

BOAS PRÁTICAS NO DESENVOLVIMENTO DE SMART CONTRACTS

André Ferreira
Rafael Capaci





André Ferreira

- Desenvolvedor na **FOHAT**
- Co-organizador do **Meetup Blockchain Curitiba**



Rafael Capaci

- Desenvolvedor na **FOHAT**
- Co-organizador dos meetups **Blockchain Curitiba** e **Curitiba.js**

FOHAT

fohat.co



WE ARE THE
FIRST LATIN
AMERICAN
COMPANY TO
CONCILIATE
ENERGY AND
BLOCKCHAIN

Blockchain

Banco de dados distribuído que registra todas as transações e as mantém imutáveis

FOHAT

Ethereum

Plataforma de aplicações descentralizadas que
executa contratos inteligentes em blockchain

FOHAT



Ether (*gas*)



Smart Contract

FOHAT

Smart Contract

- Similar à uma classe, que possui métodos e atributos
- Identificado por um endereço próprio (**0x71c20d...9ec0**)
- Executa regras contratuais definidas em código
- Escritos em linguagem de alto nível (**Solidity** ou outras)
- Compilados em bytecode
- A **EVM** consome **gas** para executar as operações do contrato

Solidity

Linguagem de programação de **tipagem estática**
para o desenvolvimento de contratos inteligentes

FOHAT

Visibilidade

- **Public:** podem ser chamadas por funções do próprio contrato, de contratos herdados ou por usuários externos.
- **External:** podem ser acessadas somente externamente, não sendo possível ser chamada pelo próprio contrato.
- **Private:** podem ser acessadas somente pelo próprio contrato.
- **Internal:** podem ser acessadas pelo próprio contrato e por contratos herdados.

Visibilidade

Mantenha suas funções ***private*** ou ***internal***,
a menos que haja necessidade de interação externa.

Armazenamento de variáveis

- **Memory:** utilizada para **valores temporários**.
É apagada entre execuções distintas e é barata.
- **Storage:** utilizada para armazenar **variáveis de estado** do contrato.
É mais cara pois é gravada no blockchain.

Armazenamento de variáveis

Variáveis locais (*memory*) e de estado (*storage*) serão sempre públicas para leitura, mesmo se marcadas como *private*.

Tipos de funções

- **view:** função que **não modifica o estado** na blockchain
- **pure:** função que **não lê e não modifica o estado** na blockchain
- **payable:** permite que o contrato receba ETH na execução da função

Enviando ETH

- **transfer()** lança erro e para a execução caso haja falta de gas
- **send()** retorna false e deixa que você trate o erro



Caso usar **send()**, **NÃO ESQUEÇA DE TRATAR O RETORNO!!!**

Enviando ETH: transfer() vs. send()



```
msg.sender.transfer(balance);
```



```
require(msg.sender.send(balance));
```


Fallback function

- Aquela que não deve ser nomeada
- Não possui argumentos
- E não retorna nada!



```
contract Test() {  
    // Fallback function  
    function () {  
  
    }  
}
```

E pra que serve então???

Fallback function

- É executada quando o contrato recebe ETH
- Ou quando se tenta executar uma função de nome inválido no contrato

Fallback function

- Sem fallback function, o ETH enviado será rejeitado
- Um fallback executa com 2300 de *gas
 - Gas insuficiente para trabalhar com variáveis em STORAGE
 - Basicamente o que dá pra fazer é emitir um evento, informando que recebeu ETH

Fallback function

```
contract Test {  
    event ReceivedEther(address sender, uint value);  
  
    function () {  
        emit ReceivedEther(msg.sender, msg.value);  
    }  
}
```

Fallback function

- Um fallback pode executar com mais gas, caso isso fique explícito
- No caso de um contrato enviar ETH para outro, pode chamar a função ***addr.call.value(x)***
 - Dessa forma, o fallback pode fazer operações mais caras

Fallback function

```
contract ExpensiveTest {  
    function() {  
        // Expensive stuff  
    }  
}  
  
contract Caller {  
    ExpensiveTest test = ExpensiveTest(0x000...);  
    function sendEther(uint _value) payable {  
        test.call.value(_value)()  
    }  
}
```

Boas práticas de programação

DIAS DEPOIS, VENDO LIM
CÓDIGO DO ALONSO...

```
IF ($num_pagina<10)
{
    // Não faz nada
}
ELSE
{
    echo "Exibir paginacao";
}
```

PLOFT!



FOHAT

Style Guide

- Indentação: 4 espaços por nível
- Tabs ou Espaços? Espaços
- Tamanho máximo de linha: 79 ou 99 caracteres
- Estruturas de controle, declaração de funções e variáveis
- Funções *modifiers*
- Convenções de nomenclatura
- *imports* no topo do contrato
- Ordem de declaração das funções
- Basicamente o recomendado é seguir a **PEP 8**

Loops

- Blocos possuem um limite de gas para as transações dentro dele
- Tomar cuidado com loops de tamanho variável
- Não se aplica a **view** e **pure** functions
- **Porém, quando uma view ou pure function é chamada a partir de uma transação, ela também consome gas**

Loops

```
uint users[];  
function dynamicSizedLoop() {  
    for (uint i = 0; i < users.length; i++) {  
        // do something  
    }  
}
```

Custo

- O consumo de gas tem impacto direto na estrutura de custos da organização
- Por isso, **otimize seus contratos**

Custo - OPCODE

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	Arithmetic operation
ADDMOD/MULMOD	8	Arithmetic operation
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/SGT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	Stack operation
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/20,000	Storage operation
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

Manipulação de variáveis de storage

- Corresponde a instrução SSTORE
- 20.000 de gas para guardar novos dados
- 5.000 de gas pra alterar dados *
 - Com exceção de uma alteração que zera um dado (**Gas refund**)




Manipulação de variáveis de storage

Recomendação:

Escreva o mínimo possível na storage


FOHAT

Manipulação de variáveis de storage



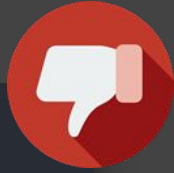
```
contract Test {  
    uint sum = 0;  
  
    function storeSum(uint x) {  
        for ( uint i = 0 ; i < x ; i++) {  
            sum += i;  
        }  
    }  
}
```

Manipulação de variáveis de storage



```
contract Test {  
    uint sum = 0;  
  
    function storeSum(uint x) {  
        uint temp = 0;  
        for ( uint i = 0 ; i < x ; i++) {  
            temp += i;  
        }  
        sum = temp;  
    }  
}
```


Custo

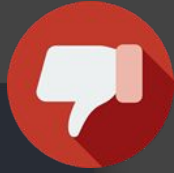


```
function doSomeStuff(uint x) {  
  if (x > 0) {  
    // ...  
    if (x != 0) {  
      // ...  
    }  
  }  
}
```



```
function doSomeStuff(uint x) {  
  if (x > 0) {  
    // ...  
    if (x != 0) {  
    // ...  
  }  
}
```

Custo

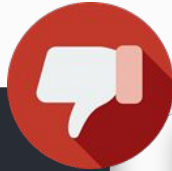


```
function doMoreStuff(uint x) {  
  if (x > 5) {  
    // ...  
    if (x*x < 20) {  
      // ... useless code  
    }  
  }  
}
```



```
function doMoreStuff(uint x) {  
  if (x > 5) {  
    // ...  
    if (x*x < 20) {  
      // ... useless code  
    }  
  }  
}
```

Custo de deploy



```
uint value;
function set(uint _value) public {
    uint x = 0;
    if (_value > 5) {
        if (_value ** 2 < 20 ) {
            x = _value * 4;
            x += _value;
        }
        x += _value;
    }
    value = x;
}
```

106.297 gas



```
uint value;
function set(uint _value) public {
    uint x = 0;
    if (_value > 5) {
        x += _value;
    }
    value = x;
}
```

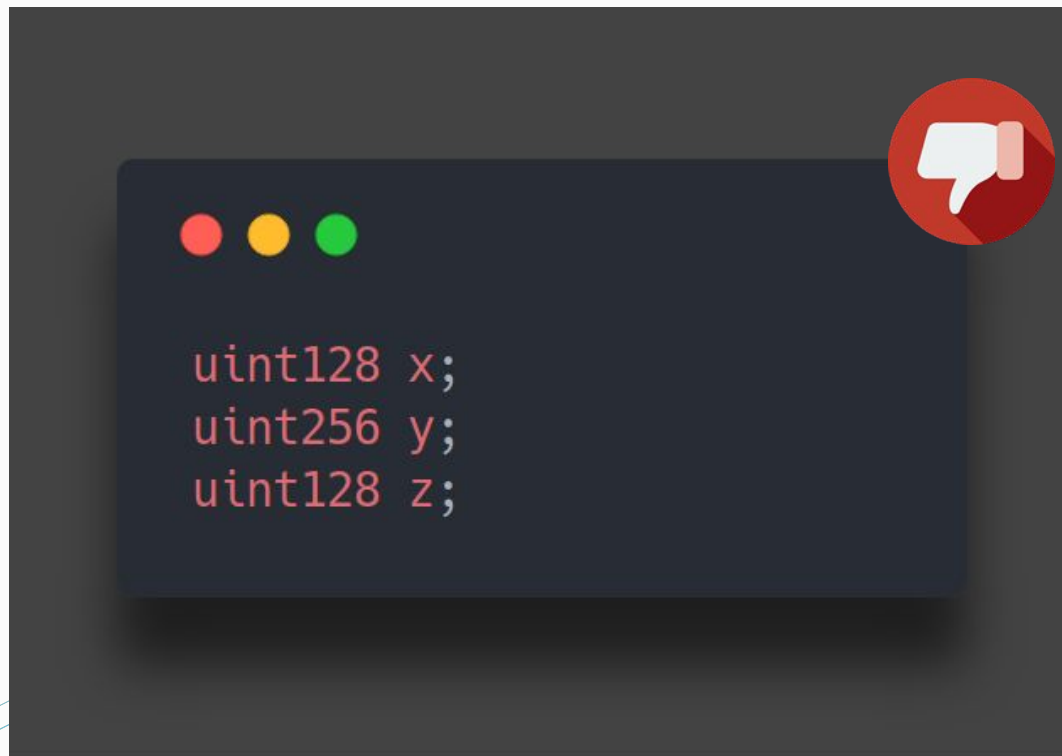
100.123 gas

FOHAT

Custo

- EVM armazena as variáveis em slots de 32 bytes
- Variáveis que ocupam espaço menor são “empacotadas”

Custo

A dark-themed terminal window with three colored window control buttons (red, yellow, green) at the top left. The code is displayed in a light red font. A red circular icon with a white speech bubble is positioned in the top right corner of the terminal window.

```
uint128 x;  
uint256 y;  
uint128 z;
```

FOHAT

Custo

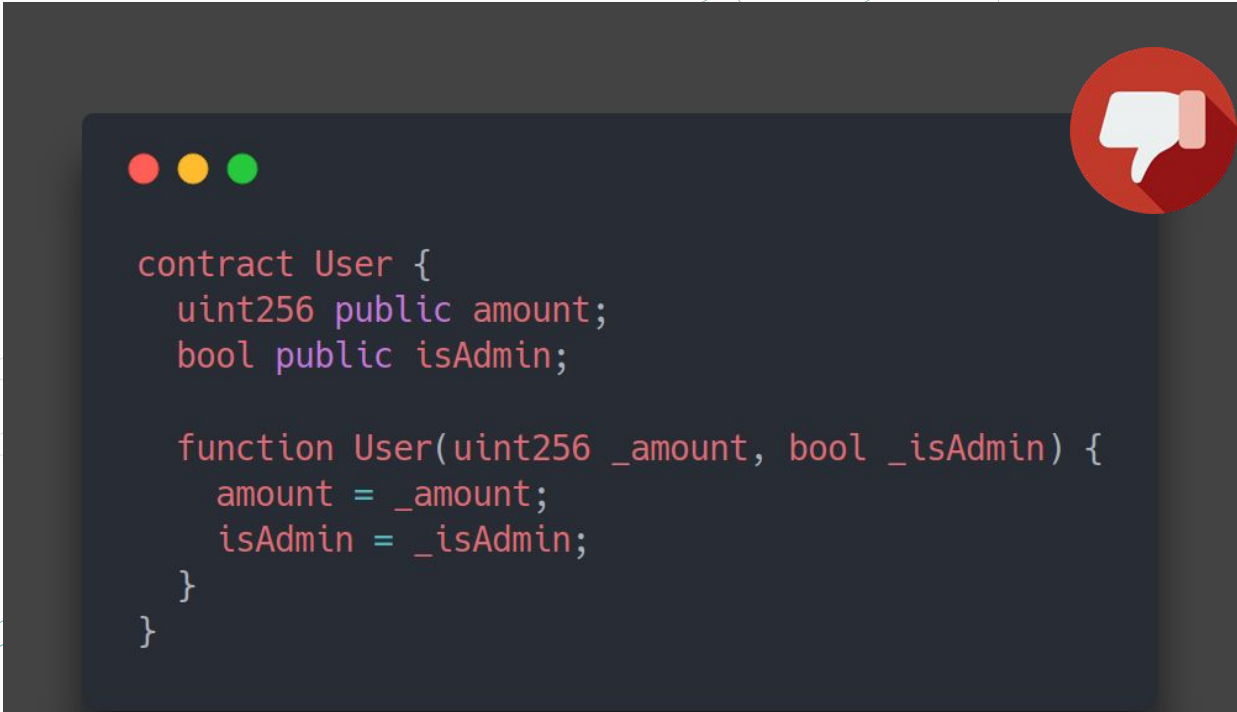
A dark-themed terminal window with three colored window control buttons (red, yellow, green) at the top left. The code is displayed in a light red font. A circular teal icon with a white thumbs-up gesture is positioned in the top right corner of the terminal window.

```
uint128 x;  
uint128 z;  
uint256 y;
```

FOHAT

Criação de contrato

- Evitar usar um contrato para armazenamento de dado somente



```
contract User {
  uint256 public amount;
  bool public isAdmin;

  function User(uint256 _amount, bool _isAdmin) {
    amount = _amount;
    isAdmin = _isAdmin;
  }
}
```

Criação de contrato



```
contract MyContract {  
    mapping(address => uint256) amount;  
    mapping(address => bool) isAdmin;  
}
```



Criação de contrato



```
contract MyContract {  
    struct {  
        uint256 amount;  
        bool isAdmin;  
    }  
  
    mapping(address => User) users;  
}
```

“Reembolso” de gas

- Operações que alteram o estado da blockchain são caras
 - Ex: Criar contratos, guardar dados em um contrato.
- Para incentivar as pessoas a liberarem espaço que não será mais utilizado, a EVM implementa o mecanismo de **Gas Refund**
- Duas operações que reembolsam gas:
 - **SUICIDE**: “matar” o smart contract. Devolve 24.000 gas.
 - **SSTORE**: Apagar dados do contrato. Devolve 15.000.

“Estoque de gas”



GasToken.io

FOHAT

Segurança



“Everyone here is a target for attack. Be paranoid.”

Martin Swende (DEVCON3)

FOHAT

Segurança

- Se houver um bug, não é possível atualizar e corrigir o código
- Qualquer erro pode levar a consequências terríveis
- Disneylândia para hackers
 - Baixo esforço, altos retornos e baixo risco
- Todo o ecossistema está amadurecendo em termos de segurança

Restrição de Acesso



```
contract Ownable {  
  
    constructor() public {  
        owner = msg.sender;  
    }  
  
    modifier onlyOwner() {  
        require(msg.sender == owner);  
        _;  
    }  
  
    function transferOwnership(address _newOwner) public onlyOwner {  
        owner = _newOwner;  
    }  
}
```

Withdraw function

- O contrato não deve enviar Ether durante uma alteração de estado
 - Recomendado: Salvar o saldo dos usuários em um **mapping (address => uint)**
 - Usuário realiza o saque do saldo através de uma função específica
- **King of Ether** Hack



Withdraw pattern

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in a light color with syntax highlighting. A red circular icon with a white speech bubble is positioned in the top-right corner of the editor area.

```
function becomeRichest() public payable returns (bool) {  
    if (msg.value > mostSent) {  
        richest.transfer(msg.value);  
        richest = msg.sender;  
        mostSent = msg.value;  
        return true;  
    } else {  
        return false;  
    }  
}
```


Withdraw pattern



```
function becomeRichest() public payable returns (bool) {
    if (msg.value > mostSent) {
        pendingWithdrawals[richest] += msg.value;
        richest = msg.sender;
        mostSent = msg.value;
        return true;
    } else {
        return false;
    }
}

function withdraw() public {
    uint amount = pendingWithdrawals[msg.sender];
    pendingWithdrawals[msg.sender] = 0;
    msg.sender.transfer(amount);
}
```

Re-entrancy

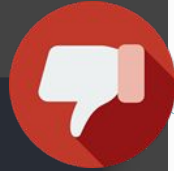
Um contrato (A) pode chamar outro contrato (B) e transferir o controle de Ether para o contrato (B), possibilitando que (B) chame (A) novamente antes que essa interação seja concluída.

Isso permite que o contrato (B) possa realizar um **loop recursivo**, utilizando a função fallback, por exemplo.

Re-entrancy

```
pragma solidity ^0.4.18;

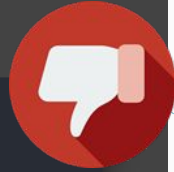
contract Fund {
    /// Mapping of ether shares of the contract.
    mapping(address => uint) shares;
    /// Withdraw your share.
    function withdraw() public {
        if (msg.sender.send(shares[msg.sender]))
            shares[msg.sender] = 0;
    }
}
```



Re-entrancy

```
pragma solidity ^0.4.18;

contract Fund {
    /// Mapping of ether shares of the contract.
    mapping(address => uint) shares;
    /// Withdraw your share.
    function withdraw() public {
        if (msg.sender.call.value(shares[msg.sender]))
            shares[msg.sender] = 0;
    }
}
```



Re-entrancy

Contrato de Ataque

The DAO hack



```
contract FundCollect {
    Fund public fund;

    constructor(address _fund) {
        fund = Fund(_fund);
    }

    function collect() payable {
        fund.put.value(msg.value)();
        fund.withdraw();
    }

    function () payable {
        if (address(fund).balance >= msg.value) {
            fund.withdraw();
        }
    }

    function kill () {
        selfdestruct(msg.sender);
    }
}
```

Re-entrancy

```
pragma solidity ^0.4.18;

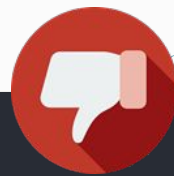
contract Fund {
    /// Mapping of ether shares of the contract.
    mapping(address => uint) shares;
    /// Withdraw your share.
    function withdraw() public {
        var share = shares[msg.sender];
        shares[msg.sender] = 0;
        msg.sender.transfer(share);
    }
}
```



Overflow/Underflow

- Tipos de dados de tamanho fixo para inteiros na EVM
- As variáveis podem ser exploradas para resultar em números fora do intervalo do tipo de dados que os armazena
- Um ***uint8***, por exemplo, só pode armazenar números no intervalo **[0..255]**. Tentar armazenar **256** em um ***uint8*** resultará em **0**

Overflow/Underflow



```
contract OverflowUnderflow {
    uint public zero = 0;
    uint public max = 2**256-1;

    function underflow() public {
        zero -= 1;
    }

    function overflow() public {
        max += 1;
    }
}
```


Overflow/Underflow

SafeMath Library

OpenZeppelin



```
pragma solidity ^0.4.24;

library SafeMath {

    function mul(uint256 a, uint256 b) internal pure returns (uint256 c) {
        if (a == 0) {
            return 0;
        }

        c = a * b;
        assert(c / a == b);
        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return a / b;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256 c) {
        c = a + b;
        assert(c >= a);
        return c;
    }
}
```

Overflow/Underflow



```
contract OverflowUnderflow {  
    using SafeMath for uint  
    uint public zero = 0;  
    uint public max = 2**256-1;  
  
    function underflow() public {  
        zero = zero.sub(1);  
    }  
  
    function overflow() public {  
        max = max.add(1);  
    }  
}
```



Ethereum Networks

- **mainnet** (default) main Ethereum network
- **kovan** or **testnet** the fast Ethereum test network
- **ropsten** the old Ethereum test network
- **classic** Ethereum Classic network
- **classic-testnet** original Morden testnet and current Ethereum Classic testnet
- **expanse** Expanse network
- **dev** a **Private development chain** to be used locally, submitted transactions are inserted into blocks instantly without the need to mine
- **musicoin** Musicoin network
- **ellaism** Ellaism network
- **tobalaba** EWF Tobalaba network

Energy Web Foundation

A central banner with a purple and black background. At the top center is a purple circle containing a white stylized 'W' logo. Below the logo is the main headline in white and purple text. Underneath is a paragraph of white text. The background of the banner shows an aerial view of a city with a grid overlay.

The Energy Web is unleashing blockchain's potential in the energy sector

Energy Web Foundation (EWF) has pioneered an enterprise-grade blockchain platform tailored to the sector's regulatory, operational, and market needs. EWF is the world's largest energy blockchain ecosystem—a growing community of over 100 energy market participants who we work with to bring energy blockchain solutions to market. The Energy Web has become the industry's largest energy blockchain ecosystem and leading choice for decentralized technology powering the world's energy future.



Testnet

- Testnet é uma rede de testes pública do Ethereum
- É gratuita, porém insegura
- Na testnet:
 - Ethers são fáceis de serem obtidos
 - Ethers não possuem valor como criptomoeda

Testnet

Faça **deploy** e **teste** os seus contratos na **testnet** antes de utilizar em produção na **mainnet**.

FOHAT

Frameworks/IDE's



openzeppelin

goo.gl/gJxR16



TRUFFLE

goo.gl/bGonkq



Ganache

goo.gl/ZXqTvNe



Embark

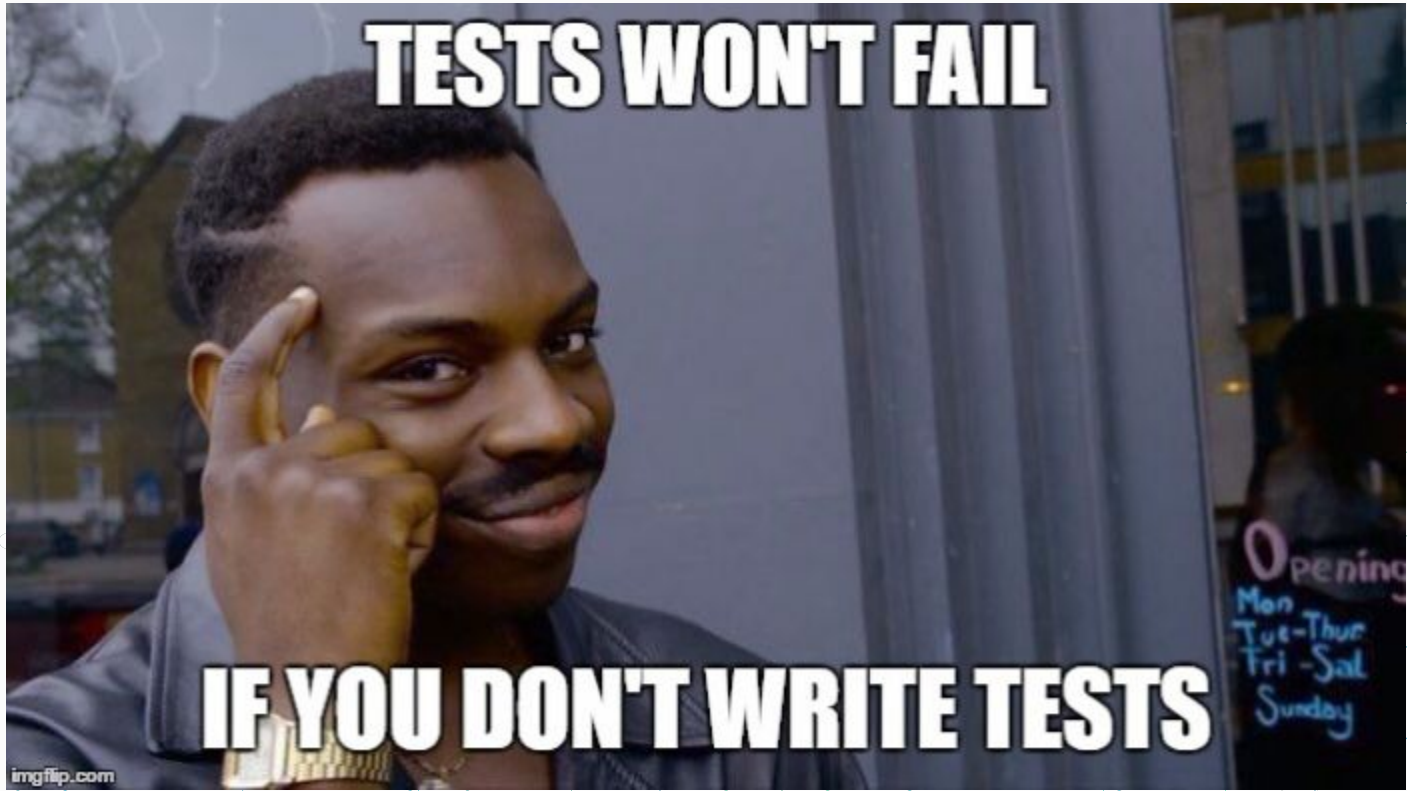
goo.gl/dELuzR



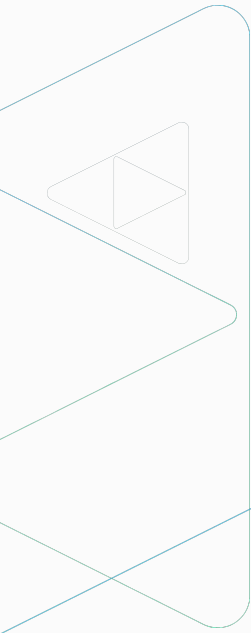
remix

goo.gl/cp6u9h

FOHAT



imgflip.com



Testes



```
import "truffle/Assert.sol";
import "../contracts/MetaCoin.sol";

contract TestMetacoin {
    function testInitialBalanceWithNewMetaCoin() {
        MetaCoin meta = new MetaCoin();

        uint expected = 10000;

        Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
    }
}
```

Testes

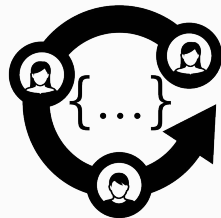
```

$ truffle test
Compiling ConvertLib.sol...
Compiling MetaCoin.sol...
Compiling truffle/Assert.sol
Compiling ../test/TestMetacoin.sol...

TestMetacoin
  ✓ testInitialBalanceWithNewMetaCoin (69ms)

1 passing (2s)
```

Auditoria



Code Review



Mythrill

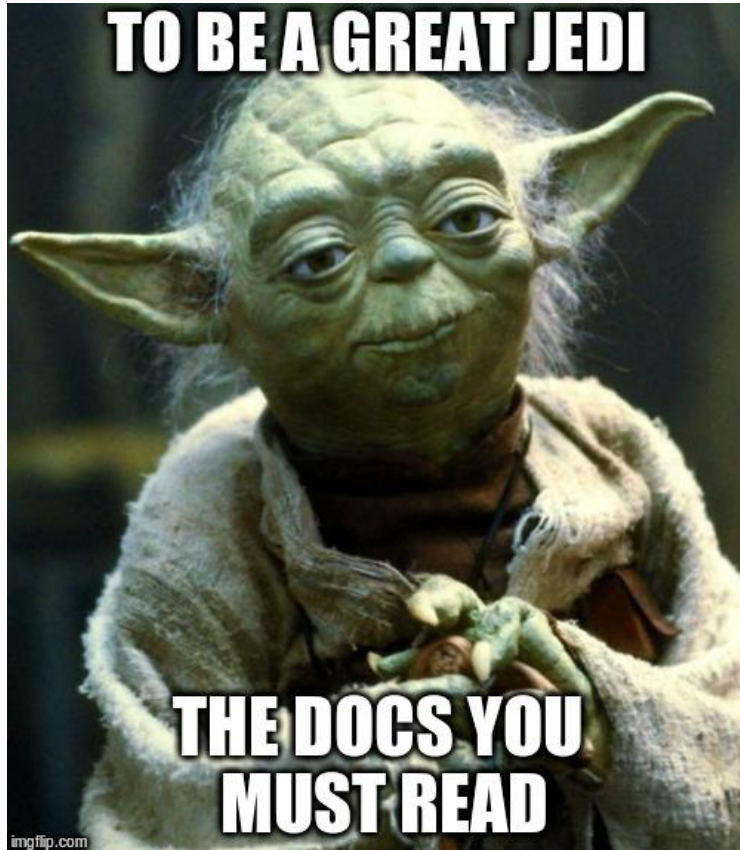


Zeppelin



SOLIDIFIED

FOHAT



- github.com/ethereum/wiki/wiki
- solidity.readthedocs.io
- ethdocs.org

FOHAT

Obrigado!

 andre@fohat.co

 @aferreira44

 @aferreira44

 @aferreira44

 rafael@fohat.co

 @capaci

 @capacirafael

 @rafaelcapaci

Link dos slides:

bit.ly/smart-contracts-tdc-2019

FOHAT

ADVANCED ENERGY SOLUTIONS

www.fohat.co