Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий
Кафедра алгебраических и информационных систем

ДОКУМЕНТАЦИЯ
по приложению

ПРИЛОЖЕНИЕ ДЛЯ СОЦИАЛЬНОГО ТРЕЙДИНГА

Студенты 4 курса очного отделения
Группы 02461–ДБ
Сивый Дмитрий Эдуардович
Забродин Вадим Алексеевич
Окунев Кирилл Сергеевич

Руководитель:
преп. Гаврилин Д.Н.

Иркутск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

ЗАПУСК ПРИЛОЖЕНИЯ 4

ЗАПУСК ТЕСТОВОЙ СРЕДЫ 5

СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ 8

СОДЕРЖАНИЕ И ВЗАИМОДЕЙСТВИЕ КОМПОНЕНТОВ 11

ФУНКЦИИ СМАРТ-КОНТРАКТОВ 14

ВВЕДЕНИЕ

Основная цель приложение это возможность пользователей передавать свой токены в управление другим трейдерам (менеджерам).

Сформированная функциональность:

1. **доверительное управление** - пользователи могут передать свои токены менеджеру, который будет торговать ими на Uniswap
2. **контроль над средствами** - смарт-контракт должен гарантировать, что менеджер может только торговать доверенными токенами и не имеет права вывести или использовать токены в личных целях

Основные требования: **функция передачи токенов от пользователя к менеджеру** и **ограничения доступа менеджера к токенам - он может только торговать ими на Uniswap**.

ЗАПУСК ПРИЛОЖЕНИЯ

Для корректного запуска приложения нужно установить модули в серверной и клиенсткой части. Из корневой директории проекта выполните команду **npm install**. Когда необходимые модули установятся необходимо выполнить следующие команды последовательно:

1. **npx hardhat compile**
2. **npx hardhat node**
3. **npx hardhat run ./scripts/lock**

После запуска необходимо установить клиентскую часть приложения, путём следующих команд:

1. **cd ./frontend**
2. **npm install**
3. **npm run serve**

ЗАПУСК ТЕСТОВОЙ СРЕДЫ

Приложение поддерживает запуск тестовой среды, после выполнения всех команд из пункта выше в консоле вы увидите адреса и их приватные ключи. Для того, чтобы добавить данные кошельки в локальный кошелёк MetaMask необходимо подключить локальную тестовую сеть.

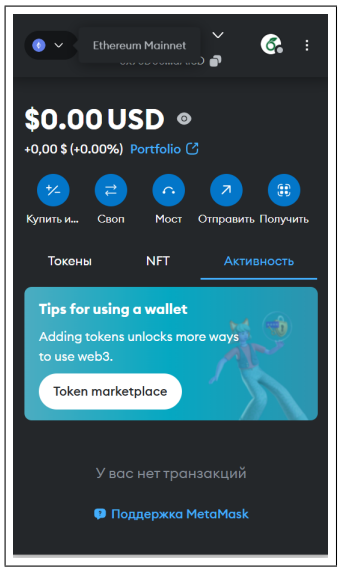

Рисунок 1. Подключение тестовый сети metamask 1

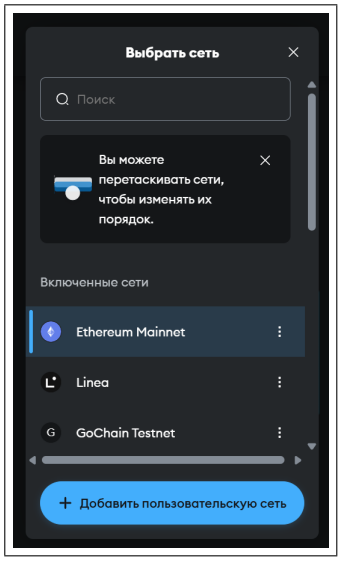

Рисунок 2. Подключение тестовый сети metamask 2

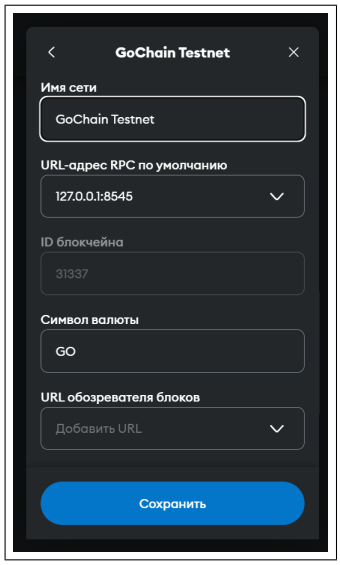

Рисунок 3. Подключение тестовый сети metamask 3

После подключения тестовый среды необходимо добавить кошелёк в приложение MetaMask следующим образом:

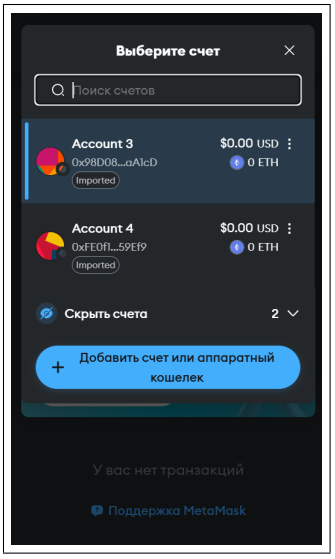

Рисунок 4. Подключение тестовый сети metamask 4

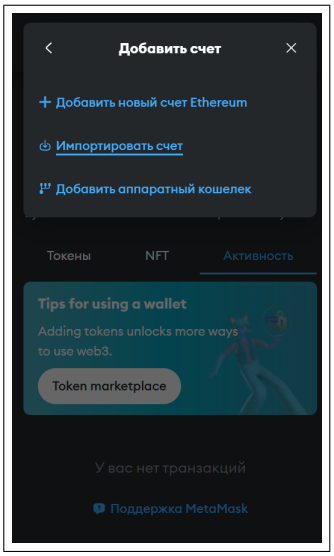

Рисунок 5. Подключение тестовый сети metamask 5

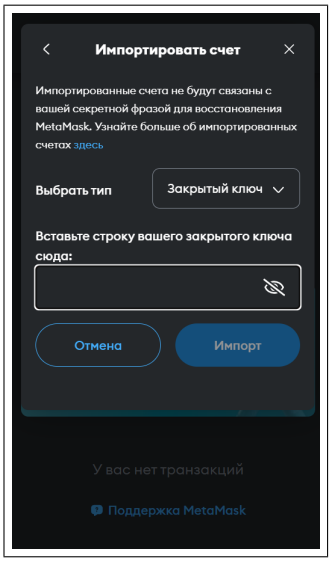

Рисунок 6. Подключение тестовый сети metamask 6

СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Пользователь при заходе на сайт видит окно 7, в котором ему необходимо авторизоваться через кошелёк metamask

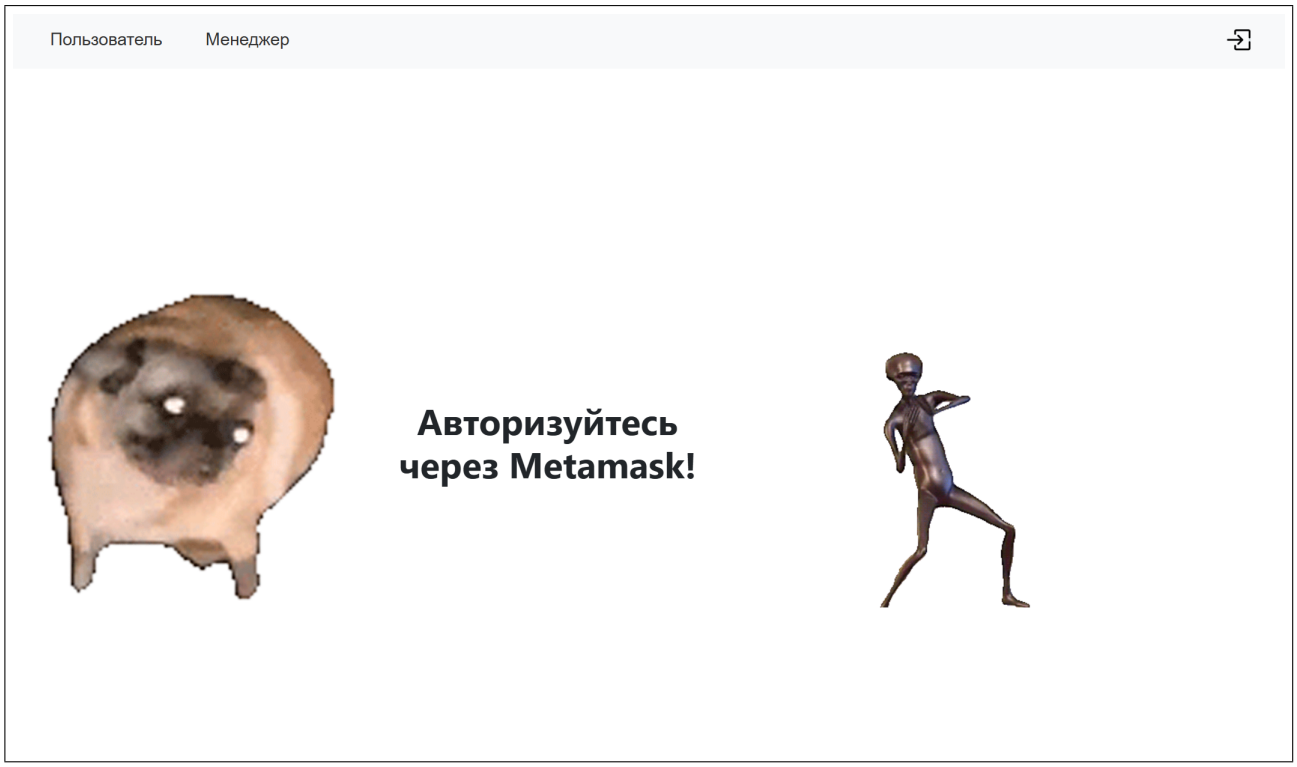

Рисунок 7. Логин в систему

Если пользователь хочет вложить свои средства, для того, чтобы другие трейдеры торговали ими, он должен авторизоваться, в кошельке метамаска, и вот так будет выглядит страница после регистрации 8.

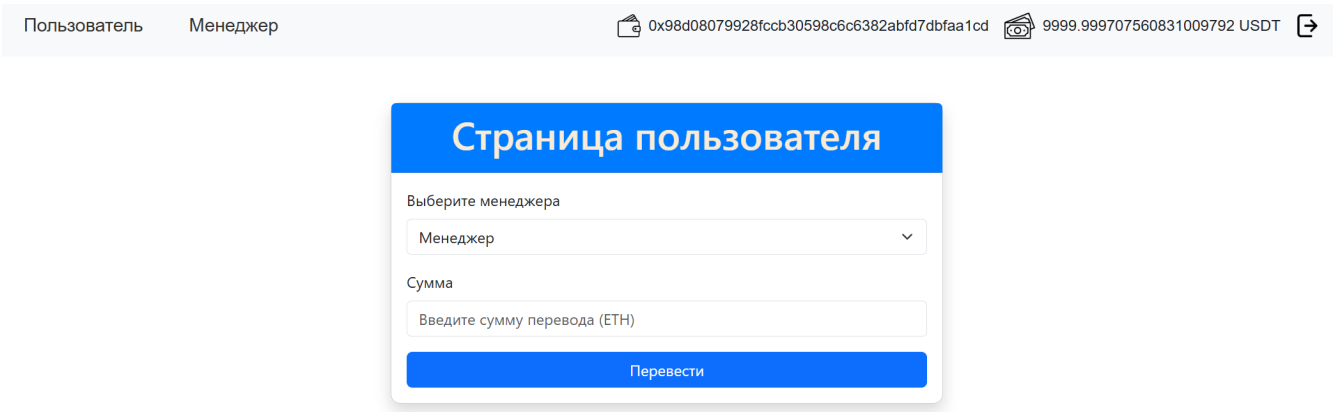

Рисунок 8. Страница пользователя

Что бы выбрать менеджера, в выпадающем списке выбираете менеджера и пишете сколько хотите отдать менеджеру.

После перевода, менеджер может открыть свою страницу 9, также зарегистрировавшись через кошелёк, и там будут написаны токены которые свободны для трейда

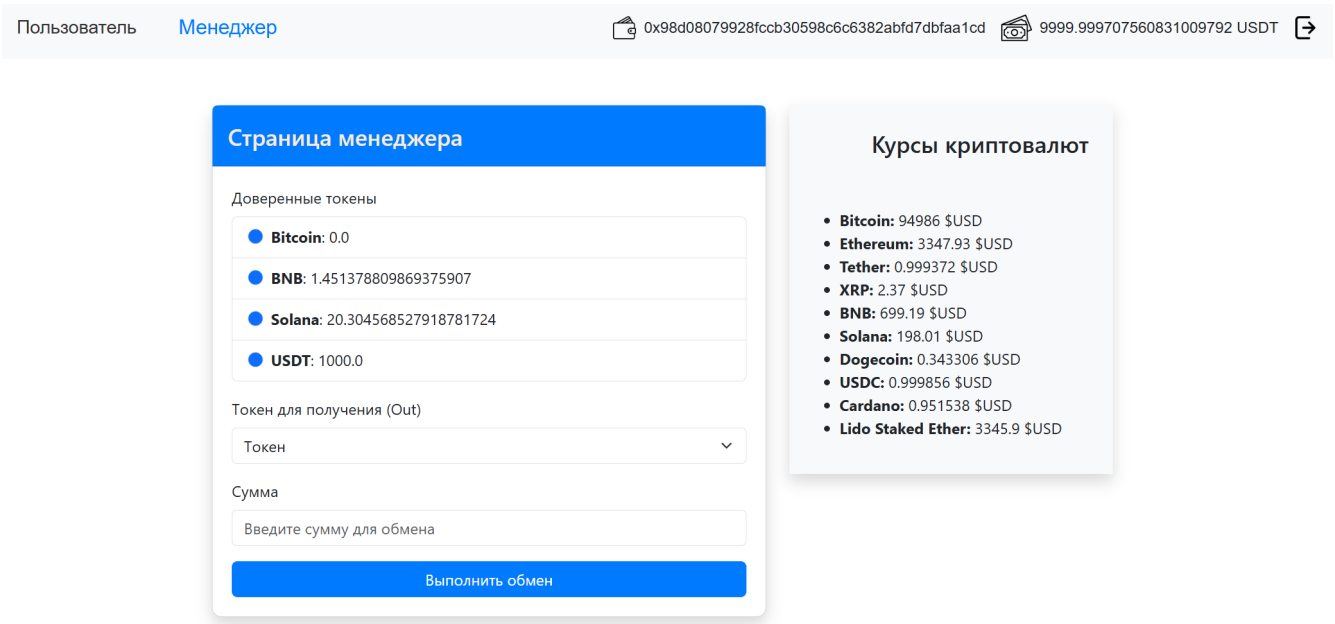

Рисунок 9. Страница менеджера

После менеджер может торговать токенами по курсу, который находится на странице, пока что количество токенов ограничено, а именно: **BTC**,**BNB**,**Solana**,**US**

СОДЕРЖАНИЕ И ВЗАИМОДЕЙСТВИЕ КОМПОНЕНТОВ

Всего в клиентской системе расположено 4 основных компонента на клиентской части: **UserPage.vue**, **ManagerPage.vue**, **HomePage.vue** и **NavBar.vue**. Рассмотрим каждый из них подробнее:

1. **UserPage.vue** - в нём расположен интерфейс для взаимодействия инвестора с системой, в нём реализовано взаимодействие контракта с текущим пользователей, а также выбор менеджера
2. **ManagerPage.vue** - в нём расположен интерфейс для взаимодействия менеджера с системой, в нём реализован функционал трейдинга валют и отображение баланса на контракте
3. **HomePage.vue** - домашняя страница, которая встречает пользователя на сайте, на ней система просит вас войти в кошелек, изображая веселые gif файлы
4. **NavBar.vue** - навигационная панель, через которую осуществляется навигация на сайте, а также отображается адрес кошелька и баланс

Всего в серверной части расположено 2 основных компонента: **Lock.js** и **Lock.sol**.

Рассмотрим каждый из них поподробнее:

1. **Lock.sol** - смарт-контракт через который происходят все обмены и передачи валют. Основные поля, с которыми взаимодействует система указана в листинге 1.

Листинг 1. Основные структуры, маппинги и ивенты

```
using SafeERC20 for IERC20;


IUniswapV2Router02 public uniswapRouter;
```

```
    IERC20 public token;
    address public baseTokenAdress;

    struct TokenBalance {
        uint256 amount; // Количество токенов
        uint256 valueInBaseToken; // Стоимость в базовом токене
        address tokenAddress;
    }



    struct User {
        uint256 balance; // Баланс в базовом токене
        address manager;
        uint256 depositTime;
        uint256 initialBalance; // Начальный баланс в базовом токене
        address[] tokenList; // Список токенов, с которыми работает пол
ьзователь
    }

    mapping(address => User) public users;
    mapping(address => uint256) public entrustedBalances;
    mapping(address => uint256) public tradingStartTime;
    mapping(address => TokenBalance) btcBalances; // Балансы других ток
енов
    mapping(address => TokenBalance) bnbBalances; // Балансы других ток
енов
    mapping(address => TokenBalance) slnBalances; // Балансы других ток
енов
    mapping(address => TokenBalance) usdBalances; // Балансы других ток
енов
    mapping(address => uint256) public tokenPrices;

    event ManagerAssigned(address indexed user, address indexed
```

```
manager);
    event TokensDeposited(address indexed manager, uint256 amount);
    event TradeExecuted(address indexed manager, address tokenIn,
address tokenOut, uint256 amountIn, uint256 amountOut);
    event FundsMovedToContract(address indexed user, uint256 amount);
    event TokensSoldAndWithdrawn(address indexed user, uint256 amount);
    event FundsReturnedAfterWeek(address indexed user, uint256 amount);
    event FundsReturnedAfterTwoWeeks(address indexed user, uint256
amount);
    event Swap(address indexed sender, address indexed tokenIn,
address indexed tokenOut, uint256 amountIn, uint256 amountOut);
```

2. **Lock.js** - файл исполняемый, который используется для локального развертывания контракта

ФУНКЦИИ СМАРТ-КОНТРАКТОВ

Функция assignManagerAdDeposit назначает менеджера и устанавливает курс токенов, а также количество токенов на контракте

Листинг 2. Выдача токенов менеджеру

```
    function assignManagerAndDeposit(address manager) external payable {
        require(users[msg.sender].manager == address(0), "Manager already
assigned");
        require(msg.value > 0, "Amount must be greater than zero");
        users[msg.sender].manager = manager;
        users[msg.sender].balance += msg.value;
        users[msg.sender].depositTime = block.timestamp;

        btcBalances[manager].amount = 94935;
        usdBalances[manager].amount = 1;
        bnbBalances[manager].amount = 689;
        slnBalances[manager].amount = 197;
        btcBalances[manager].tokenAddress =
0x7130d2A12B9BCbFAe4f2634d864A1Ee1Ce3Ead9c;
        bnbBalances[manager].tokenAddress =
0xB8c77482e45F1F44dE1745F52C74426C631bDD52;
        slnBalances[manager].tokenAddress =
0x570A5D26f7765Ecb712C0924E4De545B89fD43dF;
        usdBalances[manager].tokenAddress =
0xdAC17F958D2ee523a2206206994597C13D831ec7;
        usdBalances[manager].valueInBaseToken += msg.value;
        btcBalances[manager].valueInBaseToken = 0;
        bnbBalances[manager].valueInBaseToken = 0;
        slnBalances[manager].valueInBaseToken = 0;

        emit ManagerAssigned(msg.sender, manager);
        emit TokensDeposited(manager, msg.value);
```

```
        }
}
```

Функция trade выполняет обмен одного токена на другой, с вычетом и зачислением соответствующих счётов

Листинг 3. Обмен токенов на контрате

```
        function trade(address tokenIn, address tokenOut, uint256 amountIn,
    address manager) external {
            uint256 temp = 0;
            if(btcBalances[manager].tokenAddress == tokenIn){
                btcBalances[manager].valueInBaseToken-=amountIn;
                temp = btcBalances[manager].amount;
            }
            if(usdBalances[manager].tokenAddress == tokenIn){
                 usdBalances[manager].valueInBaseToken-=amountIn;
                 temp = usdBalances[manager].amount;
            }
            if(bnbBalances[manager].tokenAddress == tokenIn){
                bnbBalances[manager].valueInBaseToken-=amountIn;
                temp = bnbBalances[manager].amount;
            }
            if(slnBalances[manager].tokenAddress == tokenIn){
                slnBalances[manager].valueInBaseToken-=amountIn;
                temp = slnBalances[manager].amount;
            }

            if(btcBalances[manager].tokenAddress == tokenOut){

    btcBalances[manager].valueInBaseToken+=(temp*amountIn)/btcBalances[manager].amount;
            }
            if(usdBalances[manager].tokenAddress == tokenOut){

    usdBalances[manager].valueInBaseToken+=(temp*amountIn)/usdBalances[manager].amount;
```

```
            }
            if(bnbBalances[manager].tokenAddress == tokenOut){

    bnbBalances[manager].valueInBaseToken+=(temp*amountIn)/bnbBalances[manager].amount;
            }
            if(slnBalances[manager].tokenAddress == tokenOut){

    slnBalances[manager].valueInBaseToken+=(temp*amountIn)/slnBalances[manager].amount;
            }

}
```