

# Blockchain 101

## Introdução a Aplicações Descentralizadas (Dapp)

Thiago Nóbrega

<https://github.com/thiagonobrega>

# Aula Passada

# Blockchain

- Concept of a world computer

Permissionaria (permissioned)

Não permissionaria

- Características

Decentralizado

Imutável (*Temper Evident*)

Transparente/Auditável

Pode ser utilizado por adversarios (*Distrustful Parties*)

# Introdução a Aplicações Descentralizadas (Dapp)

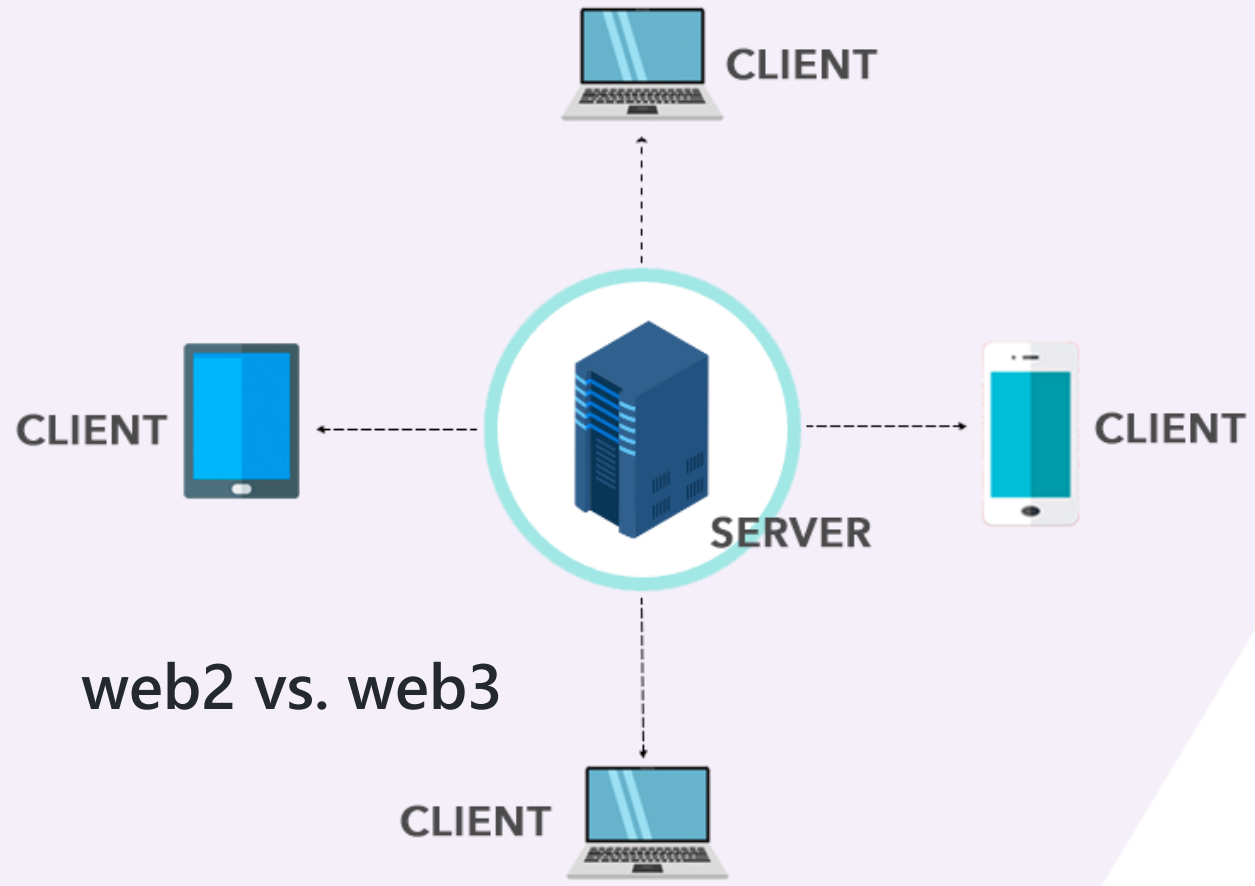
# Agenda

- Aplicações Descentralizadas
- Conceito de "world computer"
- Ethereum Virtual Machine (EVM)
- Accounts
- Blockchain properties
- Smart contracts

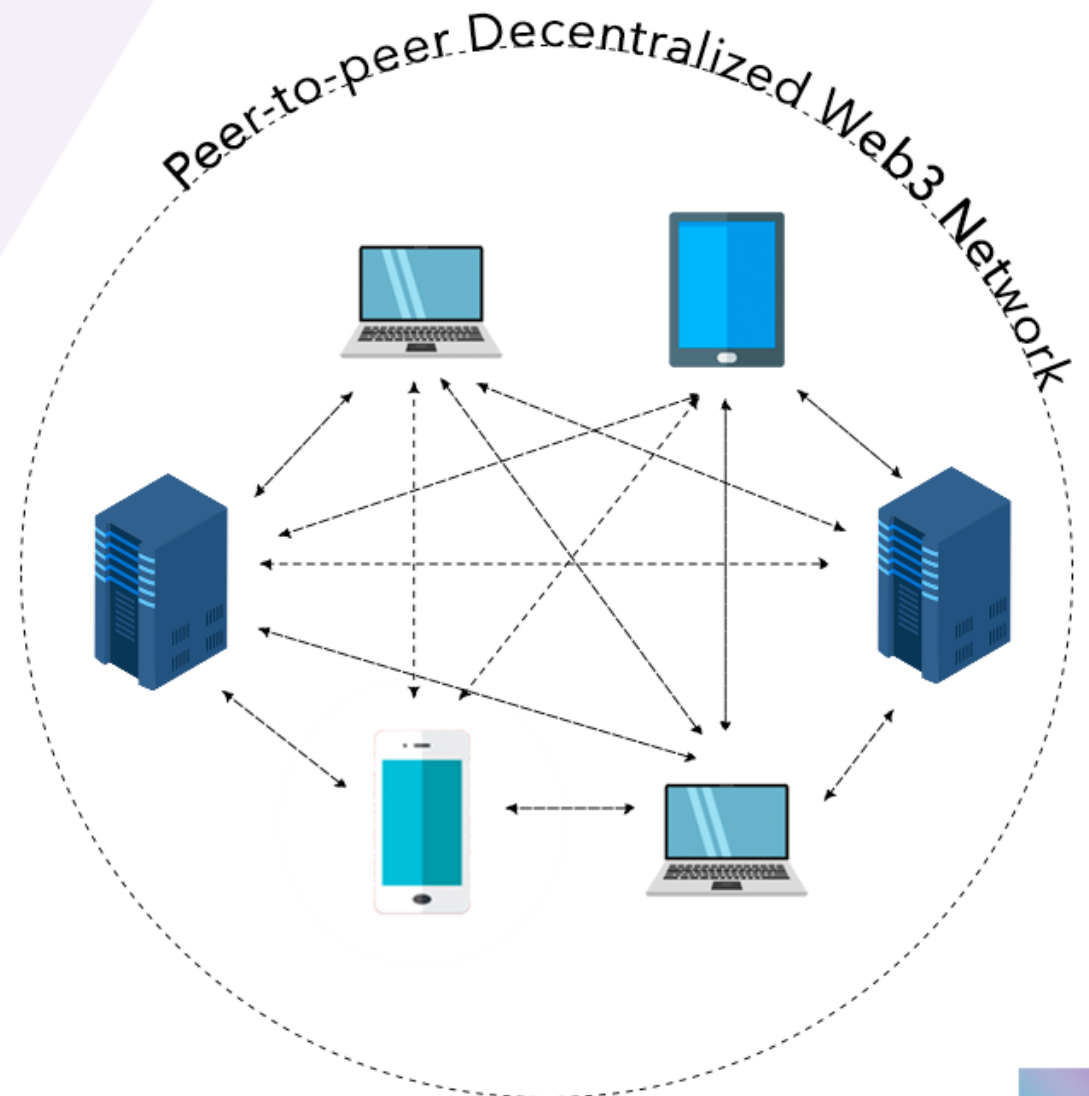
# Aplicações Descentralizadas

web3

Data & apps owned by many and stored on servers throughout the Web3 network



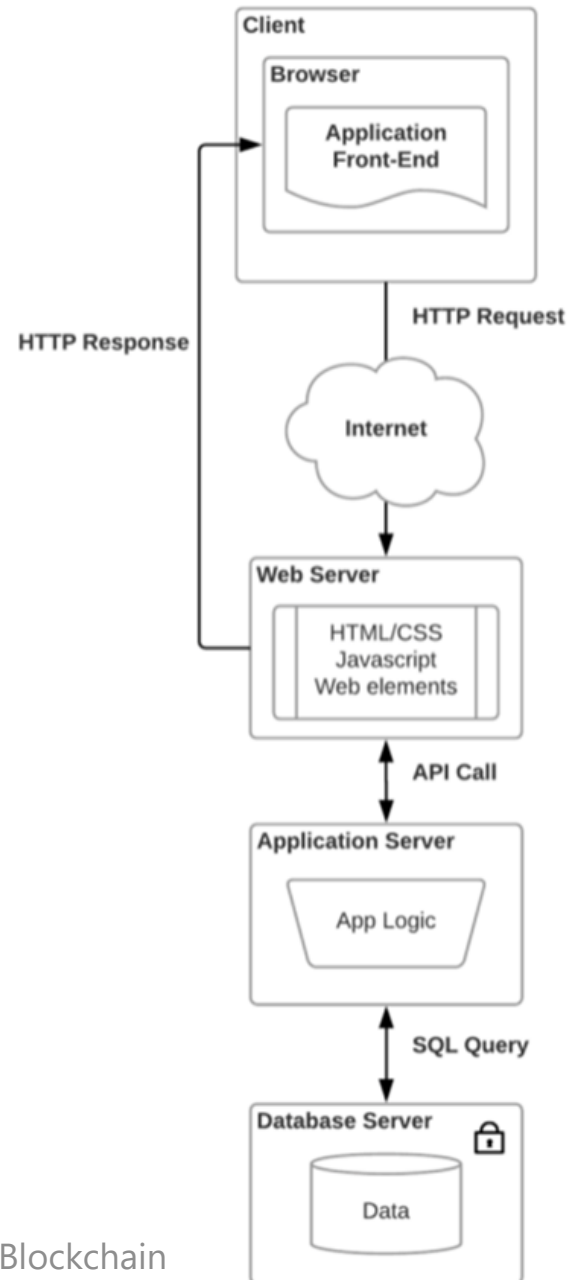
web2 vs. web3



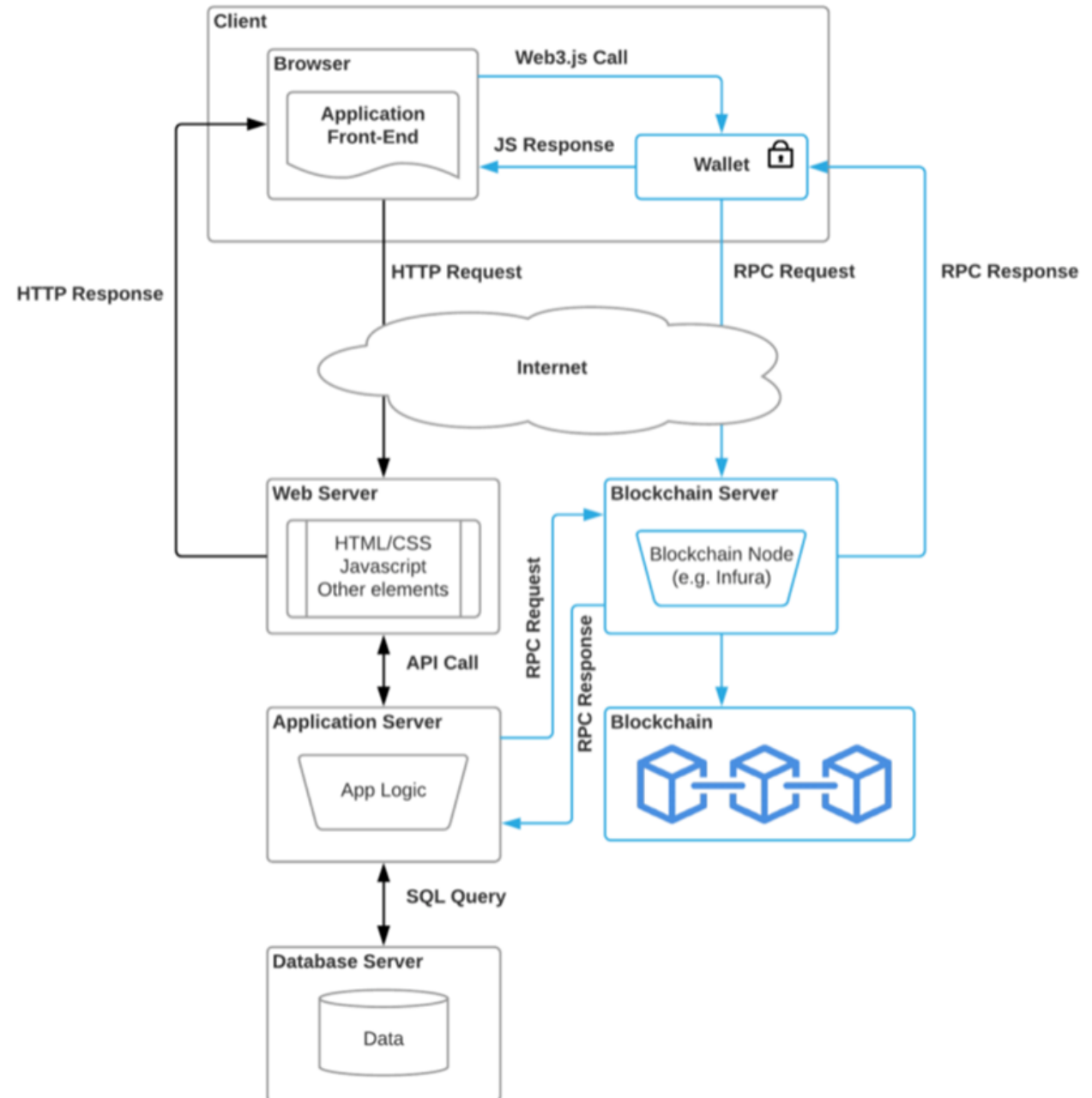
## WEB2

Introdução a Blockchain  
Data & apps stored on servers

## Web 2.0



## Web 3.0





## Web3

- Auditável
- Não depende de confiança
- Autogerida
- Não depende de permissões
- Distribuída e robusta
- Baseada em estados
- Com pagamentos integrados nativamente

## Aplicações Web3

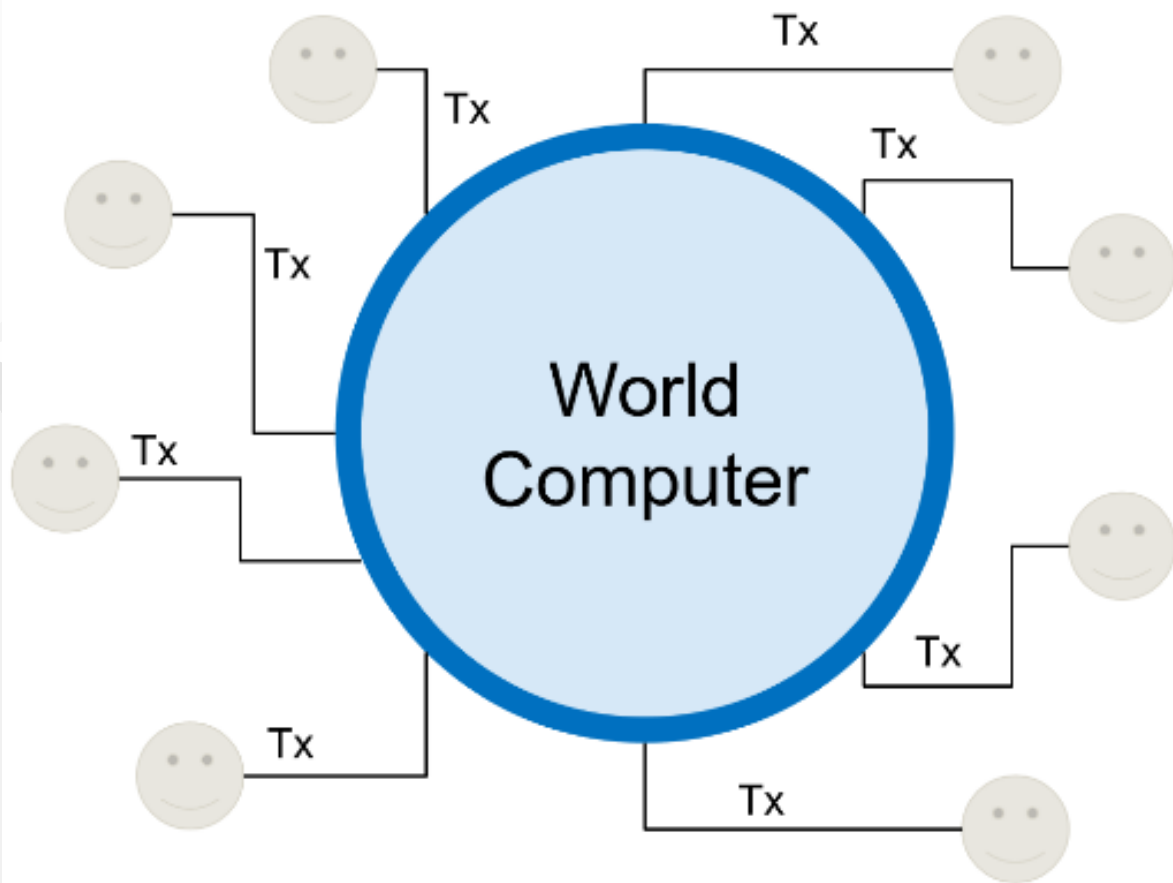
- Games
- NFTs
- DeFi (finanças descentralizadas)
  - Ethereum, [Fundo de Investimentos](#), etc.
- “Decentralized Autonomus Organization” (DAOs)

Financiar Projetos

[Leilões de obras raras](#)

[contratar profissional](#)

**"world computer"**



Tx = Transaction

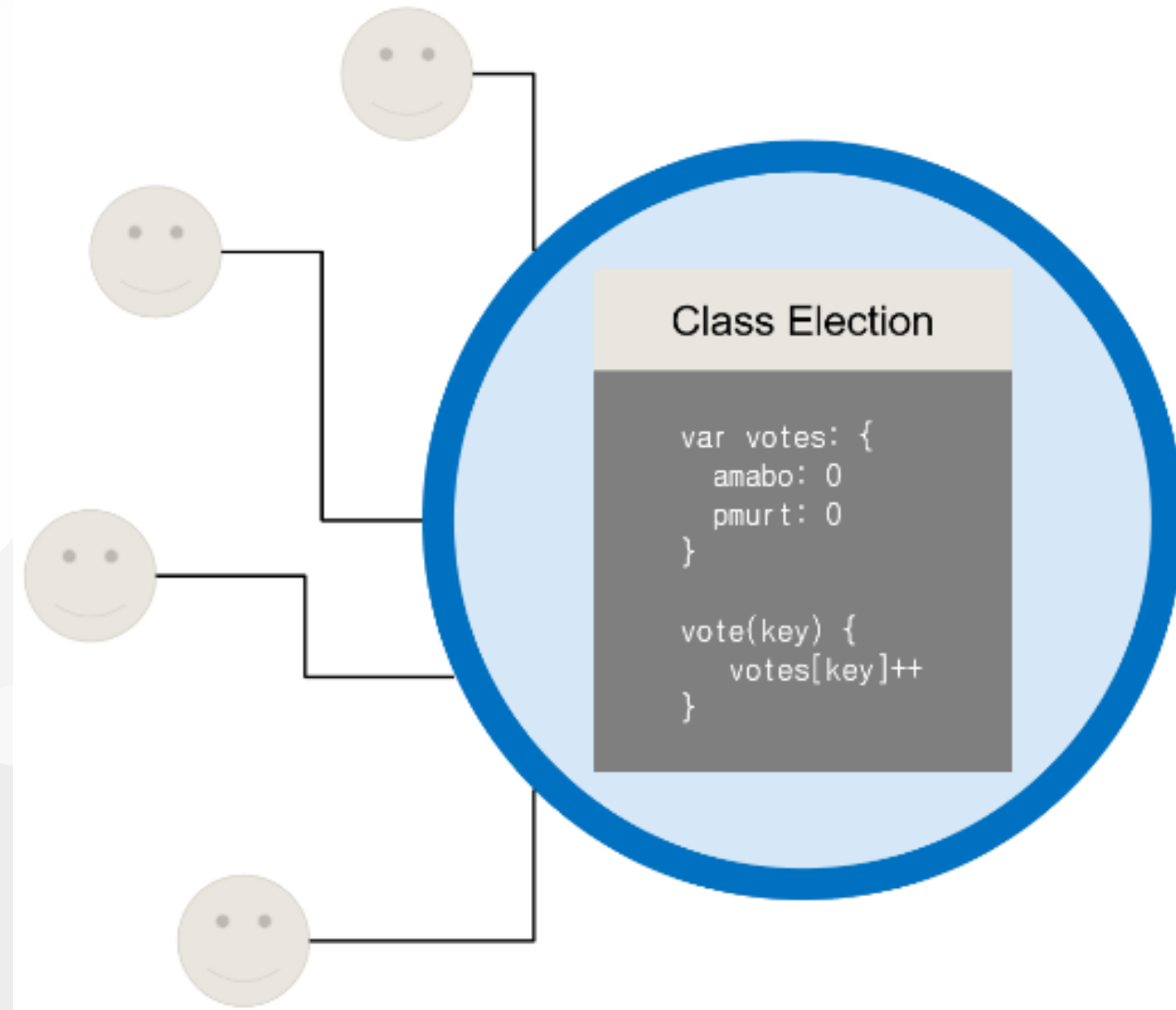
07 Ethereum Basics - Gellersdörfer, U., Holl, P., & Matthes, F. (2020). "Blockchain-based Systems Engineering". Lecture Slides. TU Munich.

## O conceito de "world computer"

### Maquina de Estado

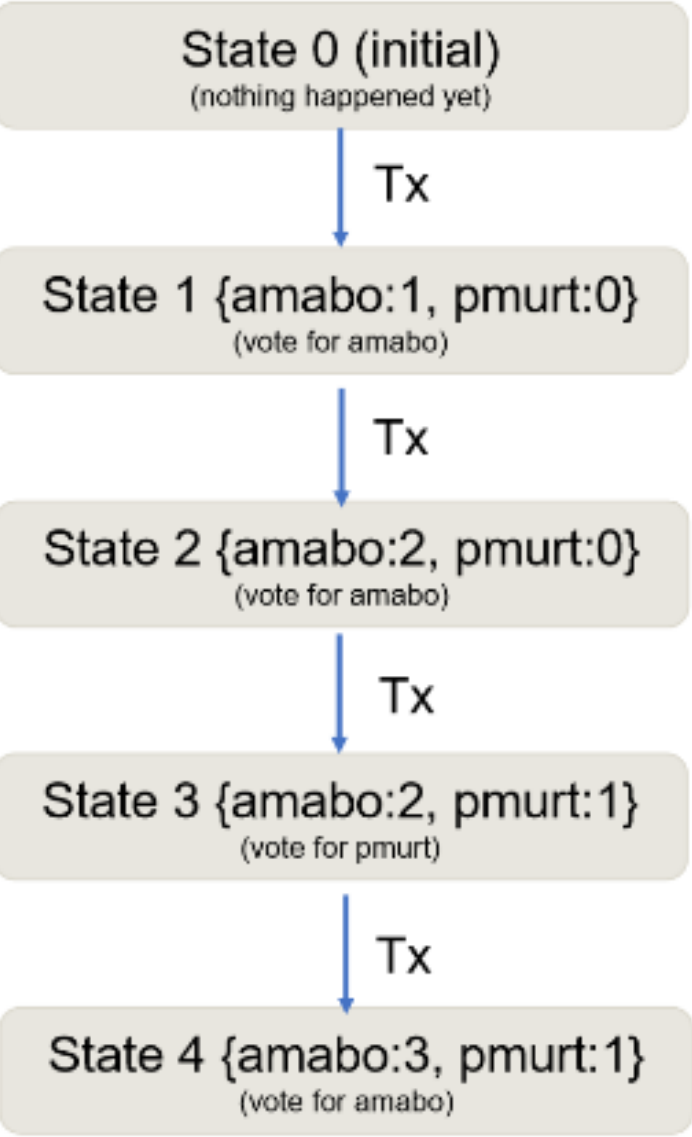
- Todos utilizam a mesma maquina
- Usuários enviam transações para invocar os programas
- Memória e compartilhada

# Programa

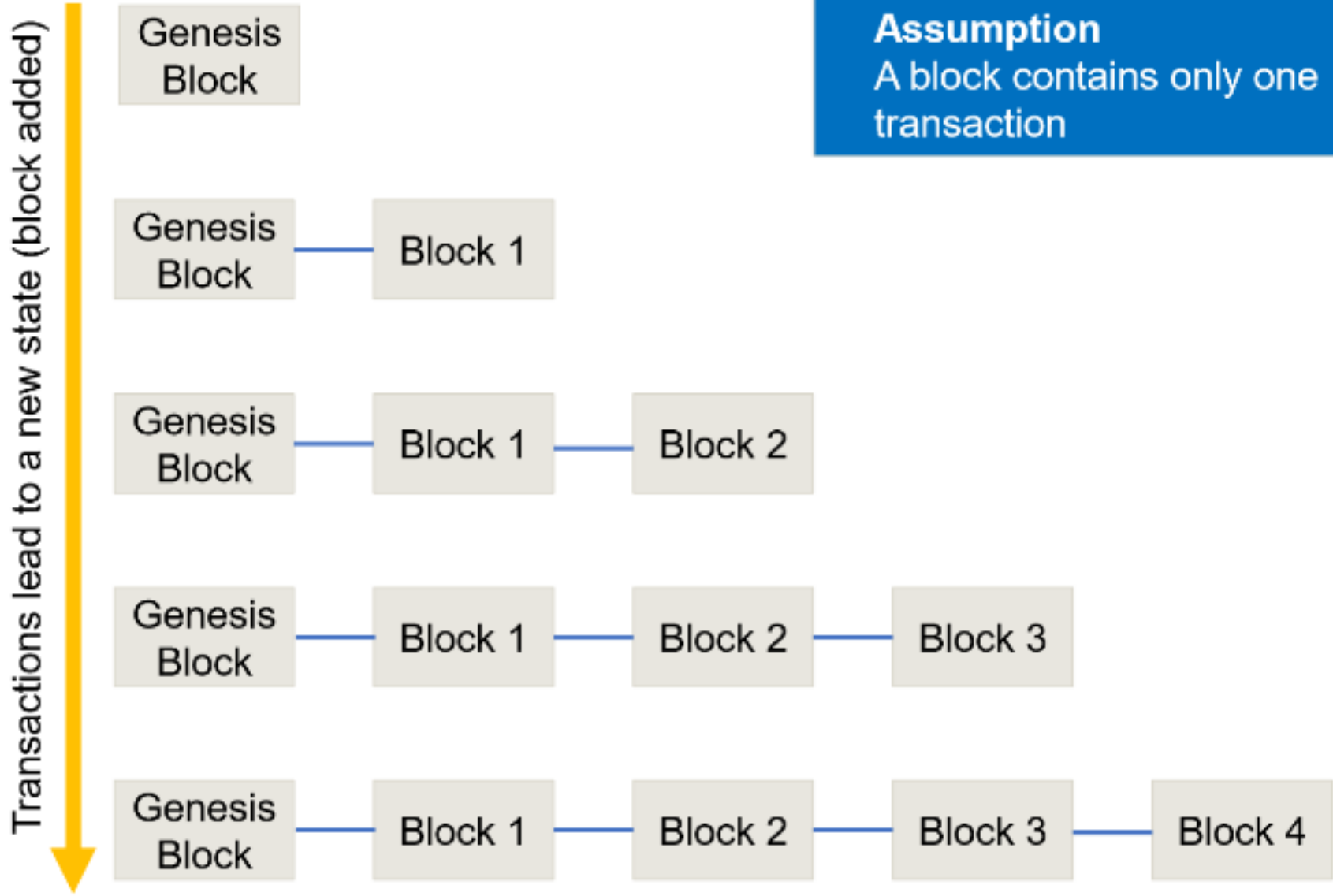


# Máquina de estado na Blockchain

# State of the world



# Blockchain



**Assumption**  
A block contains only one transaction



# Ethereum Virtual Machine (EVM)

- Proof-of-Work (main net) ---> Proof-of-Stake
- Dificuldade é ajustada a cada bloco
- Gas limit\*



# Smart Contracts



One or  
more parties



A smart  
contract



Execution of  
the contract

**CRYPTOTIPS.EU**

# Contratos Inteligentes

## Caso de uso

- Token systems ( USD, Binance + 1026 outros tokens)
- Identity and reputation systems
- Decentralized Autonomous Organization (DAO)
- Election and voting systems

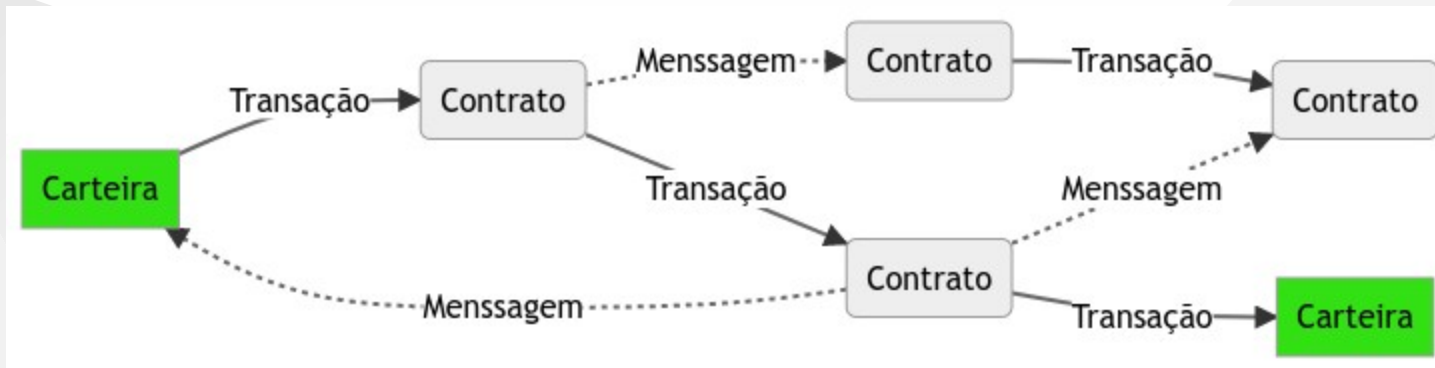
# Ethereum Virtual Machine (EVM)

## Transaction



# Ethereum Virtual Machine (EVM)

## Transactions



# Solidity



## Propriedades da Solidity

- Sintaxe similar ao JavaScript
- Estaticamente tipada
- Orientadas a Objeto
  - Suporte a herança
- Dynamic binding
  - compilada em EVM opcode

# Característica da Solidity

## Tipos nativos

inteiros (int, uint)

~~ponto flutuante~~

bytes

string\*

booleanos (bool)

array

struct

enum

mapping

# Característica da Solidity

## *Funções Built-in*

- **Tratamento de erro:** `assert()`, `require()`, `revert()`
- **Math & Crypto:** `addmod()`, `mulmod()`, `sha3()`, `keccak256()`, `sha256()`, `ripemd160()`, `ecrecover()`, `(+, -, mod, /, *)`
- **Block info:** `gasleft()`, `blockhash()`
- **Contrato:** `selfdestruct()`

## *Flow control*

- `if`, `else`, `do`, `while`, `break`, `continue`, `for`, `return`, `? ... : ...` (ternary operator)



# Característica da Solidity

## Constructor

- Executado quando o contrato é criado por uma transação;
- Não pode ser executado após a criação;
- Utilizado para configurar o contrator;
- Instruções complexas podem aumentar o custo (gas) de deployment do contrato;

# Característica da Solidity

## Funções

```
function <name>([<parameter types>])  
{internal|external|public|private}  
[pure|constant|view|payable]  
[(modifiers)]  
[returns (<return types>)]
```

- Functions are used to change the state of a contract.
- Can also be used to read the state of the contract.
- Consist of a name, a signature, a visibility, a type, a list of modifiers, and a return type.

# Visibilidade de Funções

```
function <name>([<parameter types>])  
**{internal|external|public|private}**  
[pure|constant|view|payable]  
[(modifiers)]  
[returns (<return types>)]
```

## Visibilidade

- Public
- External : não pode ser acessado pelo próprio contrato
  - Internal : ~External + contratos derivados deles
- Private : Apenas pelo contrato

## Special function types

```
function <name>([<parameter types>])  
{internal|external|public|private}  
**[pure|constant|view|payable]**  
[(modifiers)]  
[returns (<return types>)]
```

### Visibilidade

#### Leitura (não aletram a chain)

- view : pode ler estado de variaveis
- pure : não pode ler estado de variaveis

#### Escrita (a transação precisa ser inserida)

- payable :

## Exemplos

### view

```
uint state = 5;

function add(uint a, uint b)
public view
returns (uint sum) {
    return a + b + state
}
```

### pure

```
function add(uint a, uint b)
public view
returns (uint sum) {
    sum = a + b;
}
```

### playable

```
unit = total;

function add(uint a, uint b)
public {
    total = total + a + b;
}
```

# Laboratório

<https://remix.ethereum.org>

<https://github.com/thiagonobrega/bc101>