Лабораторная работа 5

Стандарт ERC-20

Теоретический блок

Токен – это всего лишь маппинг между пользователями и числом, обозначающем количество токенов.

- Покупая токен, вы, по сути, покупаете место в этом маппинге.
- *mapping (address => uint256) public balances;*

Стандарт ERC-20 – набор правил для методов и переменных токена.

Ниже приведён полный интерфейс ERC-20 токена.

```
contract ERC20Interface {
      function totalSupply() public constant returns (uint);
      function balanceOf(address tokenOwner) public constant returns (uint balance);
      function allowance(address tokenOwner, address spender) public constant returns
(uint remaining);
      function transfer(address to, uint tokens) public returns (bool success);
      function approve(address spender, uint tokens) public returns (bool success);
      function transferFrom(address from, address to, uint tokens) public returns (bool
success);

      event Transfer(address indexed from, address indexed to, uint tokens);
      event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
 }
```

Подробнее о каждом требовании:

- **totalSupply()**– возвращает количество выпускаемых токенов.
- **balanceOf** (address tokenOwner) – возвращает количество токенов у адреса.
- **allowance**(address tokenOwner, address spender) – Функция, возвращающая разрешенное к переводу количество токенов для определённых пользователей. По сути, является геттером для маппинга allowed.
- **transfer**(address to, uint tokens) – функция, позволяющая отправить токены на конкретный адрес. Перед отправкой необходимо выполнить все возможные проверки, например: to != 0x00, для перевода хватает денег.
- **approve**(address spender, uint tokens) – Функция, разрешающая перевести определённое количество токенов с адреса на адрес. Работает с массивом allowed, содержащим количество токенов, разрешённых к переводу.
- **transferFrom**(address from, address to, uint tokens) – Функция, которая позволяет кому-то постороннему вызвать перевод денег с аккаунта на аккаунт. Функция transferFrom обращается к allowed, чтобы проверить и редактировать количество токенов, разрешённое к переводу.

Пример реализации токена ERC-20: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

Практический блок

1. Реализовать свой токен на основе стандарта ERC-20.

!!!! Требования к смарт-контракту:

- реализовать основные функции интерфейса ERC-20. Внимание!: в примере выше (*ссылка на репозиторий OpenZeppelin*) реализованы ещё доп. функции, которые не нужны!
- добавить приватную переменную состояния, которая бы хранила **имя токена** и инициализировалась в **конструкторе**. Инициализировать данную переменную значением **«Имя студента + Coin»** (например, «JolieCoin»)
- добавить функцию, которая бы возвращала имя токена
- инициализировать _totalSupply в конструкторе значением **номер студента в списке * 1000**.

2. Задеплоить свой токен в тестовую сеть **Ropsten** (использовать те же шаги, что и в предыдущих лабораторных работах)

3. Предоставить **адрес** задеплоенного токена.

4. Перевести 10 токенов на адрес «0x09EB14D29c2F856b958f68C57165A2891cA5C43E»

Задание

1) Изучить теоретический материал

2) Выполнить практическую часть лабораторной работы

3) Оформить **отчёт**

4) Оформить **файл смарт-контракта с кодом** (файл имеет расширение **.sol**) **!!!!**

Защита лабораторной работы 5: **отчёт** и **файл с кодом смарт-контракта** должны быть отправлены на портал. **То есть на портал должно быть отправлено 2 файла!**

Общие требования к отчёту

Отчёт должен содержать следующие элементы:

1) номер и название лабораторной работы,

2) ФИО студента, группу и курс

3) выполненные задания практического блока: скриншоты с etherscan.io и **!!!!!адрес смарт-контракта**

4) краткие ответы на контрольные вопросы.

В случае обнаружения плагиата (в том числе и в ответах на контрольные вопросы) отчёт не будет принят преподавателем!