МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ

по производственной практике, технологической

(проектно-технологической)

направления 09.03.04 «Программная инженерия»

Выполнил: _____
студент группы КЭ-303

Дегтярев В.А.

Проверил:_____
Доцент кафедры СП, к.ф.-м.н.
Турлакова С.У.
Дата: _____
Оценка: _____

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский

ЗАДАНИЕ
по производственной практике

1. Тема работы

Разработка смарт-контракта для работы с биометрическими токенами.

2. Исходные данные к работе

2.1. Нараян Прасти. Блокчейн. Разработка приложений, 2018 г.;
2.2. Документация Solidity [Электронный ресурс]. URL: https://docs.soliditylang.org/.

3. Перечень подлежащих разработке вопросов

3.1. Изучить разработку смарт-контрактов;
3.2. Спроектировать смарт-контракт;
3.3. Реализовать смарт-контракт;
3.4. Протестировать смарт-контракт.

4. Сроки

Дата выдачи задания: «27» 06 2022 г.
Срок сдачи законченной работы: «24» 07 2022 г.

Руководитель практики со стороны ЮУрГУ:

_____	_____	_____
должность, ученая степень	подпись	ФИО ответственного

Руководитель практики со стороны предприятия:

_____	_____	_____
должность, ученая степень	подпись	ФИО ответственного

Задание принял к исполнению:

	_____	_____
	подпись	ФИО студента

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Постановка задачи

Целью производственной практики является разработка смарт-контракта для управления биометрическими токенами. Для достижения поставленной цели необходимо решить следующие задачи:

1) изучить литературу по разработке смарт-контрактов;
2) спроектировать смарт-контракт;
3) реализовать смарт-контракт;
4) протестировать смарт-контракт.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область проекта

Смарт-контракт представляет собой программный код, работающий в среде виртуальной машины Ethereum. Он запускается на всех узлах сети, и результаты его работы также реплицируются на все узлы. В более узком смысле под смарт-контрактом понимается набор функций и данных (текущее состояние), находящихся по определенному адресу в блокчейне [1].

Блокчейн — выстроенная по определённым правилам непрерывная последовательная цепочка блоков, содержащих информацию. Связь между блоками обеспечивается не только нумерацией, но и тем, что каждый блок содержит свою собственную хеш-сумму и хеш-сумму предыдущего блока [2].

Невзаимозаменяемый токен (NFT) — это токен на базе блокчейна, который предоставляет право собственности на конкретный актив, например, изображение, видео или биометрику человека [3].

1.2. Анализ аналогичных проектов

CryptoKitties [4]

CryptoKitties — онлайн-игра в форме децентрализованного приложения, которая позволяет игрокам покупать, продавать, собирать и разводить виртуальных кошек разных типов. Каждая кошка является уникальным токеном. Проект использует блокчейн Ethereum. Смарт-контракт проекта написан на языке Solidity и реализует стандарт ERC721.

PhotoChromic [5]

Сервис предназначен для создания биометрики человека и последующего хранения этих данных в виде токена в блокчейне. Проект

использует блокчейн Ethereum. Смарт-контракт проекта написан на языке Solidity и реализует стандарт ERC721.

Выводы по первой главе

После анализа источников было принято решение использовать блокчейн Ethereum и язык программирования Solidity. Разрабатываемый смарт-контракт должен реализовывать стандарт ERC721.

2. ПРОЕКТИРОВАНИЕ

2.1. Функциональные требования

Можно выделить следующий набор функциональных требований к системе:

1) система должна иметь возможность создать токен, который будет хранить биометрику человека;

2) система должна иметь возможность вывести все токены, имеющиеся у владельца;

3) система должна иметь возможность отправить токены;

4) система должна иметь возможность вывести владельца токена;

5) система должна иметь возможность сжеть токен.

2.2. Нефункциональные требования

Можно выделить следующий набор нефункциональных требований к системе:

1) система должна быть написана с помощью языка программирования Solidity;

2) система должна быть развернута в сети блокчейн Ethereum;

2.3. Функции смарт-контракта

Исходя из требований, в смарт-контракте должны быть реализованы следующие функции:

1) функция, которая создает токен с биометрикой;

2) функция, которая возвращает все токены, имеющиеся у владельца;

3) функция, которая позволяет отправить токен от одного владельца к другому;

4) функция, которая возвращает владельца токена;

5) функция, которая позволяет владельцу сжеть токен.

Выводы по второй главе

В процессе анализа требований были составлены функциональные и нефункциональные требования. Кроме того, были определены функции, которые необходимо реализовать в смарт-контракте.

3. РЕАЛИЗАЦИЯ

Каждый токен имеет уникальный числовой идентификатор. В смарт-контракте токен реализует стандарт ERC721 [1]. Это означает, что должны быть реализованы определенные функции стандарта. Код функций представлен в листинге 1.

Листинг 1 – Необходимые для реализации функции стандарта ERC721

```
//@dev Returns the number of tokens in ``owner``'s account.
function balanceOf(address owner) external view returns (uint256 balance);

//@dev Returns the owner of the `tokenId` token.
function ownerOf(uint256 tokenId) external view returns (address owner);

//@dev Transfers `tokenId` token from `from` to `to`.
function transferFrom(
        address from,
        address to,
        uint256 tokenId
    ) external;
```

Функция balanceOf возвращает количество токенов у владельца.

Функция ownerOf возвращает адрес владельца конкретного токена.

Функция transferFrom отправляет токен из одного адреса на другой.

Код реализованных функций представлен в листинге 2.

Листинг 2 – Реализованные функции balanceOf, ownerOf, transferFrom

```
function balanceOf(address owner) public view virtual override returns
(uint256) {
    require(owner != address(0), "ERC721: address zero is not a valid
owner");
    return _balances[owner];
}

function ownerOf(uint256 tokenId) public view virtual override returns
(address) {
    address owner = _owners[tokenId];
    require(owner != address(0), "ERC721: invalid token ID");
    return owner;
}

function transferFrom(
    address from,
    address to,
    uint256 tokenId
) external virtual {
    require(trueOwnerOf(tokenId) == from, "Biometric: Only the creator of
the token can send a token");
    require(to != address(0), "ERC721: transfer to the zero address");

    _beforeTokenTransfer(from, to, tokenId);
```

```
    _approve(address(0), tokenId);

    _balances[from] -= 1;
    _balances[to] += 1;
    _owners[tokenId] = to;

    emit Transfer(from, to, tokenId);

    _afterTokenTransfer(from, to, tokenId);
}
```

Функция createBiom создает токен и сохраняет в нем введенную биометрику. Код функции представлен в листинге 3.

Листинг 3 – Функция createBiom

```
function createBiom(string memory biom)
    public
    returns (uint256)
{
    uint256 newItemId = _tokenIds.current();
    address person = _msgSender();
    _mint(person, newItemId);
    _setTokenURI(newItemId, biom);

    _tokenIds.increment();
    return newItemId;
}
```

В функции _mint реализовано первоначальное присвоение токена владельцу. В функции _setTokenURI реализовано сохранение биометрики у токена. Код функций представлен в листинге 4.

Листинг 4 – Функции _mint и _setTokenURI

```
function _mint(address to, uint256 tokenId) internal virtual {
    require(to != address(0), "ERC721: mint to the zero address");
    require(!_exists(tokenId), "ERC721: token already minted");

    _beforeTokenTransfer(address(0), to, tokenId);

    _balances[to] += 1;
    _trueOwnerIdsCount[to] += 1;
    _owners[tokenId] = to;
    _trueOwners[tokenId] = to;

    emit Transfer(address(0), to, tokenId);

    _afterTokenTransfer(address(0), to, tokenId);
}

function _setTokenURI(uint256 tokenId, string memory _tokenURI) internal
virtual {
    require(_exists(tokenId), "ERC721URIStorage: URI set of nonexistent
token");
    _tokenURIs[tokenId] = _tokenURI;
}
```

В функции burn реализован процесс «сжигания» токена, то есть отправка токена на адрес 0x0000000000000000000000000000000000000000, к которому ни у кого нет доступа. Код функции представлен в листинге 5.

Листинг 5 – Функция burn

```
function burn(uint256 tokenId) internal virtual {
    address owner = ERC721.ownerOf(tokenId);
    address trueOwner = trueOwnerOf(tokenId);

    _beforeTokenTransfer(owner, address(0), tokenId);

    _approve(address(0), tokenId);

    _balances[owner] -= 1;
    _trueOwnerIdsCount[trueOwner] -= 1;
    delete _owners[tokenId];
    delete _trueOwners[tokenId];

    emit Transfer(owner, address(0), tokenId);

    _afterTokenTransfer(owner, address(0), tokenId);
}
```

В функции getTokenIdsByTrueOwner реализован возврат всех идентификаторов токенов конкретного владельца. Код функции представлен в листинге 6.

Листинг 6 – Метод getTokenIdsByTrueOwner

```
function getTokenIdsByTrueOwner(address _owner) public view returns(uint[]
memory) {

    uint[] memory result = new uint[](_trueOwnerIdsCount[_owner]);
    uint counter = 0;
    for (uint i = 0; i < _tokenIds.current(); i++) {
        if (_trueOwners[i] == _owner) {
            result[counter] = i;
            counter++;
        }
    }
    return result;
```

Выводы по третьей главе

В данной главе были реализованы все необходимые функции смарт-контракта. Адрес смарт-контракта в тестовой сети Rinkeby: 0xb72Bb84Ff57031dfbD3750034A0B191fb47eBc45 [6].

4. ТЕСТИРОВАНИЕ

Для тестирования смарт-контракта использовалось функциональное тестирование. Для тестирования была использована тестовая сеть Rinkeby [7]. В таблице 1 приведено тестирование смарт-контракта.

Таблица 1 – Тестирование смарт-контракта

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
1	Создать биометрику	Вызвать функцию createBiom с параметром b6dc933311bc2357;	В блокчейн создается токен, в котором хранится b6dc933311bc2357	Да
2	Сжечь токен	Вызвать функцию burn с параметром 0 (идентификатор ранее созданного токена)	Токен с идентификатором 0 отправляется на нулевой адрес	Да
3	Получить идентификаторы всех токенов, хранящихся на адресе	Вызвать функцию getTokenIdsByTrueOwner с параметром 0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4	Функция возвращает массив идентификаторов токенов адреса 0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4 [0, 1, 2]	Да
4	Получить количество токенов у владельца	Вызвать функцию balanceOf с параметром 0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4	Функция возвращает количество токенов адреса 0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4 [3]	Да
5	Получить адрес владельца по идентификатору токена	Вызвать функцию ownerOf с параметром 1	Функция возвращает адрес владельца токена [0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4]	Да

№	Название теста	Действие	Ожидаемый результат	Тест пройден?
6	Отправить токен на другой адрес	Вызвать функцию transferFrom с параметрами 0x89A92800Eb21ABA784239CD6cc16e74fa0FB90A4 (отправитель), 0xb5a9ae7c75107AF18d40922cC079A0a58cd9D9a4 (получатель), 1 (идентификатор токена)	Токен с идентификатором 1 отправится на адрес 0xb5a9ae7c75107AF18d40922cC079A0a58cd9D9a4	Да

Выводы по четвертой главе

В данной главе было проведено тестирование смарт-контракта. Все тесты были пройдены успешно.

ЗАКЛЮЧЕНИЕ

В ходе производственной практики был разработан смарт-контракт для управления биометрическими токенами. При этом были решены следующие задачи:

1) изучена литература по разработке смарт-контрактов;
2) спроектирован смарт-контракт;
3) реализован смарт-контракт;
4) протестирован смарт-контракт.

В будущем планируется оптимизировать смарт-контракт.

ЛИТЕРАТУРА

1. Solidity Documentation [Электронный ресурс]. URL: https://docs.soliditylang.org/ (дата обращения: 10.07.2022)

2. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System.

3. Н. Прасти, Блокчейн. Разработка приложений., 2018г..

4. Официальный сайт проекта CryptoKitties [Электронный ресурс]. URL: https://www.cryptokitties.co/ (дата обращения: 10.07.2022).

5. Официальный сайт проекта PhotoChromic [Электронный ресурс]. URL: https://photochromic.io/ (дата обращения: 10.07.2022).

6. «Смарт-контракт» [Электронный ресурс]. URL: https://rinkeby.etherscan.io/address/0xb72bb84ff57031dfbd3750034a0b191fb47ebc45

7. «Rinkeby: Network Dashboard» [Электронный ресурс]. URL: https://www.rinkeby.io/ (дата обращения: 10.07.2022).