

COM241 - Administração e Gerenciamento de Redes de Computadores

Tutorial Trabalho 1

Índice

1. O que é Blockchain?
 2. O que é um Contrato Inteligente?
 - 2.1. Como funciona um Contrato Inteligente?
 3. Ethereum
 - 3.1. Solidity
 4. Proposta do Trabalho 1
 - 4.1 Diagrama da aplicação
 - 4.2 Algumas especificidades do mecanismo Ethereum
 5. Preparando o ambiente
 6. Utilizando a aplicação
-

1. O que é Blockchain?

A Wikipédia oferece uma definição simples de Blockchain

A **blockchain** (também conhecido como “o protocolo da confiança”) é uma tecnologia de registro distribuído que visa a descentralização como medida de segurança. São bases de registros e dados distribuídos e compartilhados que têm a função de criar um índice global para todas as transações que ocorrem em um determinado mercado. Funciona como um livro-razão, só que de forma pública, compartilhada e universal, que cria consenso e confiança na comunicação direta entre duas partes, ou seja, sem o intermédio de terceiros.

Disponível em: [Blockchain](#)

2. O que é um Contrato Inteligente?

De acordo com o site oficial da IBM, um contrato inteligente é definido como

Contratos inteligentes são simplesmente programas armazenados em uma blockchain que são executados quando condições pré-determinadas são atendidas. Usualmente, eles são usados para automatizar a execução de um **acordo** de forma que todos os participantes podem ter imediatamente certeza do resultado, sem intermediação de terceiros ou perda de tempo. Eles também podem automatizar fluxo de trabalho, disparando a próxima ação quando condições são atendidas.

2.1 Como funciona um Contrato Inteligente?

Ainda no site da IBM, extraímos informações sobre o funcionamento de um contrato inteligente

Contratos inteligentes funcionam seguindo simples "if/when... then..." que são escritos em códigos colocados na blockchain. Uma rede de computadores executam as ações quando as condições pré-determinadas são atingidas e verificadas. Essas ações podem incluir: a liberação de fundos para o agente participante apropriado, o registro de um veículo, envio de notificações ou emissão de ingressos. Então a blockchain é atualizada quando a transação é completa. Isso significa que a transação não pode ser alterada, e apenas os agentes que têm permissão podem ver os resultados.

Dentro de um contrato inteligente podem haver quantas estipulações forem necessárias para satisfazer os agentes que a tarefa será completada de forma satisfatória. Para estabelecer os termos, os participantes devem determinar como a transação e seus dados serão representados na blockchain, concordar sobre as regras de "if/when... then..." que governam essas transações, explorar as exceções possíveis e definir um escopo para resolução de disputas.

Então, o contrato inteligente pode ser programado por um desenvolvedor - embora organizações que usam blockchain para negócios também podem providenciar templates, interfaces e outras ferramentas online para simplificar a estruturação de contratos inteligentes.

Os dois trechos foram traduzidos diretamente da página: [What are smart contracts on blockchain?](#)

3. Ethereum

A definição da Wikipédia é o suficiente para nosso propósito, ela define da seguinte forma

Ethereum é uma plataforma descentralizada capaz de executar **contratos inteligentes** e aplicações descentralizadas usando a tecnologia **blockchain**: São aplicações que funcionam exatamente como programadas sem qualquer possibilidade de censura, fraude ou interferência de terceiros, isso porque o contrato é imutável. Ele possui uma máquina virtual descentralizada Turing complete, a Ethereum Virtual Machine (EVM), que pode executar scripts usando uma rede internacional de nós públicos.

O *Ether* (ETH) é uma moeda digital utilizada dentro da plataforma do Ethereum para rodar os contratos inteligentes, serviços computacionais dentro da rede e para pagar taxas aos mineradores.

Disponível em: [Ethereum](#)

3.1 Solidity

Solidity é uma linguagem de alto nível orientada a objetos que visa a implementação de contratos inteligentes. Contratos inteligentes são programas que governam o comportamento de contas dentro da rede Ethereum. Solidity é uma linguagem de curly-bracket (chaves). É influenciada por C++, Python e JavaScript, e desenhada para ter como alvo o Ethereum Virtual Machine (EVM).

Traduzido diretamente da página oficial do projeto: [Solidity](#)

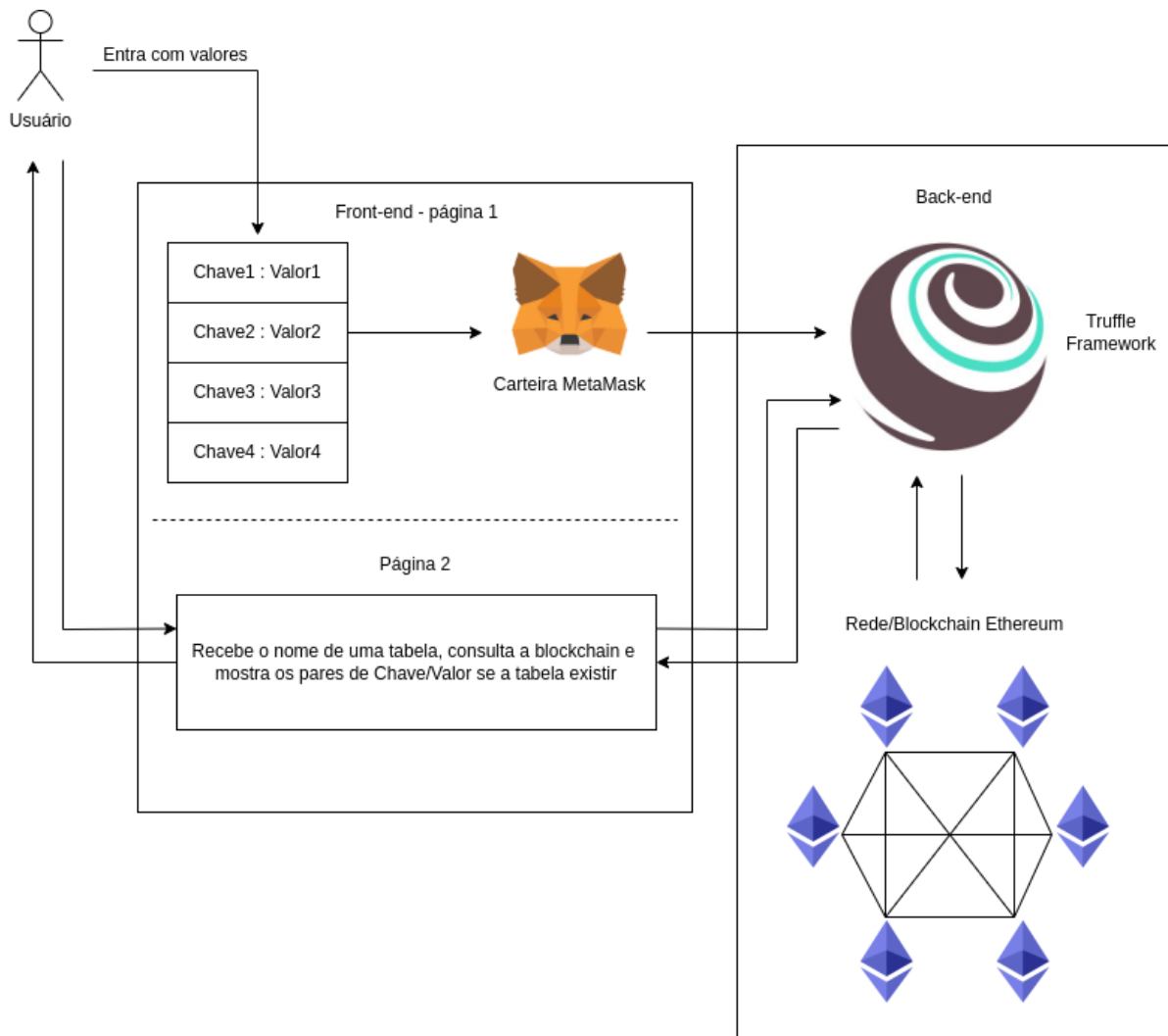
4. Proposta do Trabalho 1

A ideia do trabalho 1 consiste na criação de uma interface web onde um usuário pode entrar com valores em uma tabela que se assemelha a um dicionário, isto é, pares de chave e valor. A tabela pode ter um tamanho indeterminado e podem haver edições e remoções dos pares. A nossa aplicação também possibilita o usuário resgatar esses valores, que quando resgatados são mostrados na interface.

Uma vez que o usuário escolheu os valores desejados, o sistema irá registrar esses valores na blockchain Ethereum através de um **contrato inteligente**. Esse registro envolve um pagamento de "gas fee" utilizando a criptomoeda ETH, portanto o usuário também interage com a carteira MetaMask, que realiza o débito necessário para efetuar a operação.

A carteira MetaMask interage com funções específicas do kit de ferramenta Truffle para transmitir as informações de taxas e contas, então o Truffle realiza a intermediação entre o front-end e a blockchain. O Truffle também disponibiliza funções para interagir com o contrato inteligente utilizando linguagens de programação como JavaScript. Dessa forma, é possível passar argumentos para as funções dos contratos inteligentes e resgatar valores da blockchain com uma sintaxe familiar.

4.1 Diagrama da aplicação



4.2 Algumas especificidades do mecanismo Ethereum

Toda modificação na blockchain Ethereum envolve uma cobrança de "gas fee" (por modificação entende-se, por exemplo, alterar o valor de uma variável em um contrato inteligente ou a implementação de um contrato inteligente na blockchain) essa taxa é direcionada para os mineradores de ether que mantêm a rede funcionando.

Tudo que é registrado na blockchain é imutável, portanto o usuário da aplicação deve realizar todas as edições e remoções nas tabelas antes da aplicação registrá-las na blockchain, por esta razão implementamos essas duas funcionalidades no front-end e elas não têm relação com a blockchain.

O resgate de informações da blockchain não envolve cobrança de "gas fee", porque nada na blockchain é alterado durante uma consulta.

5. Preparando o ambiente

Nessa sessão iremos apresentar as ferramentas necessárias para o desenvolvimento e utilização da aplicação, assim como as instalações no ambiente Ubuntu 20.

Clone o repositório da aplicação no link <https://github.com/andrewunifei/blockchain-ethereum>

1) NodeJS

Esse software permite rodar programas escritos em JavaScript no back-end (normalmente apenas o navegador interpreta JavaScript).

Instalação no Ubuntu e derivados:

```
sudo apt install nodejs
```

Vamos usar a versão v12.18.2

Para obter a versão instalada no computador:

```
nodejs --version
```

2) npm

Esse software é usado para instalar bibliotecas para o NodeJS

Instalação no Ubuntu e derivados:

```
sudo apt install npm
```

Vamos usar a versão 6.14.8. Para obter a versão:

```
npm --version
```

3) JQuery e Bootstrap

Na construção do front-end utilizamos Bootstrap, as alterações dinâmicas do front-end e parte da comunicação entre o front-end e o back-end foram feitas com JQuery.

Para utilizar esses serviços adicione na tag `<head>` do index.html os seguintes script:

```
<script type="text/javascript"
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js">
</script>
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

4) Ganache

Ganache simula a blockchain Ethereum no nosso computador. Vamos implementar os nossos contratos inteligentes nessa blockchain.

Instalação no Linux: baixar o arquivo .AppImage no [site do ganache](#).

No meu caso o arquivo veio sem possibilidade de execução. Para verificar isso no terminal `ls -l` na pasta do arquivo.:

```
-rw-rw-r-- 1 andrew andrew 153130048 out  9 14:28  ganache-2.5.4-linux-x86_64.AppImage
```

A permissão é `-rw-rw-r--`, está faltando a possibilidade de execução. Para adicionar essa funcionalidade:

```
sudo chmod +x ganache-2.5.4-linux-x86_64.AppImage
```

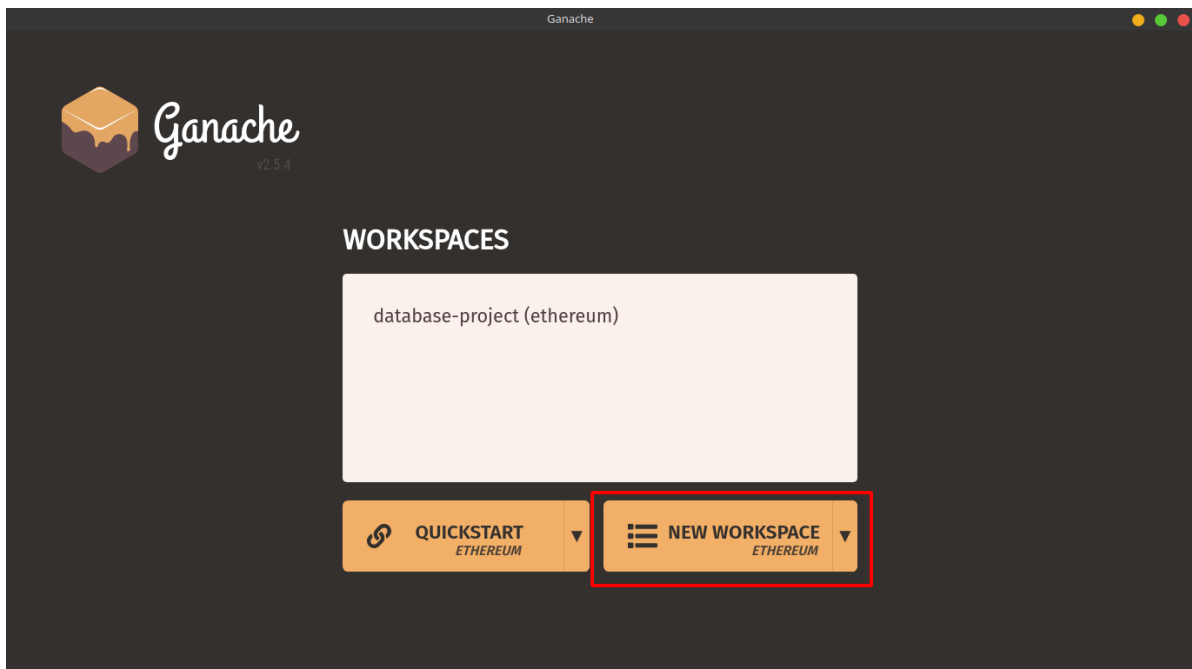
```
-rwxrwxr-x 1 andrew andrew 153130048 out  9 14:28  ganache-2.5.4-linux-x86_64.AppImage
```

Para executar o software, vá para a mesma pasta que está o arquivo e use o comando:

```
./ganache-2.5.4-linux-x86_64.AppImage
```

Uma vez aberto, é necessário configurar o Ganache:

Na tela inicial do Ganache escolha a opção NEW WORKSPACE:



A seguinte tela irá abrir:

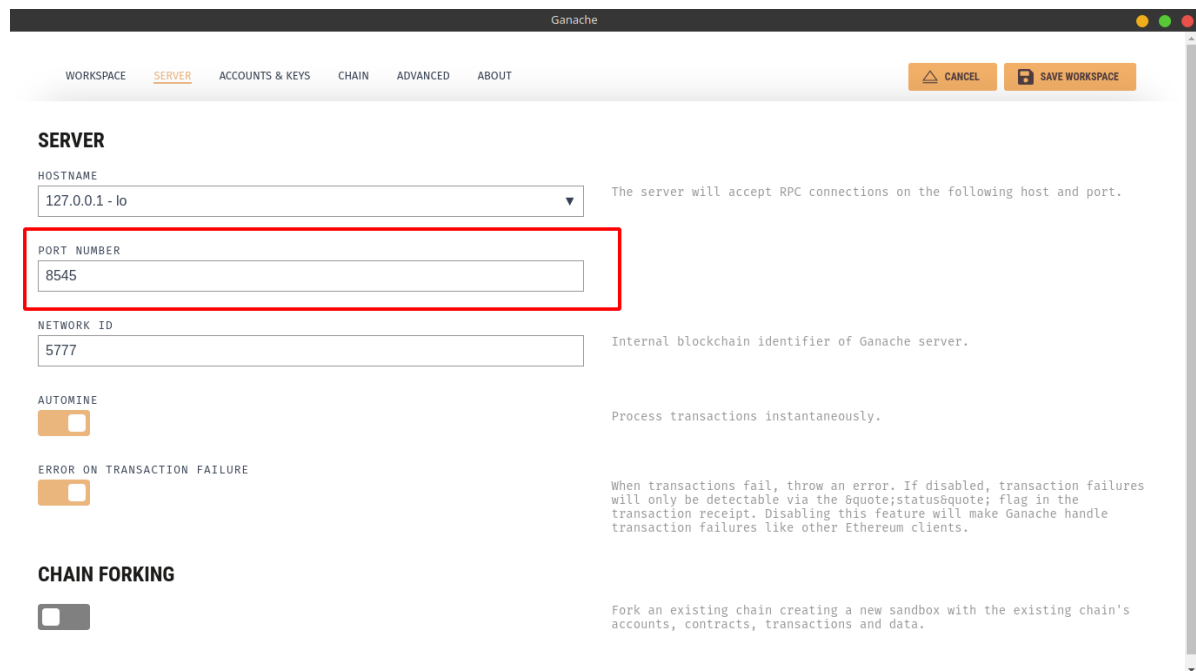
The image shows the 'NEW WORKSPACE' configuration screen. At the top is a navigation bar with tabs: 'WORKSPACE', 'SERVER', 'ACCOUNTS & KEYS', 'CHAIN', 'ADVANCED', and 'ABOUT'. On the right side of the navigation bar are 'CANCEL' and 'SAVE WORKSPACE' buttons. The main area is titled 'WORKSPACE'. It contains two input fields: 'WORKSPACE NAME' with the value 'living-answer' and a description 'A friendly name for this workspace.'; and 'TRUFFLE PROJECTS' with a description 'Link Truffle projects to this workspace by adding their truffle-config.js or truffle.js file to this workspace. This will show useful contract and event data to better understand what's going on under the hood.' Below the 'TRUFFLE PROJECTS' field are 'ADD PROJECT' and 'REMOVE PROJECT' buttons.

Nela, escolha o nome do WORKSPACE, podendo ser qualquer um de preferencia. O importante é o campo de TRUFFLE PROJECTS. Nele, é necessário entrar com o caminho absoluto do arquivo `truffle-config.js` que está presente no repositório clonado. No meu caso esse campo ficou da seguinte forma:

TRUFFLE PROJECTS

`/home/andrew/Documentos/UNIFEI/2021.2/Administração e Gerência de Redes de Computadores/Trabalhos/1/Project/truffle-config.js`

Na aba SERVER é necessário garantir que o PORT NUMBER seja o mesmo presente no LocalHost do MetaMask. Isso será reiterado no item 6) MetaMask dessa sessão.



The screenshot shows the Ganache application window with the 'SERVER' tab selected. The 'PORT NUMBER' field is highlighted with a red rectangle and contains the value '8545'. Other visible fields include 'HOSTNAME' (127.0.0.1 - lo), 'NETWORK ID' (5777), 'AUTOMINE' (checked), and 'ERROR ON TRANSACTION FAILURE' (checked). The 'CHAIN FORKING' section is also visible with a checkbox.

5) Truffle framework

O truffle framework é usado para configurações e manipulações dos contratos inteligentes, por exemplo, implementá-los no Ganache. **Ele também integra um interpretador de Solidity.**

No meu computador foi necessário uma configuração de permissão:

```
sudo chown -R $(whoami) ~/.npm
```

```
sudo chown -R $(whoami) /usr/local/lib/node_modules
```

Instalação global:

```
npm install -g truffle @5.4.14
```

6) MetaMask

A nossa aplicação é web e ela utiliza contratos inteligentes. Portanto, precisamos de uma carteira Ethereum no navegador para realizar a comunicação entre o navegador e a blockchain e validade as transações que são feitas. A carteira tem um endereço de usuário com a quantidade de Ether que esse usuário possui. Cada ação que muda o estado do contrato inteligente envolve uma cobrança de taxa em Ether, essa taxa é debitada do balanço da carteira.

Vamos usar a carteira MetaMask, ela é uma extensão de navegador. Os navegadores que suportam a MetaMask são Chrome, Firefox, Brave e Edge. [Site para instalação](#)

Depois de criar uma conta na MetaMask, é necessário conectar o Ganache com a Metamask:

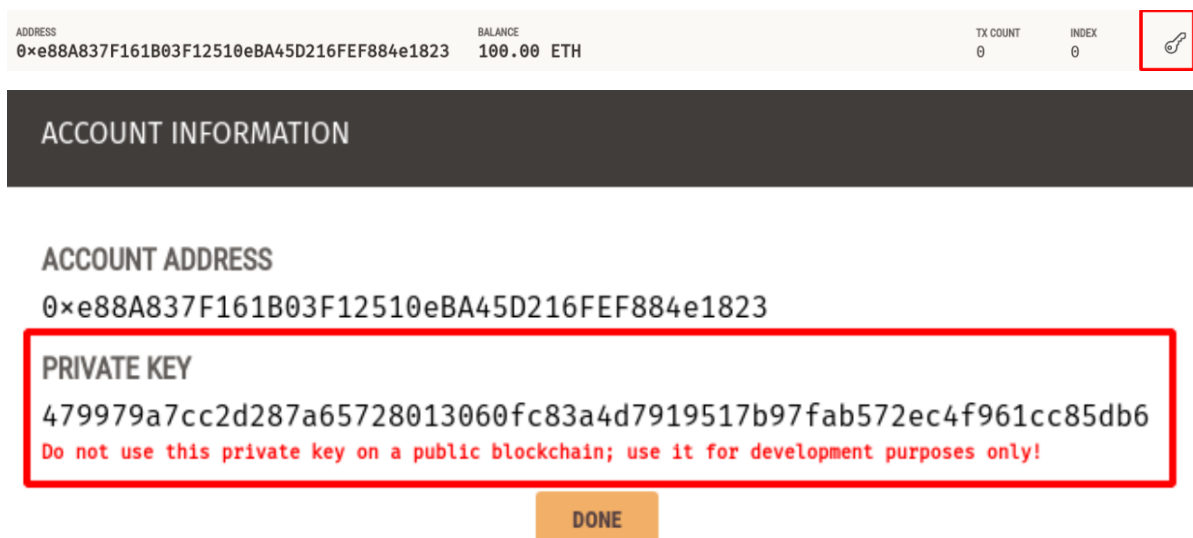
1. Abra o Ganache e selecione a sua sessão

2. Clique na engrenagem no canto esquerdo
3. Vá na aba "Server"
4. Mude o "Port Number" para o valor do Localhost da MetaMask. No meu computador é 8545

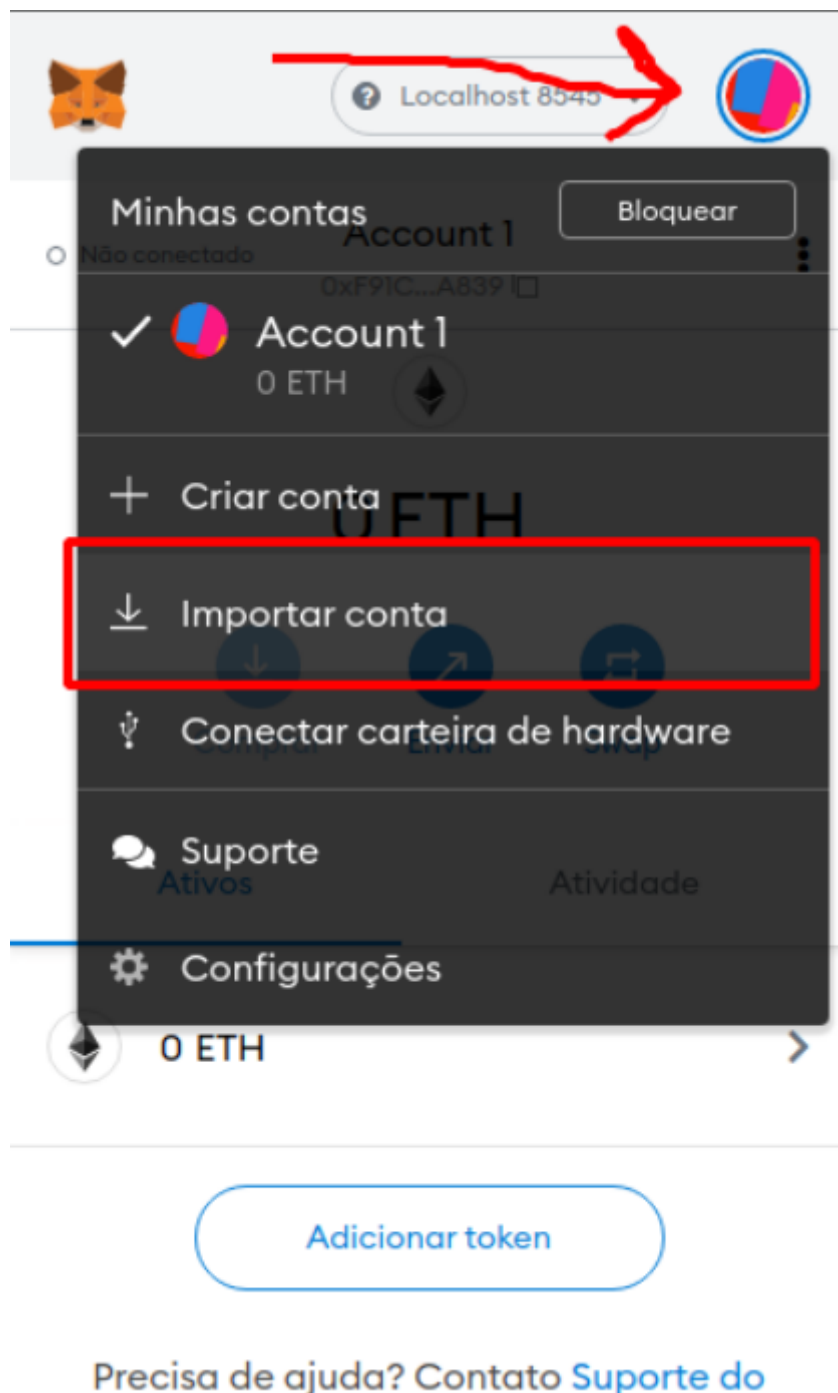


É necessário adicionar uma conta na MetaMask. Vamos usar uma conta do Ganache:

1. Copie a chave privada da conta:



2. Agora para adicionar a conta na MetaMask vá para essa opção e cole a chave no campo indicado:



6. Utilizando a aplicação

1) Ganache

Abra o Ganache e selecione o workspace criado de acordo com o item 4) da sessão 5. Preparando o Ambiente.

2) Instalando as dependências npm

Assumindo que a aplicação foi clonada do repositório no GitHub no link <https://github.com/andre-wunifei/blockchain-ethereum>, a primeira coisa a se fazer é instalar as dependências presentes no `package.json`:

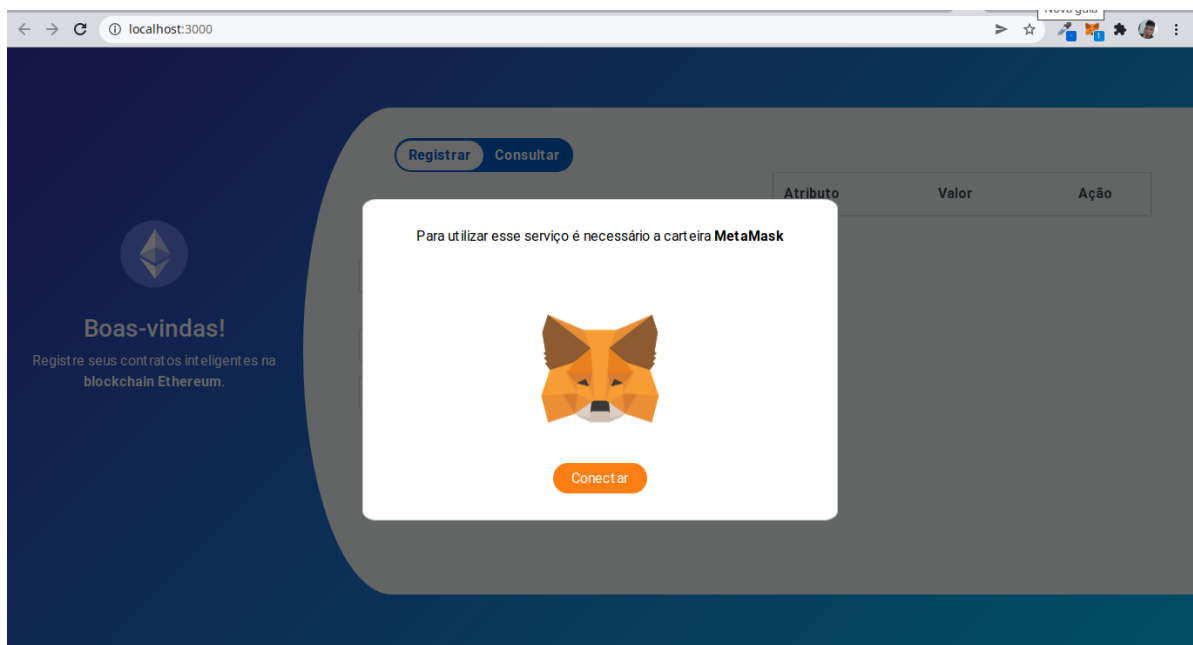
```
"devDependencies": {  
  "bootstrap": "^5.1.3",  
  "lite-server": "^2.3.0",  
  "nodemon": "^2.0.13",  
  "truffle": "^5.4.14",  
  "truffle-contract": "^4.0.31",  
  "web3": "^1.6.1"  
}
```

Para isso, no terminal aberto no diretório clonado, entre com o comando `npm install`

3) lite-server

Após instalar as dependências, para o funcionamento da nossa aplicação precisamos utilizar a aplicação **lite-server**. No mesmo arquivo `package.json`, foi definido um script que roda o lite-server, para rodar o script entre com o comando `npm run dev` no terminal aberto no diretório clonado.

Se tudo funcionar, o browser irá abrir automaticamente na página da aplicação:



4) Utilizando a aplicação

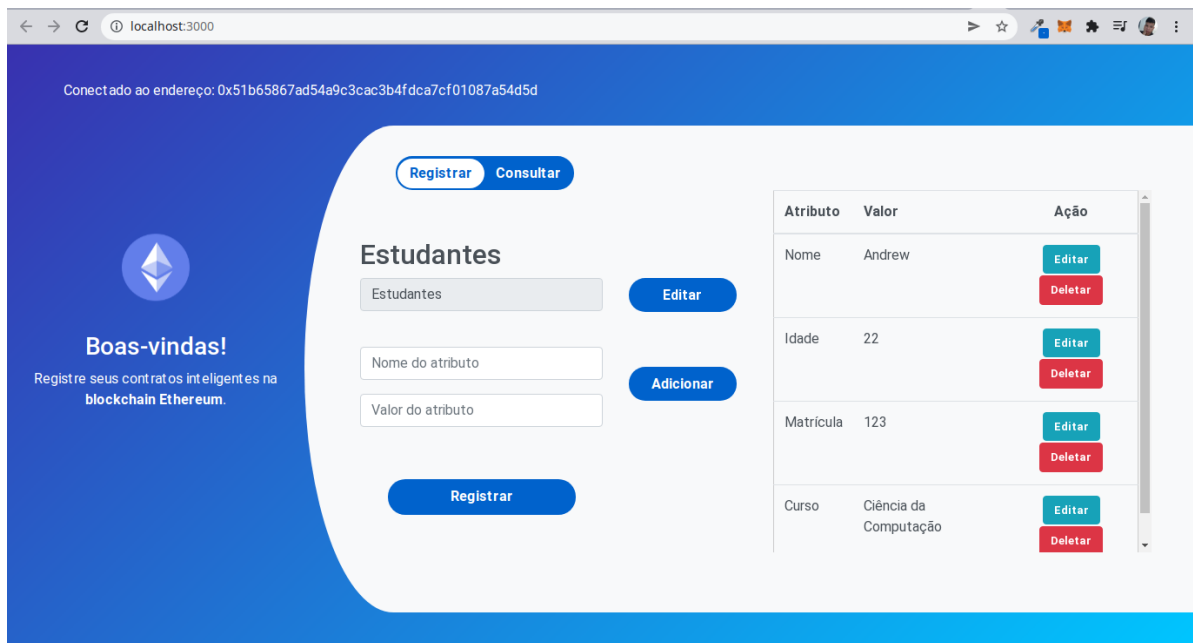
Primeiramente é necessário a conexão com a carteira MetaMask: Clique em Conectar na tela do Metamask, a seguinte tela irá abrir:



Selecione a conta associada ao Ganache, aperte em Próximo e depois Conectar. Se tudo der certo a tela do MetaMask irá sumir e você poderá utilizar a aplicação. A aplicação tem duas páginas:

Registrar e Consultar.

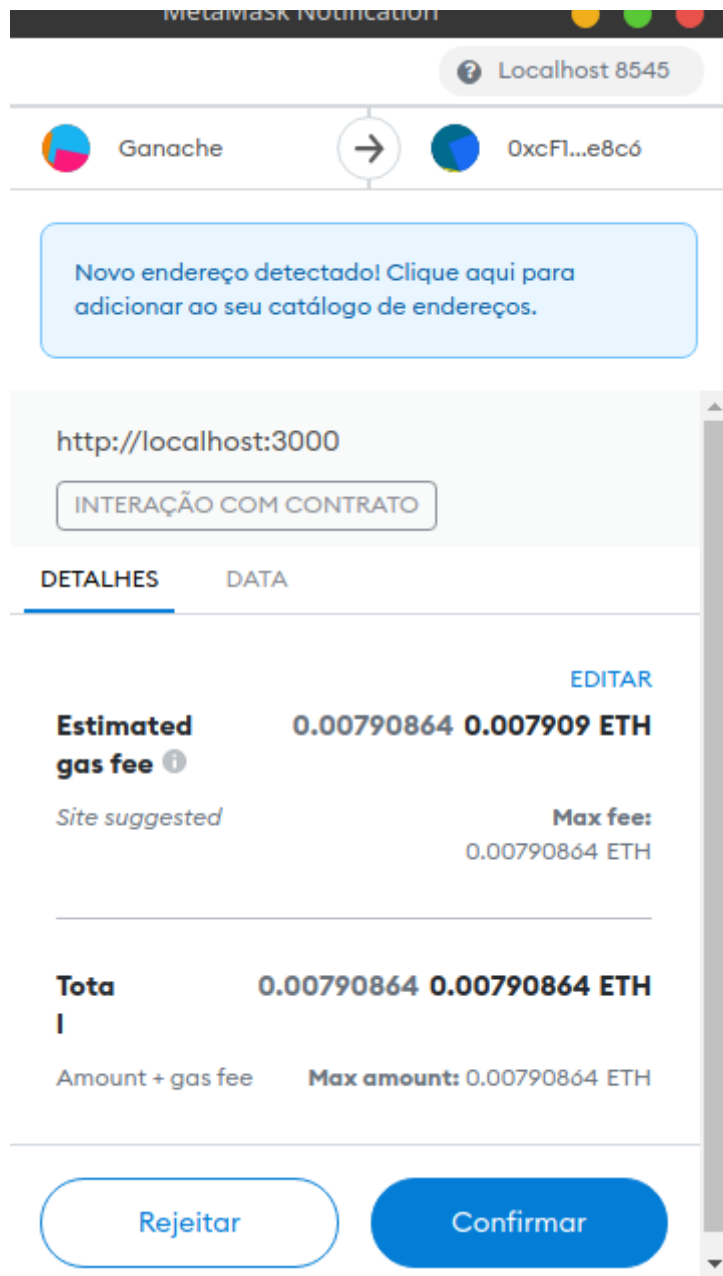
Na página Registrar ocorre os registros das chaves:valores na tabela. A página tem três campos de entrada: Nome da tabela, Nome do atributo, Valor do atributo. Os três campos precisam ser preenchidos: o nome da tabela recebe apenas um valor, os outros dois recebem um valor, porém vários valores podem ser entrados de cada vez. Exemplo de valores em uma tabela chamada Estudantes:



O atributo "Nome" e valor "Andrew", por exemplo, foram entrados nos campos "Nome do atributo" e "Valor do atributo", respectivamente. Com os valores inseridos foi clicado em Adicionar e o processo se repete para os pares subsequentes.

É possível editar ou deletar os valores de atributo e valor nos botões de Ação. Toda edição ou remoção devem ser feitas antes de se clicar no botão de Registrar, porque não é possível alterar um dado da blockchain.

Depois de realizar as revisões necessárias e apertar no botão de Registrar, uma página da MetaMask irá abrir para ser realizada a assinatura do contrato inteligente mostrando todas as condições e taxas:




Nesse caso o total de taxa foi de aproximadamente 0.007 Ether. Se estiver de acordo aperte em Confirmar e os registros serão escritos na blockchain.

Para consultar o registro aperte no botão Consultar na parte superior. Nessa página é necessário entrar com o nome da tabela. Se existir um endereço de usuário associado a esse nome de tabela, os dados serão retornados. Essa ação não envolve cobrança de taxas:

← → ↻ ⓘ localhost:3000

⌕ ☆ ⚙ 🐙 ⚙ ⌵ 👤 ⋮

Conectado ao endereço: 0x51b65867ad54a9c3cac3b4fdca7cf01087a54d5d



Boas-vindas!
Registre seus contratos inteligentes na
blockchain Ethereum.

Registrar

Consultar

Estudantes

Consultar

| Atributo | Valor |
|-----------|-----------------------|
| Nome | Andrew |
| Idade | 22 |
| Matrícula | 123 |
| Curso | Ciência da Computação |