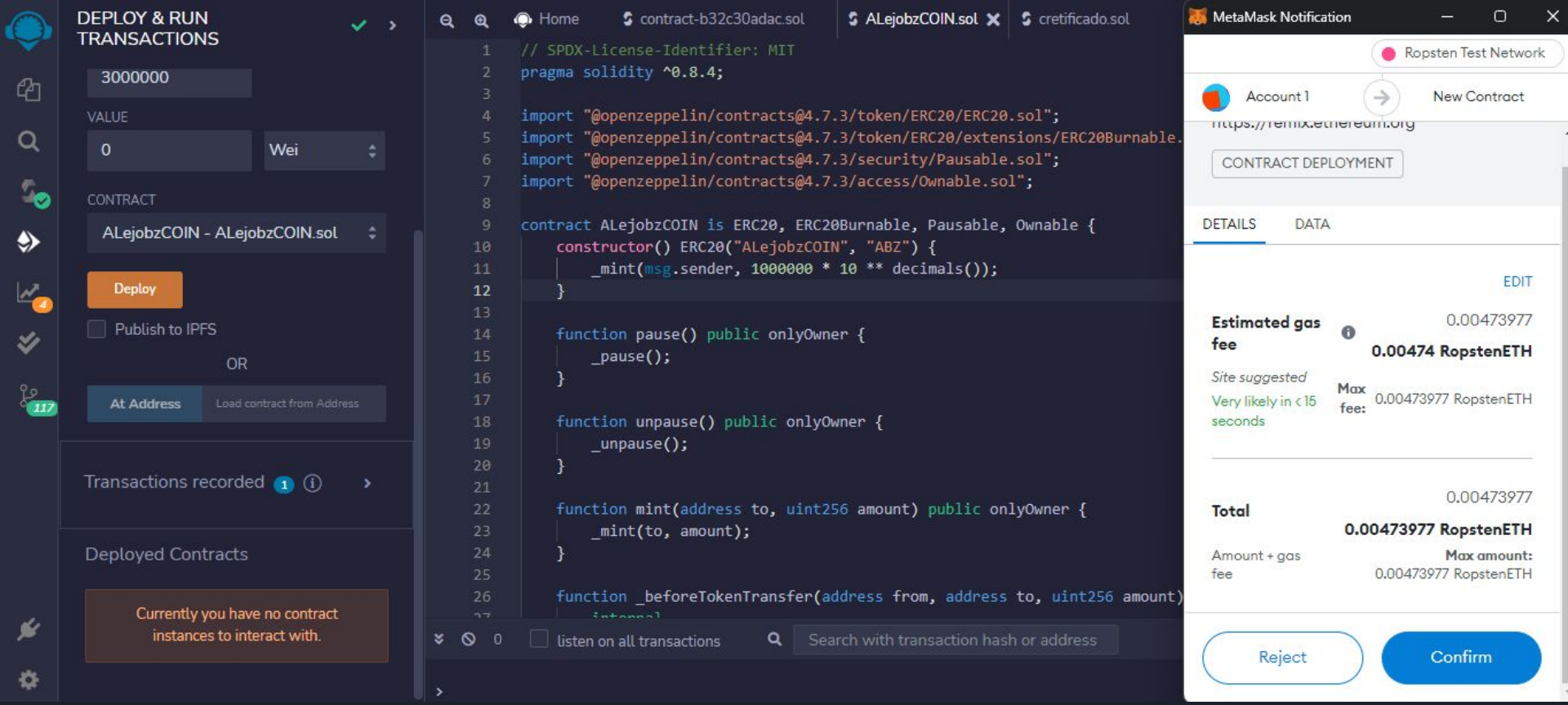
# Desarrollo de Tarea Web 3.0 - Blockchain

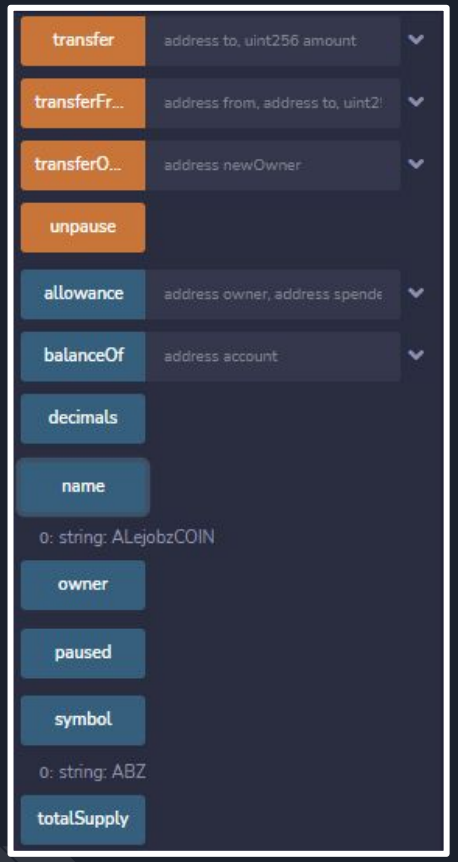
Alejandro Beltran
Victor Manchola

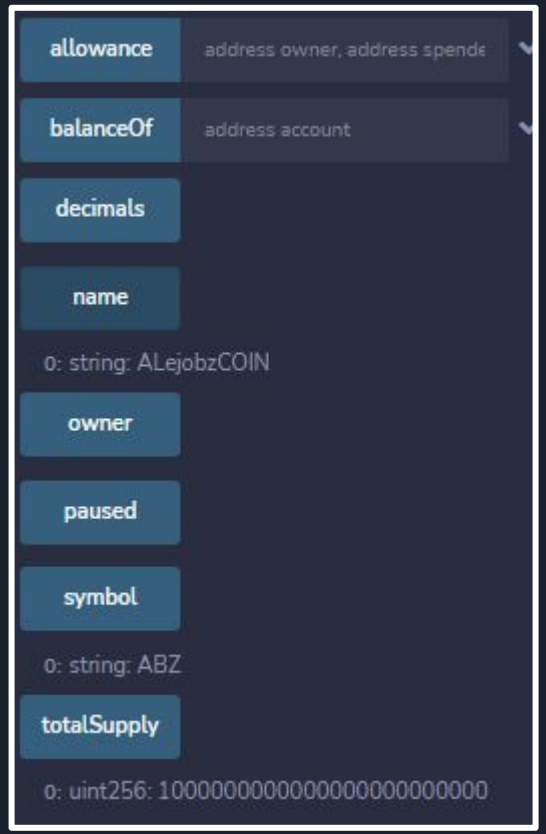
# Proceso de Diseño - Parte 1 (Invidividual)

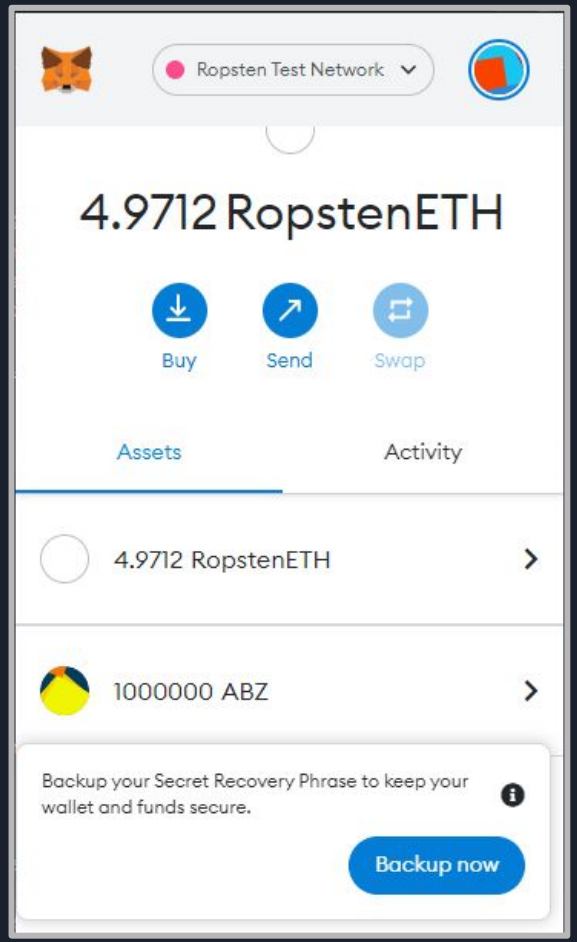
- Toma de notas de clases, revisando grabaciones
- Contrato en *Solidity* siguiendo estándar ERC-20
- Decidiendo un nombre para nuestras criptomonedas
- Escribiendo el contrato en código utilizando *OpenZeppelin*
- Funciones de:
  - Constructor
  - Pausa y Reanudo
  - Mint
  - Antes de Transferencia



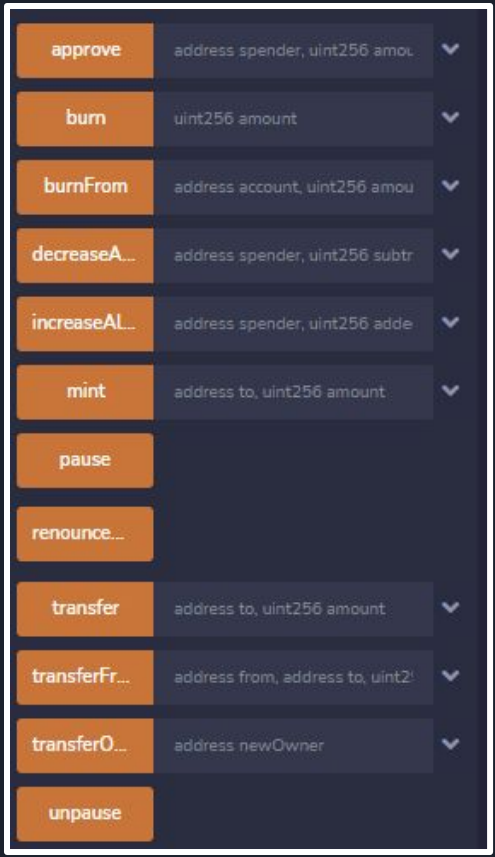

# Código en Funcionamiento (ALejobzCOIN)

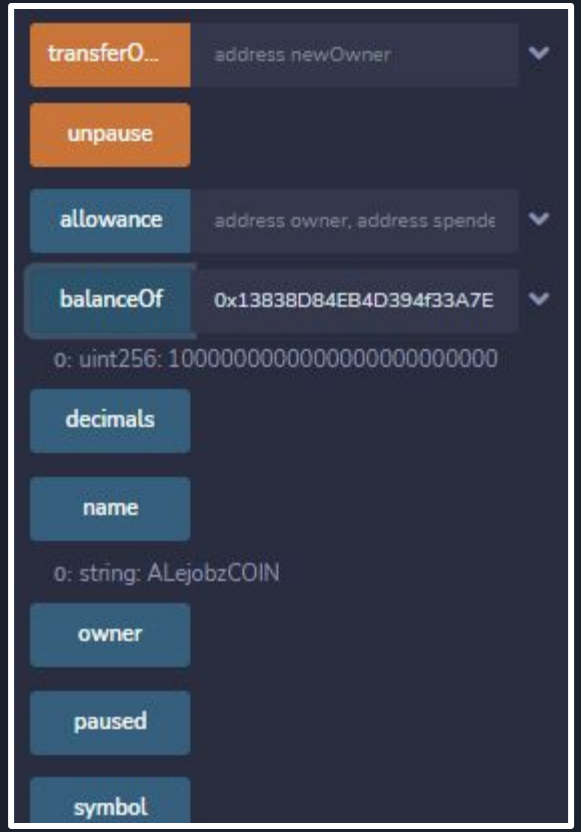


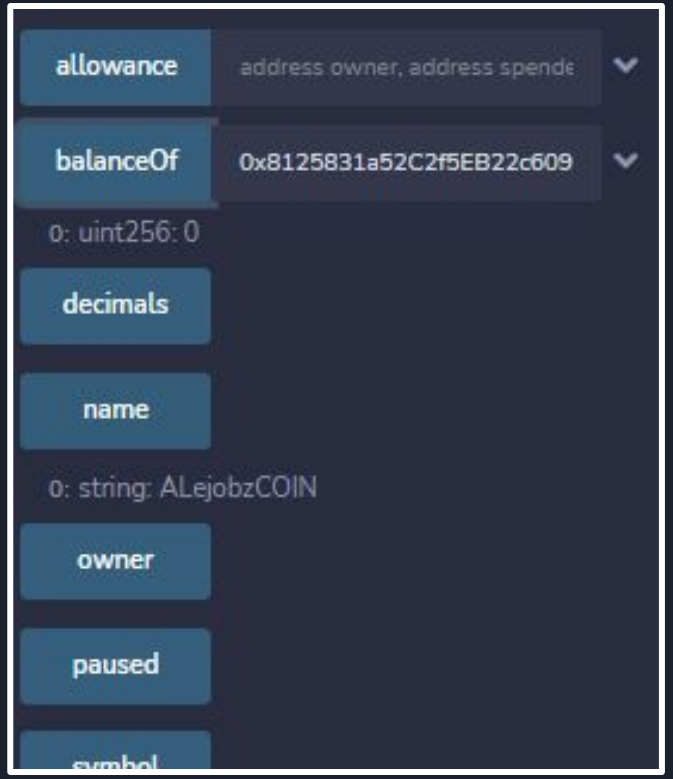



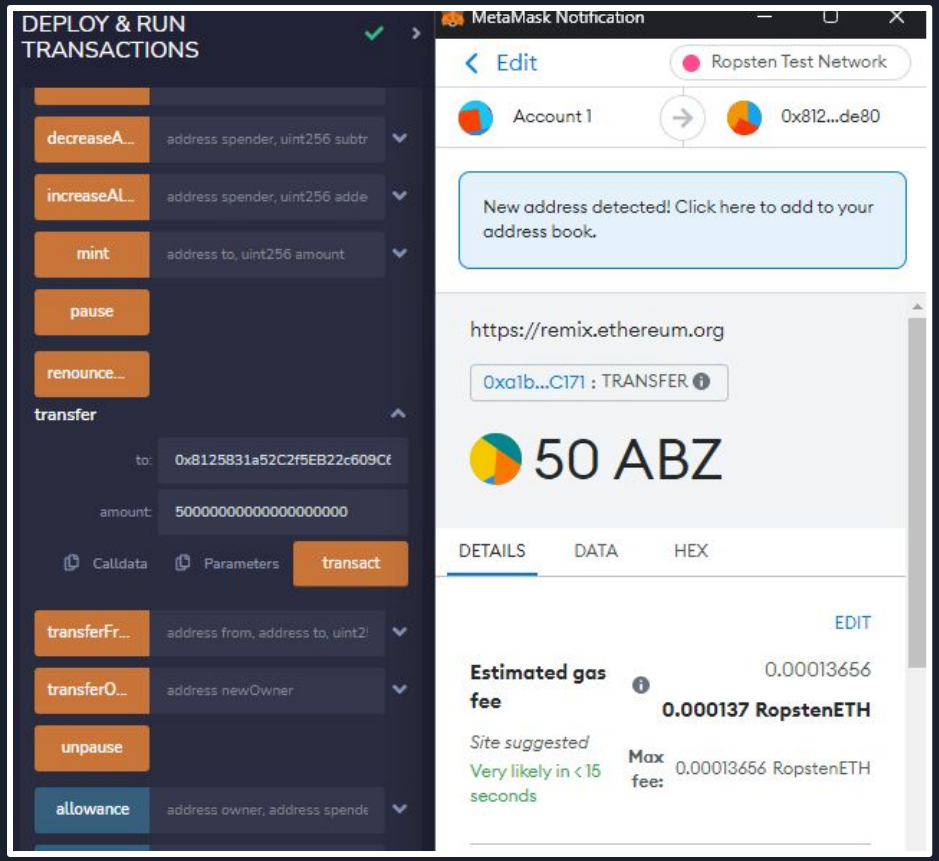

Identificador, símbolo de la moneda y suministro total
la conexión con METAMASK

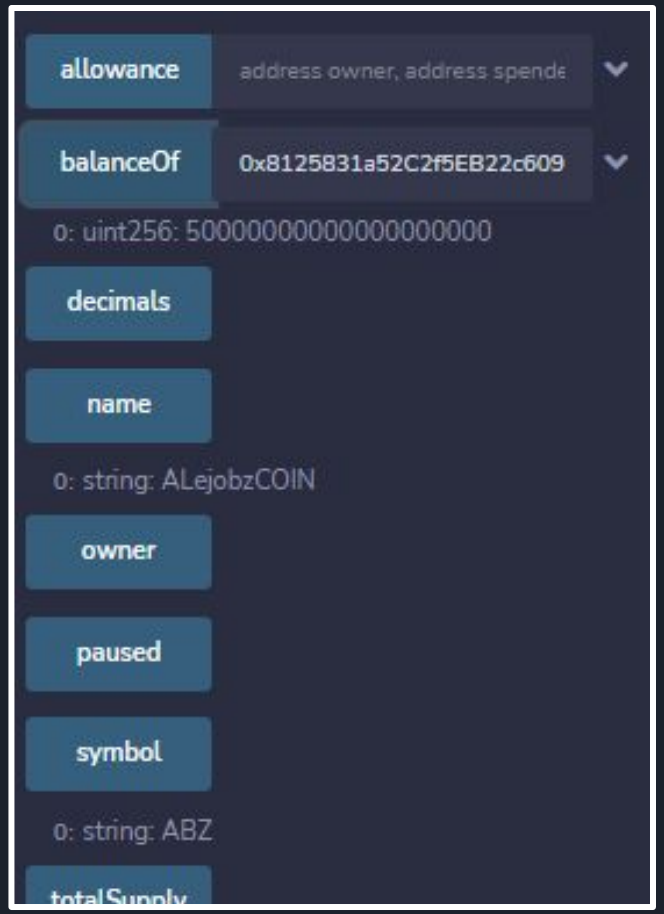



Control de emisiones, balance de mi address y de la address de mi compañero




Transfiriendo la moneda

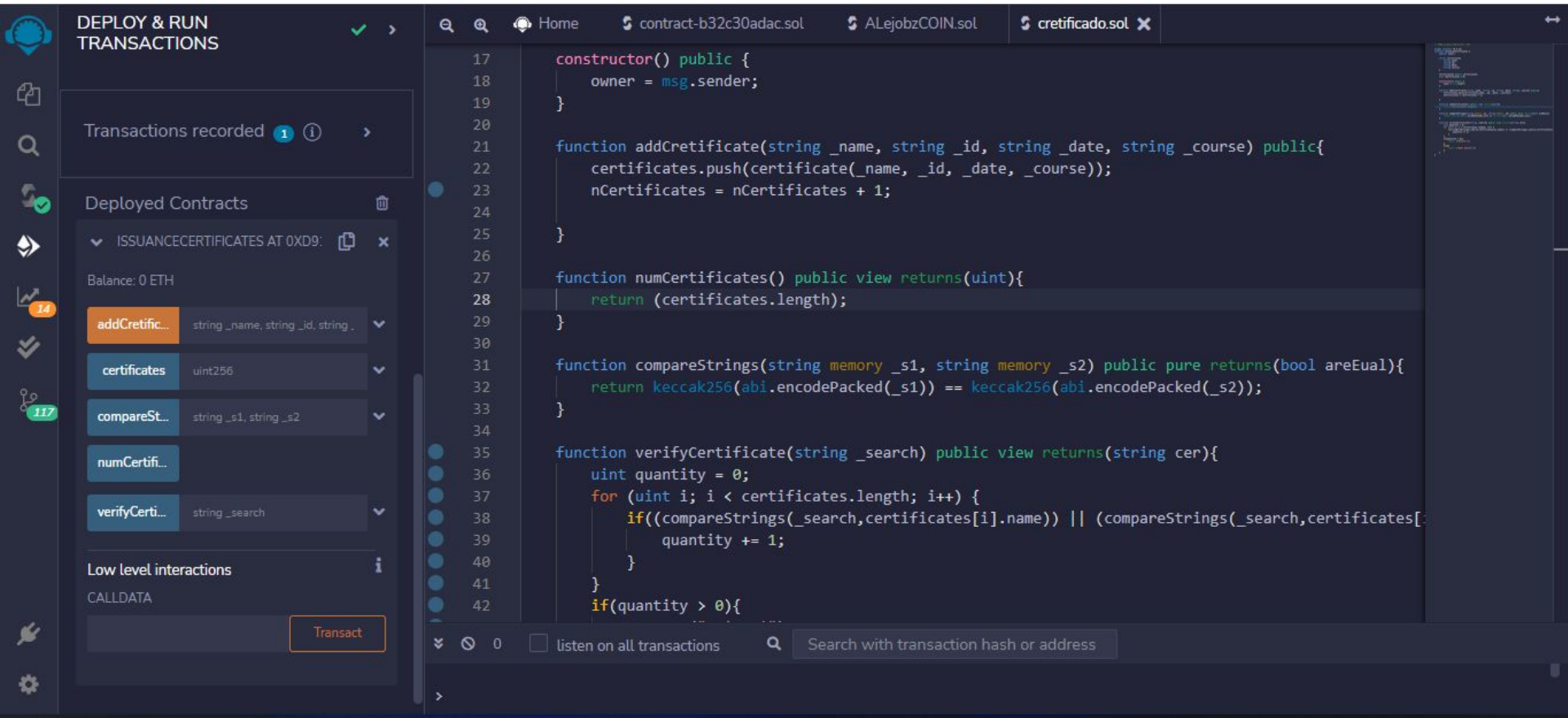
# Código en Funcionamiento (VCoin)

```
// Desarrollo de Tarea 3 de Tecnologias Emergentes
// Victor Manchola
// Licensia del programa:
//    SPDX-License-Identifier: MIT

// Pragma, Imports
pragma solidity ^0.8.4;
import "@openzeppelin/contracts@4.7.3/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts@4.7.3/token/ERC20/extensions/
  ERC20Burnable.sol";
import "@openzeppelin/contracts@4.7.3/security/Pausable.sol";
import "@openzeppelin/contracts@4.7.3/access/Ownable.sol";

// Contrato
contract VCoin is ERC20, ERC20Burnable, Pausable, Ownable {
    constructor() ERC20("VCoin", "VAM") {
        _mint(msg.sender, 1000000 * 10 ** decimals());
    }

    function pause() public onlyOwner {
        _pause();
    }

    function unpause() public onlyOwner {
        _unpause();
    }

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

    function _beforeTokenTransfer(address from, address to, uint256
      amount)
        internal
        whenNotPaused
        override
    {
        super._beforeTokenTransfer(from, to, amount);
    }
}
```

# Proceso de Diseño - Parte 2 (Grupos)

- Desarrollo de la interfaz en *JavaScript* primero
  - Campos de entrada de texto
  - Busqueda
  - Botones con acciones
- Desarrollo del contrato en *Solidity*
  - Funcionalidad para cada entrada en Blockchain
  - Funcionalidad de búsqueda
- Creación de un servidor con express
- Uniendo el contrato con la interfaz

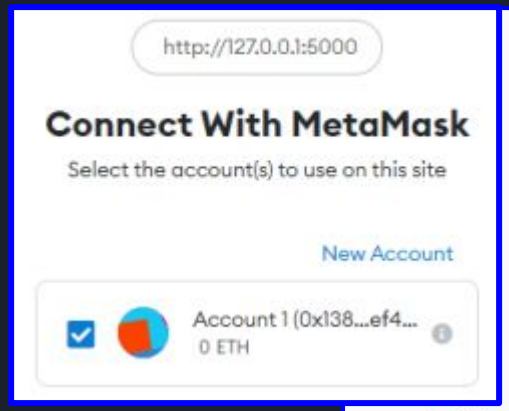
# DISEÑO

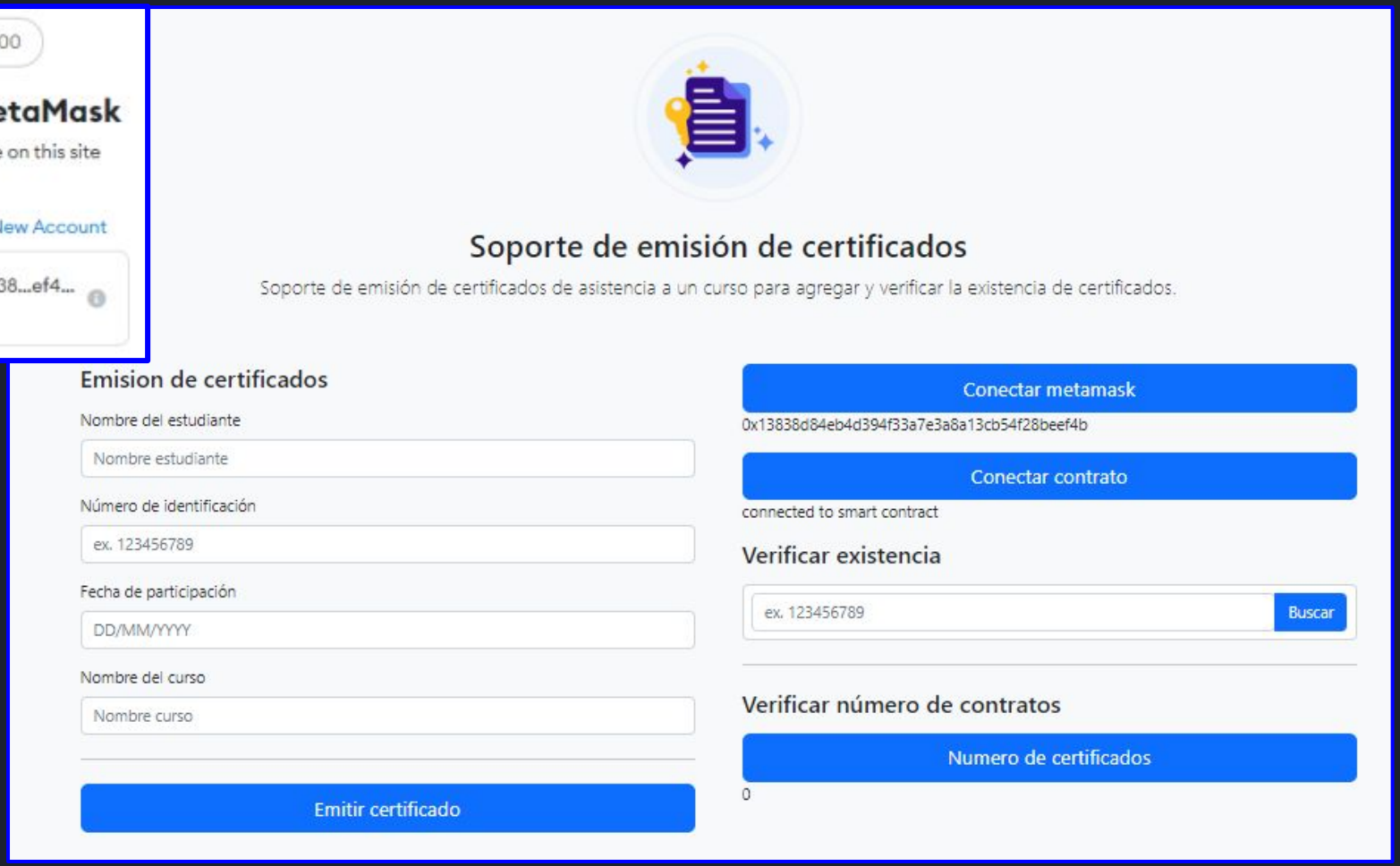

DEPLOY & RUN TRANSACTIONS
Transactions recorded 1
Deployed Contracts
ISSUANCECERTIFICATES AT 0XD9:
Balance: 0 ETH
addCretific...
string _name, string _id, string _
certificates
uint256
compareSt...
string _s1, string _s2
numCertifi...
verifyCerti...
string _search
Low level interactions
CALLDATA
Transact
Home
contract-b32c30adac.sol
ALejobzCOIN.sol
cretificado.sol
constructor() public {
owner = msg.sender;
}
function addCretificate(string _name, string _id, string _date, string _course) public{
certificates.push(certificate(_name, _id, _date, _course));
nCertificates = nCertificates + 1;
}
function numCertificates() public view returns(uint){
return (certificates.length);
}
function compareStrings(string memory _s1, string memory _s2) public pure returns(bool areEual){
return keccak256(abi.encodePacked(_s1)) == keccak256(abi.encodePacked(_s2));
}
function verifyCertificate(string _search) public view returns(string cer){
uint quantity = 0;
for (uint i; i < certificates.length; i++) {
if((compareStrings(_search,certificates[i].name)) || (compareStrings(_search,certificates[
quantity += 1;
}
}
if(quantity > 0){
0
listen on all transactions
Search with transaction hash or address

# DISEÑO

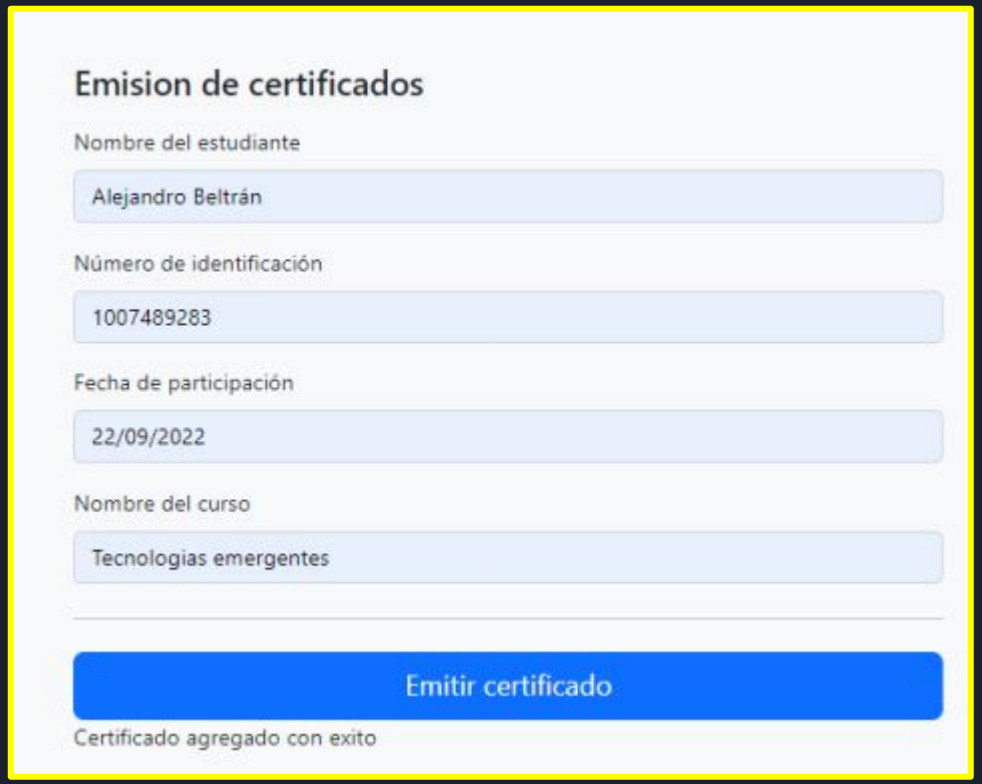
```
//Deploy del contrato en la interfaz
const Address = "0x74B8e13C1Fc19da6d1a2816bf0DA72625832ea80";
window.web3 = await new Web3(window.ethereum);
window.contract = await new window.web3.eth.Contract( ABI, Address);
document.getElementById("contractArea").innerHTML = "connected to smart contract";
```

```
const ABI = [
{
  "constant": false,
  "inputs": [
    {
      "name": "_name",
      "type": "string"
    },
    {
      "name": "_id",
      "type": "string"
    },
    {
      "name": "_date",
      "type": "string"
    },
    {
      "name": "_course",
      "type": "string"
    }
  ],
  "name": "addCretificate",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
},
```



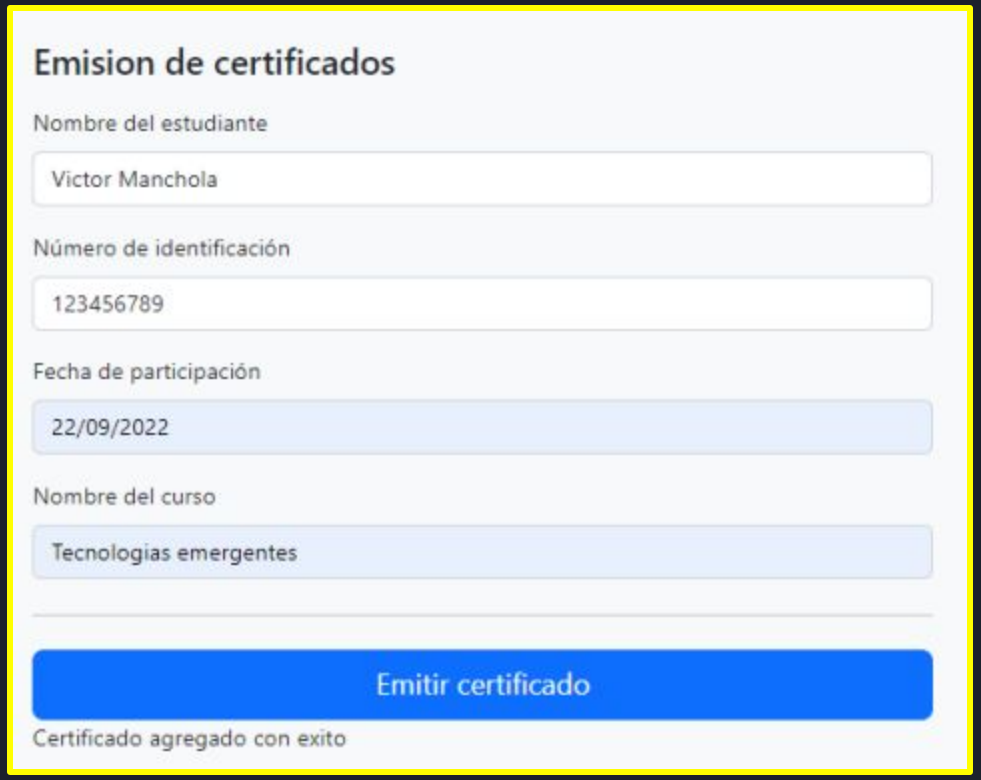



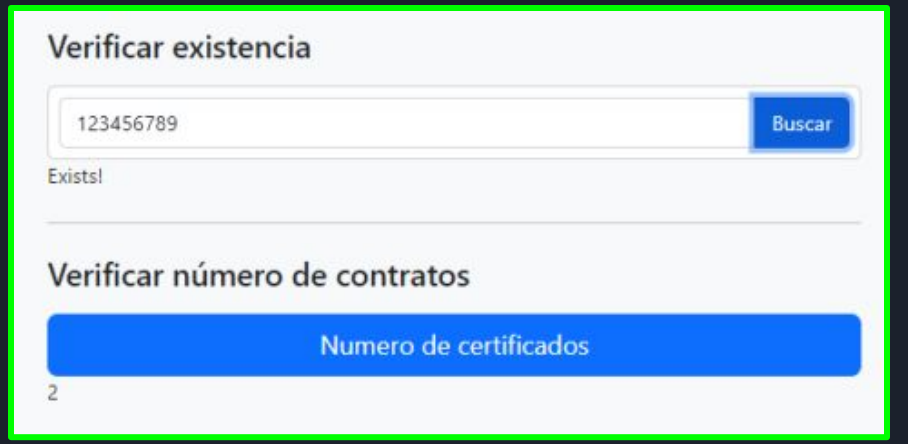

# Código en Funcionamiento






# Código en Funcionamiento





# Código en Funcionamiento

# Problemas Encontrados

- Pocas dificultades en Parte 1 gracias a *OpenZeppelin*
- Comparación de strings en *Solidity* por lo que se dificulto realizar la parte de la creación del contrato inteligente
- **Integración** (interfaz web gráfica y contrato inteligente) fue la parte que tomó más tiempo
    - Diferentes problemas en cuanto a la **testnet**
    - Integración diferente a la vista en clase
    - Problemas al agregar nuevos certificados

# CONCLUSIONES

Muchas gracias...