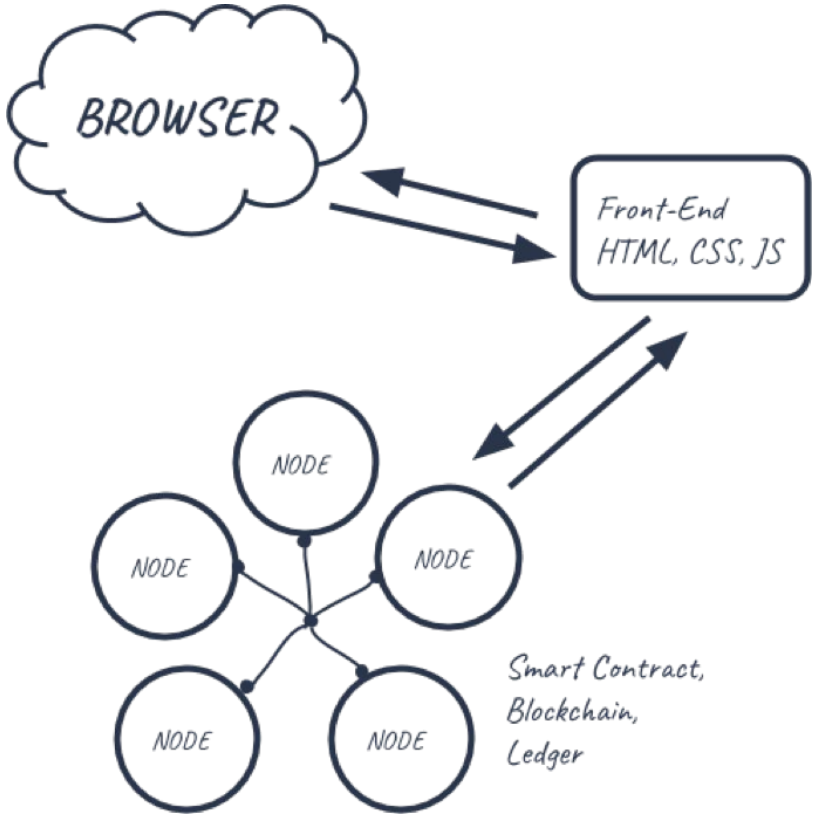
IMD0913

ĐApps

ĐApp

backend distribuído + frontend


BROWSER
Front-End
HTML, CSS, JS
NODE
NODE
NODE
NODE
NODE
Smart Contract,
Blockchain,
Ledger

Tokens

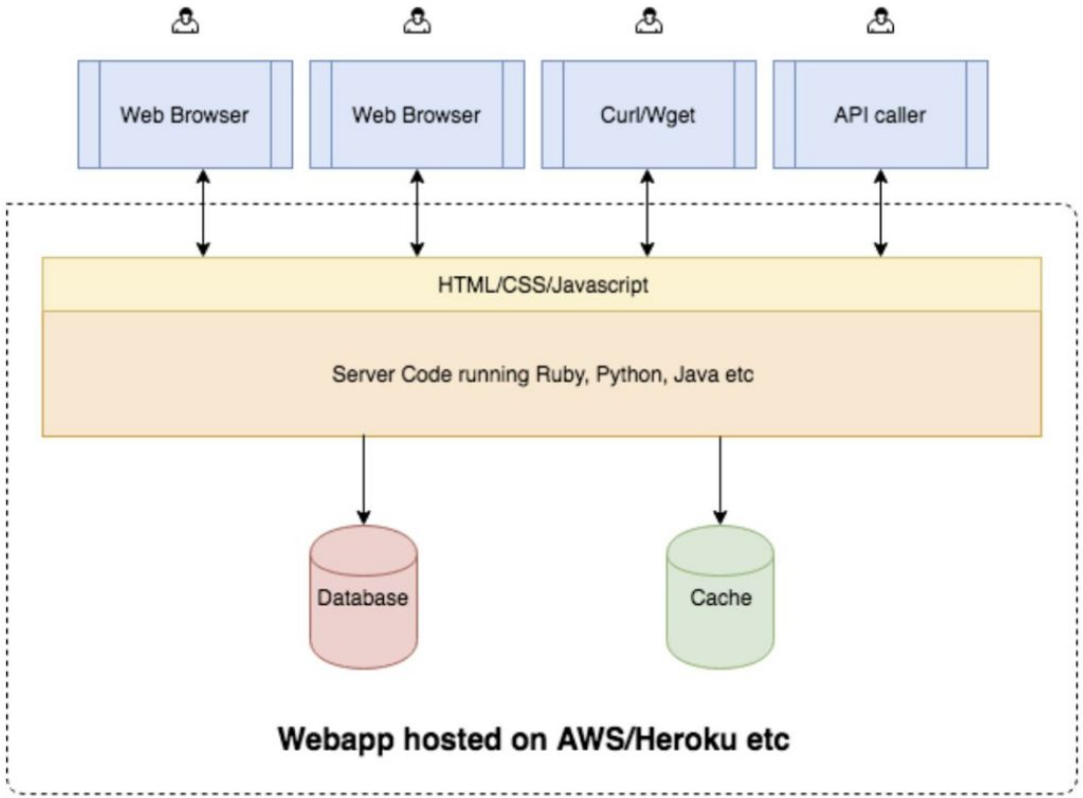
`web3.js`

Ethereum JavaScript API

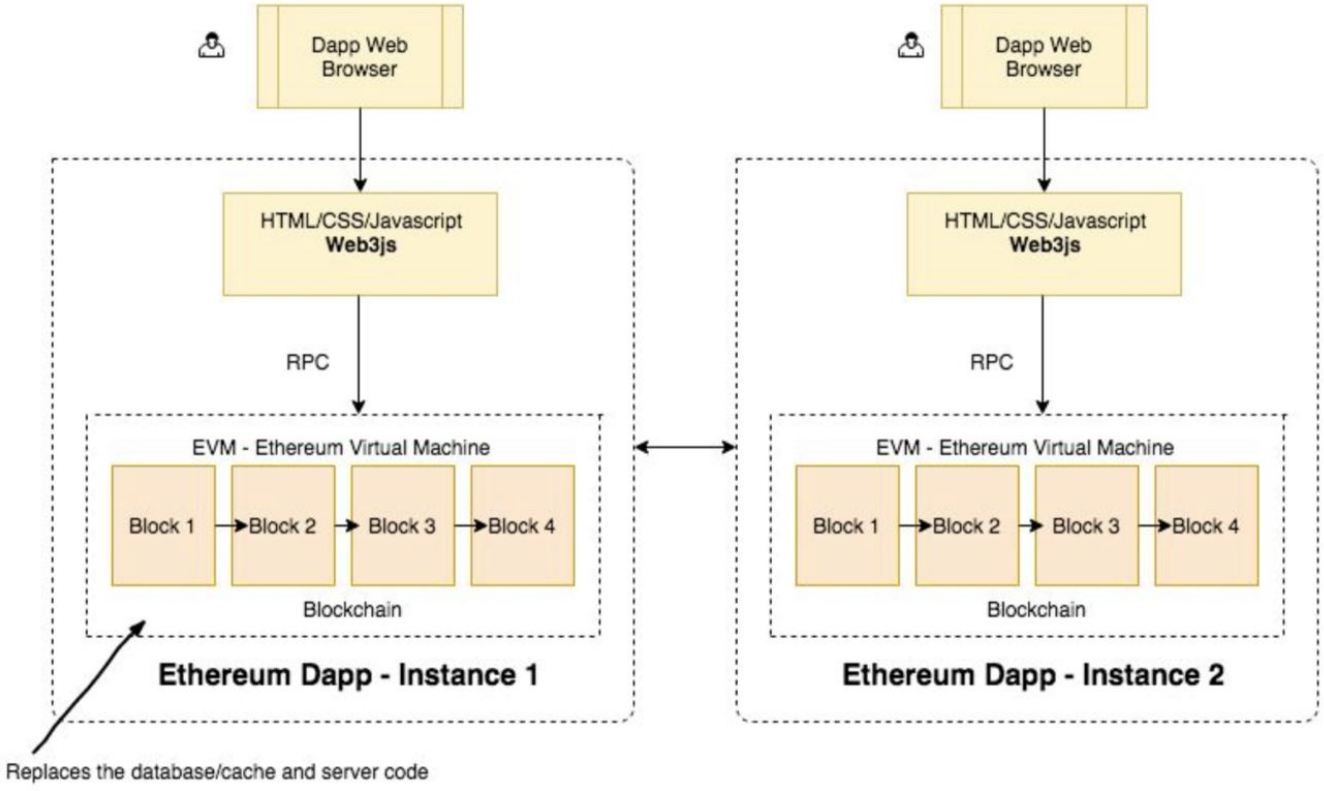
Coleção de bibliotecas que permitem um usuário **interagir** com a plataforma *Ethereum* e com *Smart Contracts*

https://web3js.readthedocs.io

Web convencional



Web 3.0



com o que o usuário interage

o que transforma a interação do usuário em interações com o "software" e recebe os resultados

a lógica de negócio (*smart contracts*)

web3

web3.js é o ponto de entrada do blockchain Ethereum do lado cliente de um ĐApp

Comunica com os nós na rede utilizando RPC

web3 contém:

- **web3.eth** : para interação com o blockchain do Ethereum

Ethereum ABI: Expondo os métodos do contrato

ABI: Application Binary Interface

Uma ABI é como você pode chamar funções de um contrato e obter os dados de retorno

> Determina como funções são chamadas e em que formato a informação deve ser passada de um componente do programa para outro

Por que é necessário?

> Você precisa de um jeito para especificar qual função será invocada do contrato e também garantir o tipo de dado que é retornado

Não é parte do protocolo Ethereum, você pode definir seu próprio ABI - porém é mais fácil obedecer o formato definido pelo **web3.js**

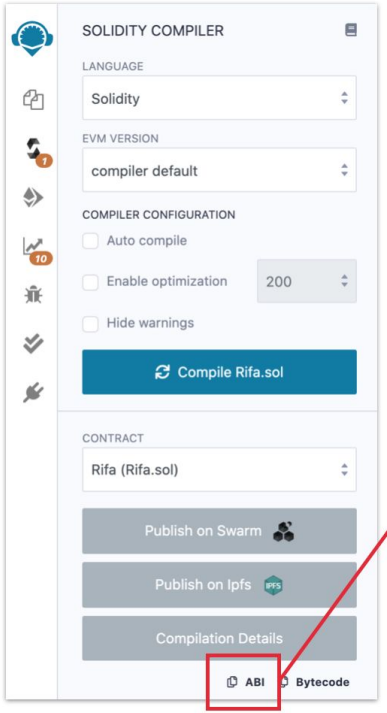
Ethereum ABI: Expondo os métodos do contrato

```
contract Test {
    uint a;
    address d = 0x1234567890123456789012345678901234567890;

    constructor(uint testInt)  { a = testInt;}

    event Event(uint indexed b, bytes32 c);
    event Event2(uint indexed b, bytes32 c);

    function foo(uint b, bytes32 c) returns(address) {
        Event(b, c);
        return d;
    }
}
```

Ethereum ABI: Expondo os métodos do contrato

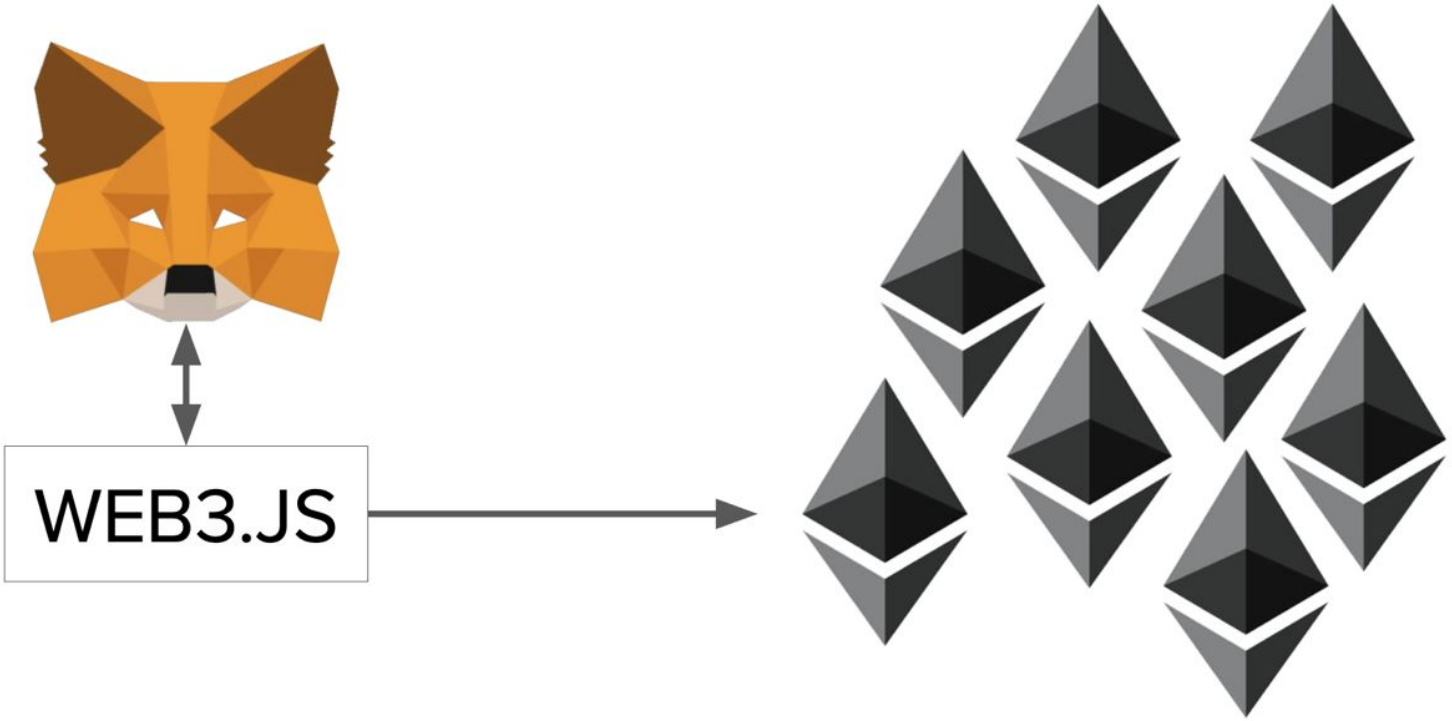
```
[{
    "type":"constructor",
    "payable":false,
    "stateMutability":"nonpayable",
    "inputs":[{"name":"testInt","type":"uint256"}],
  },{
    "type":"function",
    "name":"foo",
    "constant":false,
    "payable":false,
    "stateMutability":"nonpayable",
    "inputs":[{"name":"b","type":"uint256"}, {"name":"c","type":"bytes32"}],
    "outputs":[{"name":"","type":"address"}]
  },{
    "type":"event",
    "name":"Event",
    "inputs":[{"indexed":true,"name":"b","type":"uint256"}, {"indexed":false,"name":"c","type":"bytes32"}],
    "anonymous":false
  },{
    "type":"event",
    "name":"Event2",
    "inputs":[{"indexed":true,"name":"b","type":"uint256"},{"indexed":false,"name":"c","type":"bytes32"}],
    "anonymous":false
}]
```

ABI no Remix




```
{
    "inputs": [
        {
            "internalType": "uint256",
            "name": "_quant",
            "type": "uint256"
        }
    ],
    "name": "comprarRifa",
    "outputs": [],
    "stateMutability": "payable",
    "type": "function"
},
{
    "inputs": [],
    "stateMutability": "nonpayable",
    "type": "constructor"
},
{
    "anonymous": false,
    "inputs": [
        {
            "indexed": false,
            "internalType": "address",
            "name": "comprador",
            "type": "address"
        },
        {
            "indexed": false,
            "internalType": "uint256",
            "name": "quant",
            "type": "uint256"
        }
    ],
    "name": "RifaComprada",
    "type": "event"
},
{
    "inputs": [],
    "name": "sacarPremio",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
},
{
    "inputs": [],
    "name": "sortearRifa",
    "outputs": [],
    "stateMutability": "nonpayable",
```

Juntando tudo...



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Dapp front-end</title>
    <script language="javascript" type="text/javascript" src="web3.min.js"></script>
    <script language="javascript" type="text/javascript" src="abi.js"></script>

</head>

<body>
    <script>
        window.addEventListener('load', function () {
            var web3;
            var account;

            if (typeof window.ethereum !== "undefined") {
                try {
                    const accounts = await window.ethereum.request({
                        method: "eth_requestAccounts", // Requisita primeiro acesso ao Metamask
                    });
                    account = accounts[0];
                } catch (error) {
                    console.error("Usuário negou acesso ao web3!");
                }
                web3 = new Web3(window.ethereum);
            } else {
                console.error("Instalar MetaMask!");
            }

            startApp();
        });
    </script>
</body>
</html>
```

```
var abi = [
    {
        "constant": false,
        "inputs": [],
        "name": "sortearRifa",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "verGanhador",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "verTotalDeRifas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
```

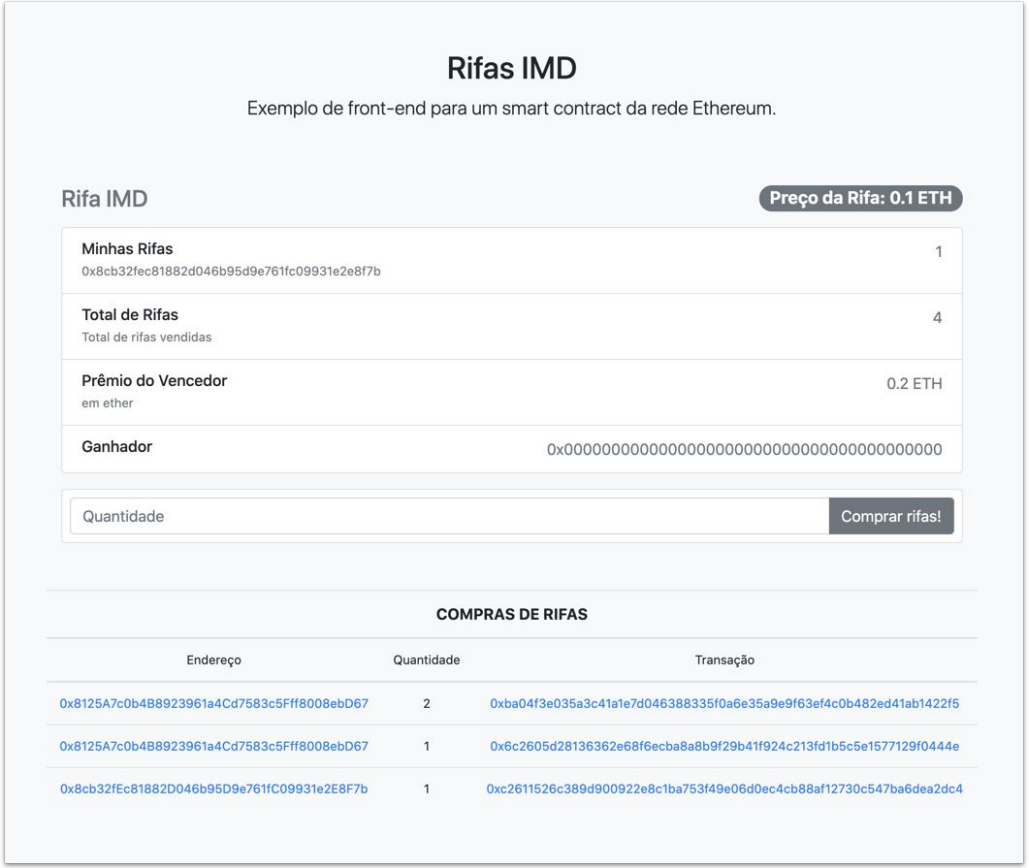
```
var enderecoContrato = "0x38996ACE62d2a5C147e04f0bbB36a518be8aa164";
var contrato = null;

function startApp() {
  contrato = new web3.eth.Contract(abi, enderecoContrato);

  web3.eth.getAccounts().then(function (result) {
    userAccount = result[0]; // Tudo com web3.js retorna Promises!
  });
}

function verTotalDeRifas() { // function verTotalDeRifas() public view returns (uint)
  contrato.methods.verTotalDeRifas().call().then((resposta) => { /* o que você quiser fazer! */ });
}

function comprarRifa() { // function comprarRifa(uint _quant) public payable
  let quant = document.getElementById("quantidade").value;
  let preco = 100000000000000000 * quant;
  return contrato.methods.comprarRifa(quant).send({ from: account, value: preco }).then(atualizaInterface);
}
```

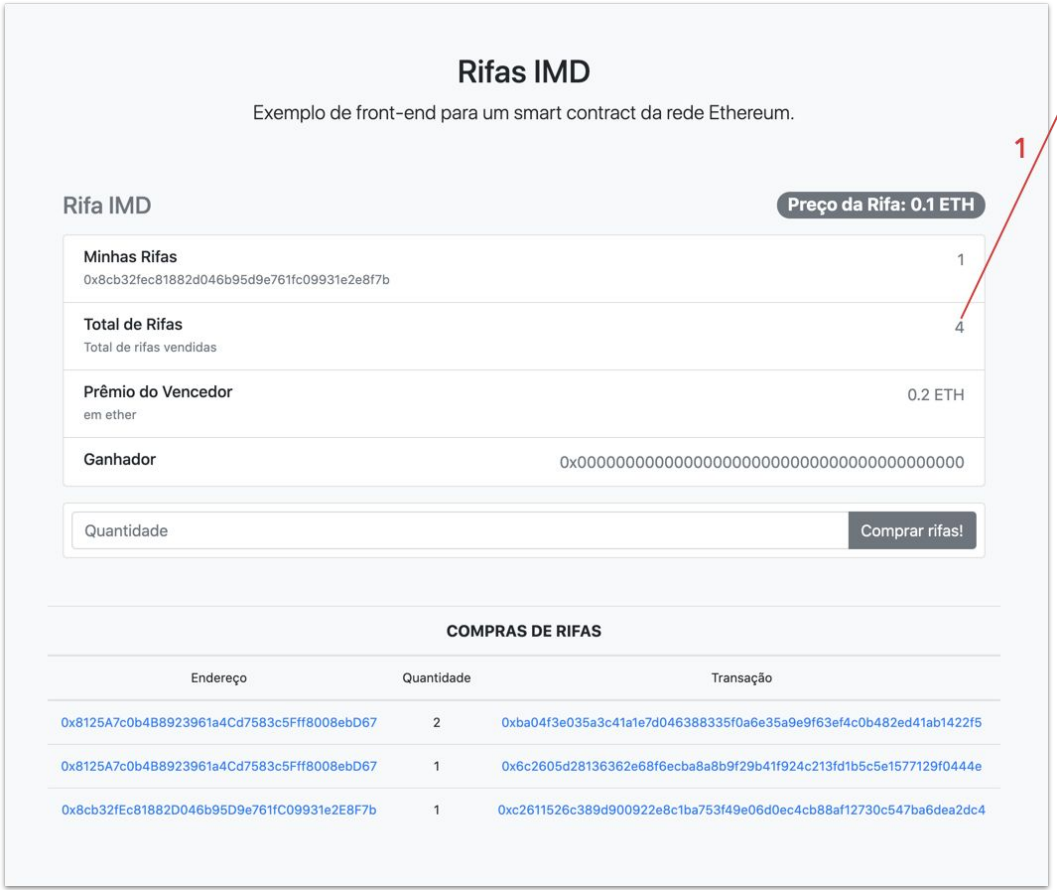
Tokens

Como *tokens* não tem seu próprio *blockchain*, por trás dos panos eles usam ***smart contracts***

Todo *token* disponível no blockchain do Ethereum tem seu próprio *smart contract*

- Registram os donos dos *tokens* para aquele tipo de *token*
- Como a EVM não sabe o que são *tokens*, a transação tem valor de 0 ETH
- Endereço de destino e quantidade de *tokens* é especificada como dados de entrada (*input data*)
- A transação é enviada para o contrato do *token*


Rifas IMD
Exemplo de front-end para um smart contract da rede Ethereum.
Rifa IMD
Preço da Rifa: 0.1 ETH
Minhas Rifas
0x8cb32fec81882d046b95d9e761fc09931e2e8f7b
1
Total de Rifas
Total de rifas vendidas
4
Prêmio do Vencedor
em ether
0.2 ETH
Ganhador
0x0000000000000000000000000000000000000000
Quantidade
Comprar rifas!
COMPRAS DE RIFAS
Endereço
Quantidade
Transação
0x8125A7c0b4B8923961a4Cd7583c5Fff8008ebD67
2
0xba04f3e035a3c41a1e7d046388335f0a6e35a9e9f63ef4c0b482ed41ab1422f5
0x8125A7c0b4B8923961a4Cd7583c5Fff8008ebD67
1
0x6c2605d28136362e68f6ecba8a8b9f29b41f924c213fd1b5c5e1577129f0444e
0x8cb32fEc81882D046b95D9e761fC09931e2E8F7b
1
0xc2611526c389d900922e8c1ba753f49e06d0ec4cb88af12730c547ba6dea2dc4



1

```
function verTotalDeRifas() {
  contrato.methods.verTotalDeRifas()
                  .call()
                  .then((resultado) => { /* */ });
}
```

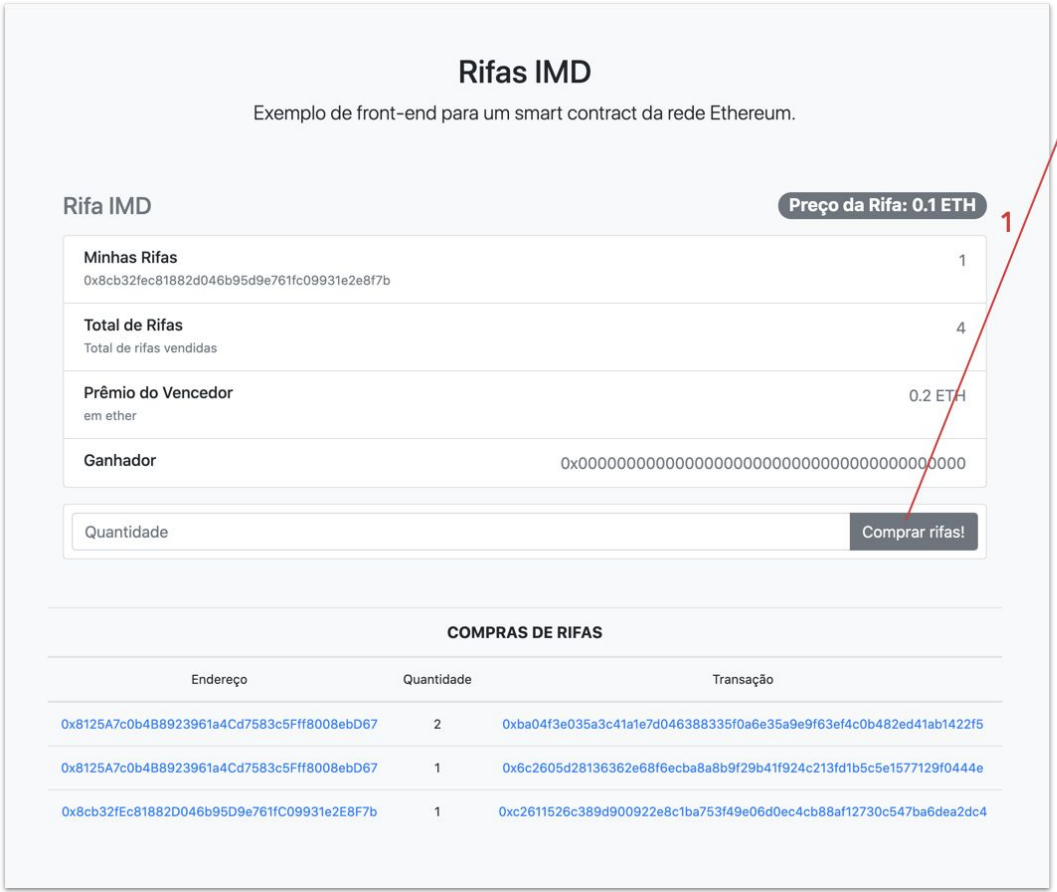
JAVASCRIPT - FRONTEND

2

3

```
function verTotalDeRifas() public view returns (uint) {
    return rifas.length;
}
```

SOLIDITY - EVM - BLOCKCHAIN - BACKEND



1

```javascript
function comprarRifa() {
  let quant = document.getElementById("quantidade").value;
  let preco = 100000000000000000 * quant;
  contrato.methods.comprarRifa(quant)
                  .send({ from: account, value: preco })
                  .then(atualizaInterface);
}
```

JAVASCRIPT - FRONTEND

2

```solidity
function comprarRifa(uint _quant) public payable {
    require(msg.value == _quant*valorDaRifa);

    for (uint i = 0; i < _quant; i++) {
        rifas.push(msg.sender);
        rifasPorPessoa[msg.sender]++;
    }

    emit RifaComprada(msg.sender, _quant);
}
```

SOLIDITY - EVM - BLOCKCHAIN - BACKEND

TRUFFLE

Ganache