



**CSBC** 2021  
XLI CONGRESSO DA  
SOCIEDADE BRASILEIRA  
DE COMPUTAÇÃO



40º Jornada de Atualização de Informática

## BLOCKCHAIN E CONTRATOS INTELIGENTES PARA APLICAÇÕES EM IOT, UMA ABORDAGEM PRÁTICA

Jauberth Abijaude  
jauberth@uesc.br

Péricles Sobreira  
pericles.delimasobreira@uqo.ca

Fabiola Greve  
fabiola@ufba.br

1

## AUTORES



**Fabiola Greve**  
Doutora em Informática pela  
Université Rennes I e Laboratórios  
IRISA-INRIA, França  
Professora UFBA



**Jauberth Weyll Abijaude**  
Doutorando em Ciência  
da Computação – UFBA  
Professor UESC



**Péricles Sobreira**  
Doutor em Ciência da  
Computação pela Université  
Grenoble Alpes, França  
Professor University of Quebec at  
Outaouais e Granby College

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

2

2

# BLOCKCHAIN E CONTRATOS INTELIGENTES PARA APLICAÇÕES EM IOT, UMA ABORDAGEM PRÁTICA

## Introdução

Blockchain, arquitetura e protocolos de Consenso

IoT e Aplicações Blockchain-IoT

Plataforma Ethereum e Contratos Inteligentes

Aplicações e Prática

Como montar um curso de Blockchain, IoT e Sistemas Web

Desafios de Pesquisa e Conclusões

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

3

# BLOCKCHAIN

- Vamos apresentar:
  - Blockchain
  - Características
  - Protocolos de Consenso

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

4

4

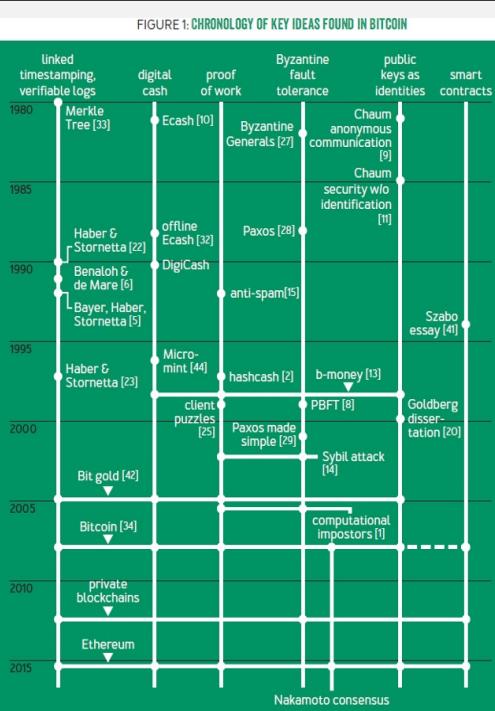
2

# Blockchain: Rede de Confiança Digital



5

5



## HISTÓRICO

- M. E. Hellman e Whitfield Diffie, 1976, [Criptografia de chave pública](#).
- Lamport, 1982, [Consenso dos Generais Byzantinos](#)
- N. Szabo, 1994, [Smart Contracts](#)
- Satoshi Nakamoto, 2008, "Bitcoin: A Peer-to-Peer Electronic Cash System".
- Bitcoin, 2009
- Vitalik Buterin, 2014, "A next generation smart contract and decentralized application platform"
- Ethereum, 2015

6

6

## COMPONENTES DE UMA BLOCKCHAIN

**Rede P2P**

**Criptografia**

**Protocolos de Consenso**

**Máquina de Estados**

**Cadeia de blocos  
(blockchain)**

**Mecanismos de Incentivo**

Blockchain Aberta

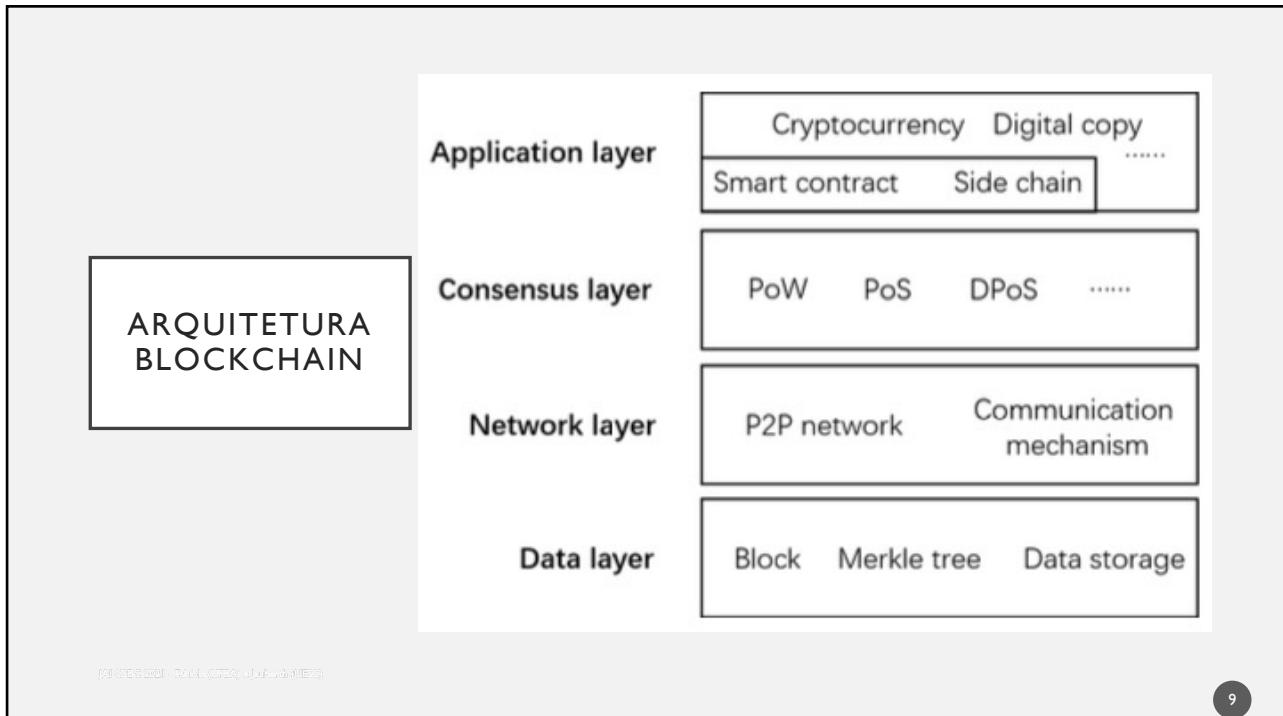
7

7

**PROPRIEDADES DA BLOCKCHAIN**

- Descentralização
- Disponibilidade e Integridade
- Transparência e Auditabilidade
- Imutabilidade e Irrefutabilidade
- Privacidade e Anonimidade
- Desintermediação
- Cooperação e Incentivos

8



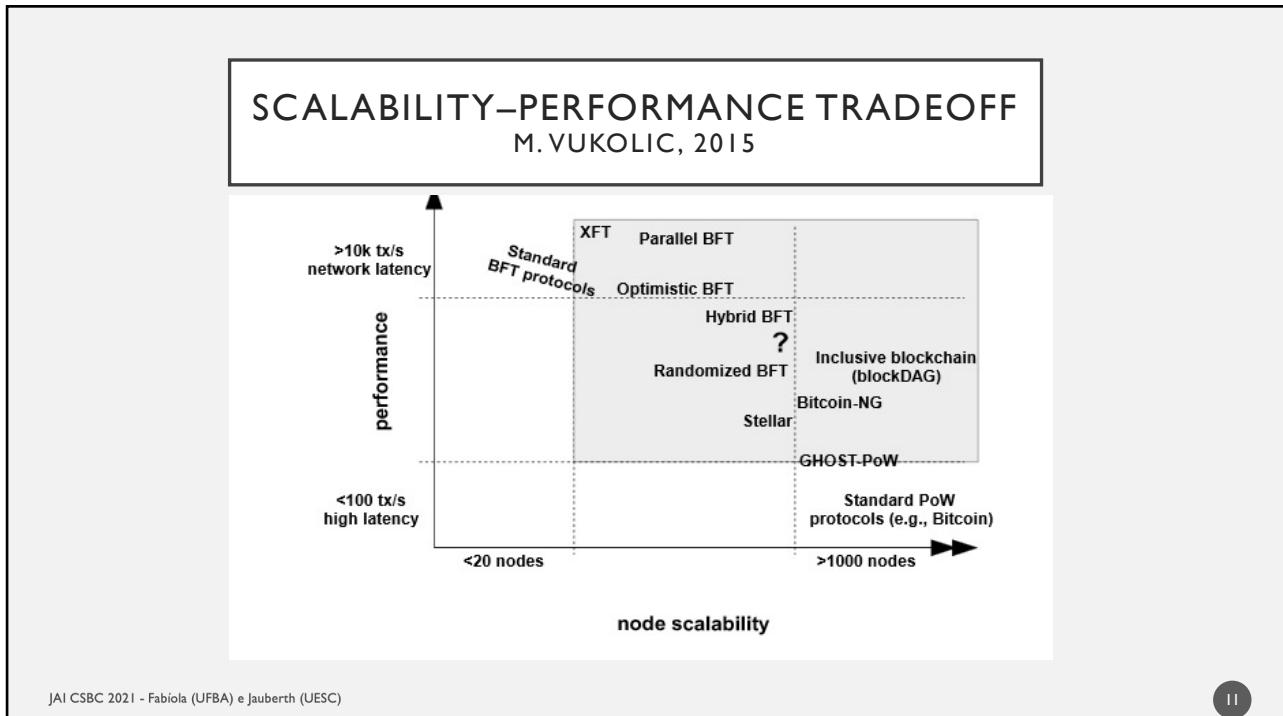
9

9



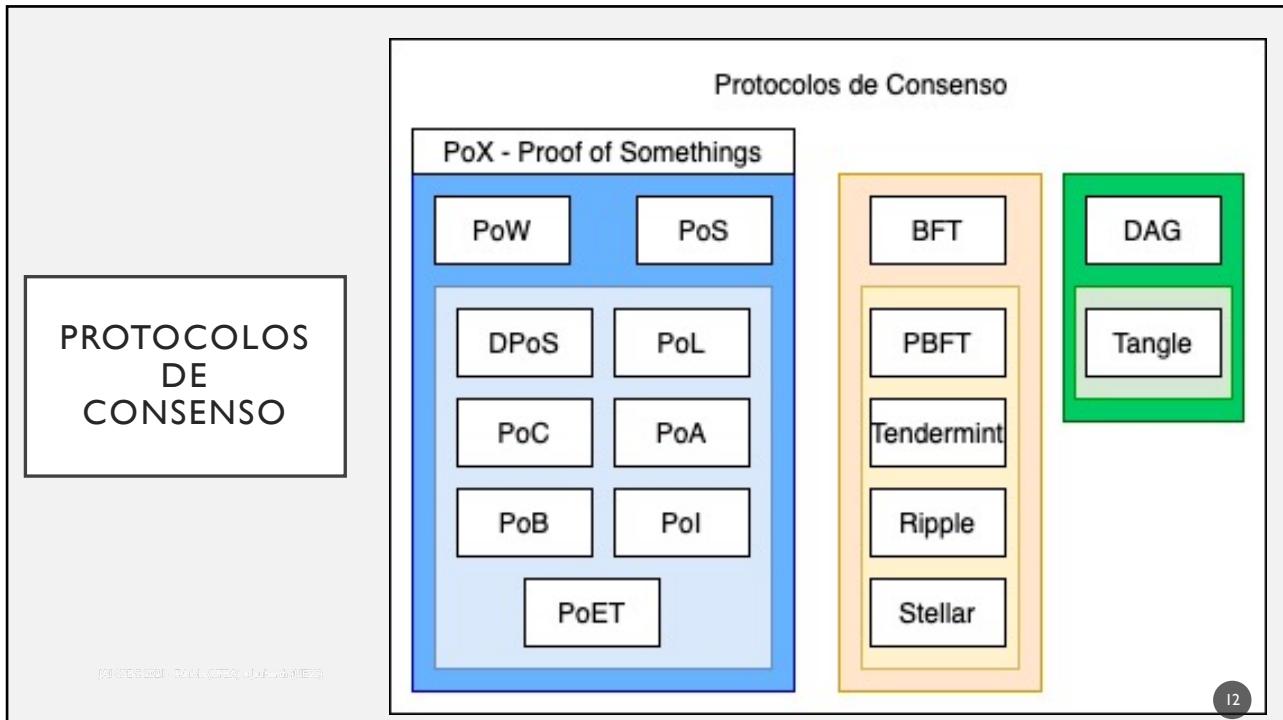
10

10



11

II



12

I2

## POW – PROOF OF WORK

### PROVA DE TRABALHO



Cada nó da rede precisa resolver um desafio computacional para poder propor à rede um bloco de transações

Dois princípios básicos: *a regra da cadeia mais longa e a regra de incentivo.*

Evita-se assim o duplo gasto e ataques de maliciosos

Estas premissas são a base de funcionamento da rede Bitcoin

Hybrid-IoT - Sagirlar et al. [Sagirlar et al. 2018] propõe o uso da blockchain na IoT com intenção de alcançar um sistema IoT baseado em consenso distribuído

Exemplos mais conhecidos: Bitcoin (BTC) e Ethereum 1.0 (ETH)

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

13

## BFT – PROTOCOLOS BYZANTINOS TOLERANTES A FALHAS

### PBFT – PRACTICAL BYZANTINE FAULT TOLERANCE

Garante Acordo num sistema onde os processos podem falhar de forma arbitrária: Problema Gerais Bizantinos [Lamport 1982], se qtdé falhos é menor que  $1/3$

PBFT é o primeiro algoritmo prático a tolerar falhas bizantinas, considerando ambientes assíncronos

O BFT-Smart é oferecido pelo Hyperledger Fabric como camada de acordo e validação

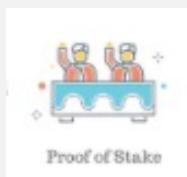
JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

14

14

## POS – PROOF OF STAKE

### PROVA DE PARTICIPAÇÃO



Considera a porcentagem do número total de moedas (ou recursos) que um nó envolvido na competição detém, e eventualmente considera o tempo que o nó leva com o montante de moedas (ou recursos) para estabelecer uma porcentagem de direito de participação no consenso.

Elimina o processo exaustivo de resolução do *puzzle* criptográfico.

Os usuários com mais moedas por um longo período têm maior possibilidade de serem selecionados pelo sistema para gerar o próximo bloco.

Utilizado pelas plataformas Cardano (ADA), Algorand (ALGO) e Ethereum 2.0 (ETH), por exemplo

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

15

## VARIANTES DO POW E POS

### PROOF OF X (PROVA X)

Apesar de eficientes, os protocolos baseados em PoW e PoS tem alguns pontos negativos:

- A alta concentração do controle dos nós que possuem mais recursos
- O alto consumo energético
- As constantes atualizações de hardware

#### Algumas alternativas

- Prova de Participação Delegada - DPoS (*Delegated Proof of Stake*)
- Prova de Sorte - PoL (*Proof of Luck*)
- Prova de Capacidade ou Prova de Espaço - PoC (*Proof of Capacity* ou *Proof of Space*)
- Prova de Autoridade - PoA (*Proof of Authority*)
- Prova de Queima - PoB (*Proof of Burn*)
- Prova de Importância - PoI (*Proof of Importance*)
- Prova de Tempo Decorrido - PoET (*Proof of Elapsed Time*)

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

16

16

## DPOS (DELEGATED PROOF OF STAKE)

Resolve o problema de centralização através da introdução do mecanismo de delegação

Os nós da rede elegem nós especiais que passam a gerar e assinar os blocos

Elimina a necessidade de se aguardar a confirmação de nós não confiáveis.

Principais plataformas: BitShares, Steem e EoS.



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

17

## POC (PROOF OF CAPACITY)

Usa o espaço disponível no disco rígido para definir privilégios ao invés do poder computacional dos nós concorrentes

A probabilidade de propor um bloco é proporcional ao espaço de armazenamento cedido à rede por um nó minerador



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

18

18

## POB (PROOF OF BURN)

Os mineradores devem comprovar que queimaram algumas moedas, enviando-as para alguns endereços onde não podem ser gastos.

A quantidade destas moedas destruídas determina a probabilidade de um minerador emitir um novo bloco



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

19

## POL (PROOF OF LUCK)

O protocolo gera um número aleatório e bloqueia as plataformas que podem ser usadas para processamento das operações como forma de limitar o poder desigual da computação.

Após a geração do número, o PoL escolhe um líder para o consenso, com isto, obtém-se economia no consumo de energia, baixa latência para confirmação de transações e equidade na mineração.



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

20

## POET (*PROOF OF ELAPSED TIME*)

Os nós eleitos estocasticamente aguardam um tempo de espera aleatório criado pelo sistema.

O nó que primeiro esgotar o tempo será eleito o líder para a criação do novo bloco

O PoET é um algoritmo semelhante a uma loteria

Para evitar trapaças, dois requisitos precisam ser verificados:

O primeiro é que o líder realmente espera por um tempo aleatório em vez de um curto período de tempo para vencer.

O segundo é que o líder realmente aguarda pelo tempo determinado pelo protocolo.



JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

21

21

## POA (*PROOF OF AUTHORITY*)

Contam com um conjunto de N nós confiáveis chamados de autoridades, eleitos em um esquema de rotação para chegar a um consenso. Consideram-se honestos pelo menos  $N / 2 + 1$  nós.

Baseado na reputação dos nós e não na quantidade de moedas possuídas

O tempo é dividido em etapas, cada uma das quais tem uma autoridade eleita como líder de mineração.

Principais protocolos: Clique e Aura



JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

22

22

## POI (PROOF OF IMPORTANCE)

A quantidade de moedas que possui e o número de transações realizadas são considerados para medir a capacidade de uma conta minerar um bloco

Também se considera o volume de transação realizada.

Principal Plataforma: NEM



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

23

23

## TENDERMINT (BFT)

Baseado em validadores, propondo blocos de transações e votando neles.

Requer apenas duas rodadas de votação para chegar a um consenso.

Em cada rodada, há três etapas:

Propor  
Prevenir  
Pré-comprometer

Quando mais de 2/3 dos votos pré-comprometidos forem recebidos para alcançar o consenso em uma rodada, o consenso para a próxima rodada começará.



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

24

24

## RIPPLE (BFT)

Utiliza sub-redes confiáveis coletivamente dentro da rede maior para chegar a um Consenso Bizantino

Unique Node List (UNL) é um conjunto de outros servidores mantidos por cada servidor

Apenas os votos dos servidores na UNL são considerados na determinação do consenso

A premissa é que cada servidor confia nos outros servidores da UNL e acredita-se que eles não entrarão em conluio.



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

25

## STELLAR (BFT)

É um protocolo do Acordo Bizantino Federado (FBA) para sistemas com composição aberta (open membership).  
Quóruns para garantia do acordo no consenso podem ser diferentes, a depender da federação que o servidor faz parte.

Primeiro mecanismo de consenso comprovadamente seguro a desfrutar simultaneamente de quatro propriedades principais: controle descentralizado, baixa latência, confiança flexível e segurança assintótica



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

26

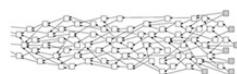
26

## DAG – DIRETAC ACYCLIC GRAPH

### Tangle

- Cada transação fica vinculada aos dois registros de transações anteriores através de um grafo. Desta forma, a conformidade da transação atual pode ser comprovada referenciando-se às transações anteriores

### Tangle



### IOTA

### Blockchain



### Bitcoin Ethereum

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

27

27

## PRINCIPAIS CARACTERÍSTICAS DE PROTOCOLOS DE CONSENSO PARA IOT

- Limitações computacionais, energéticas e de armazenamento
- Ambiente distribuído para validação e consistência
- O Tangle é um dos protocolos de consenso indicados nesse cenário

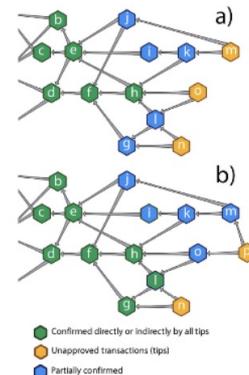
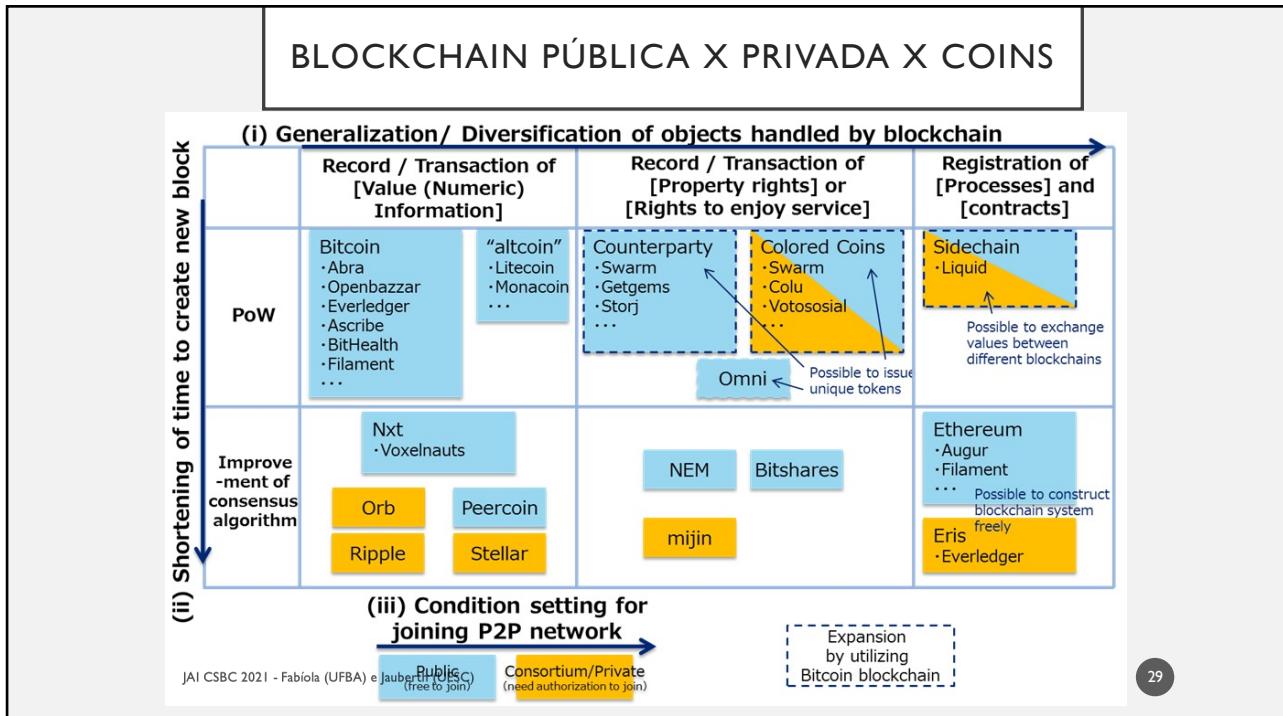


Figura 4.6. Grafo ilustrando um estado do Tangle. Em (a) um estado aleatório. Em (b) um estado após uma transação p. Fonte: [Silvano and Marcelino 2020]

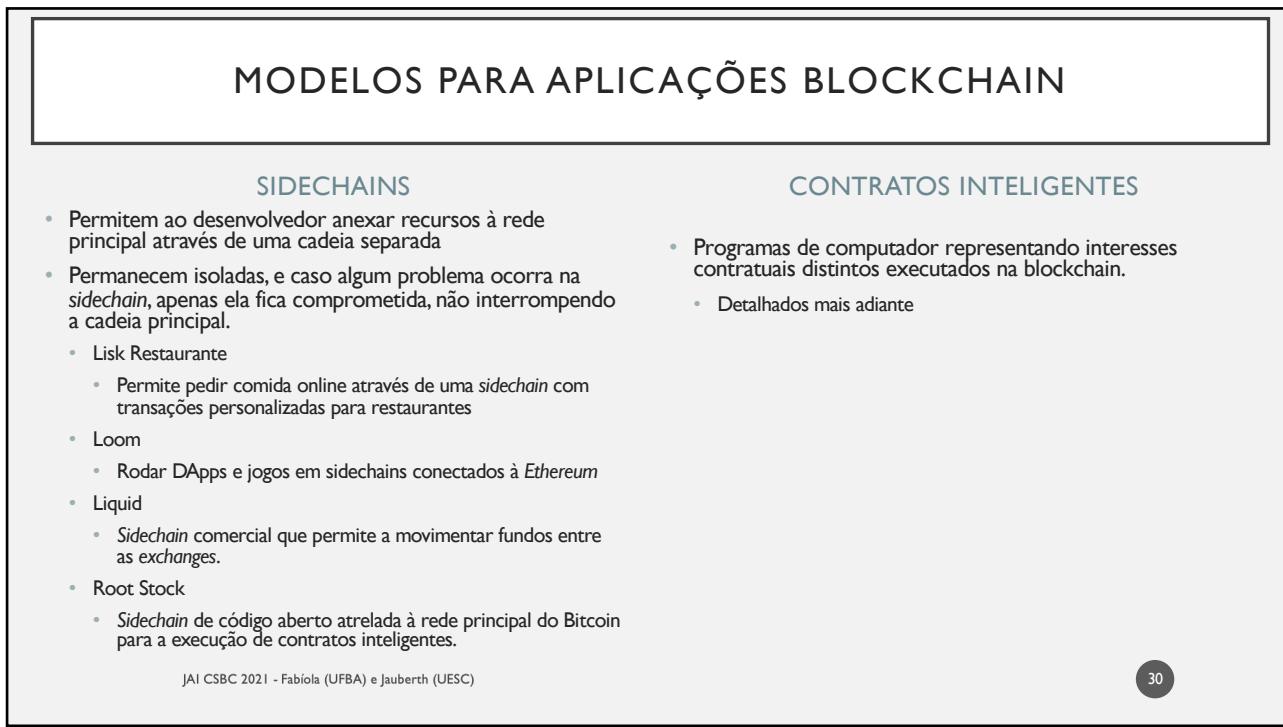
JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

28

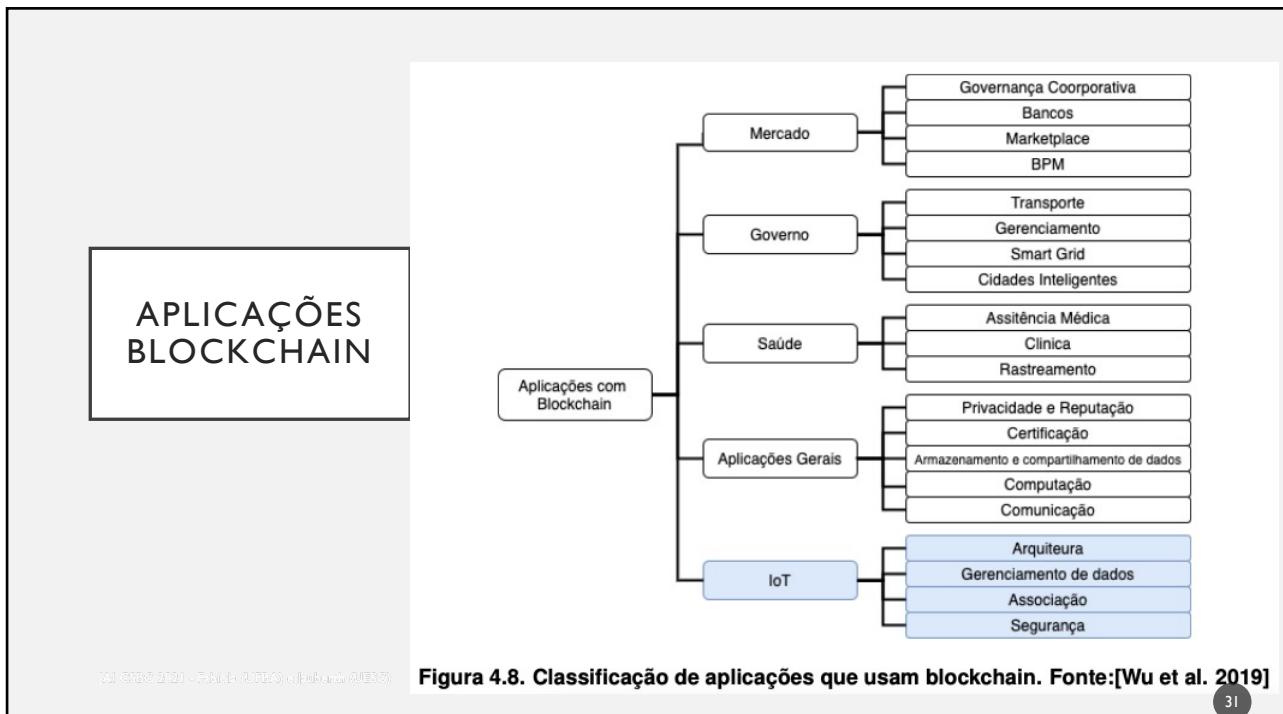
28



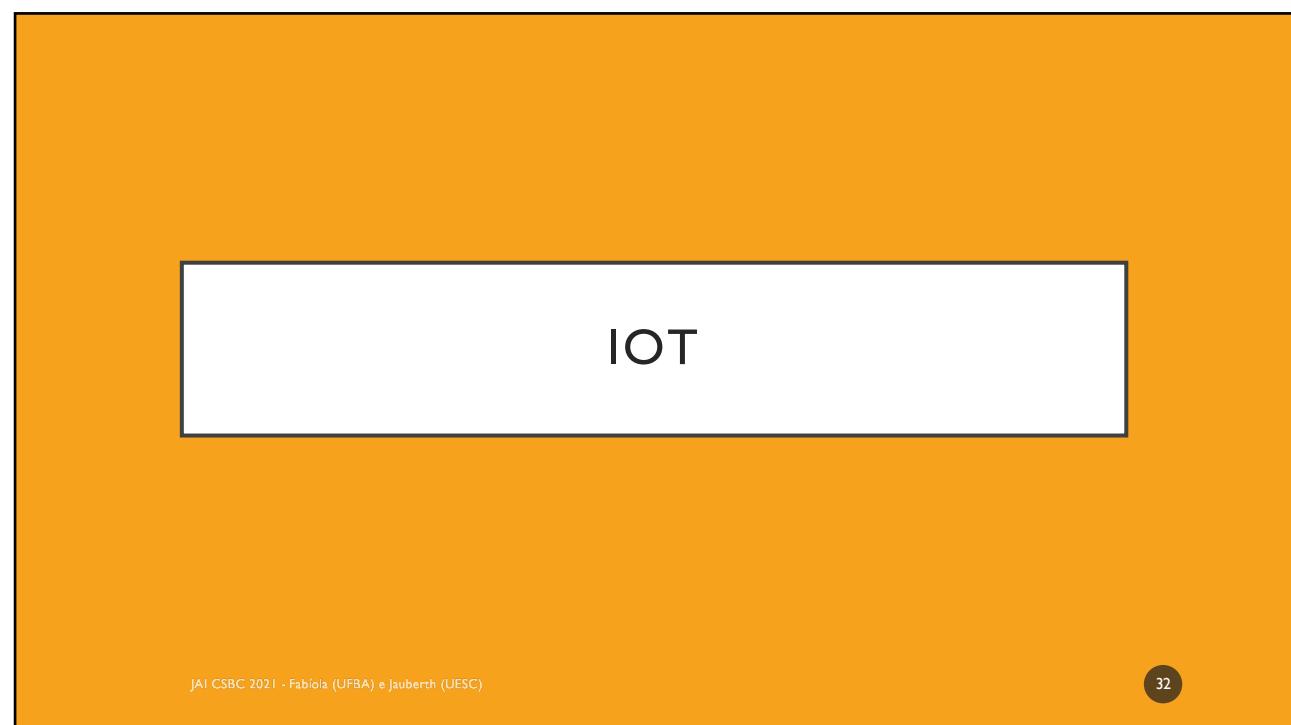
29



30



31



32

## IOT

Vamos apresentar neste seção:

- IoT;
- Características
- Arquitetura para IoT
- Arquitetura e aplicações Blockchain e IoT

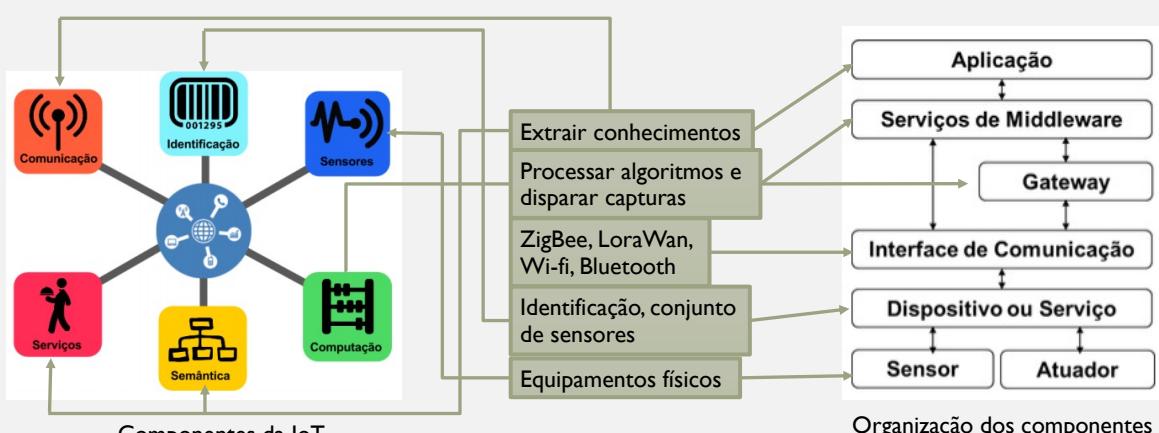
## IOT

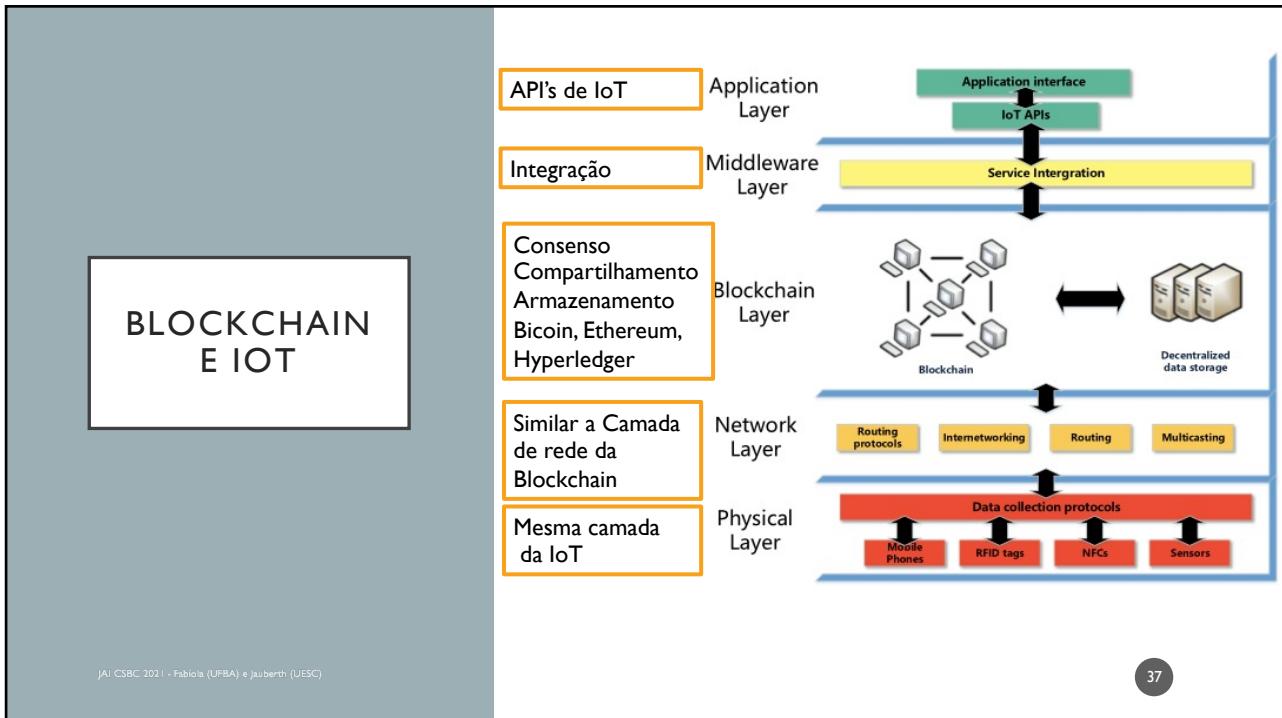
# IOT

Vamos apresentar nesta seção:

- IoT
- Características
- Arquitetura para IoT
- Arquitetura e aplicações Blockchain e IoT

# IOT





37

### EXEMPLOS DE ARQUITETURAS BLOCKCHAIN IOT

**Tabela 4.1. Arquiteturas de Blockchain-IoT. Fonte [Lao et al. 2020]**

| Blockchain-IoT             | Aplicação               | Middleware            | Blockchain                     | Rede                   | Física                    | Classe                  |
|----------------------------|-------------------------|-----------------------|--------------------------------|------------------------|---------------------------|-------------------------|
| Smart Home                 | Smart Home App          | Gerenciamento         | Blockchain comercial           | P2P                    | Dispositivos inteligentes | Aplicação específica    |
| LO3 Energy                 | Energy Shopping         | Energy Token          | Blockchain pública             | Rede de baixa latencia | Painéis solares           | Aplicação específica    |
| Slock.it                   | DApp                    | Não usa               | Ethereum                       | Rede comercial         | Travas eletrônicas        | Aplicativo específico   |
| Hybrid-IoT                 | Aplicação IoT           | Plataforma Hybrid-IoT | POW blockchain, BFT blockchain | P2P                    | Sensores                  | Aplicativo como serviço |
| PBIoT                      | DApp                    | C                     | Blockchain                     | P2P                    | Dispositivos de IoT       | Aplicativo como serviço |
| JD.COM                     | JD.com                  | Blockchain Gateway    | BFT blockchain                 | P2P                    | Dispositivos de IoT       | Aplicativo como serviço |
| IoT Data Service Framework | Aplicação para Usuários | Framework             | Ethereum                       | P2P                    | Dispositivos de IoT       | Aplicativo como serviço |
| IoT Chain                  | Acesso com Autorização  | Framework             | Ethereum                       | Rede Comercial         | Dispositivos de IoT       | Aplicativo como serviço |

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

38

## APlicações Blockchain e IoT

- Segundo [Lao et al. 2020] as aplicações de blockchain e IoT podem ser categorizadas:

### Pagamentos Digitais

Atualmente blockchains como Bitcoin e Ethereum podem ser utilizadas em smartphones



### Serviços de Contratos Inteligentes

Sistemas de automação e controle com base na IoT, eliminando a terceira parte de confiança [Christidis and Devetsikiotis 2016]



### Armazenamento

Aplicativos de armazenamento de dados que vêm a blockchain como um banco de dados seguro e distribuído



**FACTOM**

## PLATAFORMA ETHEREUM E CONTRATOS INTELIGENTES

## PLATAFORMA ETHEREUM E CONTRATOS INTELIGENTES

- Plataforma Ethereum
  - Redes Ethereum
  - Transações
  - Consenso
  - Contratos Inteligentes

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

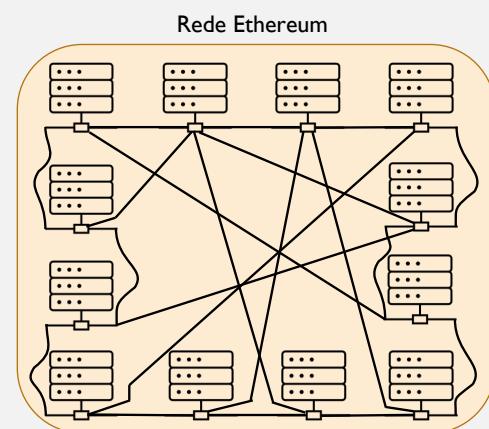
41

41

## REDE ETHEREUM

### Características

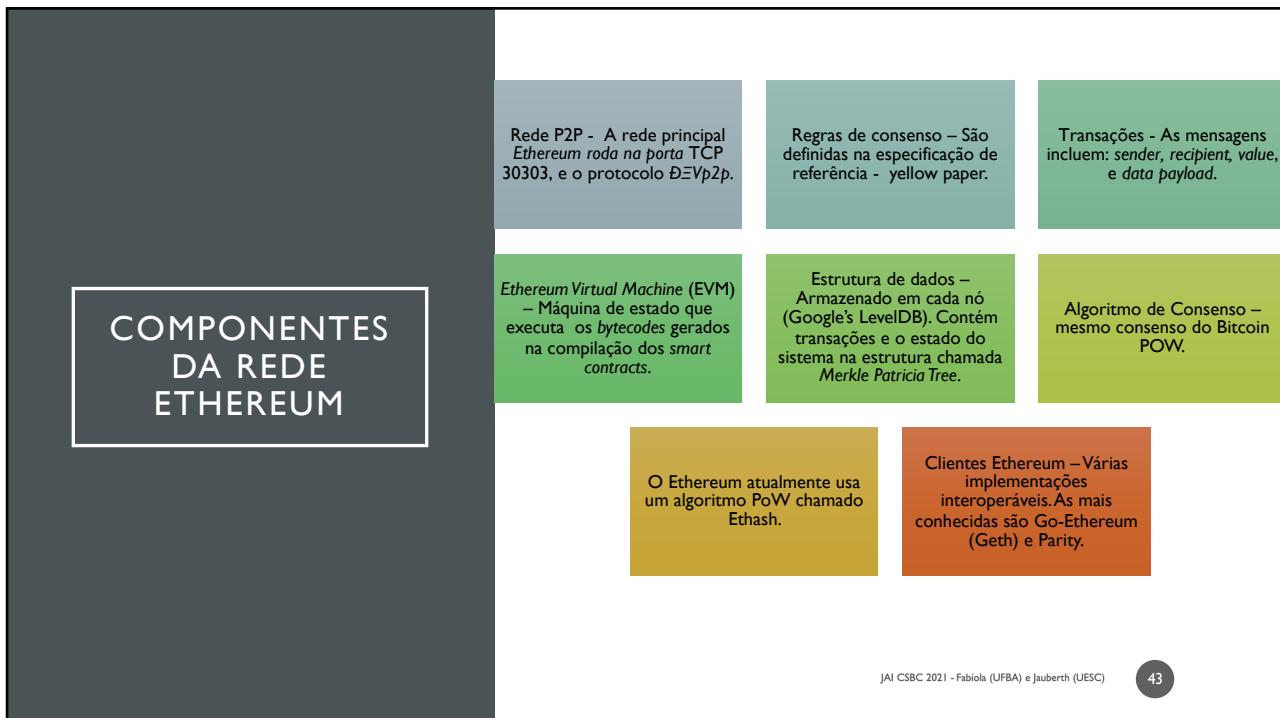
- Usada para transferência de dinheiro digital
- Existe mais de uma rede Ethereum
- Executa programas chamados de smart contracts
- Cada nó é uma máquina que possui um cliente Ethereum rodando
- Qualquer um pode rodar um nó – associação de mineradores
- Cada nó pode conter uma cópia completa da blockchain



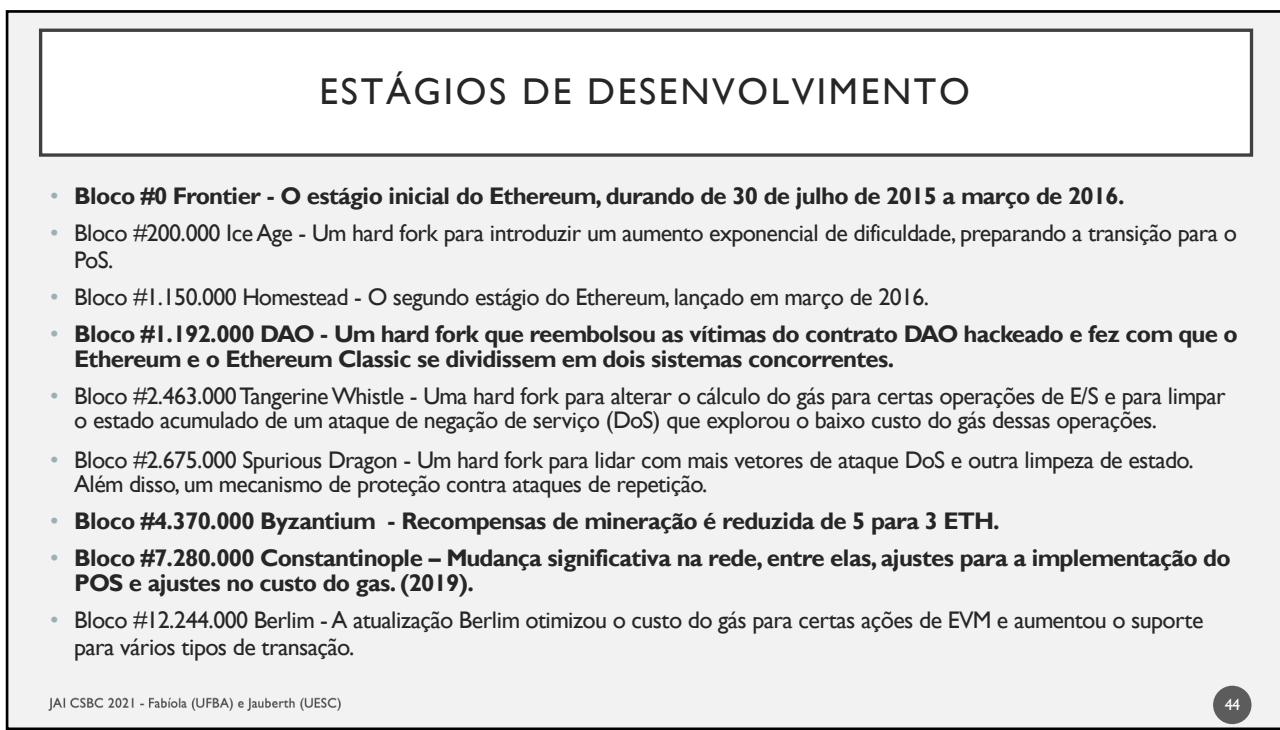
JAI CSBC Brasileiro de Sistemas de Jaqueira (UESC) - Módulo Blockchain, Contratos Inteligentes e Sistemas Web: Teoria e Prática

42

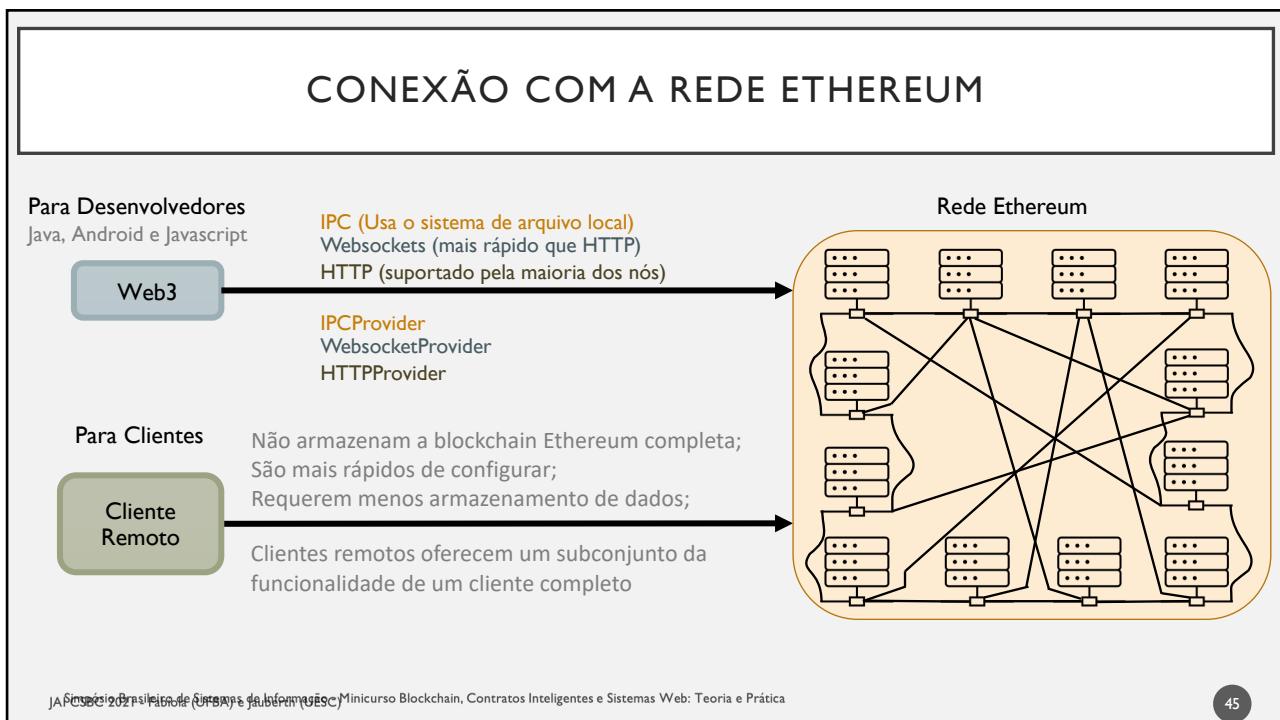
42



43



44



45

45

| CLIENTES ETHEREUM   |  |  |
|---|--|--|
| <b>Nós Clientes</b> <ul style="list-style-type: none"> <li>• Geth (Go)</li> <li>• Parity (Rust)</li> <li>• Cpp-ethereum(C++)</li> <li>• Pyethereum(Python)</li> <li>• Mantis(Scala)</li> <li>• Harmony(Java)</li> </ul> | <b>Clientes remotos</b> <ul style="list-style-type: none"> <li>• Jaxx</li> <li>• Status</li> <li>• Trust Wallet</li> <li>• Coinbase</li> </ul> | <b>Extensões</b> <ul style="list-style-type: none"> <li>• Metamask</li> <li>• Jaxx</li> <li>• MyEtherWallet</li> <li>• Nifty Wallet</li> <li>• MyCrypto</li> </ul> |
| JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)  |  |  |

46

46

## CLIENTES REMOTOS

Gerenciar chaves privadas e endereços Ethereum em uma carteira;

Criar, assinar e transmitir transações;

Interagir com contratos inteligentes, usando a carga útil de dados;

Navegar e interagir com DApps;

Oferecer links para serviços externos, como exploradores de blocos;

Converter unidades de ether e recuperar taxas de câmbio de fontes externas;

Injetar uma instância web3 no navegador web como um objeto Javascript;

Usar uma instância web3 fornecida/injetada no navegador por outro cliente;

Acessar os serviços RPC em um nó Ethereum local ou remoto.

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

47

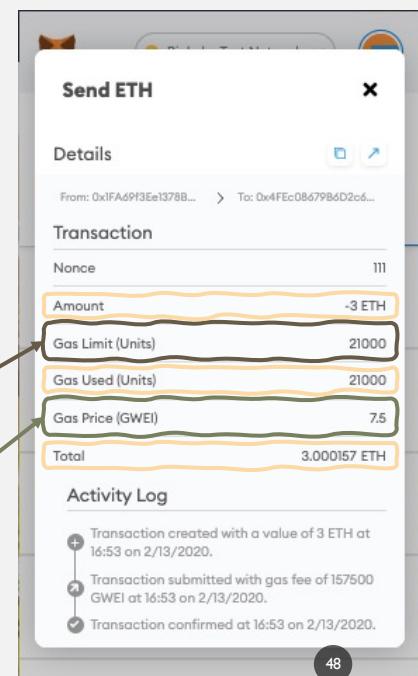
47

## GASPRICE X GASLIMIT

- O gas não é ether, é uma moeda separada usada para pagamento das transações na rede
- Possui cotação própria em relação ao ether

**GasLimit** - indica qual o limite de gas a ser consumido pela transação.

**GasPrice** - permite que o emissor da transação defina o preço que está disposto a pagar para adquirir o gas.



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

48

48

## EVM – ETHEREUM VIRTUAL MACHINE

Implementa uma máquina de turing virtual para processar os contratos Inteligentes.

Usa o `gas` para controlar o uso da Máquina de Turing

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

49

## REDES ETHEREUM

**METAMASK**

**Rede Principal** Ethers comprados com dólar

**Redes de Teste**

- Ropsten
- Rinkeby
- KOVAN
- localhost 8545** A opção conecta-se a um nó em execução no mesmo computador que o navegador.

**Custom RPC** Conecta a qualquer nó com uma interface de Chamada de Procedimento Remoto (RPC) compatível com Geth

**Ethers sem valor financeiro**

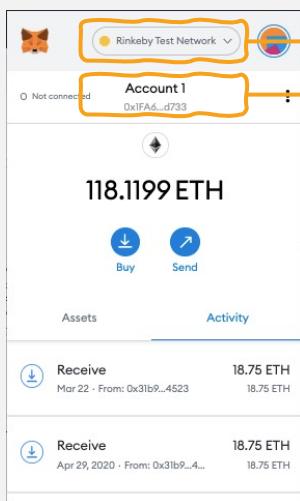
Única conta, Todas as redes.

0x1FA69f3Ee137fB2159bb  
A852F6f29Cb53e11d733

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

50

## METAMASK



- Rede ativa
- Conta ativa
- Ele fornece acesso a todas as redes da plataforma com uma única conta.
- Instalado como extensão nos principais navegadores
- Ao instalar, você receberá 12 palavras mnemônicas (Guarda-sob sigilo)
  - A ordem apresentada é importante
  - Usadas para recuperar a sua conta
  - Realizar transações usando a web3
- O procedimento para instalação do Metamask pode ser acessado na página do curso

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

51

51

## ESTRUTURA DA TRANSAÇÃO

Acesse <https://andersbrownworth.com/> para ver um exemplo

| Campo    | Descrição  |
|----------|--|
| nonce    | Número de sequência, emitido pela origem, usado para evitar a reprodução da mensagem |
| to       | Endereço para onde será enviada uma quantidade de ether                              |
| value    | Quantidade de ether a ser enviada  |
| GasPrice | Valor que será pago por unidade de gas para que a transação seja processada          |
| GasLimit | Quantidade de GAS que esta transação pode consumir                                   |
| v        | Peças criptográficas usadas para a validação do bloco.                               |
| r        |  |
| s        |  |

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

52

52

## DATA E VALUE

Apenas consome gas

```
nonce: 7
to: 0x1FA69f3Ee1378B1159b0A852F6f29Cb53e11d833
value:
gasPrice: 12,3
gasLimit: 30000
data:
```

Indica uma invocação

```
nonce: 7
to: 0x1FA69f3Ee1378B1159b0A852F6f29Cb53e11d833
value:
gasPrice: 12,3
gasLimit: 30000
data: batimentos(80)
```

Indica um pagamento

```
nonce: 7
to: 0x1FA69f3Ee1378B1159b0A852F6f29Cb53e11d833
value: 2,3
gasPrice: 12,3
gasLimit: 30000
data:
```

Indica uma invocação e um pagamento

```
nonce: 7
to: 0x1FA69f3Ee1378B1159b0A852F6f29Cb53e11d833
value: 0,5
gasPrice: 12,3
gasLimit: 30000
data: pag_exame()
```

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

55

55

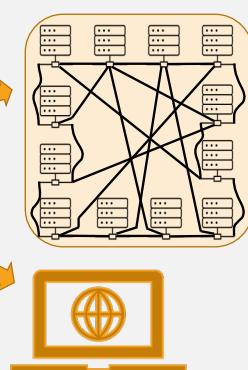
## CONTRATOS INTELIGENTES

Idealizados por Nick Szabo representam "um conjunto de promessas, especificado em formato digital, incluindo protocolos nos quais as partes cumprem estas promessas"

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.7.4;
3
4 contract Inbox{
5     string public message;
6     constructor(string memory initialMessage){
7         message = initialMessage;
8     }
9     function setMessage(string memory newMessage) public{
10        message = newMessage;
11    }
12    function getMessage() public view returns(string memory){
13        return message;
14    }
15 }
```

Bytecodes  
ABI

Rede Ethereum



JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

56

56

## CONTRATOS INTELIGENTES

Cl's são simplesmente programas de computador. A palavra contrato não tem significado legal neste contexto.

Eles são imutáveis, por que uma vez implementado em uma rede Ethereum, o código não pode ser alterado nem substituído

É preciso prever uma função de autodestruição para apagá-lo

São determinísticos, pois o resultado de sua execução é sempre o mesmo para todos os que o executam, conservando-se o contexto no momento da execução.

Linguagens de programação: são *LLL*, *Serpent*, *Vyper*, *Bambu* e *Solidity*

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

57

57

## CATEGORIAS DE CONTRATOS INTELIGENTES

Segundo [Bartoletti and Pompianu 2017], os contratos podem ser classificados em cinco categorias

### Financeiro

- Gerenciam, reúnem ou distribuem dinheiro como característica principal.
  - DAO
  - Ethersic
  - Pixelmap

### Notário

- Exploram a imutabilidade da blockchain para certificar a propriedade e procedência.

### Jogo

- implementam jogos de azar

### Carteira

- trata de chaves, envia transações, gerencia dinheiro, implanta e monitora contratos

### Biblioteca

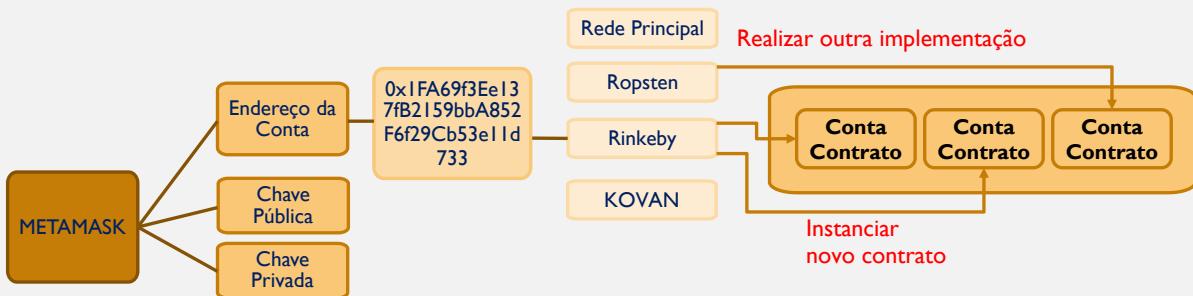
- implementam operações de propósito geral

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

58

58

## CONTRATOS INTELIGENTES



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

59

## TIPOS DE DADOS

| Tipo                 | Descrição   |
|----------------------|---|
| int                  | Inteiros positivos ou negativos (int) declarados em incrementos de 8 bits (int8, int16, ... int256).  |
| uint                 | Inteiros positivos declarados em incrementos de 8 bits (uint8, uint16... uint256).  |
| bool                 | Valor lógico, verdadeiro ou falso, com operadores lógicos ! (não), && (e),    (ou), == (igual) e != (diferente).  |
| fixed/<br>ufixed     | Números de ponto fixo, declarados com (u) fixedMxN em que M é o tamanho em bits (incrementos de 8 até 256) e N é o número de decimais após o ponto (até 18); por exemplo, ufixed32x2.           |
| address              | Usado para armazenar endereços Ethereum de 20 bytes. O objeto de endereço tem muitas funções membro úteis, como balance (retorna o saldo da conta) e transfer (transfere ether para uma conta). |
| byte array (fixed)   | Matrizes de bytes de tamanho fixo, declaradas com bytes.  |
| byte array (dynamic) | Matrizes de bytes de tamanho variável, declaradas com bytes ou string.  |
| enum                 | Tipo definido pelo usuário para enumerar valores discretos enum name {rotulo1, rotulo2...}.   |
| array                | Um array de qualquer tipo, fixo ou dinâmico.  |
| struc                | Containers de dados definidos pelo usuário para agrupar variáveis struct Car { String year; int color; }.   |
| Mapping              | Tabelas de pesquisa de hash para pares chave => mapping (key_type=>value_type).   |

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

60

60

## LINGUAGEM SOLIDITY

- Quando criamos uma transação e a enviamos para a rede, algumas informações podem ser encapsuladas na mensagem. Estas opções são pré-definidas pelo *Solidity* e agrupadas em 3 categorias:

### Objeto msg :

- msg.sender: endereço que iniciou a chamada de contrato
- msg.value: O valor de *ether* enviado (em *wei*).
- msg.gas: A quantidade de *gas* restante no suprimento de *gas* desse ambiente de execução. Isso foi descontinuado no *Solidity* v0.4.21 e substituído pela função *gasleft()*.
- msg.data: A carga útil de dados desta chamada no contrato.

### Objeto tx :

- tx.gasprice: o preço do gas na transação de chamada.
- tx.origin: O endereço da conta Ethereum de origem para esta transação.

### OBJETO block :

- block.blockhash(blockNumber): O *hash* de um bloco específico. Em desuso e substituído pela função *blockhash()* no *Solidity* v0.4.22.
- block.coinbase: O endereço do destinatário das taxas do bloco atual e da recompensa do bloco.
- block.difficulty: A dificuldade (prova de trabalho) do bloco atual.
- block.gaslimit: A quantidade máxima de *gas* que pode ser gasta em todas as transações.
- block.number: O número do bloco atual.
- block.timestamp: O carimbo de data/hora colocado no bloco atual pelo minerador.

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

61

61

## MANIPULANDO ENDEREÇO DOS CONTRATOS

### address.balance:

- O saldo do endereço, em *wei*.

### address.transfer(quantidade)

- Transfere o valor (em *wei*) para este endereço.

### address.send(quantidade)

- Semelhante ao *transfer*, ele retorna falso em caso de erro.

### address.call(payload):

- pode construir uma chamada de mensagem arbitrária com uma carga de dados.

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

62

62

**FUNÇÕES CONSTRUTORAS, MODIFICADORAS E AUTODESTRUTIVAS**

**Construtoras**

- São executadas apenas uma vez durante a criação do contrato e possuem a palavra chave `constructor()`

**Modificadoras**

- São usadas para criar condições que se aplicam a muitas situações em um contrato. Desviam temporariamente a execução de uma função para uma rotina particular e possuem a palavra chave `modifier()`

**Autodestrutivas**

- As funções de autodestruição possuem a palavra-chave `destroy()` e são utilizadas para destruir o contrato implementado

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

64

64

## CONSTRUINDO O PRIMEIRO CONTRATO USANDO O REMIX

• Manipular mensagens

- Uma variável pública para receber mensagens
- Uma função para definir uma mensagem inicial
- Uma função para modificar a mensagem inicial

**Editor Remix**

**Código do Contrato**

**Compilação**

**Implementação**

**Emulação de Browser**

**Instância do Contrato**

```

SOLIDITY COMPILER
COMPILE IR
0.7.4+commit.3f05b7f0
INCLUDE nightly builds
LANGUAGE Solidity
VM VERSION compiler default
COMPILE CONFIGURATION Auto compile
Enable optimization 200
Hide warnings Compile inbox.sol
CONTACT Inbox (inbox.sol)
Publish on Swarm Publish on IPFS Compilation Details

```

```

18 // SPDX-License-Identifier: MIT
19 pragma solidity >=0.7.4;
20
21 /**
22  * Declara o nome do Contrato e as variáveis
23  * necessárias para armazenar a mensagem
24 */
25 contract Inbox {
26     string message;
27 }
28
29 /**
30  * Declara a Função construtora
31  * que é executada quando o contrato é criado
32 */
33 constructor(string memory initialValue) {
34     message = initialValue;
35 }
36
37 /**
38  * Declara uma função para modificar o valor da variável message
39 */
40 function setMessage(string memory newValue) public {
41     message = newValue;
42 }
43
44 /**
45  * Declara uma função para recuperar o valor da variável message
46 */
47 function getMessage() public view returns(string memory) {
48     return message;
49 }
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

The Remix interface shows the Solidity code for a contract named 'Inbox'. It includes a constructor that initializes the 'message' variable with an initial value. There are two functions: 'setMessage' which allows changing the message, and 'getMessage' which returns the current message. The Remix interface also shows the compiled IR (Intermediate Representation) and various deployment options like Swarm and IPFS.

<https://remix.ethereum.org/>

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

65

65

## INBOX.SOL

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.7.4;
3
4 contract Inbox{
5     string public message;
6     constructor(string memory initialMessage){
7         message = initialMessage;
8     }
9     function setMessage(string memory newMessage) public{
10        message = newMessage;
11    }
12    function getMessage() public view returns(string memory){
13        return message;
14    }
15 }
```

The diagram illustrates the structure of the INBOX.SOL smart contract. It shows the code on the left and its corresponding annotations on the right:

- Line 1: `// SPDX-License-Identifier: MIT` → Identificador da licença do contrato
- Line 2: `pragma solidity ^0.7.4;` → Informa a versão do compilador
- Line 4: `contract Inbox{` → Informa o nome do contrato e declara as variáveis
- Line 5: `string public message;` → Declara o construtor
- Line 6: `constructor(string memory initialMessage){` → Declara a função para modificar o valor da variável
- Line 7: `message = initialMessage;` → Declara a função para ler o valor da variável
- Line 9: `function setMessage(string memory newMessage) public{` → Declara a função para modificar o valor da variável
- Line 12: `function getMessage() public view returns(string memory){` → Declara a função para ler o valor da variável
- Line 13: `return message;` → Declara a função para ler o valor da variável
- Line 15: `}` → Declara a função para ler o valor da variável

Os contratos somente executam funções se elas forem chamadas por uma transação.

Os contratos nunca podem chamar a si próprios ou atuarem em *background*, mas podem chamar outros contratos em cadeia.

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

66

66

## DESENVOLVIMENTO DE CONTRATOS INTELIGENTES

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

69

69

## DESENVOLVIMENTO DE CONTRATOS INTELIGENTES

- Página do minicurso
- Ambientes de desenvolvimento
- Construindo um contrato inteligente

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

70

70

## PÁGINA DO MINICURSO

- <https://github.com/lifuesc/jai2021>
- Códigos, instruções e material complementar
- Sempre em atualização

The screenshot shows a GitHub repository page for 'jai2021'. The repository has the following details:

- README.md** content: Adicionando práticas e tutoriais, .gitignore, LICENSE, README.md.
- Commit History:**
  - tutoriais (2 days ago)
  - .gitignore (2 days ago)
  - LICENSE (2 months ago)
  - README.md (2 days ago)
- Contributors:** 2
- Forks:** 0
- Stars:** 3
- Issues:** 6 OPEN
- LICENSE:** MIT

**CSBC 2021**  
**JAI**

Blockchain e Contratos Inteligentes para Aplicações em IoT, uma Abordagem Prática

**Autores**

- Jauberth W. Abijaude (UFBA, UESC)
- Henrique Serra (UFSC)
- Péricles de Lima Sobreira (University of Quebec Outaouais, Cégep de Saint-Hyacinthe)
- Fabiola Greve (UFBA)

A Internet das Coisas agrupa dispositivos capazes de capturar informações e interferir no ambiente, de maneira a obter, gerar e enviar dados em larga escala para sistemas de domínios de aplicações diferentes, tais como agricultura, indústria, comércio e governos. Estes sistemas precisam de uma camada de segurança para garantir, dentre outras características, a irrefutabilidade das transações e a integridade dos dados manipulados. Neste sentido, a integração com a blockchain, através dos contratos inteligentes, atenderia a esta necessidade. A blockchain é uma tecnologia disruptiva que oferece uma rede de confiança digital para a realização de transações entre pares, muitas vezes desconhecidos. Este capítulo apresenta

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

71

71

## REMIX X AMBIENTE LOCAL

**REMIX**

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.7.4;
3
4 contract Inbox{
5     string public message;
6     constructor(string memory initialMessage){
7         message = initialMessage;
8     }
9     function setMessage(string memory newMessage) public{
10        message = newMessage;
11    }
12    function getMessage() public view returns(string memory){
13        return message;
14    }
15 }
```

**AMBIENTE LOCAL**

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

72

72

## CONTRATO SENSOR

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.4;
3 pragma abicoder v2;
4 // Estrutura da configuração de medição
5 struct MeasureSetting{
6     string idv;
7     uint8 decimalPlace;
8     uint defaultValue;
9     uint max;
10    uint min;
11 }
12 // Estrutura de medição
13 struct Measure{
14     uint[] value;
15     uint64 timestamp;
16 }
17 contract Sensor{
18     // Nome do sensor
19     string public name;
20     // Aceita diversas configurações
21     MeasureSetting[] settings;
22     // Dados de medições
23     Measure[] measures;
```

```

24 // Endereço do proprietário do contrato
25 address public owner;
26 // Cria a configuração inicial do sensor
27 constructor(
28     string memory _name,
29     MeasureSetting[] memory _settings
30 ) {
31     require(_settings.length > 0, "Settings empty");
32     owner = msg.sender;
33     name = _name;
34     for(uint8 i; i < _settings.length;i++){
35         settings.push(_settings[i]);
36     }
37 }
38 // Cadastra as medições
39 function insertMeasure(Measure[] memory newMeasure) public {
40     for(uint i; i < newMeasure.length; i++){
41         require(newMeasure[i].value.length == settings.length, "SettingsSize");
42         measures.push(newMeasure[i]);
43     }
44 }
45 // Recupera todas medições
46 function getAllMeasure() public view returns (Measure[] memory measure) {
```

73

## COMPILE.JS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

```

1  const path = require("path");
2  const fs = require("fs");
3  const solc = require("solc");
4  const sensorFile = "Sensor.sol";
5  // Pega o diretório do arquivo do sensor
6  const SensorPath = path.resolve(__dirname, "contracts", sensorFile);
7  const datas = [
8    {
9      file: sensorFile,
10     name: "Sensor",
11   },
12 ];
13 const sourceSensor = fs.readFileSync(SensorPath, "utf8");
14 // Configuração do SOLC
15 var input = {
16   language: "Solidity",
17   sources: {
18     [sensorFile]: {
19       content: sourceSensor,
20     },
21   },
22   settings: {
23     outputSelection: {
24       "*": {
25         "*": ["*"],
26       },
27     },
28   },
29 }

```

74

## DEPLOY.JS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

```

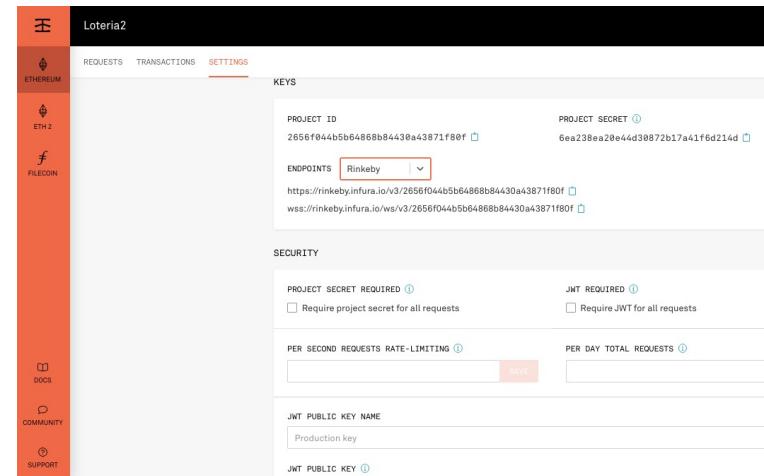
1  require("dotenv").config();
2  const HDWalletProvider = require("@truffle/hdwallet-provider");
3  const Web3 = require("web3");
4  const { abi, evm } = require("./compile");
5  // Configuração do provider do infura
6  const provider = new HDWalletProvider({
7    mnemonic: { phrase: process.env.mnemonic },
8    providerOrUrl:
9      "https://rinkeby.infura.io/v3/62f5fcfdebcc4042b082bdf179fc641c",
10  });
11 const web3 = new Web3(provider);
12 const deploy = async () => {
13   // Pega as contas do navegador
14   const accounts = await web3.eth.getAccounts();
15   // Pega a primeira conta que será utilizada no deploy
16   const deploymentAccount = accounts[0];
17   // Gera a chave primária a partir da carteira
18   const privateKey =
19     provider.wallets.accounts[0].toLowerCase().privateKey.toString("hex");
20   // Mostra carteira utilizada para deploy
21   console.log("Conta usada para o deploy ", accounts[0]);
22   try {
23     // Prepara uma instância do contrato para assinatura
24     let contract = await new web3.eth.Contract(abi)
25       .deploy({
26         data: evm.bytecode.object,
27         arguments: [
28           "dht11",
29           [
30             ["temperatura", 2, 0, 0, 0],
31             ["umidade", 2, 0, 0, 0],
32           ],
33         ],
34       })
35     const tx = await contract.deployed();
36     console.log(`Contrato ${tx.address} criado com sucesso!`);
37   } catch (err) {
38     console.error(err);
39   }
40 };
41
42 deploy();

```

75

75

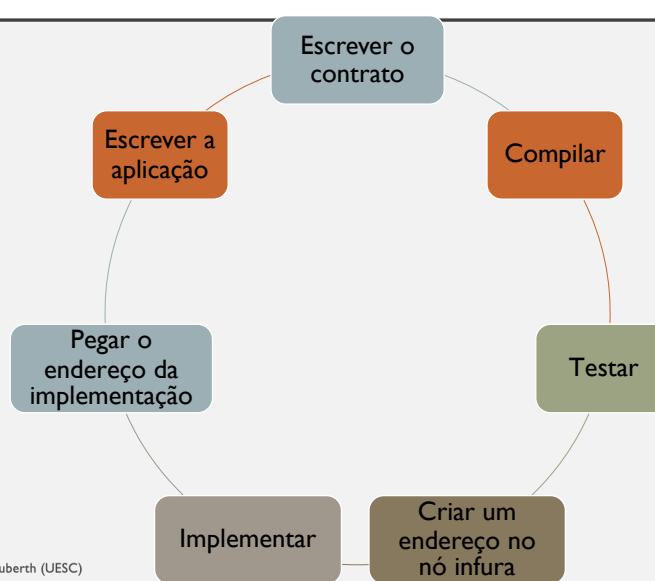
## ACIONANDO UM PROJETO



76

76

## CICLO PARA DESENVOLVER UM CONTRATO



JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

77

77

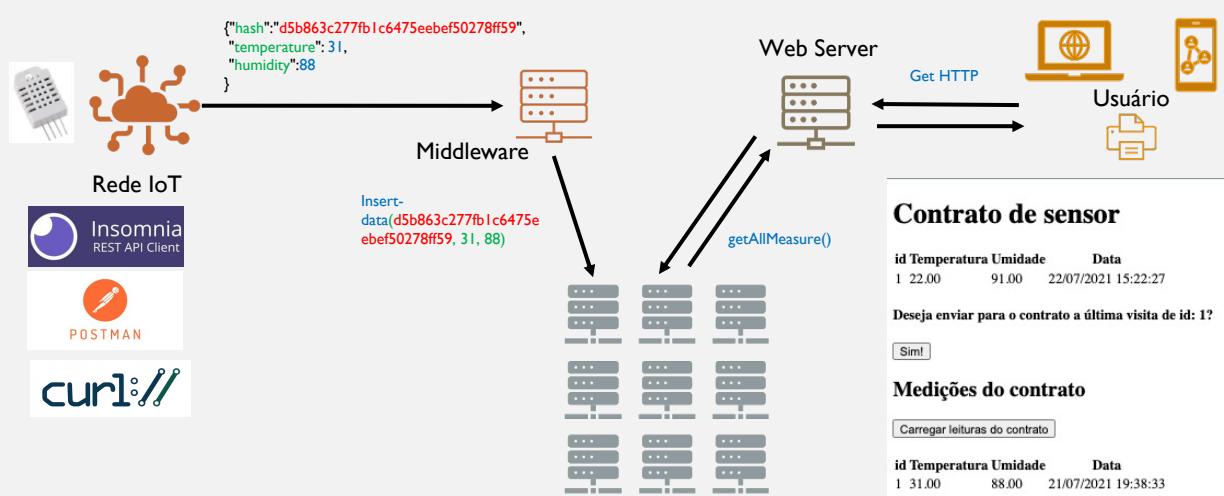
## DESENVOLVIMENTO DE DAPPS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

78

78

## DAPP VACINA



JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

79

## ARQUIVOS DO FRONTEND

src/contracts/vacina.contracts.js

- Contém o endereço do contrato implementado na rede Ethereum e a ABI do contrato.

src/contrats/web3.js

- Importa a web3 para a aplicação

src/app.js

- Contém o código javascript que acessa a DApp e cria a tela da aplicação

src/index.js

- arquivo principal

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

80

80

## WEB3.JS

```

1 // Importa módulo web3
2 import web3 from "web3";
3 // Busca o provider do metamask onde quer que esteja
4 const _web3 = new web3(web3.givenProvider);
5 // Ativa o ethereum no navegador
6 window.ethereum.enable();
7 // Exporta o web3 com o provider
8 export default _web3;

```

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

81

81

## SENSOR.JS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

```

1 // importa o web3
2 import web3 from "./web3";
3 // Endereço do contrato gerado no deploy
4 //const address = "0xfc272950a29c2f2307174e82C07566c1B231C951";
5 const address = "0xbC264f4C5cb198a6566470f8f4a89f9D779a073d";
6 // Abi gerada no deploy do contrato
7 const abi = [
8   {
9     inputs: [
10       {
11         internalType: "string",
12         name: "_name",
13         type: "string",
14       },
15       {
16         components: [
17           {
18             internalType: "string",
19             name: "idv",
20             type: "string",
21           },
22           {
23             internalType: "uint8",
24             name: "decimalPlace",
25             type: "uint8",
26           },
27           {
28             internalType: "uint256",
29             name: "defaultValue",

```

82

82

## APP.JS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

```

58 // retorna a última leitura
59 const data = readings[readings.length - 1];
60 // retorna contas do metamask
61 const contas = await web3.eth.getAccounts();
62 // Insere medidas no contrato passando a temperatura, humidade e o timestamp
63 const leitura = await sensor.methods
64   .insertMeasure([
65     [
66       [data.temperature.toFixed(0), data.humidity.toFixed(0)],
67       data.timestamp,
68     ],
69   ])
70   .send({ from: contas[0] });
71 // Altera o estado da transaction com o valor da leitura
72 setTransaction(leitura);
73 // Volta estado de loading para false
74 setLoading(false);
75 } catch (error) {
76   // Volta estado de loading para false
77   setLoading(false);
78   console.log(error);

```

83

83

## COMO MONTAR UM CURSO DE DESENVOLVIMENTO WEB COM BLOCKCHAIN E CONTRATOS INTELIGENTES

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

84

84

### COMO MONTAR UM CURSO

Não é uma tarefa trivial

- Solidity
- Front-end
- Back-end

Uma alternativa são as plataformas  
MOOC (*Massive Open Online Course*)

- Coursera
- Udemy
- edX

Curso de Extensão

- Referencias no livro texto

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

85

85

## EXPERIÊNCIA DOS AUTORES

- 3 cursos - UFBA, UESC e UESB
- 3 minicursos – SBRC 2018, SBSI 2021 e SBCAS 2021
- Módulo Básico
  - Teoria e conceitos básicos
  - Criação de um contrato simples
  - Criação de uma DApp single-page
- Módulo Avançado
  - Emprego de conceitos de engenharia de software
  - Mais recursos da linguagem solidity
  - DApp multipágina
- Módulo com IoT
  - Uso de IoT para gerar dados
  - Emprego de um middleware como intermediário
  - O Contrato pode gerar alertas
- Hardware
  - Ambiente misto
  - processadores que vão desde Core2 Duo com 4Gb de RAM até Core i7 com 32 Mb de RAM.
- Software
  - Node.js
  - Bibliotecas e Editores gratuitos
- Carga Horária
  - 16h para o módulo Básico
  - 16h para o módulo Avançado
  - 20h para o módulo com IoT

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

86

86

MAIS RECURSOS

CryptoZombies

Ethernaut

Vyper Tutorials

Ethereum Studio

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

87

87

## DESAFIOS E PERSPECTIVAS

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

88

88

### DESAFIOS DE PESQUISA

#### Atividades ilegais

- Por exemplo, o site *Silk Road* era um mercado negro online com uma plataforma para vendedores e compradores fazerem, entre outras coisas, o tráfico ilegal de drogas [Matthews 2015].
- Meiklejohn et al. [Meiklejohn et al. 2013] explora a heurística para agrupar carteiras de Bitcoin e classificar os operadores e quem busca usar Bitcoin para fins criminosos ou fraudulentos em grande escala.

#### Vazão de transações

- Novos protocolos aumentam a vazão para geração de novos blocos
- A presença de IoT intensifica o uso da Blockchain e pressiona por este aumento

JAI CSBC 2021 - Fabíola (UFBA) e Jauberth (UESC)

89

89

## DESAFIOS DE PESQUISA

### Armazenamento

- Com o aumento das transações o espaço de armazenamento aumenta
- uma estimativa simples, supondo que cada transação tenha 512 bytes e a unidade da taxa seja 0,0004 / KB, o tamanho dos dados da transação anual ficara muito alto
- A VISA em 2015 fez um total de 92.064 milhões de transações de pagamento ao longo do ano. Se convertermos esse volume de dados de transações da rede em Bitcoins, o tamanho anual poderia ser 47 TB, o que esta muito além da capacidade da atual

Tabela 4.3. Tamanho esperado dos dados em transações com o aumento do throughput. Fonte [Wu et al. 2019]

| TPS     | Tamanho do bloco | Taxas      | Tamanho anual |
|---------|------------------|------------|---------------|
| 1       | 0.3MB            | 0,12 BTC   | 15 GB         |
| 3       | 0.9MB            | 0,36 BTC   | 47 GB         |
| 10      | 3 MB             | 1.2 BTC    | 150 GB        |
| 100     | 30 MB            | 12 BTC     | 1.5 TB        |
| 1.000   | 300 MB           | 120 BTC    | 15 TB         |
| 10.000  | 3 GB             | 1.200 BTC  | 150 TB        |
| 100.000 | 30 GB            | 12.000 BTC | 1.500 TB      |

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

90

## DESAFIOS DE PESQUISA

### Combinação com outras tecnologias

- Além dos desafios de combinação com IoT, a blockchain pode ser combinada com IA, Big Data, Realidade Virtual, computação em nuvem/névoa e computação quântica

### Padronização

- Algumas organizações ou institutos internacionais têm se dedicado ativamente ao estabelecimento de padrões de blockchain para regular e facilitar seu desenvolvimento

JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

91

## PERGUNTAS

Dúvidas e mais informações:

**Jauberth Abijaude**  
[jauberth@uesc.br](mailto:jauberth@uesc.br)

**Fabiola Greve**  
[fabiola@ufba.br](mailto:fabiola@ufba.br)



JAI CSBC 2021 - Fabiola (UFBA) e Jauberth (UESC)

92