Intro to Smart Contracts & Solidity module

Learning the language and deploying smart contracts

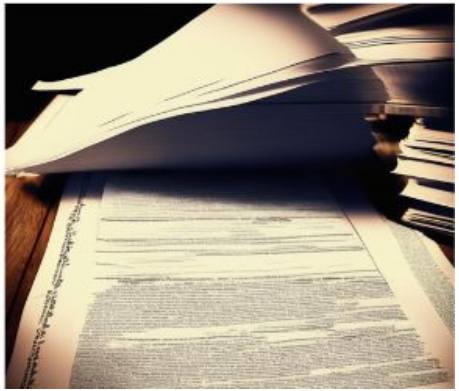
Github con diapositiva y los ejercicios

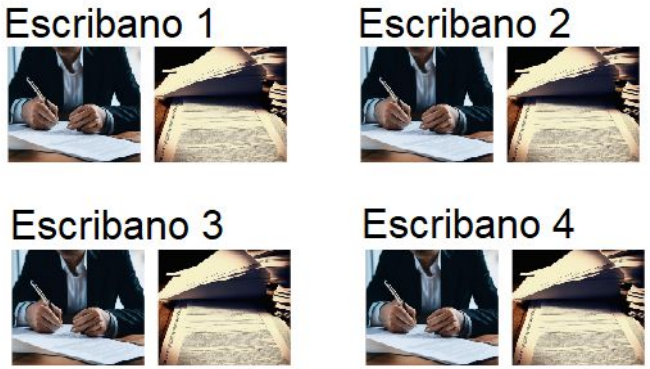
https://github.com/DigiCris/alephSolidity


COMUNY·T
the unstoppable
investment chain
Sé parte de la comunidad de inversores globales
de Real Estate y entrá en un loop imparable

Contacting Cristian Marchese

Sistemas distribuidos

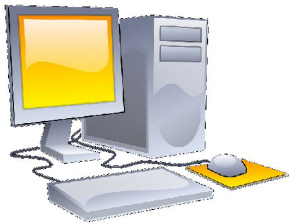




- Lossing Data
- Access to Data
- Integrity of Data
- Good will

Sistemas distribuidos 2

Nodo 1

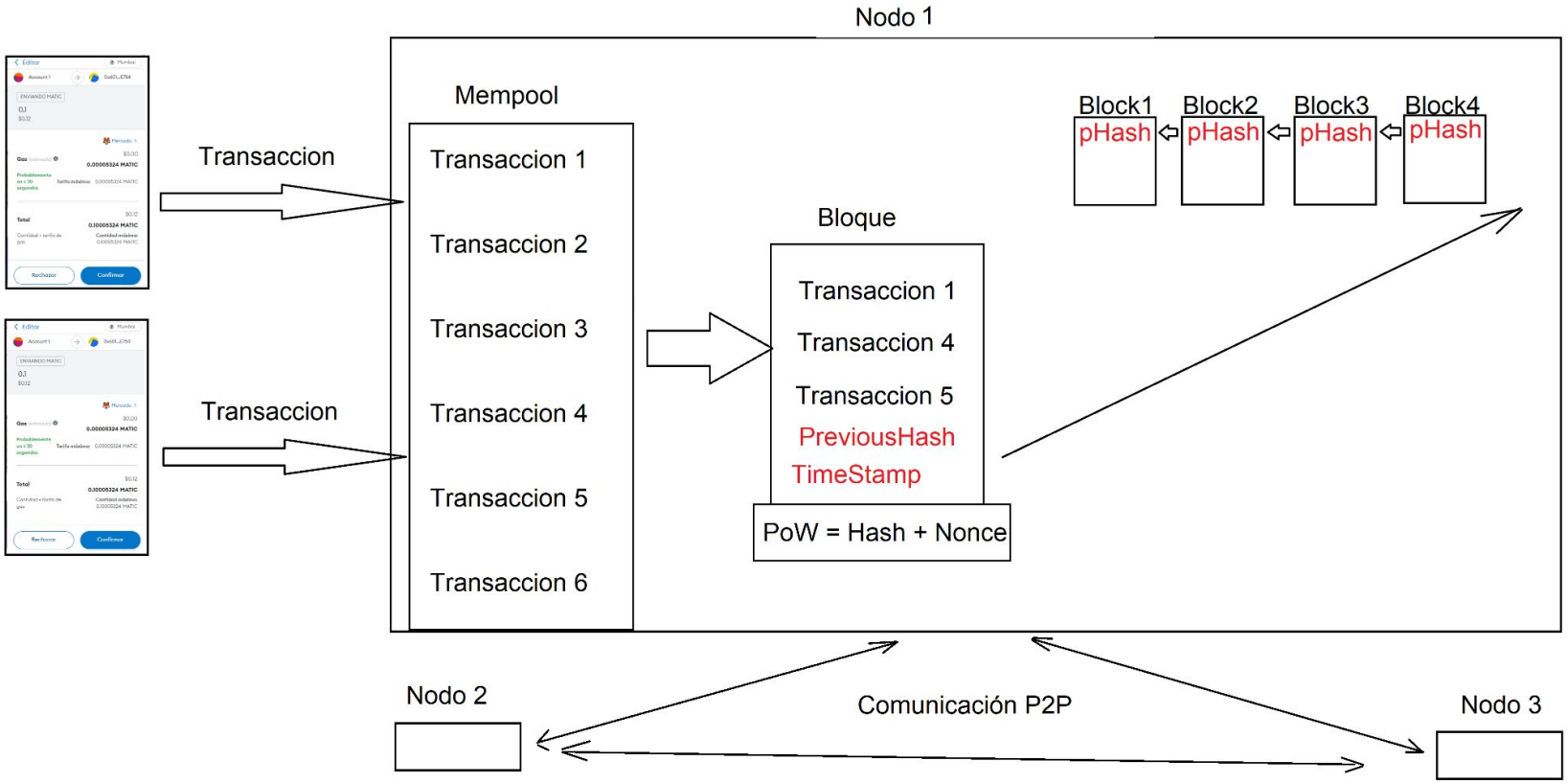


Nodo 2

Nodo 3

Nodo 4

Blockchain


Nodo 1
Mempool
Transaccion 1
Transaccion 2
Transaccion 3
Transaccion 4
Transaccion 5
Transaccion 6
Transaccion
Transaccion
Bloque
Transaccion 1
Transaccion 4
Transaccion 5
PreviousHash
TimeStamp
PoW = Hash + Nonce
Block1
pHash
Block2
pHash
Block3
pHash
Block4
pHash
Nodo 2
Comunicación P2P
Nodo 3

External Owned Accounts (EOA)

Private Key:

- Sign message to assure identity.
- Random 256 bit number.
- Don’t lose it.

Public key:

- Verify Identity.
- Derived using ECDSA from the private key
- Can’t sign messages.

Address:

- Use to receive transactions
- Derived from public key using keccak256. Bit_96_256(keccak256(public_key))
- It’s is not the same than the public key..

Metamask

Pasos para la instalación

1. Abrir ***Chrome***.
2. Abrir la ***web store***.
3. Buscar ***MetaMask***.
4. ***Añadir a Chrome***.
5. ***Agregar Extensión***.
6. ¡Listo!

Metamask 2

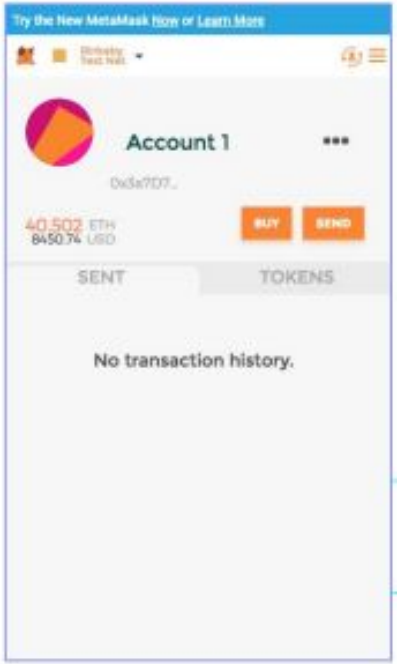
Pasos para la configuración inicial

1. Abrir MetaMask.
2. Aceptar los términos y condiciones.
3. Setear una clave.
4. Guardar de manera segura la clave de restore.
5. ¡Listo!

Metamask 3

Al completar la configuración han pasado las siguientes cosas:

1. Se ha creado un ***Account Address*** (0xcf....).
2. Se ha generado una ***Public Key***.
3. Se ha generado una ***Private Key***.
4. ¡Ya podemos operar sobre la Blockchain!



Sepolia Testnet

Network name

Ethereum Sepolia

New RPC URL

https://rpc.sepolia.org

Chain ID

11155111

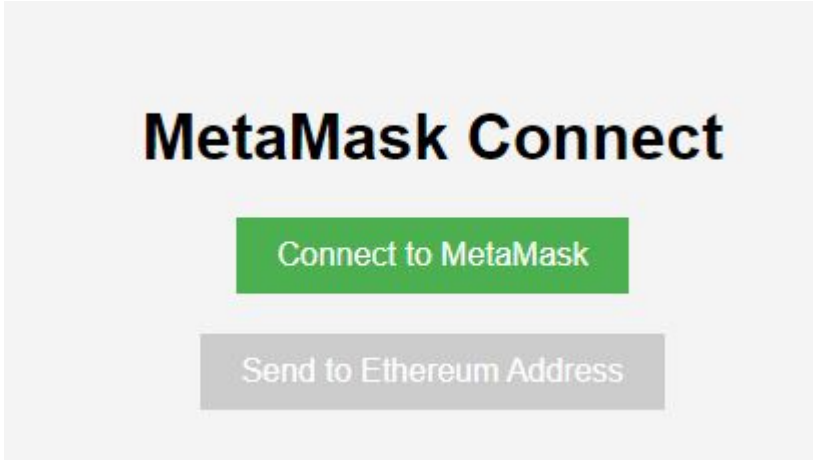
Currency symbol

ETH

Block explorer URL (Optional)

https://sepolia.etherscan.io

Faucet


MetaMask Connect
Connect to MetaMask
Send to Ethereum Address


https://comunyt.com/faucet

Envío de Ethers

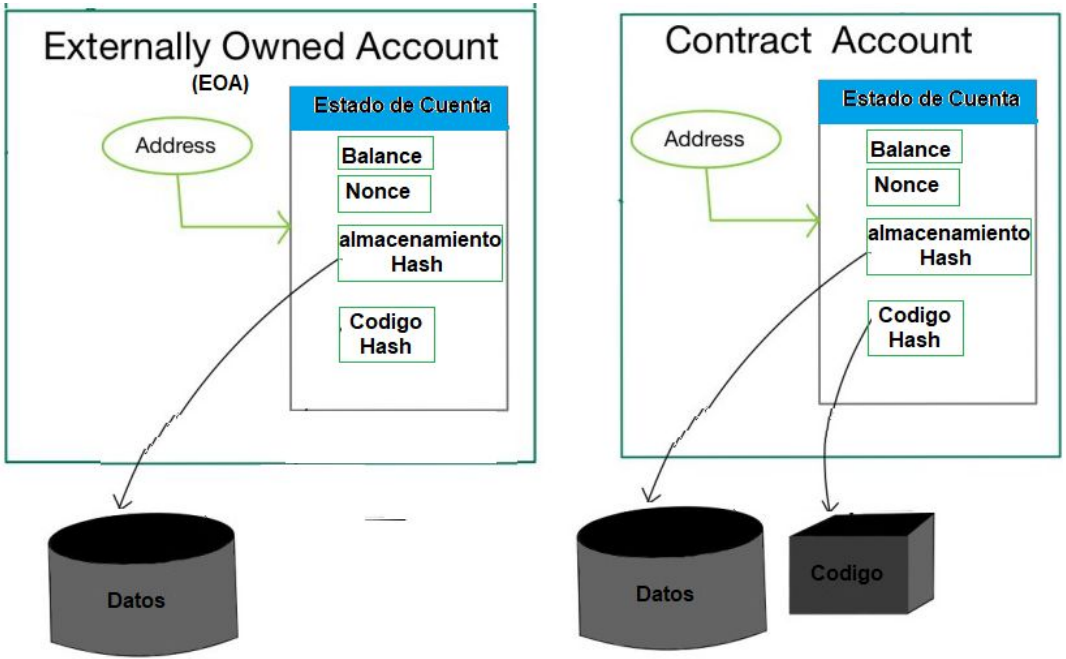
Enviar ether en sepolia a la siguiente dirección:

0xF3772Ea009e0580c01b3b6cadf3101c261a3c758

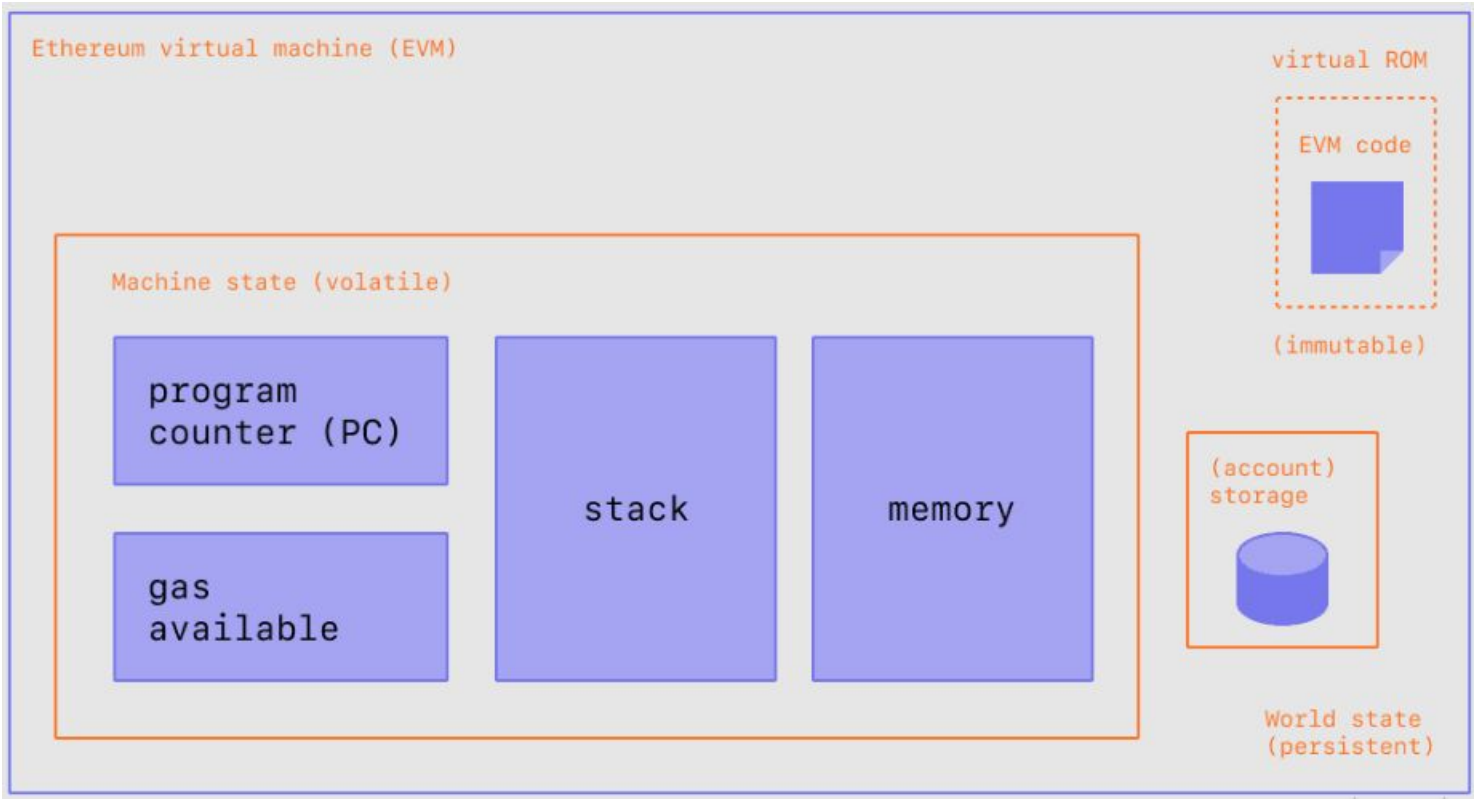
Transaction and Block explorer

https://sepolia.etherscan.io/tx/0xb49ecc90df992e80d04b57bdbf98b8c653cafa291c4097f01ba1604fb76b87b4

EOA vs Smart Contracts



EVM and Gas


Ethereum virtual machine (EVM)
virtual ROM
EVM code
(immutable)
Machine state (volatile)
program counter (PC)
gas available
stack
memory
(account) storage
World state (persistent)

SC1: Hello world

```
pragma solidity ^0.8.0;

contract HelloWorld {
    string public message;

    constructor() public {
        message = "Hello, World!";
    }

    function updateMessage(string memory newMessage)
      public {
        message = newMessage;
    }
}
```

SC 2: Aprendiendo el lenguaje Solidity

https://github.com/DigiCris/EducationIT/blob/main/clase2/solidity1.sol

- Tipos de Datos
- Propiedades de Bloques y Transacciones
- Visibilidad de Funciones y Variables
- Modificadores
- Constructor
- Underflow y Overflow
- Efectos Secundarios
- Transferencia de ethers
- Unidades y Literales
- Manejo de Cadenas de Texto
- Pseudoaleatoriedad
- Aserciones y Validaciones
- Llamada entre contratos

ERC20

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
function approve(address _spender, uint256 _value) public returns (bool success)
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

```
event Transfer(address indexed _from, address indexed _to, uint256 _value)
event Approval(address indexed _owner, address indexed _spender, uint256 _value)
```

Openzeppelin Wizzard

https://wizard.openzeppelin.com/

Codigo ERC20

```
// SPDX-License-Identifier: MIT
// Compatible with OpenZeppelin Contracts ^5.0.0
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract AlephToken is ERC20, Ownable {
    constructor(address initialOwner)
        ERC20("AlephToken", "ATK")
        Ownable(initialOwner)
    {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }

    function mint(address to, uint256 amount) public onlyOwner {
        _mint(to, amount);
    }
}
```

Desafío ICO

Crear un contrato que sea capaz de vender un token de Aleph utilizando como medio de pago un USDT propio creado.

El USDT debe poder ser minteado por cualquiera para que cualquiera pueda crearselo y usar para comprar el token de aleph.

El token de aleph solo podrá ser creado por el contrato ICO cuando reciba tokens usdt.

Solución ICO

https://comunyt.com/ico/

ICO=0xD941204eE96ff96605050b9D42f620Ccd7feC6fC

USDT= 0x365305FaDDdd0E2EacdCB73473193fc4c7493459

ATK=0xD08188fe25f90E2DAFA8a9Fb37C82510b96CA00A