|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №6**

по дисциплине «Разработка клиент-серверных приложений»

**Студент группы** ИКБО-20-19 Анваржонов Ж. Т.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** ассистент Зарипов Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Москва 2022

# Задание

Написать и задеплоить смарт-контракты моделирующие процессы покупки и обмена ценными бумагами (возможно выполнение индивидуального варианта).

Написать unit-тесты.

Продемонстрировать работу через встроенные инструменты Remix IDE.

# Ход работы

Для начала напишем смарт контракт, Листинг 1.

Листинг 1 - donations.sol

// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.4;

contract donations{

struct Donation {

uint id;

uint amount;

string donor;

string message;

uint timestamp; //seconds since unix start

}

uint amount = 0;

uint id = 0;

mapping(address => uint) public balances;

mapping(address => Donation[]) public donationsMap;

function donate(address \_recipient, string memory \_donor, string memory \_msg) public payable {

require(msg.value > 0, "The donation needs to be >0 in order for it to go through");

amount = msg.value;

balances[\_recipient] += amount;

donationsMap[\_recipient]

.push(Donation(id++,amount,\_donor,\_msg,block.timestamp));

}

function withdraw() public {

amount = balances[msg.sender];

balances[msg.sender] -= amount;

require(amount > 0, "Your current balance is 0");

(bool success,) = msg.sender.call{value:amount}("");

if(!success){

revert();

}

}

function balances\_getter(address \_recipient) public view returns (uint){

return balances[\_recipient];

}

function getBalance() public view returns(uint) {

return msg.sender.balance;

}

}

Контракт позволяет жертвовать эфир, оставляя получателю сообщения, и выводить пожертвования на счет.

Затем скомпилируем и задеплоим контракт, Рисунок 1.

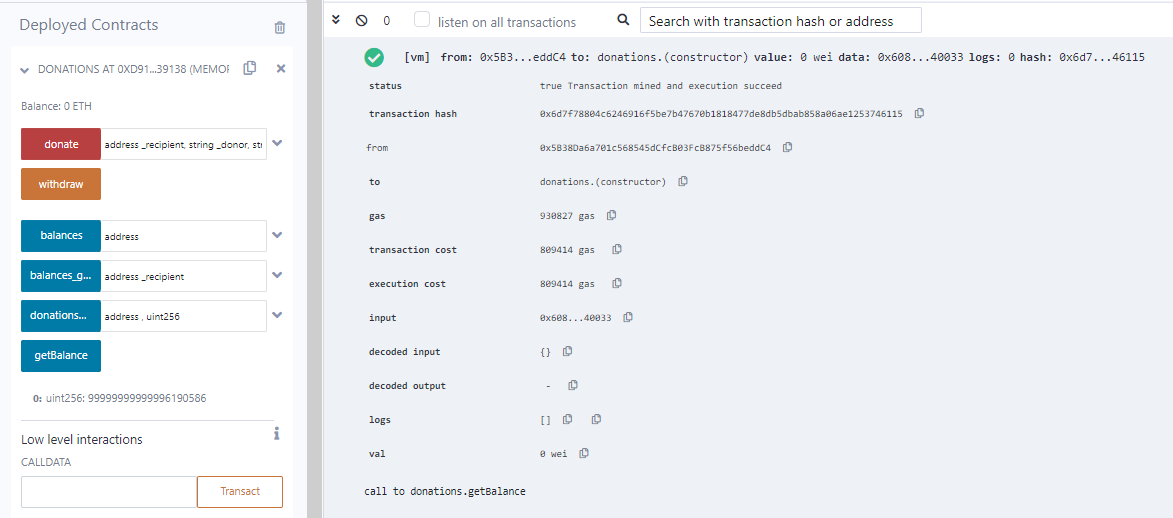


Рисунок 1 – Результат деплоймента

Затем напишем юнит-тесты и проведем тестирование при помощи Solidity Unit Testing, Листинг 2.

Листинг 2 – donations\_tests.sol

pragma solidity ^0.8.4;

import "remix\_tests.sol";

import "remix\_accounts.sol";

import "../contracts/donations.sol";

contract testSuite is donations {

address acc0 = TestsAccounts.getAccount(0); //owner by default

address acc1 = TestsAccounts.getAccount(1);

address acc2 = TestsAccounts.getAccount(2);

address acc3 = TestsAccounts.getAccount(3);

address recipient = TestsAccounts.getAccount(4); //recipient

/// #value: 1000000

/// #sender: account-1

function donateAcc1AndCheckBalance() public payable{

Assert.equal(msg.value, 1000000, 'value should be 1 Eth');

donate(recipient, "Mario", "Are you a bird?");

Assert.equal(balances\_getter(recipient), 1000000, 'balances should be 1 Eth');

}

/// #value: 1000000

/// #sender: account-2

function donateAcc2AndCheckBalance() public payable{

Assert.equal(msg.value, 1000000, 'value should be 1 Eth');

donate(recipient, "Tom", "Are you a plane?");

Assert.equal(balances\_getter(recipient), 2000000, 'balances should be 2 Eth');

}

/// #value: 2000000

/// #sender: account-3

function donateAcc3AndCheckBalance() public payable{

Assert.equal(msg.value, 2000000, 'value should be 2 Eth');

donate(recipient, "Maria", "Are you a car?");

Assert.equal(balances\_getter(recipient), 4000000, 'balances should be 4 Eth');

}

/// #sender: account-4

function withdrawDonations() public payable{

uint initialBal = getBalance();

withdraw();

uint finalBal = getBalance();

Assert.equal(finalBal-initialBal, 4000000, 'balances should be 4 Eth');

}

}

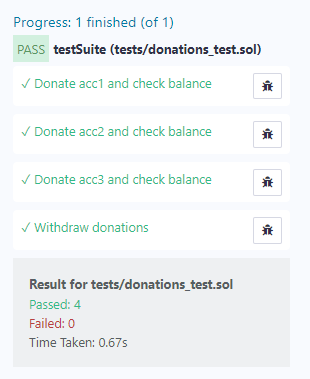


Рисунок 2 - Результаты тестирования

# Вывод

В ходе выполнения практической работы были получены навыки создания смарт-контрактов на solidity. Также были получены навыки размещения контрактов и их тестирования в IDE Remix.