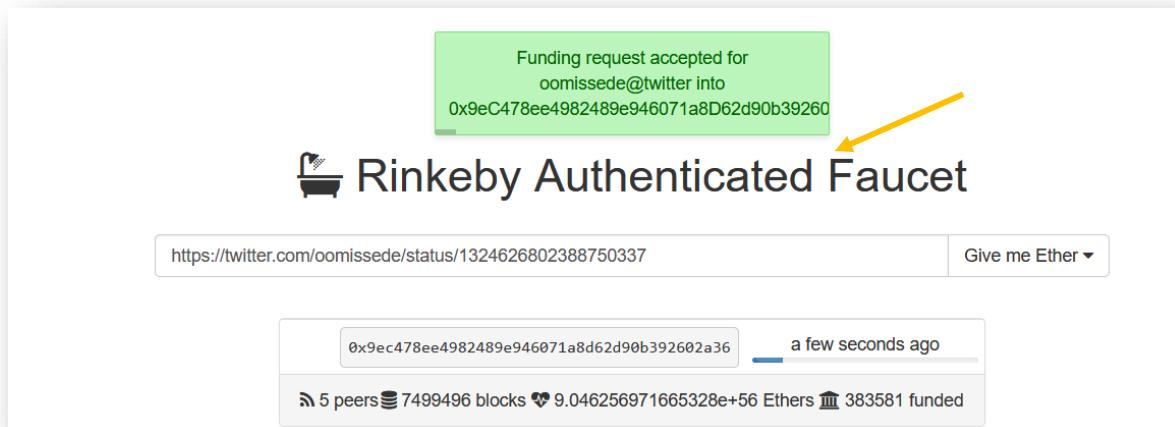
No sítio <https://faucet.rinkeby.io/> copiamos o link do twitter e no botão "Give me Ether" escolhemos a quantidade desejada.

The screenshot shows the Rinkeby Authenticated Faucet website at <https://faucet.rinkeby.io>. A dropdown menu titled "Give me Ether" is open, showing three options:

- 3 Ethers / 8 hours
- 7.5 Ethers / 1 day
- 18.75 Ethers / 3 days

A yellow arrow points to the input field where the Twitter URL was pasted. Another yellow arrow points to the dropdown menu. Below the input field, there is a section titled "How does this work?" with instructions for Twitter and Facebook users. At the bottom, there is a note about reCaptcha protection.

Finalmente, se tudo estiver correcto, daremos um anúncio de que o depósito foi aceite e dependendo da carga de trabalho do sistema de rede Rinkeby o depósito será dentro de alguns minutos ou poderá demorar mais tempo.



Agora que temos Ethers na nossa conta, podemos começar a testar na plataforma Ethereum.

Temos duas extensões para interagir com a cadeia de bloqueio Ethereum.

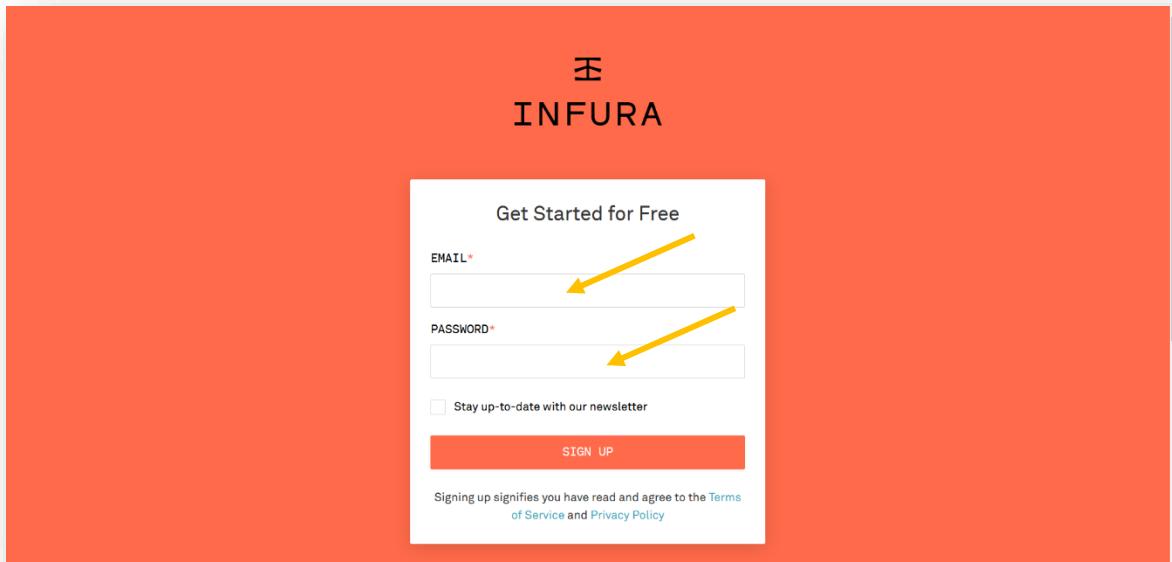
A extensão da **co-soldado. BlockoinETHEREUM.aix** contém as funções para desempenhar as seguintes funções:

- Criação de novas contas (endereços) em blockchain Ethereum (privateKey, publicKey, Address)
- Armazenamento de dados de conta (endereços) em ficheiros binários.
- Importação de contas de ficheiros binários (endereços)
- Obtenção de uma conta (endereço) através da PrivateKey.
- Criação e envio de transacções entre contas (endereços) "online".
- Criação, assinatura e envio de transacções PushRaw offline.
- Consulta dos detalhes da transacção Tx.
- Verificação do saldo de endereços e contratos inteligentes.
- Compilador para contratos Smart.
- Criação, publicação e execução de contratos inteligentes.
- Criação, publicação e execução do Token ERC20 (token cryptomoney).
- Obtenção do código ABI a partir de um contrato inteligente.
- Verificação da ligação à rede.
- Consulta do valor do éter no mercado criptográfico em qualquer país do mundo (moeda nativa do país) - Taxas de câmbio.
- Consulta sobre o preço do gás.
- Consulta de "nonce" de uma conta específica.

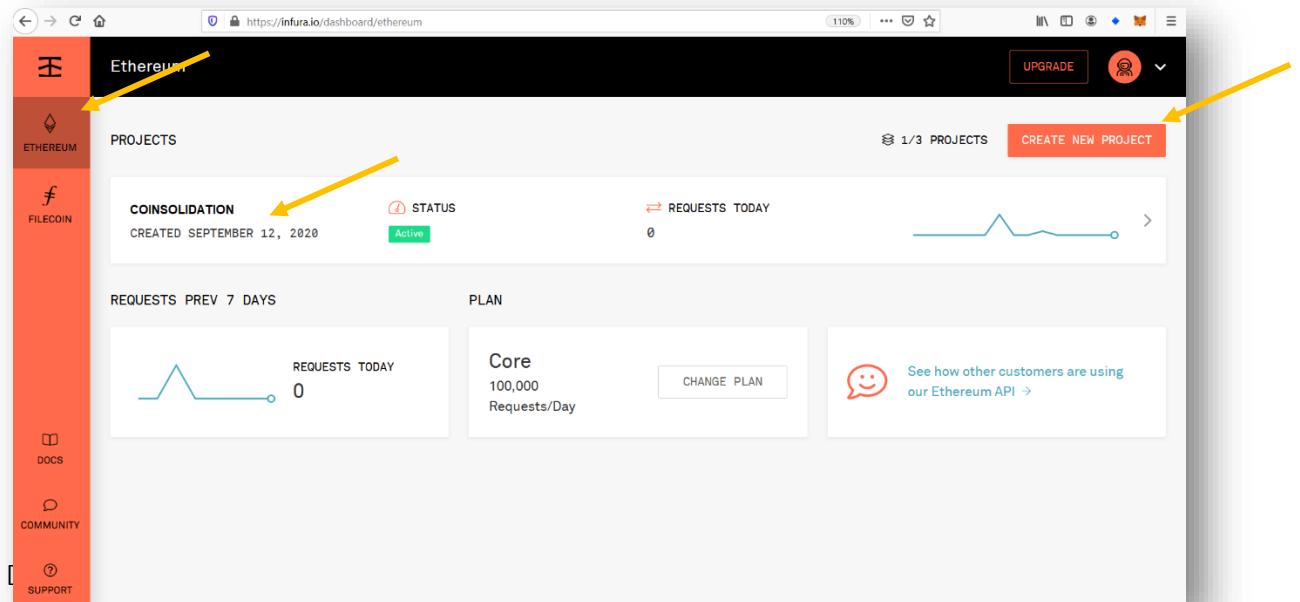
A extensão da **coinsoidação**. **BlockoinINFURA.aix** fornece-nos as 40 funcionalidades da plataforma Infura.io. Para mais detalhes, consulte directamente a documentação json-rpc em <https://infura.io/docs>

Para utilizar a extensão INFURA é necessário criar uma conta no site infura.io, uma vez que precisamos de uma API KEY para podermos enviar consultas à rede Ethereum, bem como para podermos utilizar as redes de teste Ethereum.

A forma como abrimos uma conta é simples, como se mostra abaixo.

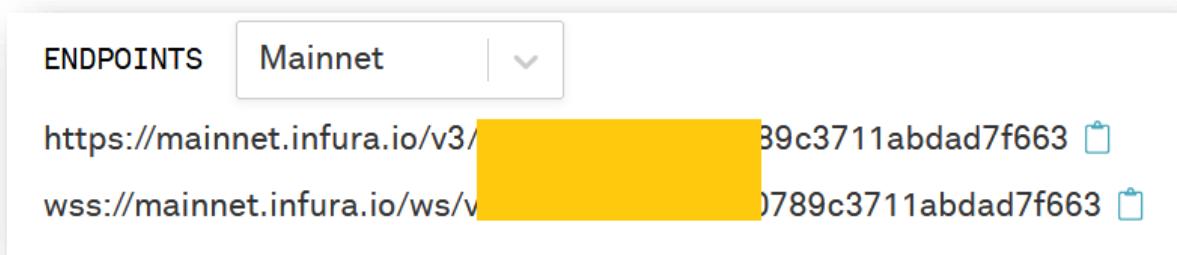


Uma vez criada a conta, entramos e podemos ter a KEY API das diferentes redes. Vamos para o canto superior esquerdo e clicamos no Ethereum, depois vemos os projectos, **criamos** um novo projecto e introduzimo-nos a esse projecto.



Dentro do projecto podemos rever a CHAVE API (PROJECT ID) e seleccionar em que rede iremos trabalhar ou as ligações completas do ENDPOINTS a escolher.

Por exemplo, para utilizar a rede principal do Ethereum, tomaríamos a seguinte liga:

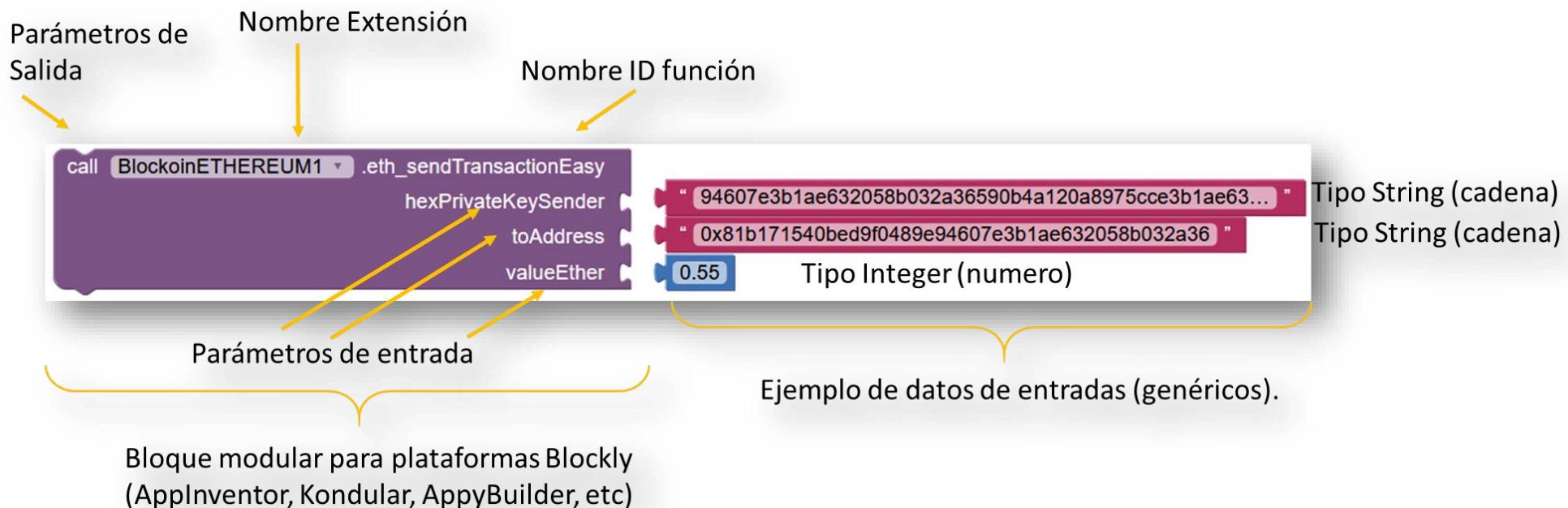


**NOTA:** As extensões foram testadas nos sistemas AppInventor, Kondular, Thunkable e AppyBuilder.

## 7. Definição e utilização de blocos (função genérica)

Começaremos por explicar a distribuição dos dados que todos os blocos terão, a sua sintaxe de utilização e configuração.

No exemplo seguinte podemos ver um bloco modular e os seus parâmetros de entrada e saída, assim como os tipos de dados de entrada, estes dados podem ser do tipo String (cadeia de caracteres) ou Inteiro (inteiro ou decimal). Mostramos como é utilizado e configuramos para o seu bom funcionamento.



Cada bloco de módulos terá a sua descrição e será nomeado caso tenha alguma dependência(s) obrigatória(s) ou opcional(s) de outros blocos utilizados como parâmetros de entrada, o processo de integração será anunciado.

## 8. Intercâmbio de características e eventos de Extensão de Etéreo (EEE).

\*\*\*Rede de teste que utilizaremos **Rinkeby**, como urlNetwork, quando quiser fazer transações reais só terá de mudar a rede urlNetwork para **rede principal**.

Bloco para verificar a ligação à Internet - (**CheckInternetConnection**).

call BlockoinETHEREUM1 .CheckInternetConnection

Parâmetros de entrada: Não aplicável.

Parâmetros de saída: Retorna "Verdadeiro" se tiver ligação ou retorna "Falso" se não houver ligação.

Descrição: Bloco para verificar a ligação à Internet e enviar dados (transacções).

Bloco para gerar um novo endereço "Offline" - (**GenerateNewAddressEthereum**)

call BlockoinETHEREUM1 .GenerateNewAddressEthereum  
phraseHex "exchange ethereum extension for systems blockly"

Parâmetros de entrada: **phraseHex <String>**.

Parâmetros de saída: Evento (**OutputGenerateNewAddressEthereum**)

Saídas: **PrivateKey<String>**, **PublicKey<String>** , **addressEthereum<String>**.

when BlockoinETHEREUM1 .OutputGenerateNewAddressEthereum  
privKeyEther pubKeyEther addressEthereum  
do

Descrição: Criar um novo endereço de etéreo (conta) com base numa frase ou sequência de números. O novo endereço pode ser criado sem ligação à rede ou à Internet - "Offline".

Bloco para gerar um novo endereço "Online" - (**GenerateNewAddressEthereum**)

call BlockoinETHEREUM1 .GenerateNewAddressEthereumAPI

Parâmetros de entrada: Não aplicável.

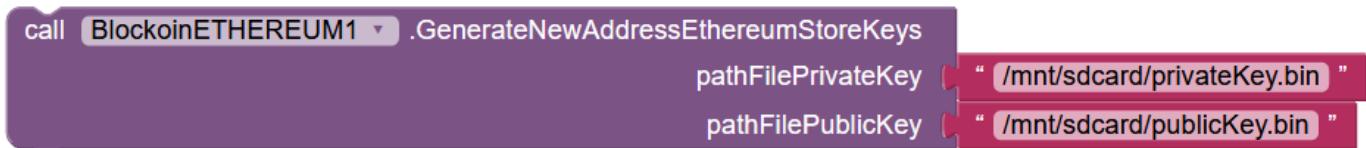
Parâmetros de saída: devolução em formato de dados JSON; privateKey, publicKey, endereço.

Exemplo de saída:

```
{
  "private": "9ab4b2fba728a55643414f26adc04eea080740860cbe39b13fe4acb43dbb9f83",
  "public": "0421d559aa98d6fc77688ae9f19b3dc502c05f0b7f70bbe924cb712d6243a489695b1c1ee03993b3b6d97277
97e780677a5469800b4d98374bdb910ed99fa2b5c8",
  "address": "92a2f157d5aec3fa79f92995fea148616d82c5ef"
}
```

Descrição: Criar um novo endereço de etéreo (conta). É necessário ter acesso ou ligação à Internet uma vez que a geração é através do serviço REST API - "Online".

Bloco para gerar um novo endereço "Offline" e guardar as chaves públicas e privadas em ficheiros binários - ([GenerateNewAddressEthereumStoreKeys](#))



Parâmetros de entrada: `pathFilePrivateKey<String>`, `pathFilePublicKey<String>`.

Parâmetros de saída: Evento ([OutputGenerateNewAddressEthereumStoreKeys](#))

Saídas: `endereçoEthereum<String>` , `privateKeyECC<String>` , `publicKeyECC<String>` , `privateKeyHex<String>` , `publicKeyHex<String>`.



Descrição: Cria um novo endereço etéreo aleatório (conta) e armazena as chaves públicas e privadas em ficheiros binários que serão utilizados para a importação e exportação de dados de contas. O novo endereço pode ser criado sem ligação à rede ou à Internet - "Offline".

Bloco para gerar chave pública - (**GeneratePublicKeyHexFromPrivateKeyHex**).



Parâmetros de entrada: **hexPrivateKey <String>**.

Parâmetros de saída: Evento (**OutputGeneratePublicKeyHexFromPrivateKeyHex**)

Saídas: **endereço<String>** , **publicKeyHex<String>**.



Descrição: Cria a chave pública com base na entrada de uma chave privada.

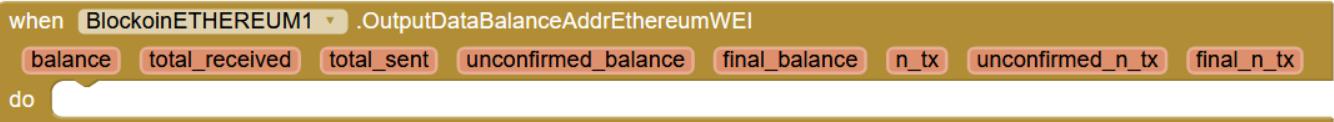
Bloco para obter o saldo de um endereço - (**GetBalanceAddrEthereum**).



Parâmetros de entrada: **hexPrivateKey <String>**.

Parâmetros de saída: Evento (**OutputGeneratePublicKeyHexFromPrivateKeyHex**)

Saídas: **balanço<Corda>** , **total\_recebido<Corda>** , **total\_enviado<Corda>** , **balanço\_finalizado<Corda>** , **balanço\_final<Corda>** , **n\_tx<Corda>** , **não-confirmado\_n\_tx<Corda>** , **final\_n\_tx<Corda>**.



Descrição: Apresenta o balanço e dados detalhados da conta (endereço).

Bloco para verificar se o seu telemóvel tem a interface de rede activada - (GetDataNetworkConnection).

```
call BlockoinETHEREUM1 .GetDataNetworkConnection
```

Parâmetros de entrada: Não aplicável.

Parâmetros de saída: Evento (OutputGeneratePublicKeyHexFromPrivateKeyHex)

Saídas: interfacename<String>, isconnected<String>.

```
when BlockoinETHEREUM1 .OutputGetDataNetworkConnection
    interfacename
    isconnected
do
```

Descrição: Apresenta o nome da interface móvel e fornece se a interface está activada ou não.

Bloco para assinar transacção "Offline" - (SignerGenericPushRawTransactionOffline)

```
call BlockoinETHEREUM1 .SignerGenericPushRawTransactionOffline
    urlNetwork " (https://rinkeby.infura.io/v3/...440789c3...)"
    hexPrivateKeySender " (9eC478ee49824890b4a120a8975cce3b1ae632058b0301f8...)"
    nonceNumber get global nonce
    gasPrice " (250000000000"
    gasLimit 21000
    toAddress " (0x92a2f157d5aec3fa79f92995fea148616d82c5ef"
    valueWei " 10000000000000000000 " x " 0.01 "
```

Dependência(ões) necessária(s): Bloco (eth\_getTransactionCount), Bloco (eth\_SendRawTransactionInfura)

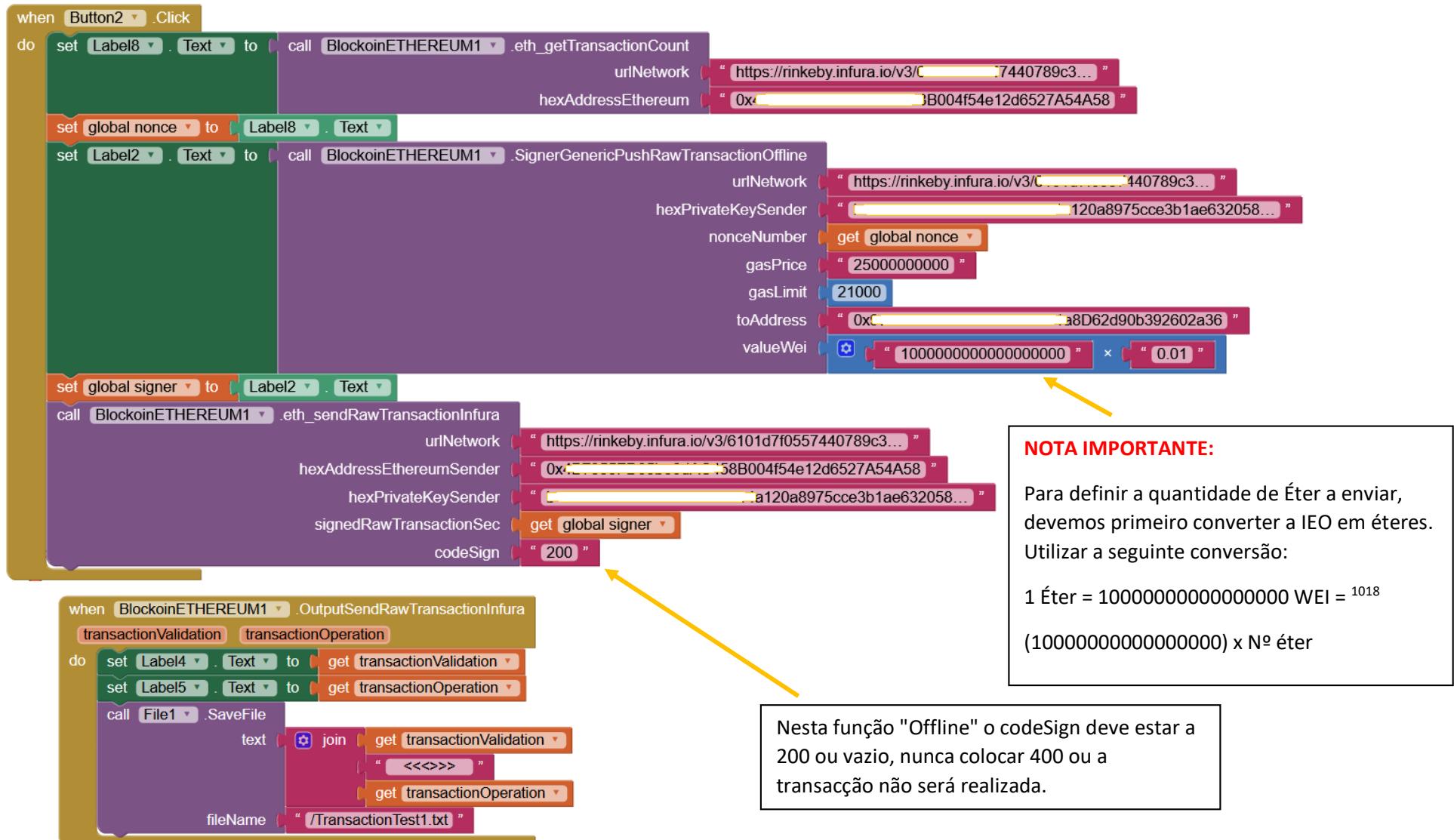
Parâmetros de entrada: urlNetwork <String>, hexPrivateKeySender <String>, nonceNumber <String>, gasPrice <String>, gasLimit <Integer>, toAddress <String>, valueWEI <Integer>.

Parâmetros de saída:

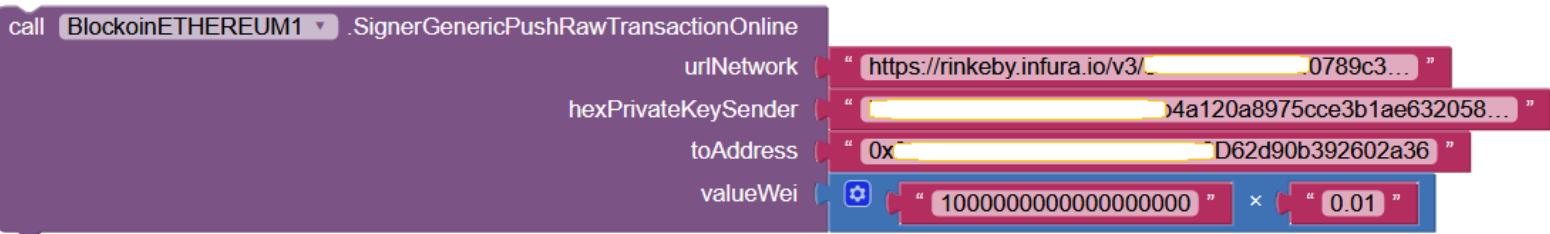
Saídas: Transacção assinada a ser enviada. <Calça>.

Descrição: Prepara uma nova transacção a ser enviada (encriptada e assinada). Isto pode ser processado sem ligação à rede ou à Internet - "Offline".

Exemplo de utilização plena com dependências de blocos (`SignerGenericPushRawTransactionOffline`).



Bloco para assinar a transacção "Online" - (**SignerGenericPushRawTransactionOnline**).

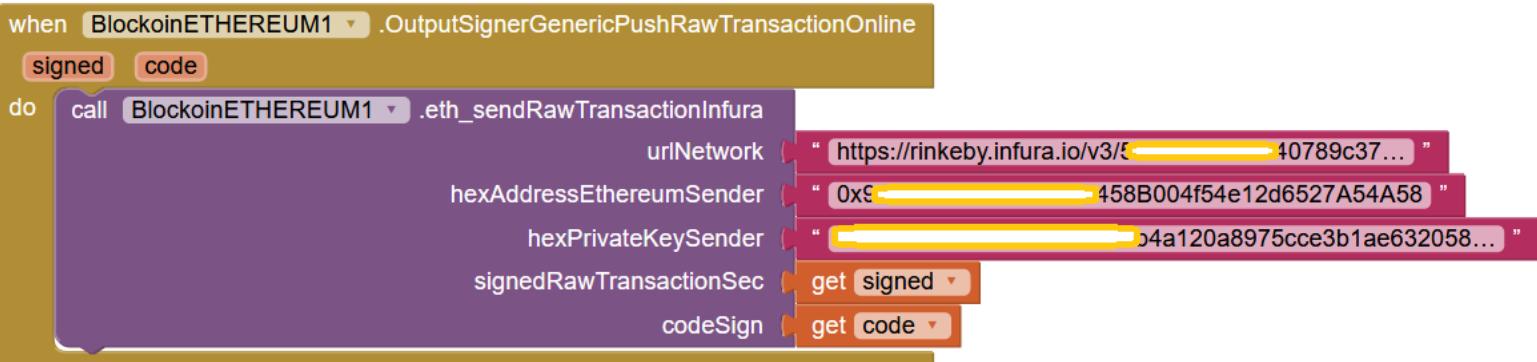


Unidade(s) obrigatória(s): Bloco (**eth\_SendRawTransactionInfura**).

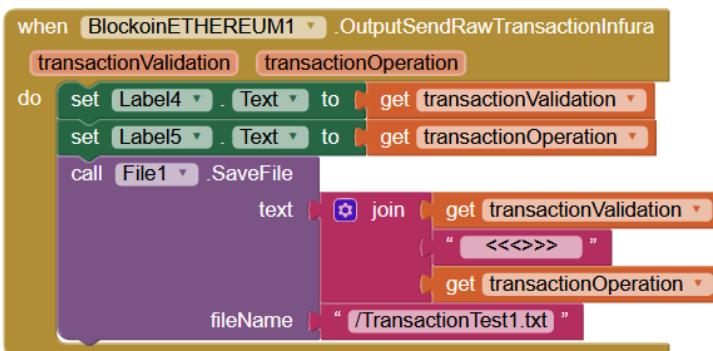
Parâmetros de entrada: **urlNetwork <String>**, **hexPrivateKeySender <String>**, **toAddress <String>**, **valueWEI <Integer>**.

Parâmetros de saída: Eventos utilizados na seguinte ordem (**OutputSignerGenericPushRawTransactionOnline**) e (**OutputSendRawTransactionInfura**).

Saídas: **assinado<Calça>**, **código<Calça>**.

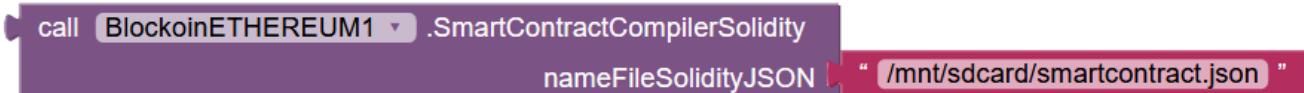


As saídas bloqueiam eth\_sendRawTransactionInfura: **transactionValidation<String>** , **transactionOperation<String>**.



Descrição: Prepara uma nova transacção a ser enviada (encriptada e assinada). É necessária uma ligação à rede ou à Internet - "Online".

Bloco para **compilação** de contrato Smart "Online" - (**SmartContractCompilerSolidity**).



Parâmetros de entrada: **nameFileSolidityJSON <String>**.

Parâmetros de saída: Mostra o contrato inteligente compilado, esta função ajuda-nos a verificar se está bem escrito antes de publicar um contrato inteligente na rede Ethereum.

Saídas: **Código compilado**.

O contrato Smart deve estar num ficheiro em formato JSON.

Exemplo de um contrato básico Smart em linguagem Solidity.

```
pragma solidez ^0.5.0;

contrato fatal {
    proprietário do endereço;
    função mortal() { proprietário = msg.sender; }
    função kill() { se (msg.sender == dono) suicídio(dono); } }
    o cumprimento do contrato é mortal {
        saudação de corda;
        função saudação pública { saudação = _ saudação; }
        função de saudação() retornos constantes (string) {retorno de saudação; }
```

Exemplo de contrato anterior Smart em formato JSON com comentários.

```
# Verificar a compilação da solidez através de teste não publicado
# Usando o exemplo de solidez contratual "greeter", o "olá mundo" do
Ethereum.
```

Ficheiro: smartcontract.json

```
{
    "solidez": "contrato mortal" {
        /* Definir proprietário variável do tipo endereço*/
        endereço proprietário; /* esta função é executada na inicialização e define o proprietário do contrato */
        função mortal() { owner = msg.sender; } /* Função para recuperar os fundos do contrato */
        função kill() { if (msg.sender === owner) suicide(owner); }
        contract greeter is mortal { /* define a saudação variável do tipo string */
            string greeting; /* este é executado quando o contrato é executado */
            função cumprimentar (string _greeting) público { saudação = _greeting; }
        }
    }
}
```

```
/* função principal */n      função cumprimentar() retornos constantes
(string) {\n    saudação de           retorno;{\n    }      ",
"params". ["Hello Coinsolidation Test"]
}
```

**NOTA IMPORTANTE:** O formato JSON deve ter sempre uma quebra de linha no final de cada linha.

Exemplo de compilação da produção de Smartcontract.

```
[
{
  "nome": " u003cstdin\u003e:greeter",
  "solidez": "contrato mortal" {\n    /* Definir proprietário variável do\n    tipo endereço*/\n    endereço proprietário;\n    /* esta função é\n    executada na inicialização e define o proprietário do contrato */\n    função mortal() { owner = msg.sender; }\n    /* Função para recuperar os\n    fundos do contrato */\n    função kill() { if (msg.sender === owner)\n        suicide(owner); }\n    contract greeter is mortal {\n        /* define a\n        saudação variável do tipo string */\n        string greeting;\n        este é\n        executado quando o contrato é executado */\n        função cumprimentar\n            (string _greeting) público {\n                saudação = _greeting; }\n        /* função principal */\n        função cumprimentar() retornos constantes\n            (string) {\n                saudação de           retorno;{\n                }      ",
    "caixote do lixo": "606060405260405161023e38038061023e8339810160405280510160008054600160a060020a031916331790558060016000509080519060200190828054600181600116156101000203166002900490600052602060002090601f016020900481019282601f10609f57805160ff19168380011785555b50608e9291505b808211560cc57600081558301607d565b50505061016e806100d06000396000f35b828001600101855582156076579182015b8281115607657825182600050559160200191906001019060b0565b509056606060405260e060020a600035046341c0e1b58114610026578063cfaf321714610068575b005b6100246000543373fffffffffffff908116911614156101375760005473fff\nfffffffffffff16ff5b6100c9600060609081526001805460a06020601f6002600019610100868816150201909416939093049283018190040281016040526080828152929190828280156101645780601f1061013957610100808354040283529160200191610164565b60405180806020018281038252838181518152602001915080519060200190808383829060006004602084601f0104600f02600301f150905090810190601f1680156101295780820380516001836020036101000a031916815260200191505b50925050506
  "abi": [
    {
      "constante": falso,
      "inputs": [],
      "nome": "matar",
      "outputs": [],
      "tipo": "função".
    },
    {
      "constante": é verdade,
      "inputs": [],
      "nome": "saudar",
      "outputs": [
        {

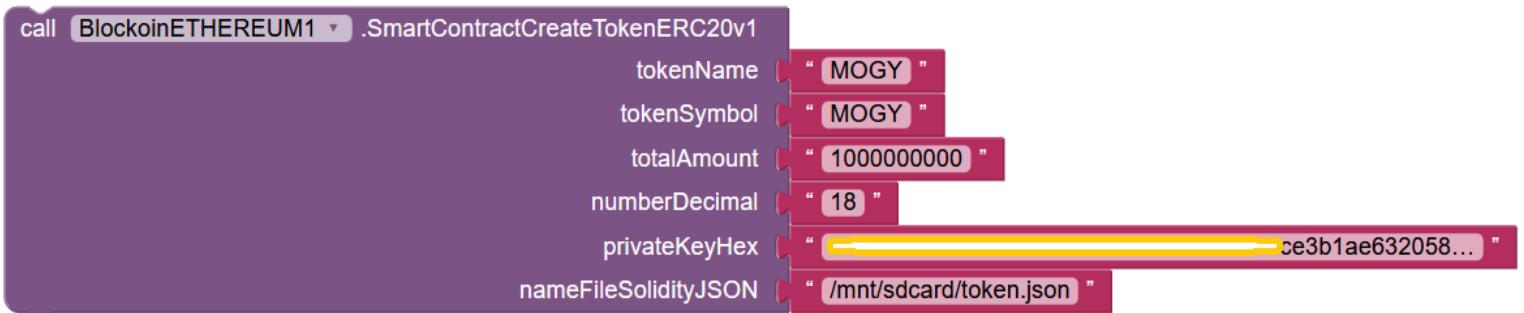
```

```

        "nome": "",
        "tipo": "corda"
    },
],
"tipo": "função".
},
{
"inputs": [
{
    "nome": "_ saudação",
    "tipo": "corda"
}
],
"tipo": "construtor"
}
],
"params": [
    "Olá Coinsolidation Test"
]
},
{
    "nome": "mortal",
    "solidez": "contrato mortal" /* Definir proprietário variável do tipo endereço*/
        endereço proprietário; /* esta função é executada na inicialização e define o proprietário do contrato */
        função mortal() { owner = msg.sender; }/* Função para recuperar os fundos do contrato */
        função kill() { if (msg.sender === owner) suicide(owner); }/*contract greeter is mortal */
        /* define a saudação variável do tipo string */
        string greeting; /* este é executado quando o contrato é executado */
        função cumprimentar(string _greeting) público { saudação = _greeting; }
        /* função principal */
        função cumprimentar() retornos constantes (string) { saudação de retorno; },
        "caixote do lixo":
"606060405260008054600160a060020a03191633179055605c8060226000396000f36060
60405260e060020a600035046341c0e1b58114601a575b005b60186000543373ffffffffffff
fffffffffffffffffffffffffffff90811691161415605a5760005473fffffffffffff
fffffffffffffffffffff16ff5b56",
    "abi": [
    {
        "constante": falso,
        "inputs": [],
        "nome": "matar",
        "outputs": [],
        "tipo": "função".
    },
    {
        "inputs": [],
        "tipo": "construtor"
    }
],
"params": [
    "Olá Coinsolidation Test"
]
}
]

```

Bloco para compilar, criar e publicar o Token ERC20 - (SmartContractCreateTokenERC20v1)

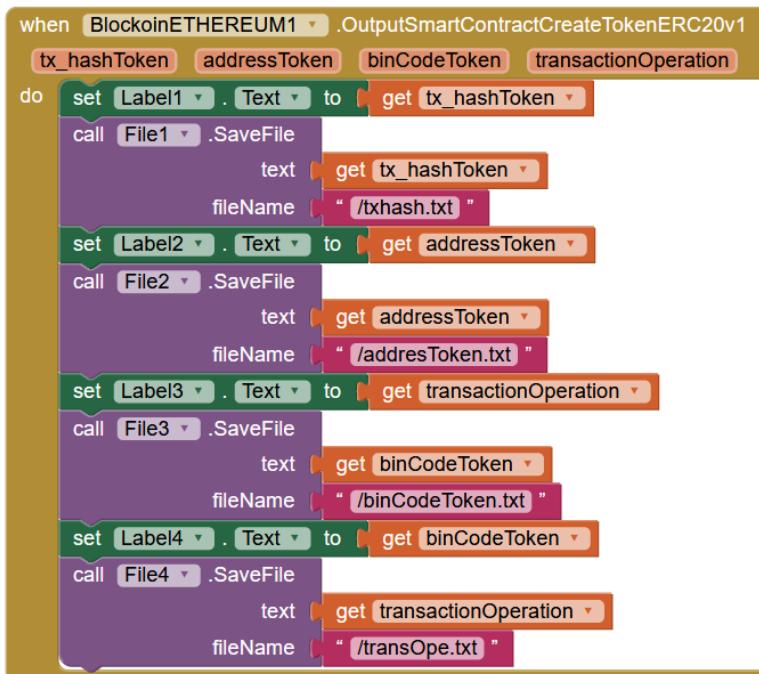


Unidade(s) obrigatória(s): Bloco (**CreateTestingFie**). **IMPORTANTE**: Primeiro deve usar este bloco para se certificar de que o caminho está correcto, isto porque se não der um caminho válido no bloco (**SmartContractCreateTokenERC20v1**) a criação do token não será executada porque o parâmetro de entrada "nameFileSolidityJSON" é usado para criar um ficheiro temporário.

Parâmetros de entrada: **tokenName <String>**, **tokenSymbol <String>**, **totalAmount <String>**, **numberDecimal <String>**, **privateKeyHex <Integer>**, **nameFileSolidityJSON <String>**. Este ficheiro é o caminho válido para **criar um ficheiro temporário**, deve certificar-se de que o caminho é válido para testar que o ficheiro é criado, pode utilizar o Bloco (**CreateTestingFile**) depois de o utilizar para verificar se foi criado com sucesso.

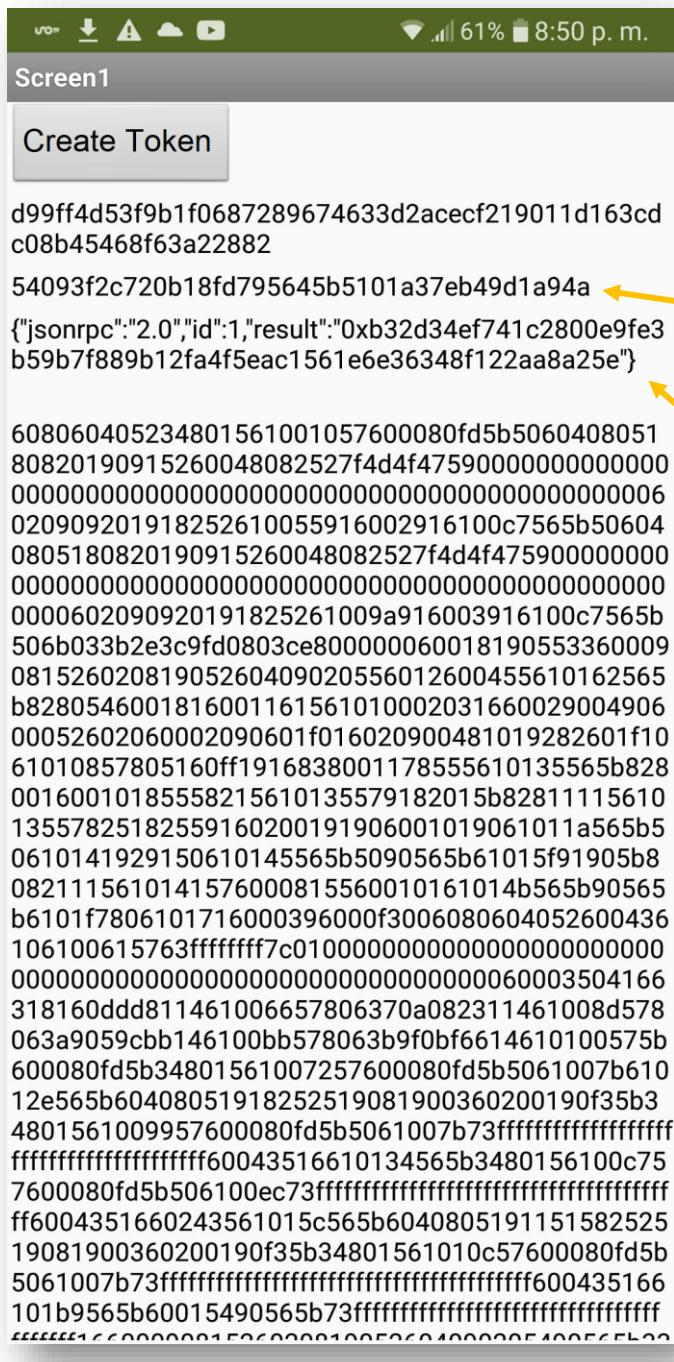
Parâmetros de saída: Evento (**OutputSmartContractCreateTokenERC20v1**).

Saídas: **tx\_hashToken<String>** , **addressToken<String>** , **binCodeToken<String>** , **trasactionOperation<String>**.



Descrição: Contrato inteligente "Token ERC20" - activo publicado na rede Ethereum. A versão 1 (v1) já tem o parâmetro Limite de gás definido em 500.000 WEI.

Exemplo de saída da função anterior `SmartContractCreateTokenERC20v1`.



Transacção (Tx\_hash) da criação da rede Ethereum. Isto pode ser consultado em: [www.etherscan.io](http://www.etherscan.io)

Endereço do novo contrato que se refere à nova ficha ERC20 criada.

Transacção (Tx\_hash) da operação  
validada no sistema  
CoinSolidation.org

Isto pode ser encontrado em:  
[www.etherscan.io](http://www.etherscan.io)

Código Binário (BIN) do novo contrato simbólico que foi processado na EVM (Ethereum Virtual Machine).

## 9. Passos para criar um Token CryptoToken ou Cryptomoney.

Passo 1.

Verificar se o caminho temporário no dispositivo móvel onde o contrato Smart foi criado é válido e se um ficheiro pode ser criado com sucesso. Isto é feito utilizando o Bloco (**CreateTestingFile**). Verificar se o ficheiro de teste dado na entrada "pathTestFile" foi criado, genericamente o caminho é dado por: [/mnt/sdcard/name\\_file.txt](/mnt/sdcard/name_file.txt)

Passo 2 (opcional).

Nesta etapa, iremos verificar se a conta (endereço) a partir da qual a operação será cobrada tem saldo suficiente para realizar a transacção de criação e publicação de um contrato Smart. Isto pode ser verificado utilizando o bloco (**eth\_VerifiBalanceForTransaccionSmartContract**). Esta utilização deste bloco é opcional, uma vez que os blocos que geram o **SmartContractCreateTokenERC20v1** ou **SmartContractCreateTokenERC20v2** já contêm internamente esta verificação.

Passo 3.

Seleccione o bloco a utilizar para criar a ficha ERC20, tem duas opções:

a.- Block **SmartContractCreateTokenERC20v1** já tem o valor implícito de Limite de Gás atribuído com um valor de 500.000 WEi.

b.- Block **SmartContractCreateTokenERC20v2** tem a opção de ser capaz de configurar o Limite de Gás à necessidade do utilizador final ou desenvolvedor. Note-se que se for dado um limite de gás muito baixo, inferior a 350.000 Wei, é muito possível que o contraste Smart.

Quarto passo.

Utilize os blocos **SmartContractCreateTokenERC20v1** ou **SmartContractCreateTokenERC20v2**, certificando-se de que ao utilizar a variável de entrada "nameFileSolidityJSON" é igual à variável de entrada "pathTestFile" do bloco (**CreateTestingFile**) já verificada no passo 1.

Passo cinco.

Antes de executar a criação de um token ERC20 com qualquer um dos blocos **SmartContractCreateTokenERC20v1** ou **SmartContractCreateTokenERC20v2**, recomenda-se guardar os valores do Evento (resultados) conforme apropriado para guardar os resultados (tx\_hashToken, addressToken, binCodeToken, transactionOperation). Ver exemplo de saída da função anterior **OutPutSmartContractCreateTokenERC20v1**.

Sexto passo.

Executar a criação da ficha ERC20 e depois publicá-la para venda. Ver secção 10.

## 10.Como colocar à venda um novo bem ou a sua cript token (Token ERC20).

Desde que criámos um ERC20 - Ficha Cryptomoney (Ver Bloco **SmartContractCreateTokenERC20v1** ou com o Bloco **SmartContractCreateTokenERC20v2**) temos de o carregar para alguma Bolsa de Valores para que qualquer pessoa no mundo o possa comprar. Um Exchange é um site na Internet onde são publicadas novas fichas.

Os intercâmbios são classificados em dois tipos, Centralizado e Descentralizado. A principal diferença é que um é governado e auditado por algum tipo de organismo internacional (Centralizado) e os Descentralizados não têm ninguém com os auditores. Embora isto possa dar mais confiança, a realidade é que recentemente os descentralizados tomaram mais força e são utilizados sobretudo sem grandes problemas.

Uma das melhores práticas no tratamento de bens de qualquer tipo é não os ter todos numa conta, mas distribuí-los em várias contas para a segurança das chaves privadas e públicas.

No nosso caso utilizaremos um Intercâmbio Descentralizado mas já com uma história nada má, utilizaremos [www.forkdelta.app](https://www.forkdelta.app)

Neste momento já devemos ter instalado a aplicação para o browser (Mozilla ou Chrome) **METAMASK** [www.metamask.io](https://www.metamask.io) isto porque a Bolsa para visitar a sua página [www.forkdelta.app](https://www.forkdelta.app) precisa de se ligar à conta que temos Ethereum.

Um ponto importante é que a nossa conta Ethereum que já temos no METAMASK deve ter um saldo de mais ou menos 10 USD isto porque quando publicamos o nosso novo Token ERC20 que criámos com o bloco **SmartContractCreateTokenERC20v1** ou com o bloco **SmartContractCreateTokenERC20v2** teremos de pagar a transacção para o publicar na Bolsa.

A fim de utilizar a Bolsa [www.forkdelta.app](https://www.forkdelta.app) devemos ter os seguintes dados simbólicos que queremos publicar para venda na Bolsa.

Endereço da nova ficha ERC20 que criámos anteriormente.

**0x54093F2C720b18Fd795645b5101A37EB49d1A94a**

Número de decimais que o nosso toque utiliza.

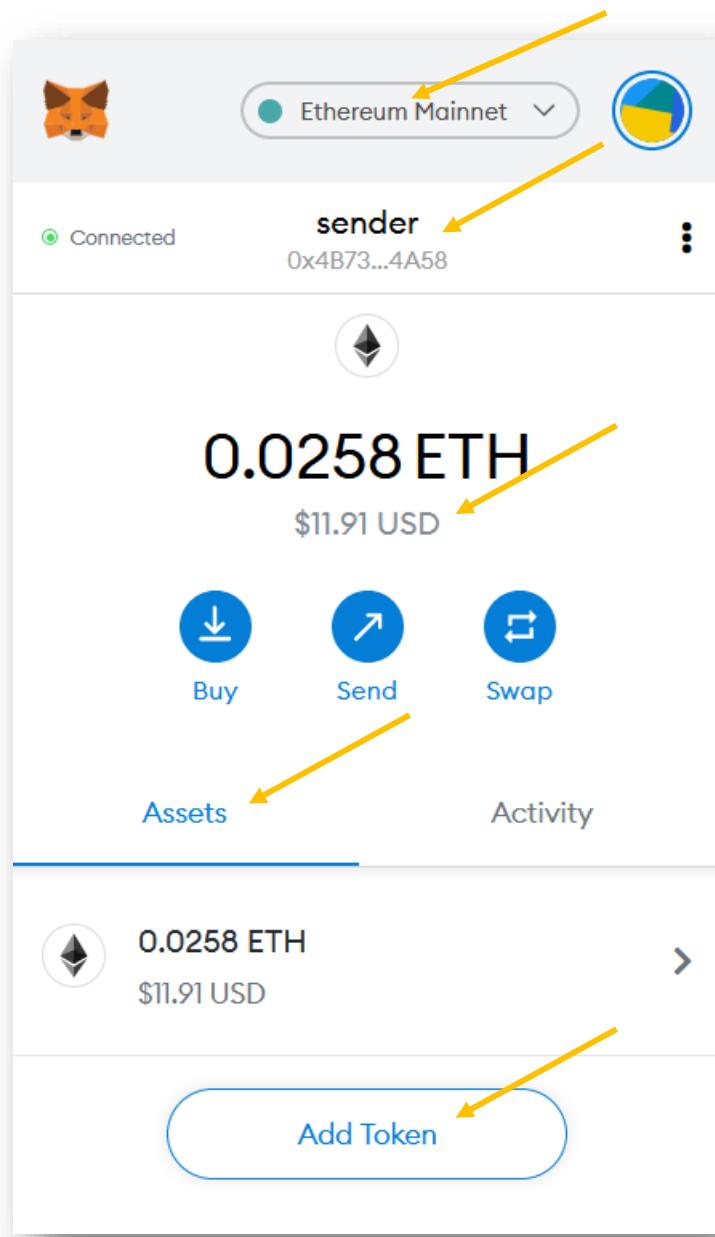
**18**

O nome que identifica o símbolo.

**MOGIA**

Comecemos por verificar se a ficha que criámos tem os parâmetros acima mencionados e se foram os parâmetros com que foi inicialmente criada.

Vamos a **METAMASK** e primeiro certificarmo-nos de que estamos na conta com a qual criámos a ficha. Depois ir para o fundo e clicar no botão "Add Token".



Vamos agora adicionar o nosso novo símbolo para ver o seu equilíbrio actual. Depois de clicar no botão "Ficha personalizada" no topo da página, introduzir as informações solicitadas nos campos seguintes e depois clicar no botão "Seguinte". Depois disso, o nosso novo símbolo aparecerá com o equilíbrio que tem. Se não tiver um saldo, verifique se está na conta com a qual criou o token. Se não estiver numa conta com a qual criou o token, terá um saldo zero porque ainda não comprou nenhum token.

The image consists of two side-by-side screenshots of the Coinsolidation.org mobile application interface.

**Screenshot 1: Add Tokens Screen**

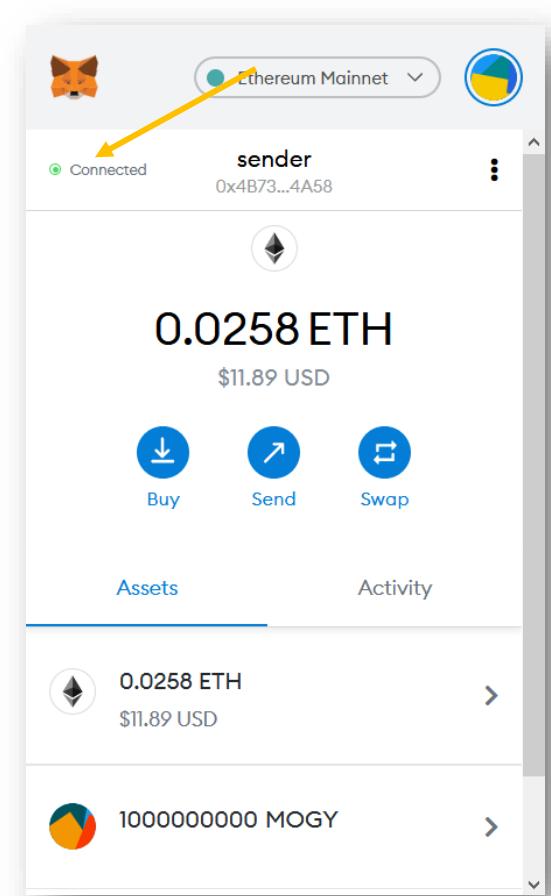
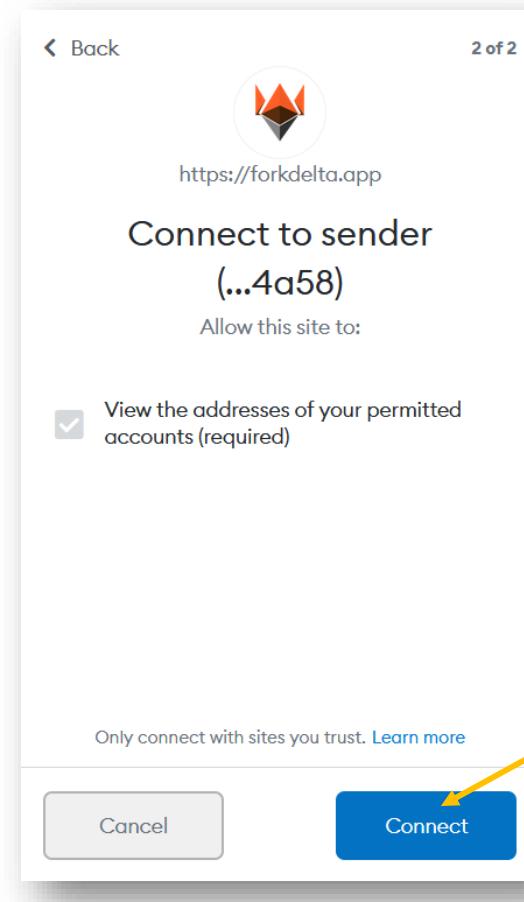
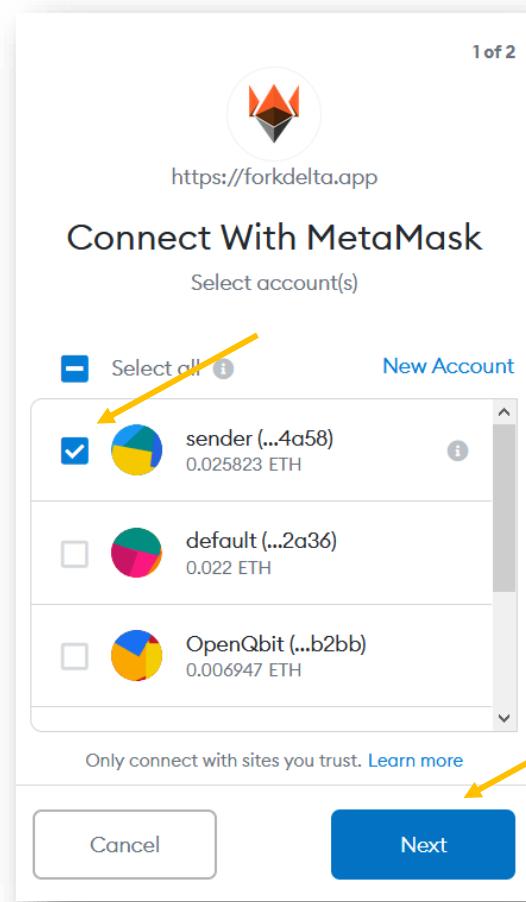
- Top navigation bar: Ethereum Mainnet dropdown, network switcher icon.
- Section title: **Add Tokens**.
- Input fields:
  - Search (disabled)
  - Custom Token** (selected tab)
  - Token Contract Address: `0x54093F2C720b18Fd795645b5101A3...`
  - Token Symbol: `MOGY`
  - Decimals of Precision: `18` (with a dropdown arrow icon)
- Buttons at the bottom: **Cancel** and **Next**.

**Screenshot 2: Token Overview Screen**

- Top navigation bar: Ethereum Mainnet dropdown, network switcher icon.
- Section title: **sender / MOGY**.
- Token icon: A circular icon divided into three segments (blue, orange, yellow).
- Token balance: **1000000000 MOGY**.
- Actions: **Send** and **Swap**.
- Text at the bottom: **You have no transactions**.

Assim que carregarmos o nosso novo símbolo para venda na Bolsa [www.forkdelta.app](https://forkdelta.app)

Quando estiver no site Exchange, ser-lhe-á pedido que se ligue ao site <https://forkdelta.app>. Clique no botão "Next" abaixo, depois em "Connect" e finalmente poderá verificar se está ligado ao site do forkdelta.app



Está na hora de lançar o novo símbolo na Bolsa <https://forkdelta.app>

Clique no menu superior "DAI" e vá para o fim do pergaminho e escolha a opção "Outro".

The screenshot shows the ForkDelta interface. At the top left, there's a dropdown menu with 'DAI' selected. A yellow arrow points from the text above to this 'DAI' button. Another yellow arrow points from the text below to the 'Other' option in the dropdown menu. The main area displays the Order Book, Price Chart, and Trades sections. The 'Order Book' shows several bids and asks for DAI/ETH, ETH, and DAI. The 'Price Chart' shows price levels from 0.00 to 1.00 over a 24-hour period. The 'Trades' section lists recent trades with columns for DAI/ETH, DAI, and ETH. Below these sections are the 'New Order' form and the 'My Transactions' history. On the right side, there's a 'Tweets' section with a tweet from @ForkDelta and a sidebar with links to Telegram, Reddit, and ForkDelta's Twitter account.

Depois registamos a nova ficha com os dados que já conhecemos.

The screenshot shows the ForkDelta application interface. A modal dialog box titled "Other token" is open in the center. It contains three input fields: "Address" with the value "0x54093F2C720b18Fd795645b5101A37EB49d1A94a", "Name" with the value "MOGY", and "Decimals" with the value "18". There are "Cancel" and "Go" buttons at the bottom right of the dialog. In the background, the main trading interface is visible, showing a "Balance" section with DAI and ETH amounts, a "Volume" section with various tokens like ASTRO, REQ, SXDT, and SNOV, and a "Trades" section listing recent trades for DAI/ETH. A yellow arrow points from the text "Depois registamos a nova ficha com os dados que já conhecemos." to the "Name" field in the dialog.

Neste momento o novo símbolo aparecerá na parte superior esquerda da Bolsa, apenas o carregámos para a Bolsa, precisamos de o publicar para que todos possam comprá-lo e vê-lo. Agora precisamos de depositar algum Éter (0,015) é suficiente da nossa conta para a Bolsa.

The image consists of two side-by-side screenshots of the ForkDelta website, both titled "ForkDelta MOGY".

**Left Screenshot:** Shows the "Balance" section. Under the "Token" tab, "MOGY" is selected. The "Wallet" column shows "1000000000.000" and the "ForkDelta" column shows "0.000". Below this, there are two input fields: "Amount" for MOGY (containing "0.000") and "Amount" for ETH (containing "0.026"). A blue "Deposit" button is located next to each. A yellow arrow points from the "Amount" field for MOGY to its value in the "Wallet" column. Another yellow arrow points from the "Amount" field for ETH to its value in the "ForkDelta" column.

**Right Screenshot:** Shows the same "Balance" section after a deposit. The "Wallet" column for MOGY remains at "1000000000.000" but the "ForkDelta" column now shows "0.000". The "Amount" field for MOGY now contains "0.015". The "Amount" field for ETH still contains "0.026". The blue "Deposit" button is visible below the ETH field. A yellow arrow points from the "Amount" field for MOGY to its value in the "ForkDelta" column. Another yellow arrow points from the "Amount" field for ETH to its value in the "ForkDelta" column. A tooltip appears over the "Deposit" button for ETH, stating: "Use this to deposit from your personal Ethereum wallet ("Wallet" column) to the current smart contract ("ForkDelta" column)."

Uma vez que fizemos o depósito, podemos depositar tantas fichas quantas quisermos na Bolsa [www.forkdelta.app](http://www.forkdelta.app)

Para depositar uma quantidade definida de fichas, precisamos de criar uma ordem de compra, isto é feito no fundo onde diz "Nova Ordem" e clicamos na opção "Vender". Depois introduzimos a quantidade de fichas que queremos disponibilizar a qualquer comprador no mundo, o preço em Éter que queremos vender cada uma (preço unitário) e o parâmetro "Expira" é a quantidade de tempo que queremos ter estas fichas à venda:

Tempo de expiração = 14 segundos X quantidade introduzida.

Exemplo: 14 segundos X 10000 = 140.000 segundos = 1,62 dias

The screenshot shows the ForkDelta app interface for trading the MOGY token. On the left, there's a 'Balance' section with tabs for Deposit, Withdraw, and Transfer, and a 'Volume' section with a search bar and a list of tokens like ASTRO, REQ, SXDT, SNOV, and POE. On the right, the 'Order Book' shows a single entry: 'There are no orders here.' In the center, the 'New Order' form is displayed with the following fields highlighted by yellow arrows:

- Sell**: The 'Sell' tab is selected.
- MOGY**: The amount input field contains '10000'.
- MOGY/ETH**: The price input field contains '0.05'.
- ETH**: The amount input field contains '500.000'.
- Expires**: The time input field contains '10000'.
- Sell**: The large orange 'Sell' button at the bottom right.

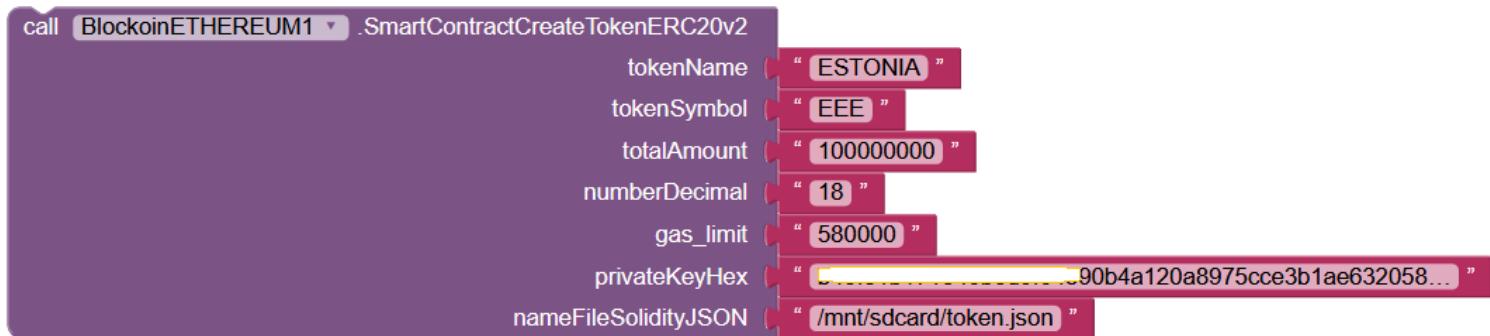
Aí, as suas novas fichas estão à venda, qualquer pessoa pode entrar e comprá-las. Por vezes não reconhece a nova ficha, pelo que terão de ser vendidos a partir da aplicação Metamask.

Exemplo de consulta no site [www.etherscan.io](https://www.etherscan.io) novo símbolo criado.

The screenshot shows the Etherscan interface for a transaction. The transaction hash is 0xd99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882. The status is Success. The block number is 11240201 with 224 block confirmations. The timestamp is 47 mins ago (Nov-12-2020 02:49:56 AM +UTC) | Confirmed within 30 secs. The transaction was from address 0x4b7355fd05be6dac458b004f54e12d6527a54a58. The recipient is a newly created contract at address [Contract 0x54093f2c720b18fd795645b5101a37eb49d1a94a Created]. The value sent was 0 Ether (\$0.00). The transaction fee was 0.011014691 Ether (\$5.06). The gas price was 0.000000041 Ether (41 Gwei). The gas limit was 500,000. The gas used by the transaction was 268,651 (53.73%). A yellow arrow points from the 'Created' status in the To field to a callout box containing the text 'Endereço do contrato ficha recém-criada.' Another yellow arrow points from the transaction fee to a callout box containing the text 'Custo da rede Ethereum apenas.' and 'Não inclui o custo da operação + 15'.

Overview	Status	Comments
② Transaction Hash:	0xd99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882	
② Status:	Success	
② Block:	11240201	224 Block Confirmations
② Timestamp:	47 mins ago (Nov-12-2020 02:49:56 AM +UTC)	Confirmed within 30 secs
② From:	0x4b7355fd05be6dac458b004f54e12d6527a54a58	
② To:	[Contract 0x54093f2c720b18fd795645b5101a37eb49d1a94a Created]	
② Value:	0 Ether (\$0.00)	
② Transaction Fee:	0.011014691 Ether (\$5.06)	
② Gas Price:	0.000000041 Ether (41 Gwei)	
② Gas Limit:	500,000	
② Gas Used by Transaction:	268,651 (53.73%)	

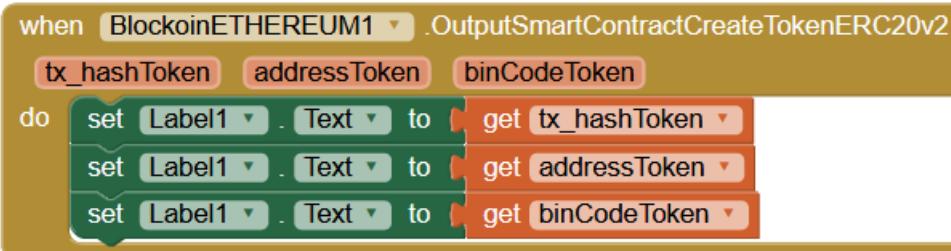
Bloco para compilar, criar e publicar o Token ERC20 - (**SmartContractCreateTokenERC20v2**)



Parâmetros de entrada: `tokenName>String</code>, tokenSymbol>String</code>, totalAmount>String</code>, numberDecimal <String>, gas_limit <Integer>, privateKeyHex <Integer>, nameFileSolidityJSON <String>.`

Parâmetros de saída: Evento (**OutputSmartContractCreateTokenERC20v2**).

Saídas: `tx_hashToken<String>`, `addressToken<String>`, `binCodeToken<String>`.



Descrição: Contrato inteligente "Token ERC20" - activo publicado na rede Ethereum. A versão 2 (v2) pode ser optimizada no valor do Limite de Gás porque dependendo do contrato inteligente, a rapidez de publicação na rede etérea. Recomenda-se que o valor mínimo seja de 350.000 WEI para a publicação sem falhas ou atrasos.

A diferença entre as funções de criação do TokenERC20v1 e de criação do TokenERC20v2 é que no caso da versão 1 o parâmetro do GasLimit já está pré-configurado e na versão 2 pode ser configurado de acordo com as necessidades do utilizador ou desenvolvedor.

Bloco para chamar ou executar a ficha ERC20 - (**SmartContractExecution**)



Parâmetros de entrada: addressSmartContract<String>, nameFileSolidityJSON<String>, funtionExecutionSmartContract<String>.

Parâmetros de saída: Execução da função especificada no Contrato Inteligente referenciado.

Saídas: **Execução de contrato inteligente.**

**NOTA:** Para ser executado, deve ser criado um ficheiro com o formato JSON que contenha os parâmetros da chave primária do endereço que pretende executar o contrato Smart, é necessário introduzir o limite de gás (WEI).

Exemplo de ficheiro JSON para executar as funções do contrato Smart compilado, no exemplo da função de compilação acima mencionada. O nome do ficheiro pode ser arbitrário.

Ficheiro: call.json

```
{
  "privado": "3ca40...",
  "gas_limit": 20000
}
```

Exemplo do resultado da execução da função "cumprimentar" do contrato Smart:

```
{
  "gas_limit": 20.000,
  "address": "0eb688e79698d645df015cf2e9db5a6fe16357f1",
  "resultados": [
    "Olá Coinsolidation Test"
  ]
}
```

ERC20 Token Property Display Block - (**SmartContractGetCreationTx**)

call BlockoinETHEREUM1 .SmartContractGetCreationTx

txSmartContract

" d99ff4d53f9b1f0687289674633d2acecf219011d163cdc0... "

Parâmetros de entrada: **txSmartContract<String>**

Parâmetros de saída: Mostra propriedades do contrato Smart referenciado com (**Tx\_hash**).

Descrição: Mostra as principais características e o código ABI do contrato Smart referenciado.

Exemplo de propriedades do token ERC20, amostra de tokenNome "MOGY" previamente criado usando a função (**martContractCreateTokenERC20v1**)

```
{
  "block_hash": "cadb8914c43ed28fb370c9e1b6f5d8818ba31a1e944d1f7049441b982902dea8",
  "block_height": 11240201,
  "block_index": 170,
  "hash": "d99ff4d53f9b1f0687289674633d2acecf219011d163cdc08b45468f63a22882",
  "endereços": [
    "4b7355fd05be6dac458b004f54e12d6527a54a58"
  ],
  "total": 0,
  "taxas": 110146910000000,
  "tamanho": 958,
  "gas_limit": 500.000,
  "gas_utilizado": 268651,
  "gas_price": 41000000000,
  "contract_creation": é verdade,
  "relayed_by": "200.77.24.87",
  "confirmed": "2020-11-12T02:49:56Z",
  "received": "2020-11-12T02:50:13.185Z",
  "ver": 0,
  "double_spend": falso,
  "vin_sz": 1,
```

Bloco para exibir código de compilação de fichas ERC20 - (SmartContractGetcodeABI)

call BlockchainETHEREUM1 .SmartContractGetcodeABI

addressSmartContract

"0eb688e79698d645df015cf2e9db5a6fe16357f1"

Parâmetros de entrada: **addressSmartContract<String>**

Parâmetros de saída: Mostra o código de contrato inteligente referenciado.

Descrição: Mostra o código ABI do contrato Smart referenciado.

Exemplo de código ABI de um contrato genérico Smart:

```
{
  "solidez": "contrato mortal" {\n    /* Definir proprietário variável do\n    tipo endereço*/\n    endereço proprietário;\n    /* esta função é\n    executada na inicialização e define o proprietário do contrato */\n    função mortal() { owner = msg.sender; }/* Função para recuperar os\n    fundos do contrato */\n    função kill() { if (msg.sender === owner)\n        suicide(owner); }\n    contract greeter is mortal {\n        /* define a\n        saudação variável do tipo string */\n        string greeting;\n        este é\n        executado quando o contrato é executado */\n        função cumprimentar(\n            string _greeting) público {\n                saudação = _greeting; }\n        /* função principal */\n        função cumprimentar() retornos constantes\n        (string) {\n            saudação de\n            retorno; }\n    },\n    \"caixote do lixo\":\n    \"606060405260405161023e38038061023e8339810160405280510160008054600160a060\n    020a031916331790558060016000509080519060200190828054600181600116156101000\n    203166002900490600052602060002090601f016020900481019282601f10609f57805160\n    ff19168380011785555b50608e9291505b8082111560cc57600081558301607d565b50505\n    061016e806100d06000396000f35b828001600101855582156076579182015b8281111560\n    7657825182600050559160200191906001019060b0565b509056606060405260e060020a6\n    00035046341c0e1b58114610026578063cfaf321714610068575b005b6100246000543373\n    ffffffffffffff\n    f908116911614156101375760005473fff\n    f16ff5b6100c9600060609081526001805460\n    a06020601f600260001961010086881615020190941693909304928301819004028101604\n    0526080828152929190828280156101645780601f10610139576101008083540402835291\n    60200191610164565b6040518080602001828103825283818151815260200191508051906\n    0200190808383829060006004602084601f0104600f02600301f150905090810190601f16\n    80156101295780820380516001836020036101000a031916815260200191505b509250505\n    06\n    \"abi\":\n    [{}\"constant\":false,\\"inputs\":[],\"name\":\"kill\",\"outputs\":[],\"ty\n    pe\":\"function\"},{}\"constant\":true,\\"inputs\":[],\"name\":\"greet\",\"y\n    outputs\": [{\"name\":\"\",\"type\":\"string\"}],\"type\":\"function\"},{}\"y\n    inputs\": [{\"name\":\"_greeting\",\"type\":\"string\"}],\"type\":\"const\n    ructor\"}]\n    \"creation_tx_hash\":\n    \"61474003e56d67aba6bf148c5ec361e3a3c1ceea37fe3ace7d87759b399292f9\",\n    \"created\": \"2016-07-20T01:54:50Z\",\n    \"address\": \"0eb688e79698d645df015cf2e9db5a6fe16357f1\"}
```

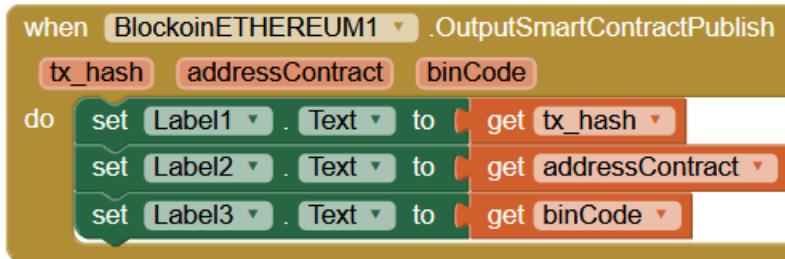
\*Antes de utilizar o seguinte Bloco (**SmartContractPublish**) deve utilizar o Bloco (**SmartContractCompilerSolidity**) para verificar se o contrato Smart está bem escrito.

Bloco para publicar a ficha ERC20 na rede Ethereum - (**SmartContractPublish**)

```
call BlockoinETHEREUM1 .SmartContractPublish
      nameFileSolidityJSON " /mnt/sdcard/PublishSmartContract.json "
```

Parâmetros de entrada: `nameFileSolidityJSON<String>`

Parâmetros de saída: Mostra as propriedades do contrato Smart referenciado. No caso (`OutputSmartContractPublish`).



Descrição: Publicar na rede Ethereum o contrato Smart referenciado no ficheiro JSON.

Exemplo de ficheiro: `PublishSmartContract.json`

```

{
  "solidez": "contrato mortal" /* Definir proprietário variável do tipo endereço*/
  /* endereço proprietário; esta função é executada na inicialização e define o proprietário do contrato */
  função mortal() { owner = msg.sender; } /* Função para recuperar os fundos do contrato */
  função kill() { if (msg.sender === owner) suicide(owner); }
  contract greeter is mortal /* define a saudação variável do tipo string */
  string greeting; este é executado quando o contrato é executado
  função cumprimentar(string _greeting) público {
    saudação = _greeting;
  }
  /* função principal */
  função cumprimentar() retornos constantes(string) {
    saudação de retorno;
  }
  "params": ["Olá Teste"],
  "publicar": ["cumprimentar"],
  "privado": "3ca40...",
  "gas_limit": 500000
}
  
```

Como mostrado no código acima, é o mesmo código JSON utilizado no exemplo da função de compilação ao mesmo código que os parâmetros foram adicionados no final do ficheiro JSON:

**Params:** Parâmetros implícitos do contrato inteligente.

**Publicar:** Nome de como o contrato Smart será publicado.

**Privado:** A chave primária da conta que irá executar o contrato Smart deve ter um saldo.

**Gas\_limit:** É o saldo na WEI que se pretende gastar para a publicação do contrato Smart.

Exemplo de saída quando executado (publicação de contrato Smart) o contrato Smart é inserido na rede etérea. Mostra a transacção realizada "**creation\_tx\_hash**" e o endereço atribuído do "**endereço**" do contrato Smart criado.

```
[
{
  "nome": "saudador",
  "solidez": "contrato mortal" /* Definir proprietário variável do tipo endereço*/
  /* endereço proprietário; esta função é executada na inicialização e define o proprietário do contrato */
  função mortal() { owner = msg.sender; }/* Função para recuperar os fundos do contrato */
  função kill() { if (msg.sender === owner) suicide(owner); }
  contract greeter is mortal /* define a saudação variável do tipo string */
  string greeting; este é executado quando o contrato é executado */
  função cumprimentar(string _greeting) público {
    saudação = _greeting;
  }
  /* função principal */
  função cumprimentar() retornos constantes (string) {
    saudação de retorno;
  },
  "caixote do lixo":
"606060405260405161023e38038061023e8339810160405280510160008054600160a060
020a031916331790558060016000509080519060200190828054600181600116156101000
203166002900490600052602060002090601f016020900481019282601f10609f57805160
ff19168380011785555b50608e9291505b8082111560cc57600081558301607d565b50505
061016e806100d06000396000f35b828001600101855582156076579182015b8281111560
7657825182600050559160200191906001019060b0565b509056606060405260e060020a6
00035046341c0e1b58114610026578063cfae321714610068575b005b6100246000543373
fffffffffffffffffffff908116911614156101375760005473fff
fffffffffffff908116911614156101375760005473fff
a06020601f600260001961010086881615020190941693909304928301819004028101604
052608028152929190828280156101645780601f10610139576101008083540402835291
60200191610164565b6040518080602001828103825283818151815260200191508051906
0200190808383829060006004602084601f0104600f02600301f150905090810190601f16
80156101295780820380516001836020036101000a031916815260200191505b509250505
06
  "abi": [
    {
      "constante": falso,
      "inputs": [],
      "nome": "matar",
      "outputs": [],
      "tipo": "função".
    },
    {
      "constante": é verdade,
      "inputs": [],
      "nome": "saudar",
      "outputs": [
        {
          "nome": "",
          "tipo": "corda"
        }
      ],
      "tipo": "função".
    },
    {
      "inputs": [

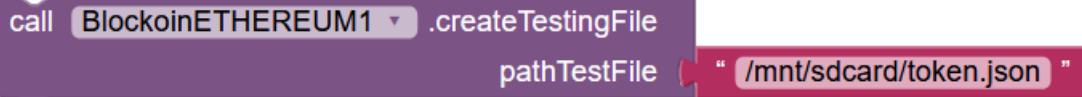
```

```

{
    "nome": "_ saudação",
    "tipo": "corda"
}
],
"tipo": "construtor"
}
],
"gas_limit": 500.000,
"creation_tx_hash":
"61474003e56d67aba6bf148c5ec361e3a3c1ceea37fe3ace7d87759b399292f9",
"address": "0eb688e79698d645df015cf2e9db5a6fe16357f1",
"params": [
    "Olá Teste"
]
},
{
    "nome": "mortal",
    "solidez": "contrato mortal" /* Definir proprietário variável do tipo endereço*/
        /* endereço proprietário*/
        /* esta função é executada na inicialização e define o proprietário do contrato */
        função mortal() { owner = msg.sender; }/* Função para recuperar os fundos do contrato */
        função kill() { if (msg.sender === owner) suicide(owner); }
    contract greeter is mortal /* define a saudação variável do tipo string */
        string greeting; /* este é executado quando o contrato é executado */
        função cumprimentar(string _greeting) público { saudação = _greeting; }
    /* função principal */
        função cumprimentar() retornos constantes (string) { saudação de retorno; },
        "caixote do lixo":
"606060405260008054600160a060020a03191633179055605c8060226000396000f36060
60405260e060020a600035046341c0e1b58114601a575b005b60186000543373ffffffffffff
ffffffffffffffffffff90811691161415605a5760005473ffffffffffff
ffffffffffffffffffff16ff5b56",
"abi": [
{
    "constante": falso,
    "inputs": [],
    "nome": "matar",
    "outputs": [],
    "tipo": "função".
},
{
    "inputs": [],
    "tipo": "construtor"
}
],
"gas_limit": 500.000,
"params": ["Olá Teste"]
}
]

```

Bloco de teste para **criar** ficheiro - (**createTestingFile**)



Parâmetros de entrada: **pathTestFile<String>**

Parâmetros de saída: Um ficheiro de teste é criado no caminho referenciado.

Descrição: Isto serve para verificar o caminho de criação do ficheiro temporário válido para garantir que está correcto quando se utiliza o Bloco (**SmartContractCreateTokenERC20v1**) ou o Bloco (**SmartContractCreateTokenERC20v2**).

Bloco para obter as taxas de Preço do Gás - (**eth\_RatesGasStationInfo**).



Parâmetros de entrada: **nameFileSolidityJSON<String>**

Parâmetros de saída: Mostra as propriedades do contrato Smart referenciado. No caso (**OutputEth\_GasStationInfo**) os valores entregues são dados em GWEI.

O Preço do Gás é o valor que será pago aos sistemas que executam as transacções na rede do Ethereum. Estes sistemas são normalmente conhecidos como "mineiros" e os valores do Preço do Gás são uma função da rapidez (tempo e prioridade) da transacção realizada na rede do Ethereum.

Os valores entregues são baseados nos seguintes tempos de execução. Estes tempos são aproximados e podem variar em função da forma como se está a proceder em relação às exigências (transacções) na rede Ethereum.

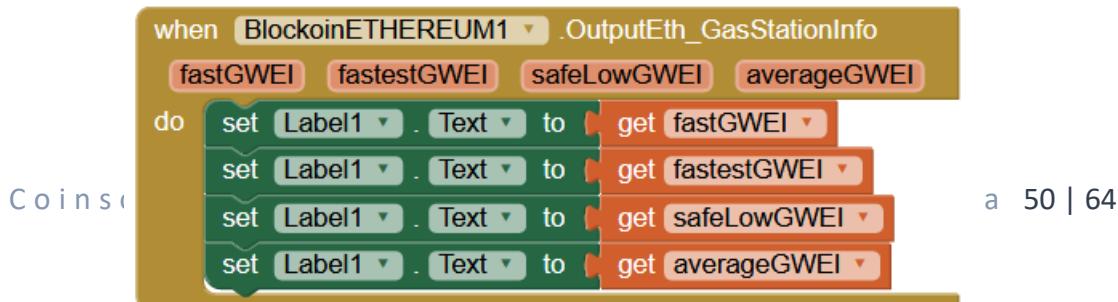
**Rápido** < 2 minutos.

**O mais rápido** < 30 segundos.

**SafeLow** < 30 minutos.

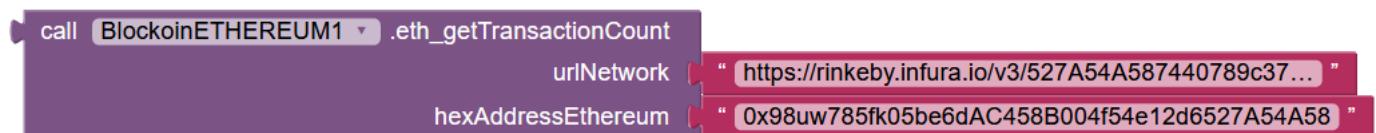
**Média** < 15 minutos.

As transacções feitas com a Troca Extensão Etérea (EEE) utilizam sempre o Preço do Gás = Média.



Descrição: Obtém o Preço do Gás actualizado no momento da consulta para criar uma nova transacção

Bloco para obter o número "nonce" - (`eth_getTransactionCount`).



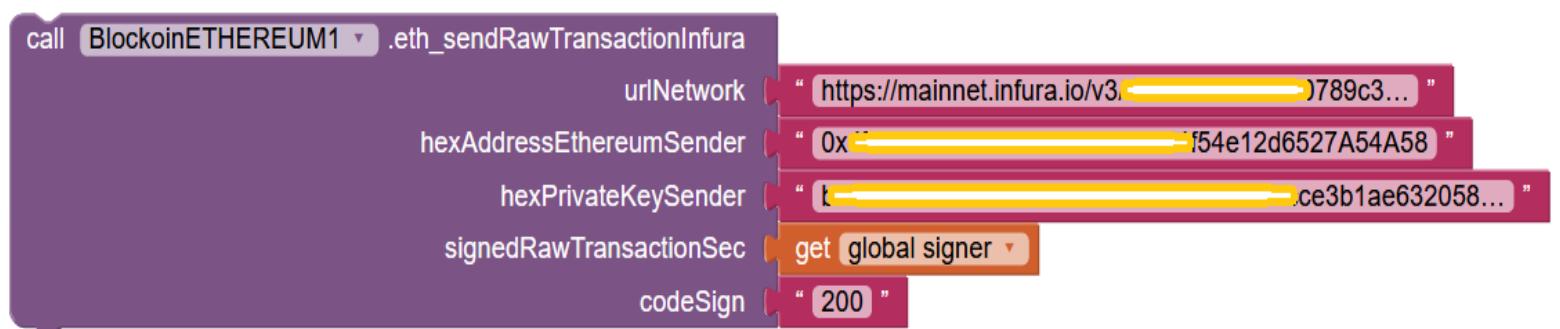
Parâmetros de entrada: `urlNetwork<String>`, `hexAddressEthereum<String>`.

Parâmetros de saída: Mostra em formato hexadecimal o número consecutivo "nonce" do endereço referenciado.

O número "nonce" é um número incremental que mantém um registo do número de transacções que foram feitas a partir de um endereço específico.

Descrição: Obtém o número "nonce" do endereço referenciado.

Bloco para enviar uma transacção assinada - (`eth_SendRawTransactionInfura`).

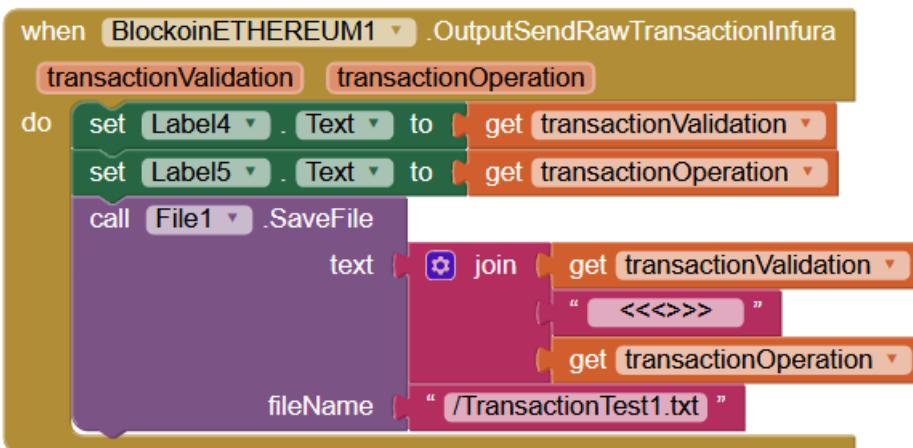


Dependência(ões) necessária(s): Bloco (`eth_getTransactionCount`), Bloco (`SignerGenericPushRawTransactionOffline`). Exemplo de revisão do bloco (`SignerGenericPushRawTransactionOffline`).

Parâmetros de entrada: `urlNetwork <String>`, `hexAddressEthereumSender <String>`, `hexPrivateKeySender <String>`, `SignedRawTrasactionSec <String>`, `codeSign <String>`.

Parâmetros de saída: Evento (**OutputSendRawTransactionInfura**)

Saídas: **transactionValidation<String>** , **transactionOperation<String>**.



Descrição: Fornece dois valores hexadecimais como resultado das transacções. O valor da transacçãoValidation é a transacção feita na rede Ethereum que contém o custo implícito do Ethereum. O valor da transacçãoOperation é a transacção efectuada na rede Coinsolidation.org com o custo de \$0,5 céntimos USD para cada transacção com base no valor do éter no momento da transacção. Este custo destina-se ao pagamento de serviços da rede Coinsolidation.org e é investido na manutenção, apoio e criação de extensões para o sector de crypto e activos em todo o mundo.

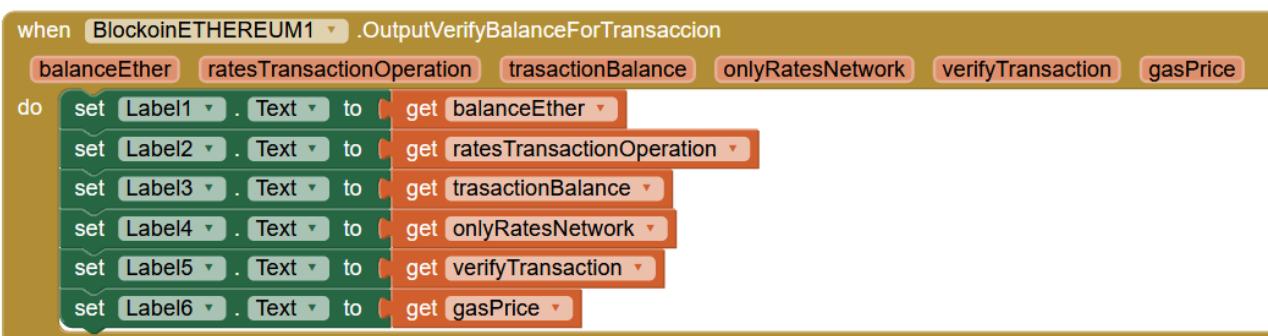
Bloco para calcular o custo de uma transacção padrão - (**eth\_VerifiBalanceForTransaction**)



Parâmetros de entrada: **enderecoEthereumSender<String>**, **valorEthertoSend<String>**.

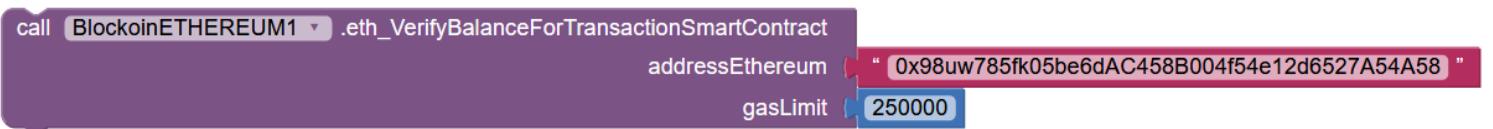
Parâmetros de saída: Evento (**OutputVerifyBalanceForTransaction**).

Saídas: **balanceEther<String>** , **ratesTransactionOperation<String>** ,  
**trasactionBalance<String>** , **OnlyRatesNetwork<String>** , **verifyTransaction<String>** ,  
**gasPrice<String>**.



Descrição: Fornece pormenores sobre o custo de uma transacção padrão com referência ao endereço de entrada. O parâmetro de saída "verifyTransaction" diz-nos se a transacção pode ser feita "True" ou se o endereço referenciado não tem saldo suficiente dar-nos-á um "False".

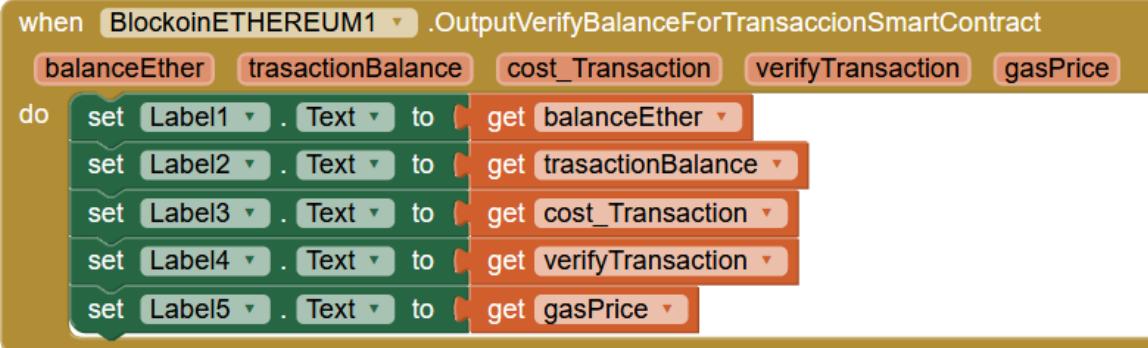
Bloco para calcular o custo de uma transacção padrão -  
(eth\_VerifyBalanceForTransaccionSmartContract)



Parâmetros de entrada: `endereçoEthereum<String>`, `gasLimit<String>`.

Parâmetros de saída: Evento (`OutputVerifyBalanceForTransaccionSmartContract`)

Saídas: `balanceEther<String>` , `trasactionBalance<String>` , `cost_Transaction<String>` , `verifyTransaction<String>` , `gasPrice<String>`.



Descrição: Fornece detalhes sobre qual seria o custo aproximado de uma transacção padrão para publicar um **contrato Smart**, referência ao endereço de entrada. O parâmetro de saída "verifyTransaction" diz-nos se a transacção pode ser feita "True" ou se o endereço referenciado não tem saldo suficiente dar-nos-á um "False".

**balanceEther**: O balanço do endereço referenciado é entregue em éteres.

**TransactionBalance**: Saldo após a realização da transacção.

**cost\_Transaction:** É o custo da transacção para publicar o contrato inteligente.

**verifyTransaction:** (balanceEther minus cost\_Transaction).

**GasPrice:** Valor actual do GasPrice utilizado pelos "mineiros", este pode variar a cada minuto.

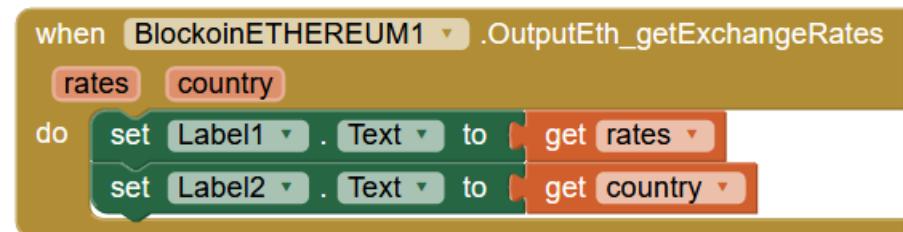
Bloco para obter o preço do Éter na moeda do país especificado - (**eth\_getExchangeRates**).



Parâmetros de entrada: **CountryRates<String>**. Verificar o parâmetro de saída "país" onde contém todos os tipos de moeda do país para escolher a desejada.

Parâmetros de saída: Evento (**OutputEth\_getExchangeRates**).

Saídas: **taxas<Corda>, países<Corda>**saída em formato JSON todas as taxas dos países do mundo.



Descrição: Fornece o preço actual de um Éter à taxa de câmbio da moeda do país de referência.

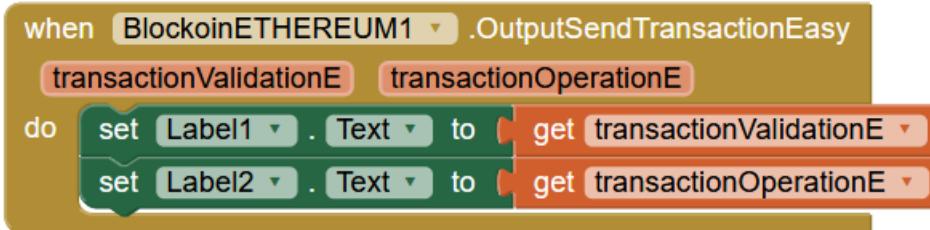
Bloco para realizar uma transacção padrão com os parâmetros mínimos de entrada - (**eth\_sendTransactionEasy**).



Parâmetros de entrada: **hexPrivateKeySender>String>, toAddress>String>, valueEther>String>**.

Parâmetros de saída: Evento (**OutputSendTransactionEasy**)

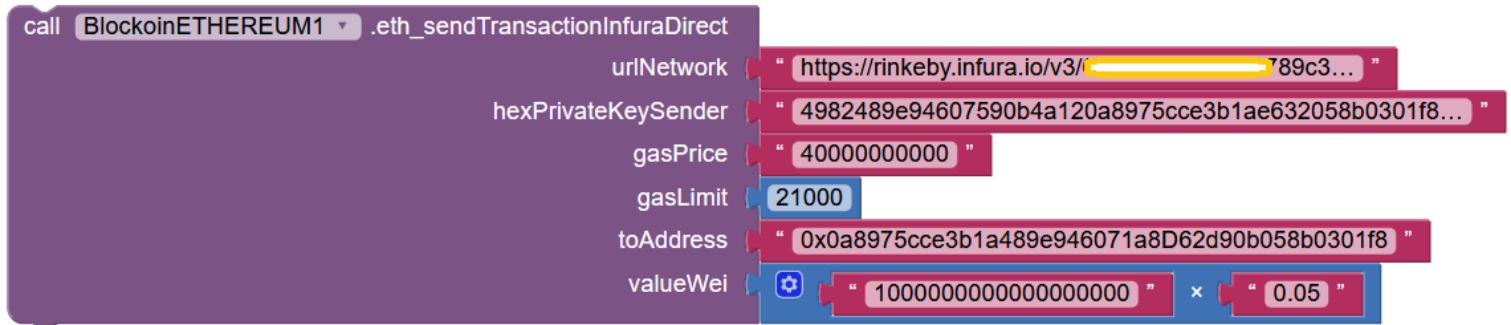
Saídas: **transactionValidation<String>**, **transactionOperation<String>**.



Descrição: Função para fazer uma transacção padrão na rede do Ethereum, esta função é de uso imediato, não precisa de ter uma conta no INFURA e só precisa de 3 parâmetros de entrada, só precisa de ter saldo suficiente para fazer a transacção desejada.

As transacções são colocadas na rede Ethereum directamente utilizando a biblioteca oficial Ethereum Web3j e a nossa rede Coinsolidation.org

Bloco para realizar uma transacção padrão com os parâmetros mínimos de entrada - (**eth\_sendTransactionInfuraDirect**).



Parâmetros de entrada: **urlNetwork<String>**, **hexPrivateKeySender>String>**, **gasPrice<String>**, **gasLimit<String>**, **toAddress<String>**, **valueWEI<String>**.

Parâmetros de saída: Evento (**OutputSendTransactionInfuraDirect**)

Saídas: **transactionValidation<String>**, **transactionOperation<String>**.



Descrição: Função para enviar uma transacção padrão que já contém a assinatura digital implícita, isto é útil para pessoas que já têm conhecimento prévio dos componentes de uma transacção e querem optimizar estes parâmetros de acordo com as suas necessidades.

## 11. Cálculo do custo de transacção padrão e transacção de contrato inteligente

Para o cálculo de uma transacção padrão, são necessários 3 parâmetros na rede Ethereum.

- 1.- Preço do gás.
- 2.- Limite de gás.
- 3.- Valor actual de um Éter (moeda digital Ethereum).

**Preço do gás:** Normalmente é dado em unidades de GWEI (GigaWEI). Isto corresponde a, se 1 éter tem 1.000.000.000.000.000 wei então 1 GWEI é igual a 1.000.000.000.000. Esta unidade serve de pagamento para os sistemas que executam todas as transacções da rede Ethereum e são chamados "mineiros", é distribuída por todo o mundo. O GasPrice não é um valor fixo e é variável e pode mudar de minuto para minuto ou segundo. Aqueles que definem o valor do GasPrice são os "mineiros" e depende de quanto saturada está a rede Ethereum.

**GasLimit:** Este valor é normalmente dado em unidades de WEI e numa transacção padrão um valor médio por defeito é de 21.000 WEI embora possa ser superior ou inferior dependendo do tipo de transacção que se pretende efectuar, numa transacção padrão utilizamos o valor de 40.000 WEI para garantir que não temos rejeição por termos sob Limite de Gás.

**Valor do Éter:** Este valor é também variável e deve-se a diferentes parâmetros de um mercado financeiro global, este valor pode ser obtido de entidades que sempre actualizaram o valor do Éter a nível mundial, chamadas trocas centralizadas e descentralizadas.

**NOTA IMPORTANTE:** Na Extensão Ethereum de Troca (EEE) utilizamos um Limite de Gás mais elevado (40.000), isto deve-se ao facto de no caso de não se atingir a quota mínima estabelecida pelos "mineiros" a TRANSACÇÃO NÃO SER EFECTUADA, no entanto a rede Ethereum, se cobrar a despesa que fez no cálculo da transacção sem a efectuar, Por esta razão, alguns utilizadores não explicam porque é que a sua transacção não é realizada, no entanto, ser-lhes-á cobrado um montante ou todo o GasLimit que foi oferecido quando o pedido de transacção foi lançado, por esta razão temos de ser sempre claros quais serão os valores do GasLimit e do Gas Price.

O GasPrice pode ser consultado com a função **eth\_GetRatesGasStation** e o GasLimit para transacções padrão deve ser superior a 21.000 WEI e as transacções para publicar e/ou executar o contrato Smart devem ser de pelo menos 500.000 WEI.

Custo de transacção padrão.

É definido pela seguinte fórmula:

Custo = (GasLimit x (GasPrice / (1.000.000.000.000.000.000)) x Valor do Éter.

Exemplo:

Assumir os seguintes valores:

**GasLimit** = 40,000, **GasPrice** = 45 GWEI, **Ether Value** = \$406.

Custo = (40.000 x (45.000.000.000.000 / (1.000.000.000.000.000.000)) x 406

**Custo da transacção padrão** = 0,0018 ether x 406 USD = \$0,73 USD

No caso de uma transacção de contrato Smart, é necessário saber que tipo de contrato Smart será publicado, uma vez que o custo é directamente proporcional à carga de trabalho que os "mineiros" terão de realizar para processar o contrato Smart, por outras palavras, a quantidade de processamento nos sistemas informáticos que os "mineiros" manuseiam é mais simples.

No caso do contrato Smart, um valor por defeito seria iniciar o GasLimit com um valor de 500.000 WEI.

Um ponto importante a ter em conta é que quando se propõe um GasLimit, os "mineiros" não tomarão necessariamente todo o montante proposto, ou seja, quando se envia uma transacção, os "mineiros" calculam o esforço de cálculo e retiram o que será retirado do GasLimit, que pode ser inferior ou igual ao GasLimit padrão de 21.000 WEI oferecido em alguns casos.

Todas as transacções poderão ser consultadas no site [www.etherscan.io](http://www.etherscan.io) onde podemos consultar os detalhes de cada transacção.

**Exemplo de uma transacção padrão**, em que a transacção foi enviada com um Limite de Gás de 40.000 WEI, no entanto os "mineiros" ao calcular o quanto o custo de processamento seria tomado apenas 52,5%, ou seja, o valor por defeito que é de 21.000 WEIs.

The screenshot shows a detailed view of a transaction on Etherscan. The transaction hash is 0x7dae251f21c616fb04a94112633c97e04d11c91f4d06dd9b85ed11112f02703a. It was successful and has 2 block confirmations. The transaction was sent from 0x4b7355fd05be6dac458b004f54e12d6527a54a58 to 0x5d2acdb34c279aa6d1e94a77f7b18ab938fb2bb at Nov-11-2020 06:17:24 AM +UTC. The value sent was 0.001084598698481562 Ether (\$0.50), and the transaction fee was 0.000672 Ether (\$0.31). The gas price was 0.000000032 Ether (32 Gwei) and the gas limit was 40,000. The actual gas used was 21,000 (52.5%).

Parameter	Value	Notes
Transaction Hash	0x7dae251f21c616fb04a94112633c97e04d11c91f4d06dd9b85ed11112f02703a	Transaction Operation or Transaction Validation
Status	Success	
Block	11234630	2 Block Confirmations
Timestamp	52 secs ago (Nov-11-2020 06:17:24 AM +UTC)	Confirmed within 38 secs
From	0x4b7355fd05be6dac458b004f54e12d6527a54a58	
To	0x5d2acdb34c279aa6d1e94a77f7b18ab938fb2bb	Custo da transacção em Éter: $21.000 \times 0,000000032 = 0,000672$ Éter (\$0,31 USD)
Value	0.001084598698481562 Ether (\$0.50)	
Transaction Fee	0.000672 Ether (\$0.31)	
Gas Price	0.000000032 Ether (32 Gwei)	Limite de gás proposto: 40.000
Gas Limit	40,000	
Gas Used by Transaction	21,000 (52.5%)	Limite real de gás utilizado na transacção: 21.000 WEI

Voltando à transacção do contrato Smart e tomando o 500.000 WEI GasLimit, obtemos o seguinte custo se o utilizarmos todo.

Custo de transacção de contrato inteligente.

É definido pela seguinte fórmula:

Custo = (GasLimit x (GasPrice / (1.000.000.000.000.000.000))) x Valor do Éter.

Exemplo:

Assumir os seguintes valores:

**GasLimit** = 500,000, **GasPrice** = 45 GWEI, **Ether Value** = \$406.

Custo = (500.000 x (45.000.000.000.000 / (1.000.000.000.000.000))) x 406

**Custo da transacção publicar contrato Smart** = 0,0225 ether x 406 USD = \$9.135 USD

Todas as transacções podem ser consultadas no sítio [www.etherscan.io](http://www.etherscan.io)

**NOTA:** Para obter o custo total da transacção deve adicioná-los:

**Custo total da transacção** = *Custo da rede Ethereum + taxa Coinsolidation.org*

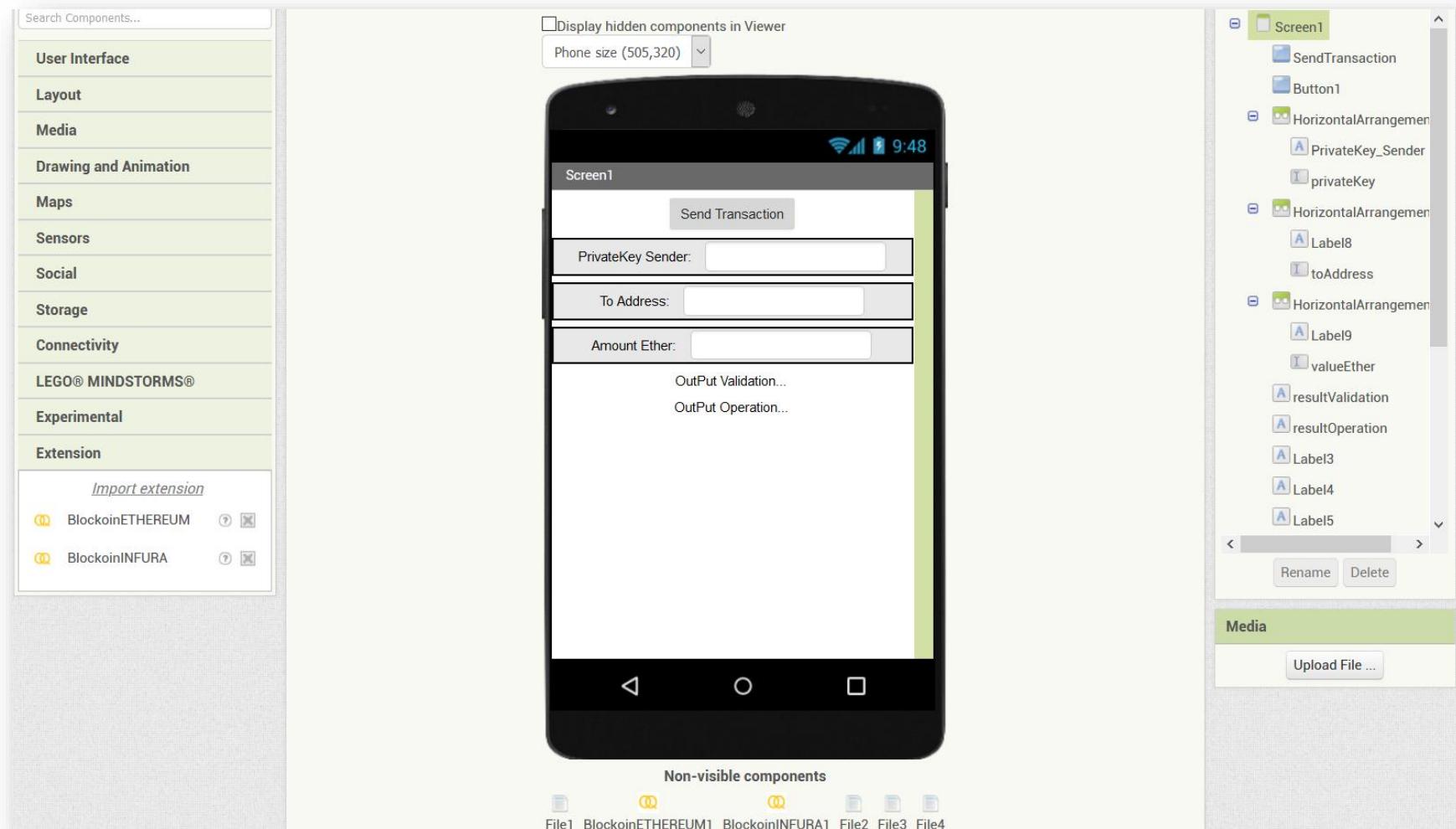
## 12. Coinsolidtion.org Taxes

Transacção padrão: \$ 0,5 cêntimos USD + custo da rede Ethereum.

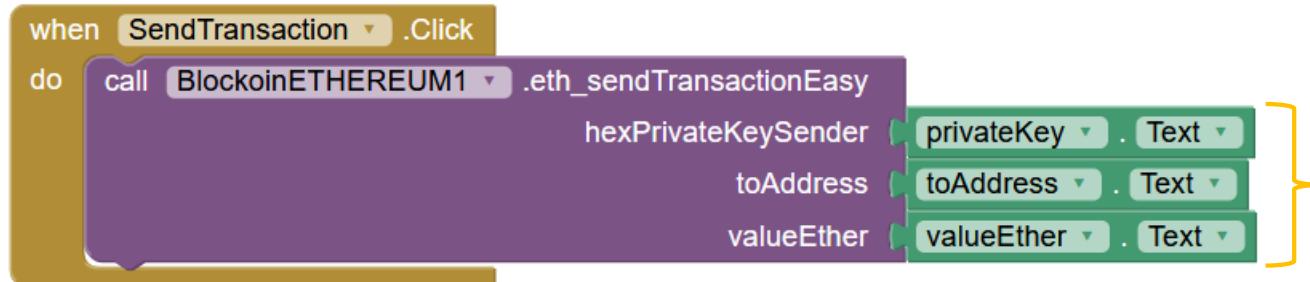
Transacção publicar e/ou executar um contrato Smart: \$15 USD + custo da rede Ethereum.

### 13. Crie a sua aplicação Android (Exchange) em 15 minutos.

Desenho em App Inventor (Screen). - 5 minutos.



Blocos de funções (`eth_SendTransactionEasy`) e evento (`OutPutSendTransactionEasy`) - 5 minutos

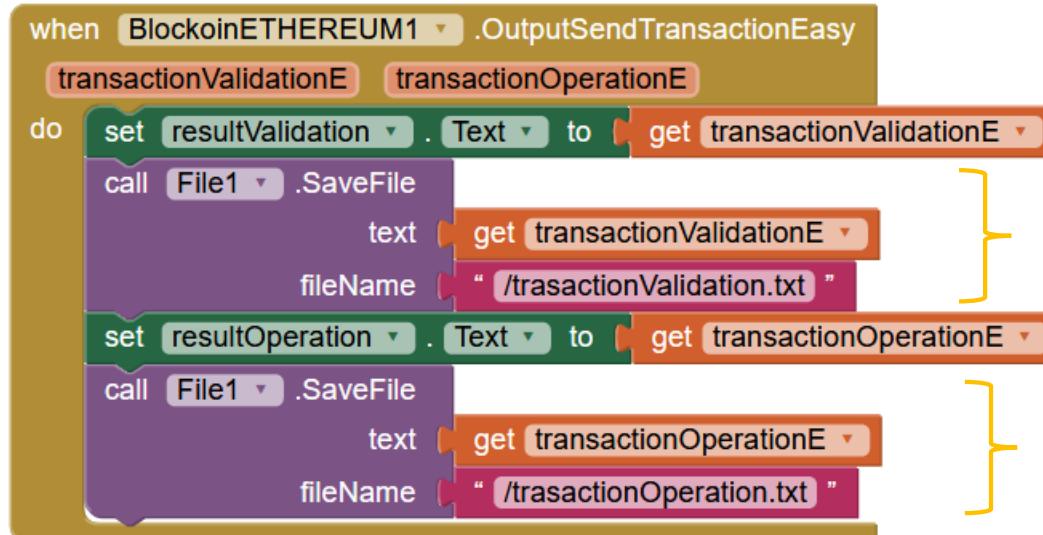


Dados de entrada:

**PrivateKey:** Chave primária para o endereço do remetente.

**Endereço:** Endereço hexadecimal do destinatário.

**valorEther:** Dê a quantidade de Éter que será enviada.



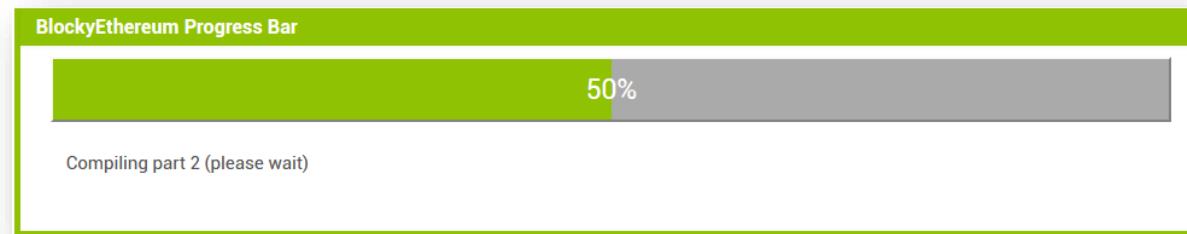
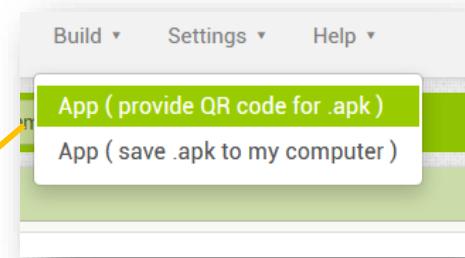
Guardar os resultados em ficheiros de texto:

Função File1: File **trasactionValidation.txt**

Guardar os resultados em ficheiros de texto:

Função File2: File **trasactionValidation.txt**

Compilamos, geramos o ficheiro APK para o instalar no dispositivo Android. - 5 minutos



**NOTA:** Quando a transacção for executada, levará aproximadamente 6 a 8 segundos para libertar o botão "Enviar Transacção". Devido ao tempo de ligação com a rede Ethereum.

## 14. Token CoinSolidation.

Token COINSolidation é um projecto com três orientações principais.

Imagine ter sua própria criptomoeda Token para expandir seus negócios. Usando COINSolidation, você pode torná-lo fácil e simples.

A primeira é criar a primeira rede de extensões baseada na metodologia de programação visual Blockly, na qual pela sua utilização fácil e intuitiva pode ser utilizada por qualquer pessoa sem conhecimento prévio de programação, as extensões que podem ser consultadas no nosso Roadmap (Livro Branco) são dirigidas a dois sectores fundamentais da economia mundial, o sector das moedas criptográficas e/ou fichas e o sector das moedas (fiat) ou utilização comum a nível mundial, como o dólar americano, o euro da UE, a libra ou qualquer outra moeda em uso.

A segunda directriz no projecto COINSolidation é a criação de um novo algoritmo para consolidar endereços de criptomontagens actuais e futuras. Estamos a desenvolver um novo modelo de algoritmo para criar um endereço que consolida diferentes endereços de diferentes cadeias de bloqueio num endereço universal ver o nosso (Livro Branco) em [www.CoinSolidation.org](http://www.CoinSolidation.org) ou <https://github.com/coinsolidation/whitepaper>

A terceira orientação é aplicar a tecnologia da Computação Quântica na segurança do ambiente COINSolidation, isto será aplicado com as extensões já desenvolvidas dos algoritmos de segurança QRNG (Quantum Random Number Generator) e PQC (Post-Quantum Computing). Estas podem ser encontradas no repositório oficial de extensões no site do Github, <https://github.com/coinsolidation>

## Características gerais da Consolidação CryptoToken COINsolidation

Nome: COINsolidation

Símbolo: CUAG

País de lançamento: Estónia

Página oficial: [www.Coinsolidation.org](http://www.Coinsolidation.org)

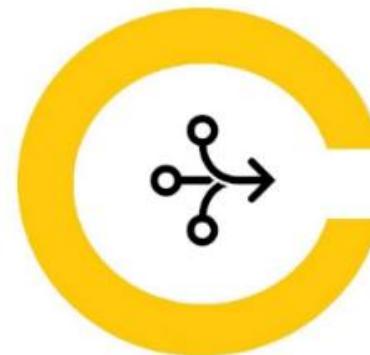
Empresa: Coinsolidation International.

Data de lançamento: 04/30/2021

Criado por: Guillermo Vidal.

Algoritmo de consenso: Prova de Quantum.

Algoritmo de endereço: Endereço Universal Consolidado.



## 15. Licenciamento e utilização de software.

Licenciamento, termos e condições de utilização ver [www.coinsolidation.org](http://www.coinsolidation.org) ou escrever para [info@coinsolidation.org](mailto:info@coinsolidation.org)