Tecnologías de registro distribuído y Blockchain

Práctica 1: Introducción a la Programación de Smart Contracts en Solidity

Álvaro Nóvoa Fernández

Índice

Ejercicio 1

Se ha creado una fábrica donde se fabrican o ensamblan diferentes productos inteligentes.

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.10;

contract FabricaContract{

	uint private idDigits = 16;

	struct Producto {
		string nombre;
		uint identificacion;
	}

	Producto[] public productos;

	function crearProducto(string memory _nombre, uint _id) private{
		Producto memory nuevoProducto = Producto(_nombre, _id);
		productos.push(nuevoProducto);
		emit NuevoProducto(productos.length-1, _nombre, _id);
	}

	function generarIdAleatorio(string memory _str) private view  returns(uint){
		uint rand = uint(keccak256(abi.encodePacked(_str)));
		uint idModulus = 10^idDigits;
		return rand % idModulus;
	}

	function crearProductoAleatorio(string memory _nombre) public{
		uint randId = generarIdAleatorio(_nombre);
		crearProducto(_nombre, randId);
	}

	event NuevoProducto(uint ArrayProductId, string nombre, uint id);

	mapping (uint => address) public productoAPropietario;

	mapping  (address => uint) propietarioProductos;

	function Propiedad(Producto memory _producto) private {
		address direccion = msg.sender;
		productoAPropietario[_producto.identificacion] =  direccion;
		propietarioProductos[direccion]++;
	}

	function deProductosPorPropietario(address _propietario) external view  returns(uint[] memory){
		uint contador = 0;
		uint contador2 = 0;
		uint[] memory resultado= new uint[](productos.length);
		for (contador; contador < productos.length; contador++)
		{
			if(productoAPropietario[productos[contador].identificacion] == _propietario){
				resultado[contador2] = (productos[contador].identificacion);
				contador2++;
			}
		}
		return resultado;
	}
}
```

Ejercicio 2

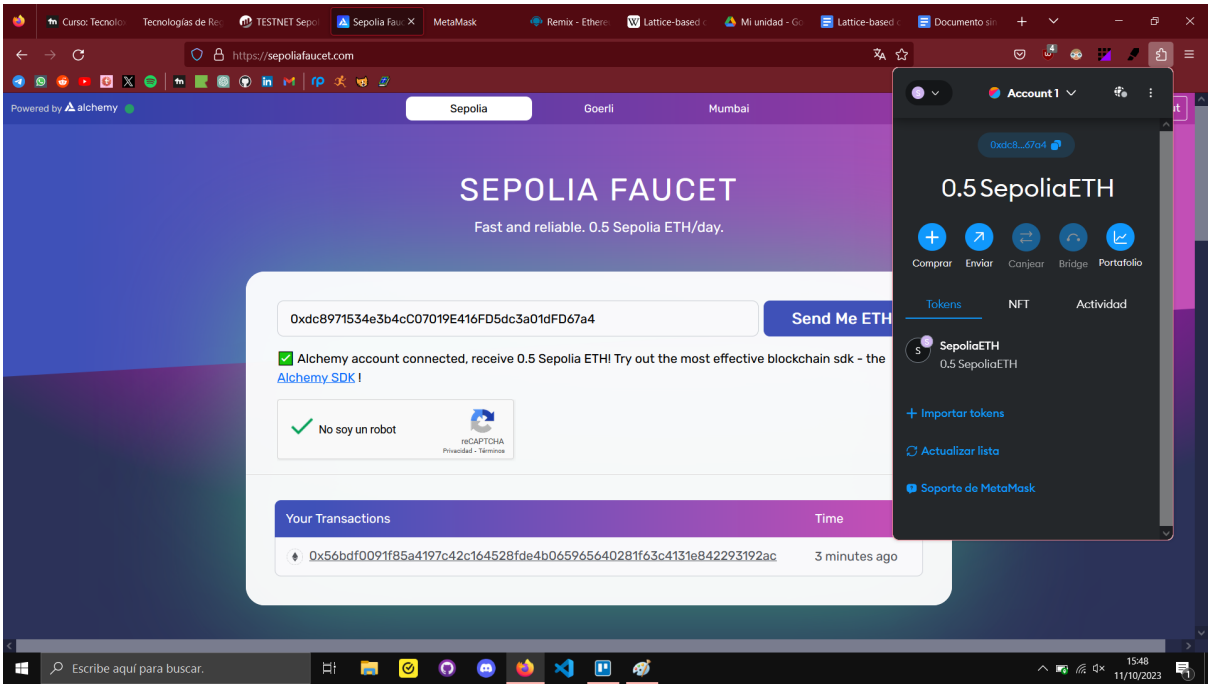
Implementación de un contrato para que cualquier token pueda ser comprado con Ether, siempre que el propietario todavía tenga suficientes tokens.

```
// SPDX-License-Identifier: Unlicenced
pragma solidity 0.8.18;
contract TokenContract {

	address public owner;

	struct Receivers {
		string name;
		uint256 tokens;
		uint256 _ether;
	}

	mapping(address => Receivers) public users;

	modifier onlyOwner(){
		require(msg.sender == owner);
		_;
	}

	constructor(){
		owner = msg.sender;
		users[owner].tokens = 100;
	}

	function double(uint _value) public pure returns (uint){
		return _value*2;
	}

	function register(string memory _name) public{
		users[msg.sender].name = _name;
	}

	function giveToken(address _receiver, uint256 _amount) onlyOwner public{
		require(users[owner].tokens >= _amount);
		users[owner].tokens -= _amount;
		users[_receiver].tokens += _amount;
	}

	function buyTocken(address _receiver, uint256 _amount, uint256 _ether) public  {
		require(users[owner].tokens >= _amount);
		require((_ether / 5) == _amount);
		require(users[_receiver]._ether >= _ether);
		users[owner].tokens -= _amount;
		users[owner]._ether += _ether;
		users[_receiver].tokens += _amount;
		users[_receiver]._ether -= _ether;
	}
}
```

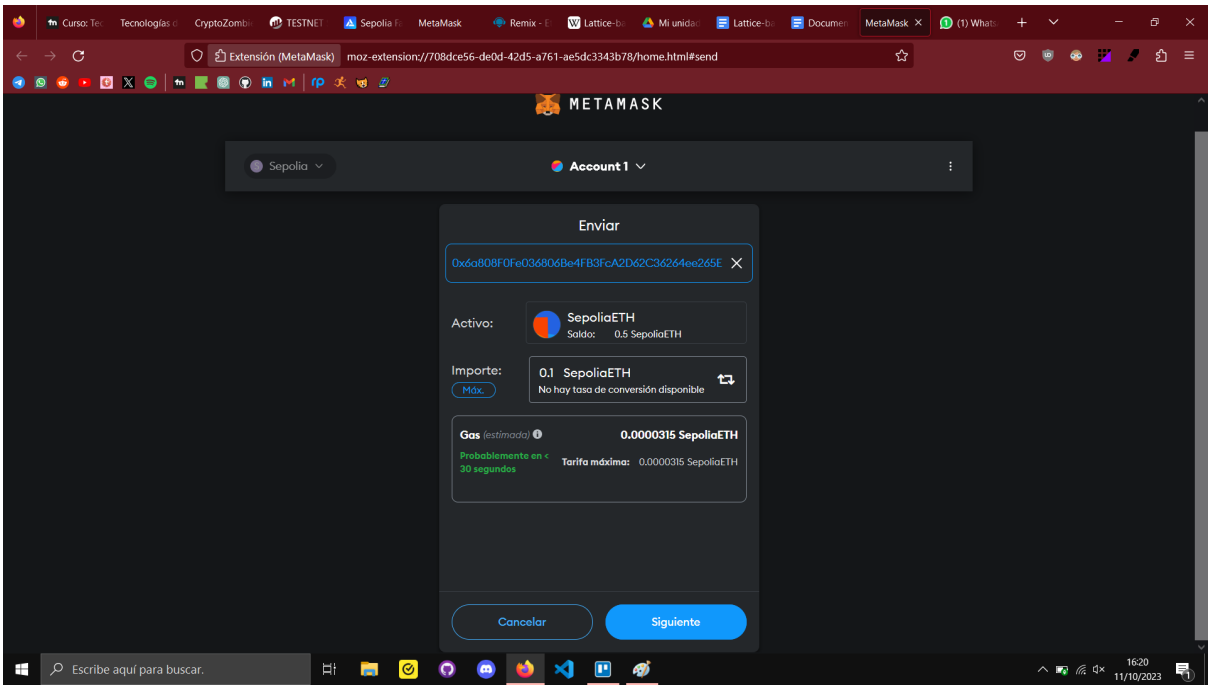
Ejercicio 3

Obtener, enviar ETH de prueba de Sepolia a un compañero y realizar un seguimiento de las transacciones en Sepolia Blockchain Explorer.

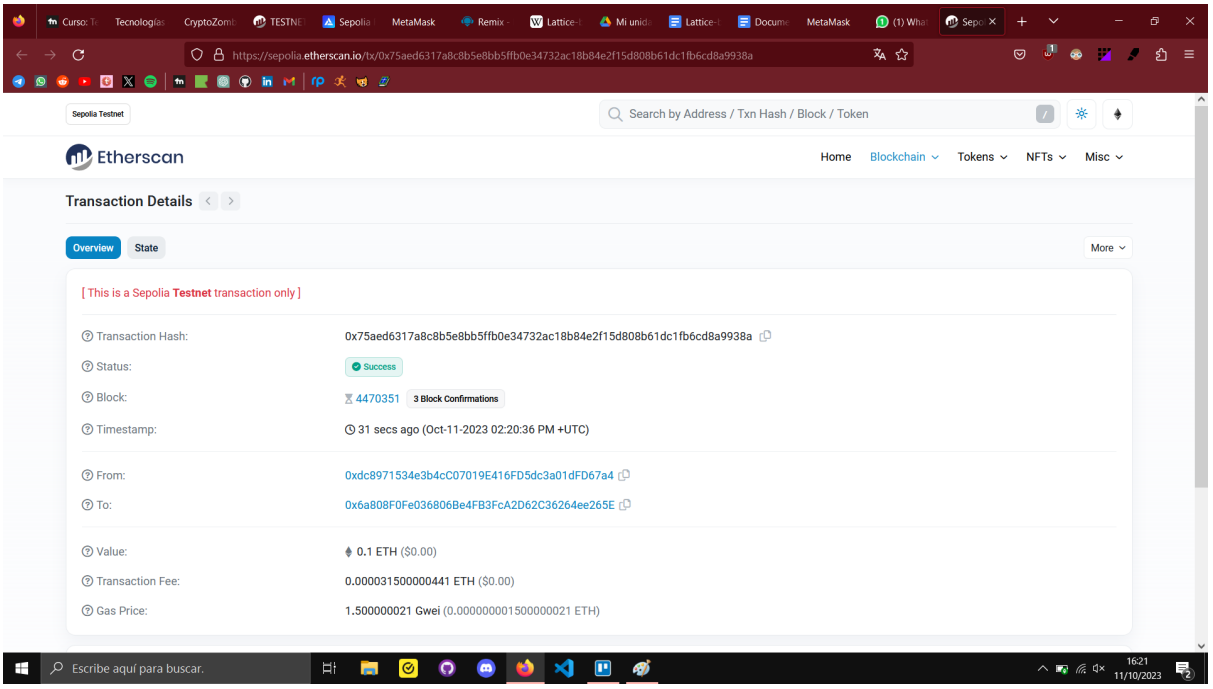
Sepolia ETH obtenido:



Envío de ETH:

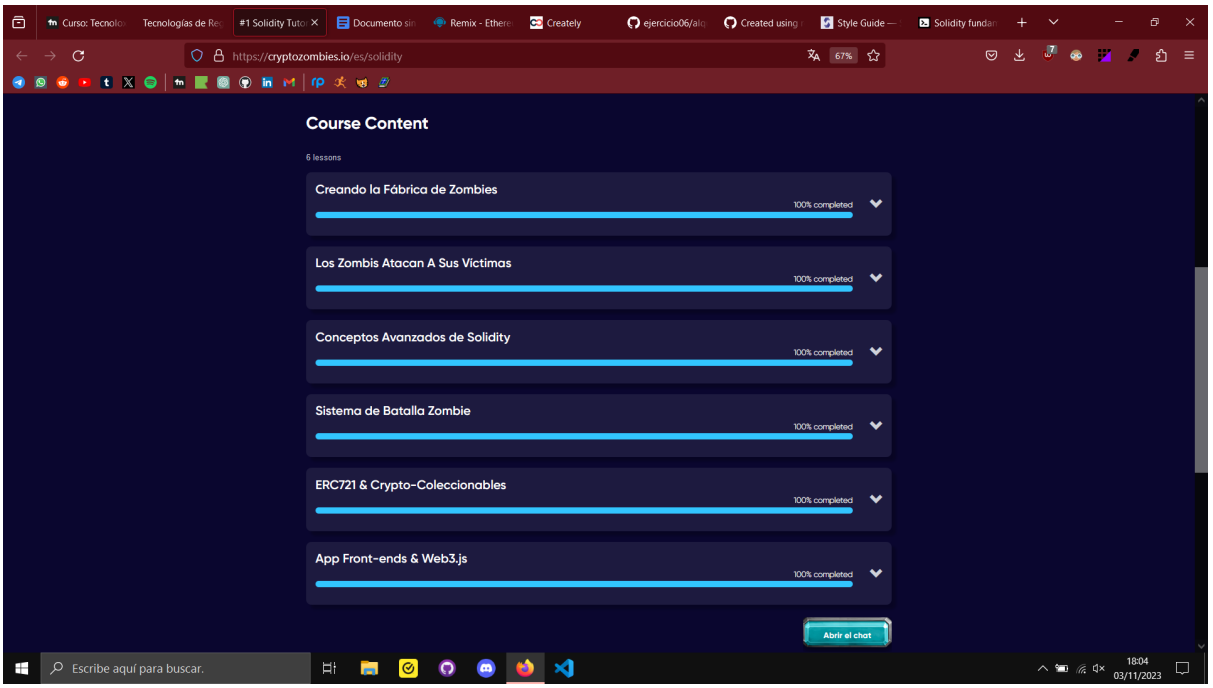


Seguimiento de transacciones:



Ejercicio 4

Completar Solidity: Beginner to Intermediate Smart Contracts.



Ejercicio 5

Escoger y probar uno de los contratos de OpenZeppelin.

```
// contracts/access-control/Auth.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Auth {
	address private _administrator;

	constructor(address deployer) {
		// Make the deployer of the contract the administrator
		_administrator = deployer;
	}

	function isAdministrator(address user) public view returns (bool) {
		return user == _administrator;
	}
}
```

Ejercicio 6

Análisis y definición del escenario

Un contrato que permite a los inquilinos reservar y acceder a una propiedad de alquiler por un período determinado después de realizar un pago en criptomonedas.

Diseño

Caso de uso

Identificador	Alquiler de una propiedad
Descripción	El sistema deberá permitir al inquilino solicitar el alquiler de una propiedad
Secuencia Normal	1.- El usuario realiza una petición al contrato
	2.- El sistema comprueba si la cantidad de criptomoneda es suficiente para la realización del contrato
	2a.- Si el usuario posee suficiente crédito, se le asigna la propiedad durante el tiempo especificado y se cobra el precio de su alquiler al usuario
	2.b- Si el usuario no posee suficiente crédito, se deniega la transacción
Importancia	De alta importancia para el funcionamiento de la aplicación

Contenido del contrato inteligente

Datos

Propiedad - Estructura de la propiedad a alquilar. Contiene:
- **nombre** - Nombre de la propiedad.
- **dirección** - Dirección física de la propiedad.
- **estado** - Estado físico de la propiedad.
- **precioDia** - Precio de la propiedad por día.

Alquiler - Estructura para almacenar información sobre cada alquiler.
- **inquilino** - Dirección del inquilino.
- **inicio** - Fecha de inicio del alquiler.
- **fin** - Fecha de finalización del alquiler.

propiedadADuenho - Mapping que relaciona una propiedad con su dueño.

duenhoAPropiedad - Mapping que relaciona a un usuario con el número de propiedades que posee.

propiedadAAlquiler - Mapping que relaciona una propiedad con un array de alquileres.

Funciones

reservar(uint _idPropiedad, uint _inicioAlquiler, uint _finAlquiler) public payable {}

Permite a un inquilino alquilar una propiedad.

renovar(uint _idPropiedad, uint _inicioAlquiler, uint _finAlquiler) public payable {}

Permite a un inquilino volver a alquilar una propiedad que ya ha alquilado anteriormente.

cancelarReserva(uint _idPropiedad, uint _inicioAlquiler, uint _finAlquiler) public {}

Permite a un inquilino cancelar un alquiler.

anhadirPropiedad(string memory _nombre, string memory _direccion, string memory _estado, uint _precioDia) public {}

Permite a un propietario añadir una propiedad.

eliminarPropiedad(uint _idPropiedad) public onlyOwnerOf(_idPropiedad) {}

Permite a un propietario eliminar una de sus propiedades.

modificarPropiedad(uint _idPropiedad, string memory _nombre, string memory _direccion, string memory _estado) public onlyOwnerOf(_idPropiedad) {}

Permite a un propietario modificar una de sus propiedades.

cancelarReservaInquilino(uint _idPropiedad, uint _inicioAlquiler, uint _finAlquiler, address _inquilino) public onlyOwnerOf(_idPropiedad) {}

Permite a un propietario cancelar el alquiler de un inquilino.

cancelarReservasInquilino(uint _idPropiedad, address _inquilino) public onlyOwnerOf(_idPropiedad) {}

Permite a un propietario cancelar todos los alquileres de un inquilino.

cancelarReservasPropiedad(uint _idPropiedad) public onlyOwnerOf(_idPropiedad) {}

Permite a un propietario cancelar todos los alquileres de una propiedad.

_terminoElAlquiler(uint _idPropiedad, address _inquilino) internal view returns (bool) {}

Comprueba si el alquiler de un inquilino ya ha terminado.

_rangoAlquileres(uint _idPropiedad, uint _inicioAlquiler, uint _finAlquiler) internal view returns (bool) {}

Comprueba si una propiedad está alquilada durante un periodo de tiempo.

_esInquilino(uint _idPropiedad, address _inquilino) internal view returns (bool) {}

Comprueba si un inquilino ha alquilado una propiedad.

_eliminarAlquiler(uint _idPropiedad, uint _id) internal {}

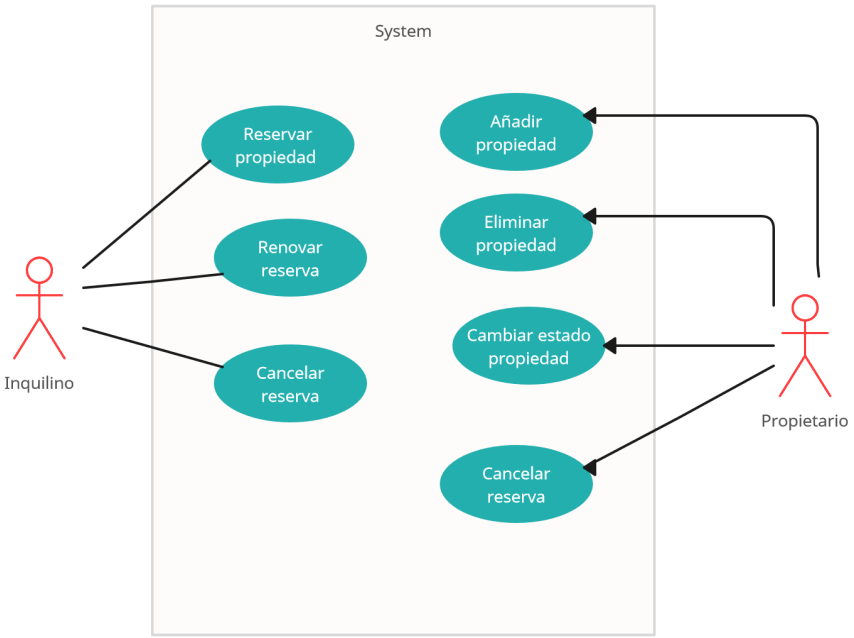
Función interna para eliminar alquileres.

transferirFondosAlDuenho() public {}

Función para acceder a los fondos del contrato.

Usuarios del sistema

Dueños de propiedades: Usuarios que ofrezcan propiedades a alquilar.
Inquilinos: Usuarios que deseen alquilar propiedades.



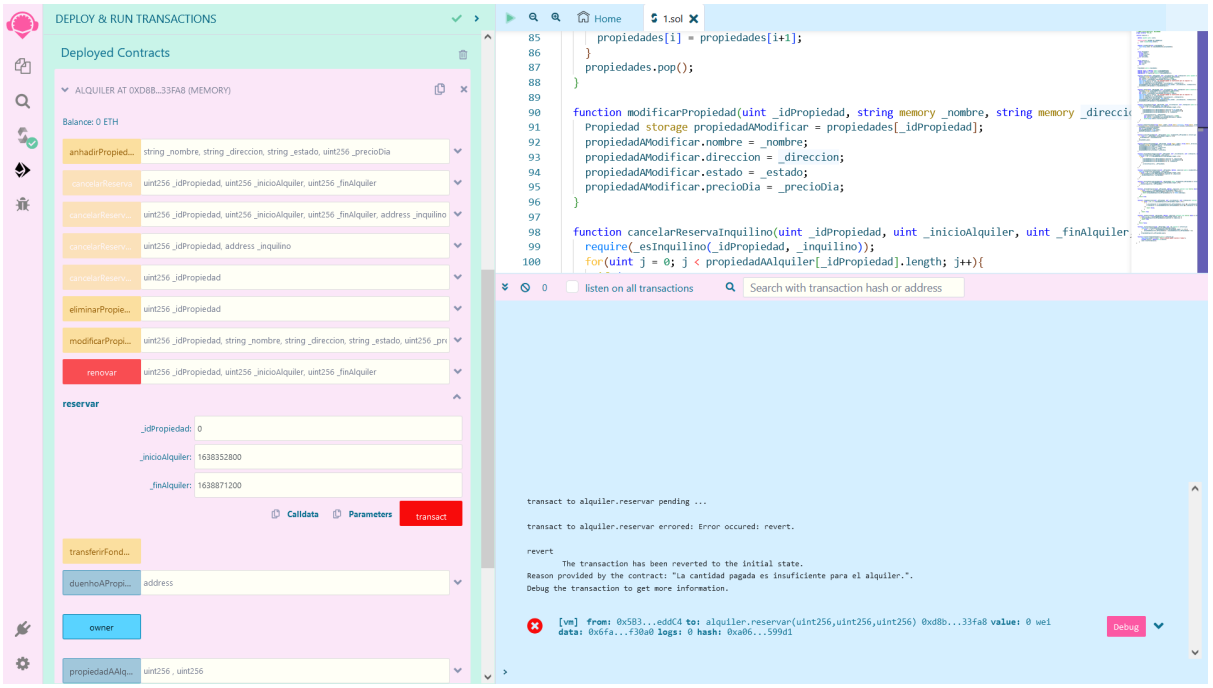
Implementación

Disponible en github.

Pruebas

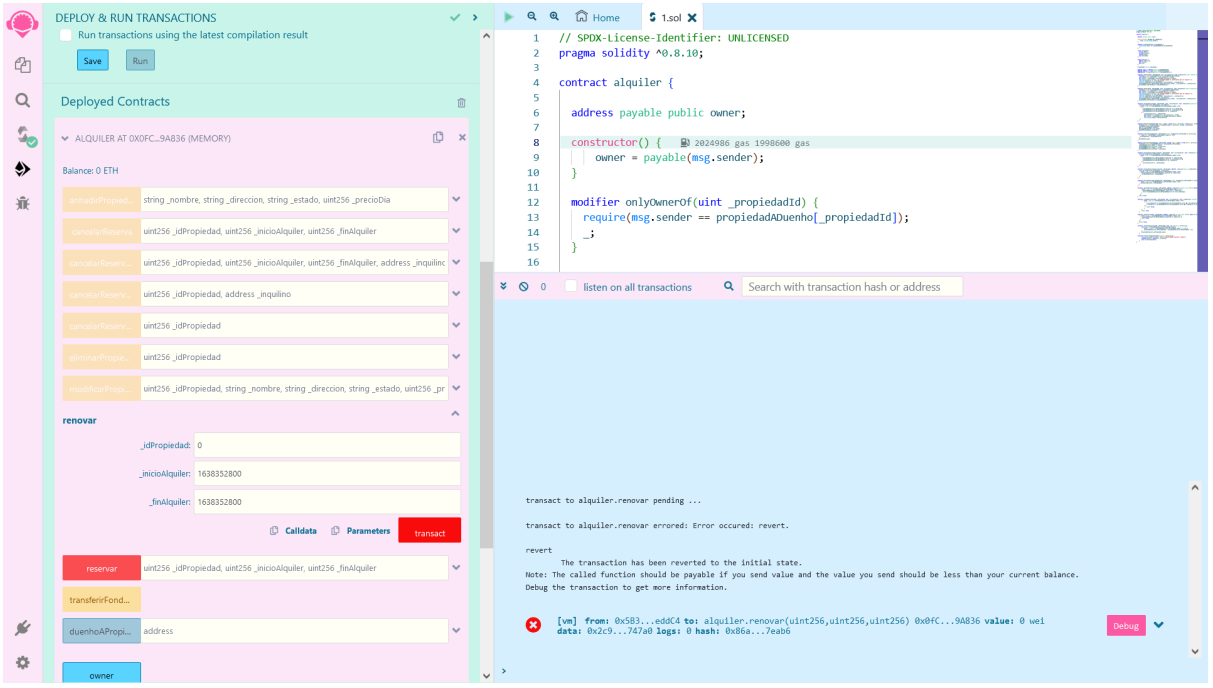
Prueba reservar

Aquí se puede observar cómo la transacción ha sido cancelada porque no tengo suficiente ETH cómo para alquilar la propiedad.

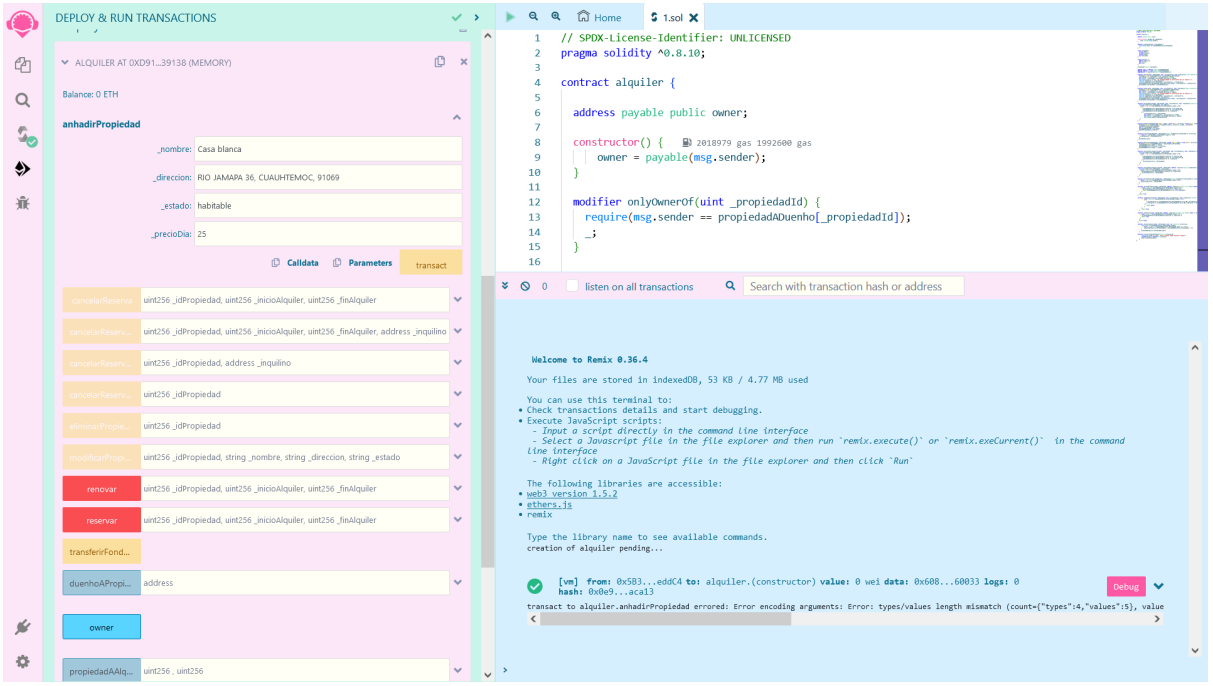


Prueba renovar

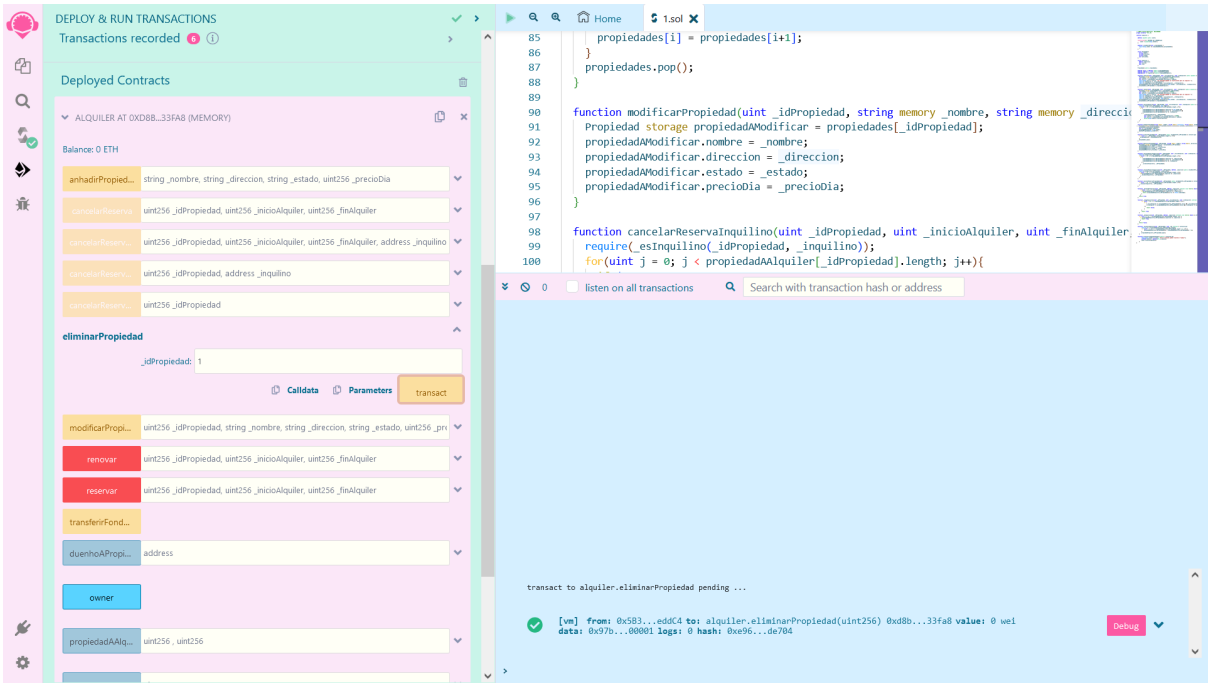
Lo mismo se puede observar con renovar.



Prueba anhadirPropiedad



Prueba eliminarPropiedad



Prueba modificarPropiedad