Лабораторна робота 6 Розробка смарт-контракту

Теоретична частина в матеріалі лекції 11

Варіант - остання цифра в порядковому номері списку в журналі групи

0. Смарт-контракт для голосування.
1. Смарт-контракт для підтвердження авторських прав.
2. Смарт-контракт для продажу цифрового контенту.
3. Смарт-контракт для приймання довільних платежів.
4. Смарт-контракт для оплати рахунків.
5. Смарт-контракт для укладення парі.
6. Смарт-контракт для зберігання дипломів.
7. Смарт-контракт для протекції угоди третьою стороною.
8. Смарт-контракт для угоди з мультипідписами.
9. Смарт-контракт для ... тему студент може обрати сам, але її потрібно попередньо обговорити з викладачем.

Слід розробити функції щонайменше для двох ролей - власник контракту і користувач контракту. У контракті мають бути присутні складні типи даних. У контракті бажано використовувати платіжні функції.

1. Сформулюйте постановку задачі до наступного заняття.
2. До кінця семестру розробіть контракт і протестуйте його на платформі Remix + MetaMask + JavaScript.

Додаток. Контракт для продажу промокодів

Мета:- створити контракт для продажу промокодів з web-сторінки.

Користувачі:

- адміністратор,
- покупець.

Функції адміністратора:

- створення контракту,
- завдання нових промокодів,
- знищення контракту.

Функції покупця:

- оплата промокоду,
- отримання оплаченого промокоду.

Змінні контракту:

- адреса творця контракту (адміністратора),
- поточний промокод,
- хеш-таблиця оплачених промокодів (ключ - адреса платника, значення - сам промокод).

Завдання нового промокоду може виконати адміністратор у будь-який момент, старі промокоди не зберігаються.

Покупець **оплачує** промокод, пересилаючи суму не менше 0.01 eth. При цьому в хеш-таблицю оплачених промокодів додається новий елемент (або змінюється старий): ключ - адреса платника, значення - сам промокод.

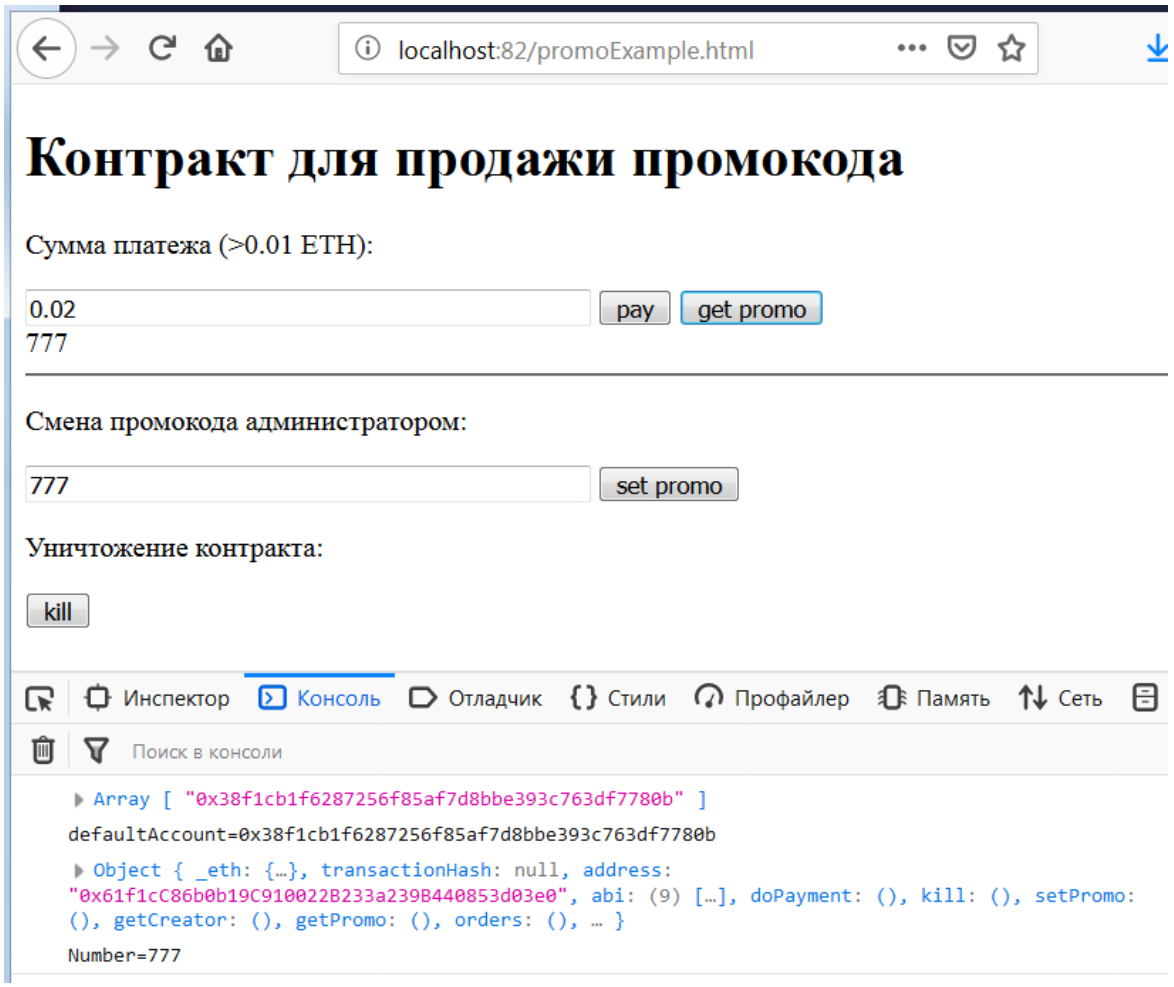
Після оплати промокоду покупець може **необмежену** кількість разів викликати функцію отримання оплаченого промокоду (поки адміністратор не змінить промокод на новий). Під час виклику функції отримання контракту перевіряється, чому дорівнює значення елемента хеш-таблиці з ключем-адресою. Якщо воно дорівнює **поточному** промокоду, це означає, що з цієї адреси надійшла оплата цього промокоду, **повертається** значення промокоду.

Переказувати оплату безпосередньо на рахунок контракту забороняється.

Видалити контракт може тільки його творець. При видаленні контракту всі накопичені на ньому ефіри переводяться на рахунок творця контракту.

```
pragma solidity ^0.5.1;

contract MyContract{
      адреса творця кредиторської
      заборгованості; uint
      myPromo;
      mapping(address => uint) public orders;

      constructor() public {
           creator=msg.sender;
           myPromo=123;
      }
      function () external payable {
     }

     function getPromo() public view returns(uint)
     {
         require(orders[msg.sender] == myPromo);
         return myPromo;
     }

     event PaymentEvent(string message);

      function doPayment() public payable
      {
          require(msg.value >= 0.01 ether);
          orders[msg.sender]=myPromo;
          emit PaymentEvent("платіж відправлено");
      }
      function getCreator() public view returns (address) {
           return creator;
      }
      function setPromo(uint promo) public {
          if(msg.sender == creator) {
          myPromo=promo;
          }
      }
```

```
    function kill() public {
        if(msg.sender == creator) {
            самознищення(creator);
        }
    }
}
```



HTML-сторінка для тестування контракту

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<script type="text/javascript">
    var abi=[
    {
        "constant": false,
        "inputs": [],
        "name": "doPayment",
        "виходи": [],
```

```
            "payable": true,
            "stateMutability": "payable",
            "type": "function"

      },
      ...
];



var address = " 0x61f1cC86b0b19C910022B233a239B440853d03e0";

window.addEventListener('load', async() => {
   // Сучасні      dapp браузери...
   if (window.ethereum) {
      window.web3 = new Web3(ethereum);
     // Запитується доступ до рахунків у Metamask
      await ethereum.enable();
    }
   // Застарілі dapp браузери...
    else if (window.web3) {
      window.web3 = new Web3(web3.currentProvider);
    }
   // Non-dapp браузери...
    else {
      console.log ('Ваш браузер не підтримує Ethereum!
Встановіть розширення MetaMask!');
            }
});

function paymentExample() {
     web3.eth.getAccounts(function(error, acc) {
          defaultAccount=acc[0];
          console.log(acc);
          console.log("defaultAccount="+defaultAccount);
          contract = web3.eth.contract(abi).at(address);
          console.log(contract);
          sum = document.getElementById("sum").value;
          sum = web3.toWei(sum, 'ether');
          contract.doPayment.sendTransaction(
               {
               from: defaultAccount,
               value: sum
               },
               function(error, data) {
                    if(error!=null) console.log(error);
```

```
console.log("Data="+data);
```

```
                });
        });
}

function kill() {
        web3.eth.getAccounts(function(error, acc) {
                defaultAccount=acc[0];
                console.log(acc);
                console.log("defaultAccount="+defaultAccount);
                contract = web3.eth.contract(abi).at(address);
                console.log(contract);
                contract.kill(function(error) {
                        console.log("Killed");
                });
        });
}
function setPromo() {
        web3.eth.getAccounts(function(error, acc) {
                defaultAccount=acc[0];
                console.log(acc);
                console.log("defaultAccount="+defaultAccount);
                contract = web3.eth.contract(abi).at(address);
                console.log(contract);
                newPromo = document.getElementById("new").value;
                contract.setPromo(newPromo, function(error, data)
{
                        console.log("Number="+data);
                });
        });
}



function getPromo() {
        web3.eth.getAccounts(function(error, acc) {
                defaultAccount=acc[0];
                console.log(acc);
                console.log("defaultAccount="+defaultAccount);
                contract = web3.eth.contract(abi).at(address);
                console.log(contract);
                contract.getPromo(function(error, data) {
                 console.log("Number="+data);
                 document.getElementById("newPromo").innerHTML  =
дані;           });
```

```
});
```

```
}


</script>
</head>
<body>
    <h1>Контракт для продажу промокоду</h1>
    <p>Сума платежу (>0.01 ETH):</p>
    <input type="text" size="50" id="sum" />
    <button type="button"
onClick="paymentExample();">оплатити</but
ton>
    <button type="button" onClick="getPromo();">get
promo</button>
     <div id="newPromo"></div>
     <hr>
     <p>Зміна промокоду адміністратором:</p>
    <input type="text" size="50" id="new" />
    <button type="button"
onClick="setPromo();">зазначити промо</button>
     <p>Знищення контракту:</p>
    <button type="button" onClick="kill();">kill</button>
</body>
</html>
```

Література

1. **Дрешер Д.** Основи блокчейна: вступний курс для початківців у 25 невеликих розділах / Даніель Дрешер. - М.: ДМК Прес, 2018. - 312 с.: іл.

2. **Свон М.** Блокчейн: схема нової економіки / Мелані Свон. - М.: Олімп-бізнес, 2017. - 240 с., іл. ISBN 978-5-9693-0360-7

3. **Поппер Н.** Цифрове Золото. Неймовірна історія біткойна або про те, як ідеалісти і бізнесмени винаходять гроші заново / Натаніель Поппер. - М.: Діалектика, 2016. - 75 с. ISBN 978-5- 8459-2079-9

4. **Даннен К.** Вступ до Ethereum і Solidity. Основи криптовалют і програмування блокчейнів для початківців (переклад Dannen Ch. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain, Programming for Beginners. - Apress, 2017.)

Інтернет-ресурси

MetaMask

https://metamask.io

Remix - Solidity IDE

http://remix.ethereum.org

Документація щодо Solidity

https://solidity.readthedocs.io/en/v0.5.4/

Підручник із JavaScript

http://www.wisdomweb.ru/JS/
javascript-first.php