

### Test sepolia / Адреса

Поиск сетей

- Ethereum**  
0x81Ad5...a8934
- Bitcoin**  
bc1qnmc...pc9lt
- Solana**  
7KovoG6...GDtiv
- Linea**  
0x81Ad5...a8934
- Base**  
0x81Ad5...a8934
- BNB Chain**  
0x81Ad5...a8934
- Hardhat local**  
0x81Ad5...a8934
- Polygon

The screenshot shows the Google Cloud Web3 interface. On the left, a sidebar menu includes 'Дом', 'Обнаружить', 'Кран' (selected), 'Протоколы', 'Web3 AI', 'События', 'Учиться', 'Универсальная бух...', 'Программа подде...', 'Обратная связь', and 'политика конфиде...'. The main content area is titled 'Кран Ethereum Sepolia' (BETA). A central modal window says 'Получите 0,05 Sepolia ETH'. It contains a green checkmark icon and the text 'Ваши тестовые токены находятся в очереди, это может занять несколько минут...'. Below this, it shows the network as 'Ethereum Sepolia', the recipient address as '0x81Ad508F00aF99719B9A8B116625C809CCa89...', and the transaction hash as '0x730e4ec16b269436d12e25e56cdf5347b6a23e58...'. There are also two small icons for copy and close.

MetaMask wallet connected to the Sepolia test network with test ETH balance.  
Testnet is used to demonstrate the application without real funds.

The screenshot shows the Remix IDE interface. The top bar includes 'REMX 1.5.1', 'Login with GitHub', 'Theme', and a gear icon. The left sidebar is labeled 'FILE EXPLORER' and shows files like '.deps', 'states', 'artifacts', 'contracts', 'RewardToken.sol', 'TokenizedCrowdfunding.sol', 'Test', '.prettierc.json', 'README.md', and 'remix.config.json'. The main workspace shows two tabs: 'Compiled' (selected) and 'Home'. The 'Compiled' tab displays the Solidity code for 'TokenizedCrowdfunding.sol':

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

interface IRewardToken {
    function mint(address to, uint256 amount) external;
}

contract TokenizedCrowdfunding {
    struct Campaign {
        address payable creator;
        string title;
        uint256 goalWei;
        uint256 deadline;
        uint256 raisedWei;
        bool finalized;
        bool successful;
    }

    IRewardToken public immutable rewardToken;

    uint256 public immutable rewardRate;
}
```

The bottom right corner of the workspace has a 'Debug' button. At the bottom of the screen, there are several status bars: 'Screen Alert', 'Initialize as git repo', 'Did you know?', 'RemixAI Copilot (disabled)', and a message about recording and replaying transactions.

Project structure in Remix IDE.

The application consists of two smart contracts: an ERC-20 reward token and a crowdfunding contract.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
5 import "@openzeppelin/contracts/access/Ownable.sol";
6
7
8 contract RewardToken is ERC20, Ownable {
9     constructor(string memory name_, string memory symbol_) payable infinite gas 876000 gas
10    ERC20(name_, symbol_)
11    Ownable(msg.sender)
12 }
13
14 function mint(address to, uint256 amount) external onlyOwner {
15     _mint(to, amount);
16 }
17 }
```

ERC-20 reward token contract.

Token minting is restricted to the contract owner using the onlyOwner modifier.

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.20;
3
4 interface IRewardToken {
5     function mint(address to, uint256 amount) external; payable - gas
6 }
7
8
9 contract TokenizedCrowdfunding {
10     struct Campaign {
11         address payable creator;
12         string title;
13         uint256 goalWei;
14         uint256 deadline;
15         uint256 raisedWei;
16         bool finalized;
17         bool successful;
18     }
19
20     IRewardToken public immutable rewardToken;
21
22
23     uint256 public immutable rewardRate;
24
25     Campaign[] private campaigns;
26 }
```

```

27
28
29
30     mapping(uint256 => mapping(address => uint256)) public contributions;
31
32     event CampaignCreated(uint256 indexed campaignId, address indexed creator, string title, uint256 goalWei, uint256 deadline);
33     event Contributed(uint256 indexed campaignId, address indexed contributor, uint256 amountWei, uint256 mintedTokenAmount);
34     event Finalized(uint256 indexed campaignId, bool successful, uint256 raisedWei);
35     event Refunded(uint256 indexed campaignId, address indexed contributor, uint256 amountWei);
36
37     constructor(address rewardTokenAddress, uint256 rewardRateTokensPerEth) payable infinite gas 1515600 gas
38     require(rewardTokenAddress != address(0), "token addr = 0");
39     require(rewardRateTokensPerEth > 0, "rate = 0");
40     rewardToken = IRewardToken(rewardTokenAddress);
41     rewardRate = rewardRateTokensPerEth;
42
43     function createCampaign(string calldata title, uint256 goalWei, uint256 durationSeconds) external returns (uint256 campaignId) {
44         require(title.length > 0, "empty title");
45         require(goalWei > 0, "goal = 0");
46         require(durationSeconds > 60, "min 60s");
47
48         uint256 deadline = block.timestamp + durationSeconds;
49
50     }
```

The image displays three vertically stacked screenshots of the REMIX IDE interface, version 1.5.1, showing the `TokenizedCrowdfunding.sol` smart contract. The IDE has a dark-themed interface with various toolbars and status indicators.

**Top Screenshot:**

```
53     campaigns.push(Campaign({
54         creator: payable(msg.sender),
55         title: title,
56         goalWei: goalWei,
57         deadline: deadline,
58         raisedWei: 0,
59         finalized: false,
60         successful: false
61     }));
62
63     campaignId = campaigns.length - 1;
64     emit CampaignCreated(campaignId, msg.sender, title, goalWei, deadline);
65 }
66
67 function getCampignCount() external view returns (uint256) { 2461 gas
68     return campaigns.length;
69 }
70
71 function getCampign(uint256 campaignId) external view returns (
72     address creator,
73     string memory title,
74     uint256 goalWei,
75     uint256 deadline,
76     uint256 raisedWei,
77     bool finalized,
78     bool successful
79 ) {
80     require(campaignId < campaigns.length, "bad id");
81     Campaign storage c = campaigns[campaignId];
82     return (c.creator, c.title, c.goalWei, c.deadline, c.raisedWei, c.finalized, c.successful);
83 }
84
85 function contribute(uint256 campaignId) external payable { infinite gas
86     require(campaignId < campaigns.length, "bad id");
87     Campaign storage c = campaigns[campaignId];
88
89     require(block.timestamp < c.deadline, "ended");
90     require(!c.finalized, "finalized");
91     require(msg.value > 0, "0 value");
92
93     c.raisedWei += msg.value;
94     contributions[campaignId][msg.sender] += msg.value;
95
96     uint256 minted = msg.value * rewardRate;
97     rewardToken.mint(msg.sender, minted);
98
99     emit Contributed(campaignId, msg.sender, msg.value, minted);
100 }
101
102 function finalize(uint256 campaignId) external { infinite gas
103     require(campaignId < campaigns.length, "bad id");
104     Campaign storage c = campaigns[campaignId];
105
106     require(!c.finalized, "already finalized");
107     require(block.timestamp >= c.deadline || c.raisedWei >= c.goalWei, "too early");
108
109     c.finalized = true;
110
111     if (c.raisedWei == c.goalWei) {
112         c.successful = true;
113         (bool ok,) = c.creator.call{value: c.raisedWei}("");
114         require(ok, "transfer failed");
115     } else {
116         c.successful = false;
117     }
118
119     emit Finalized(campaignId, c.successful, c.raisedWei);
120 }
121
122 function claimRefund(uint256 campaignId) external { infinite gas
123     require(campaignId < campaigns.length, "bad id");
124     Campaign storage c = campaigns[campaignId];
125
126     require(!c.finalized, "not finalized");
127     require(!c.successful, "successful => no refunds");
128
129     uint256 amount = contributions[campaignId][msg.sender];
130     require(amount > 0, "nothing to refund");
131 }
```

**Middle Screenshot:**

```
79     )
80     require(campaignId < campaigns.length, "bad id");
81     Campaign storage c = campaigns[campaignId];
82     return (c.creator, c.title, c.goalWei, c.deadline, c.raisedWei, c.finalized, c.successful);
83 }
84
85 function contribute(uint256 campaignId) external payable { infinite gas
86     require(campaignId < campaigns.length, "bad id");
87     Campaign storage c = campaigns[campaignId];
88
89     require(block.timestamp < c.deadline, "ended");
90     require(!c.finalized, "finalized");
91     require(msg.value > 0, "0 value");
92
93     c.raisedWei += msg.value;
94     contributions[campaignId][msg.sender] += msg.value;
95
96     uint256 minted = msg.value * rewardRate;
97     rewardToken.mint(msg.sender, minted);
98
99     emit Contributed(campaignId, msg.sender, msg.value, minted);
100 }
101
102 function finalize(uint256 campaignId) external { infinite gas
103     require(campaignId < campaigns.length, "bad id");
104     Campaign storage c = campaigns[campaignId];
105
106     require(!c.finalized, "already finalized");
107     require(block.timestamp >= c.deadline || c.raisedWei >= c.goalWei, "too early");
108
109     c.finalized = true;
110
111     if (c.raisedWei == c.goalWei) {
112         c.successful = true;
113         (bool ok,) = c.creator.call{value: c.raisedWei}("");
114         require(ok, "transfer failed");
115     } else {
116         c.successful = false;
117     }
118
119     emit Finalized(campaignId, c.successful, c.raisedWei);
120 }
121
122 function claimRefund(uint256 campaignId) external { infinite gas
123     require(campaignId < campaigns.length, "bad id");
124     Campaign storage c = campaigns[campaignId];
125
126     require(!c.finalized, "not finalized");
127     require(!c.successful, "successful => no refunds");
128
129     uint256 amount = contributions[campaignId][msg.sender];
130     require(amount > 0, "nothing to refund");
131 }
```

**Bottom Screenshot:**

```
105     require(!c.finalized, "already finalized");
106     require(block.timestamp >= c.deadline || c.raisedWei >= c.goalWei, "too early");
107
108     c.finalized = true;
109
110     if (c.raisedWei == c.goalWei) {
111         c.successful = true;
112         (bool ok,) = c.creator.call{value: c.raisedWei}("");
113         require(ok, "transfer failed");
114     } else {
115         c.successful = false;
116     }
117
118     emit Finalized(campaignId, c.successful, c.raisedWei);
119 }
120
121 function claimRefund(uint256 campaignId) external { infinite gas
122     require(campaignId < campaigns.length, "bad id");
123     Campaign storage c = campaigns[campaignId];
124
125     require(!c.finalized, "not finalized");
126     require(!c.successful, "successful => no refunds");
127
128     uint256 amount = contributions[campaignId][msg.sender];
129     require(amount > 0, "nothing to refund");
130 }
```

The screenshot shows the REMIX IDE interface with the following details:

- Toolbar:** Includes icons for Home, RewardToken.sol, TokenizedCrowdfunding.sol, Compile, Run, Deploy, and Meta.
- Header:** Shows "REMX 1.5.1", "Blank - 1", "Login with GitHub", "Theme", and a gear icon.
- Code Area:** Displays the Solidity code for the `TokenizedCrowdfunding` contract. The code handles contributions, reward token minting, campaign finalization, and refund requests. It uses external functions to interact with a `RewardToken` contract and manages a `Campaign storage` array.
- Bottom Bar:** Features "Explain contract", "Listen on all transactions" (with a count of 0), a search bar, and a "Filter with transaction hash or address" input field.

```
function claimRefund(uint256 campaignId) external { infinite gas
    require(campaignId < campaigns.length, "bad id");
    Campaign storage c = campaigns[campaignId];

    require(c.finalized, "not finalized");
    require(!c.successful, "successful => no refunds");

    uint256 amount = contributions[campaignId][msg.sender];
    require(amount > 0, "nothing to refund");

    contributions[campaignId][msg.sender] = 0;

    (bool ok,) = payable(msg.sender).call{value: amount}("");
    require(ok, "refund failed");

    emit Refunded(campaignId, msg.sender, amount);
}
```

Crowdfunding smart contract implementation.

The contract manages campaigns, accepts ETH contributions, mints reward tokens, finalizes campaigns, and handles refund

**DEPLOY & RUN TRANSACTIONS**

ENVIRONMENT 

Injected Provider - MetaMask 

Sepolia (11155111) network

ACCOUNT +  

0x81a...a8934 (0.02483344232€) 

+ Create Smart Account

GAS LIMIT

Estimated Gas

Custom **3000000**

VALUE

**0** **Wei**

CONTRACT

IRewardToken - contracts/TokenizedC

evm version: osaka

Verify Contract on Explorers

DEPLOY 

 Calldata  Parameters **transact**



## DEPLOY & RUN TRANSACTIONS

REWARDTOKEN AT 0X9E2...0C2

Balance: 0 ETH

approve address spender, uint256 value

mint address to, uint256 amount

renounceOwnership

transfer address to, uint256 value

transferFrom address from, address to, uint256 value

## TRANSFER OWNERSHIP

newOwner: 0xc92eEb53Dd4e4eA11286cD0cf6f...

Calldata Parameters transact

allowance address owner, address spender

balanceOf address account

decimals

name

owner

This screenshot shows a user interface for interacting with a smart contract named 'REWARDTOKEN' at address 0X9E2...0C2. The interface is divided into sections for deploying and running transactions.

**Deploy & Run Transactions:** This section shows the current balance as 0 ETH. It lists several functions: approve, mint, renounceOwnership, transfer, and transferFrom. Each function has its name in an orange button and its parameters in a grey box with a dropdown arrow.

**Transfer Ownership:** This section shows the new owner's address as 0xc92eEb53Dd4e4eA11286cD0cf6f... It includes tabs for Calldata, Parameters, and transact, with transact currently selected. Below these are five blue buttons for the functions allowance, balanceOf, decimals, name, and owner.

## Transfer Ownership

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0x9e212...0C2f6

### Транзакция

Одноразовый код 28

Сумма -0 SepoliaETH

Лимит Газа (Единицы) 29399

Использовано Газа (Единицы) 29051

Базовая комиссия (Гвей) 1.006376129

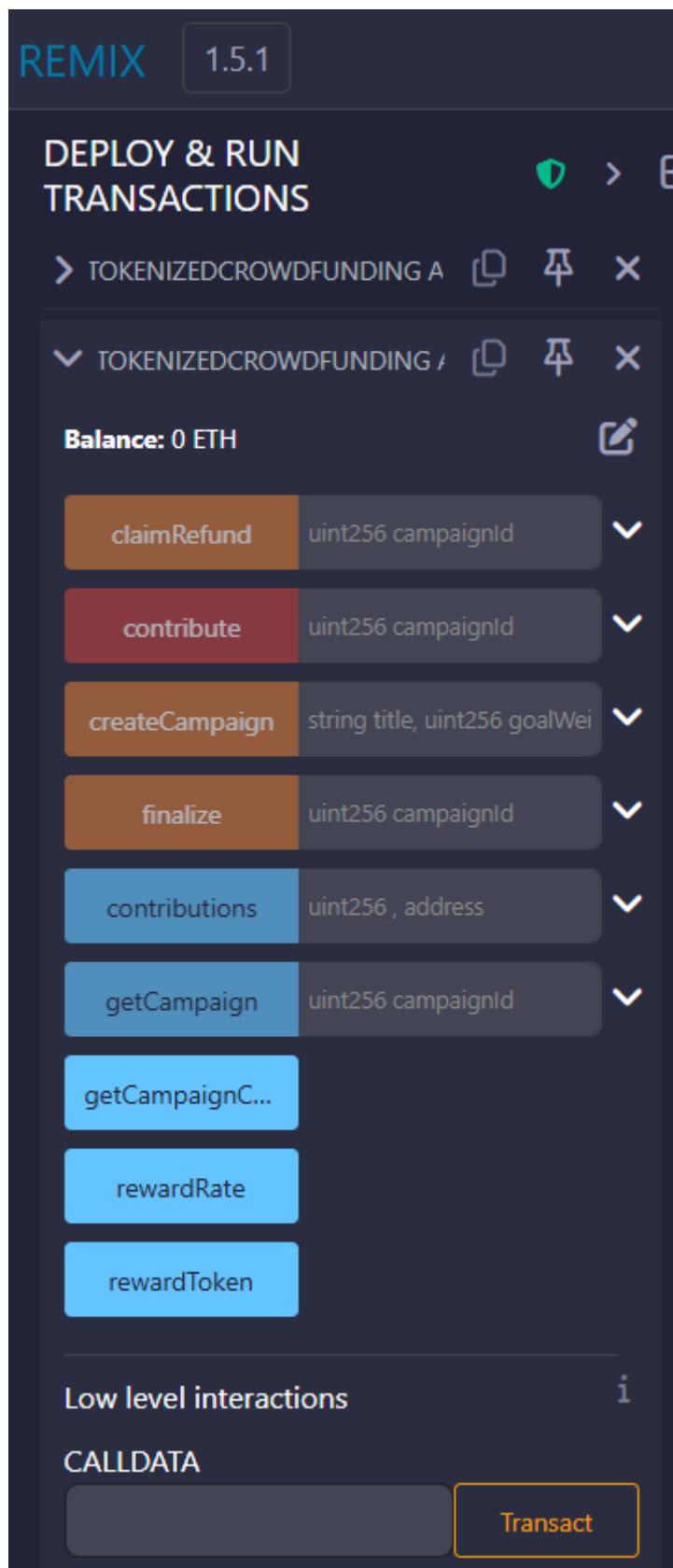
Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.000073 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого 0.00007281 SepoliaETH

+ Журнал активности



Smart contracts deployed on the Sepolia test network using MetaMask as an injected provider. Ownership of the reward token transferred to the crowdfunding contract, allowing it to mint tokens automatically during contributions.

[block:10217107 txIndex:11] from: 0x81A...a8934 to: RewardToken.(constructor) value: 0 wei data: 0x608...00000 logs: 1 hash: 0xbbe...8f8c9	
status	1 Transaction mined and execution succeed
transaction hash	0x481f24d44d19949e4a901c33c3fa8d085cdf65ebac781b76debf8f3bbbf2a563 ⓘ
block hash	0xbbea4085130a603d2bd32bac404393e87c4915cafca9ae10a3119bf1e9a8f8c9 ⓘ
block number	10217107 ⓘ
contract address	0x9e2124F845233F6a965d6BF75e9Be57D71c0C2f6 ⓘ
from	0x81Ad508F00aF99719B9A88116625C809CCa8934 ⓘ
to	RewardToken.(constructor) ⓘ
transaction cost	1094448 gas ⓘ
decoded input	{ "string name_": "CrowdReward", "string symbol_": "CRWD" } ⓘ
decoded output	- ⓘ
0 ⓘ Listen on all transactions ⓘ Filter with transaction hash or ad... ⓘ Debug ⓘ	
[block:10217120 txIndex:39] from: 0x81A...a8934 to: TokenizedCrowdfunding.(constructor) value: 0 wei data: 0x60c...003e8 logs: 0 hash: 0x808...2dcc6	
status	1 Transaction mined and execution succeed
transaction hash	0xc958ed7f3e90e45255233a0df3f0e413a5decdb1c87383417e39149598cf7f3 ⓘ
block hash	0x00837b2757fa58bf8b8a03169d006d016f34f4d4224675993d90850a492dcc6 ⓘ
block number	10217120 ⓘ
contract address	0xc92eEb530d4e4eA11286c00cf6fe313e5076f2e6 ⓘ
from	0x81Ad508F00aF99719B9A88116625C809CCa8934 ⓘ
to	TokenizedCrowdfunding.(constructor) ⓘ
transaction cost	1691772 gas ⓘ
decoded input	{ "address rewardTokenAddress": "0x9e2124F845233F6a965d6BF75e9Be57D71c0C2f6", "uint256 rewardRateTokensPerEth": "1000" } ⓘ
decoded output	- ⓘ
> 0 ⓘ Listen on all transactions ⓘ Filter with transaction hash or ad... ⓘ Debug ⓘ	
[block:10217126 txIndex:16] from: 0x81A...a8934 to: RewardToken.transferOwnership(address) 0x9e2...0C2f6 value: 0 wei data: 0xf2f...6f2e6 logs: 1	
hash	0xd93...25b6a
status	1 Transaction mined and execution succeed
transaction hash	0xa9302d79c18614133a4628b3a174a7a71f1dfa50cecd77b49dd2faa12d010c66 ⓘ
block hash	0xd931e267504e63f76b0ac4353c52c2ee34615780b8c51282a00e0520db525b6a ⓘ
block number	10217126 ⓘ
from	0x81Ad508F00aF99719B9A88116625C809CCa8934 ⓘ
to	RewardToken.transferOwnership(address) 0x9e2124F845233F6a965d6BF75e9Be57D71c0C2f6 ⓘ
transaction cost	29051 gas ⓘ
decoded input	{ "address newOwner": "0xc92eEb530d4e4eA11286c00cf6fe313e5076f2e6" } ⓘ
decoded output	- ⓘ
logs	[ ]
> [block:10217126 txIndex:16] from: 0x81A...a8934 to: RewardToken.transferOwnership(address) 0x9e2...0C2f6 value: 0 wei data: 0xf2f...6f2e6 logs: 1	
hash	0xd93...25b6a
status	1 Transaction mined and execution succeed
transaction hash	0xa9302d79c18614133a4628b3a174a7a71f1dfa50cecd77b49dd2faa12d010c66 ⓘ
block hash	0xd931e267504e63f76b0ac4353c52c2ee34615780b8c51282a00e0520db525b6a ⓘ
block number	10217126 ⓘ
from	0x81Ad508F00aF99719B9A88116625C809CCa8934 ⓘ
to	RewardToken.transferOwnership(address) 0x9e2124F845233F6a965d6BF75e9Be57D71c0C2f6 ⓘ
transaction cost	29051 gas ⓘ
decoded input	{ "address newOwner": "0xc92eEb530d4e4eA11286c00cf6fe313e5076f2e6" } ⓘ
decoded output	- ⓘ

## Tokenized Crowdfunding (Sepolia)

[Connect MetaMask](#)

Connected to Sepolia

Wallet: 0x81Ad588F00aF99719B9A8B116625C809CCa8934

Network: sepolia (11155111)

ETH Balance: 0.024833442326592714 ETH

Token Balance: 11.0 CRWD

Frontend application connected to MetaMask wallet.

User address, network, ETH balance, and token balance are displayed.

### 1) Set contract addresses

Crowdfunding contract address

0xc92eEb53Dd4e4eA11286cD0cf6fe313e5D76f2e6

Reward token address

0x9e2124F845233F6a965d6BF75e9Be57D71c0C2f6

[Load campaigns](#)

Configuration of deployed smart contract addresses in the frontend application.

These addresses are obtained after deployment in Remix.

### 2) Create campaign

Title

REFUND

Goal (ETH)

0.02

Duration (minutes)

2

[Create](#)

Campaign created!

Creating a new crowdfunding campaign by specifying the title, funding goal, and

campaign duration.

The screenshot shows a dark-themed user interface for managing campaigns. At the top, it says "3) Campaigns" and "Found 2 campaigns".

**Campaign #0 — Arlan\_777:**  
Creator: 0x81Ad508F00aF99719B9A8B116625C809CCa8934  
Goal: 0.01 ETH  
Raised: 0.01 ETH  
Deadline: 08.02.2026, 16:33:48  
Time left: 0 min  
Finalized: true, Successful: true  
Your contribution: 0.01 ETH

Contribute (ETH): 0.01  Actions

**Campaign #1 — REFUND:**  
Creator: 0x81Ad508F00aF99719B9A8B116625C809CCa8934  
Goal: 0.02 ETH  
Raised: 0.001 ETH  
Deadline: 08.02.2026, 16:32:00  
Time left: 0 min  
Finalized: true, Successful: false  
Your contribution: 0.0 ETH

Contribute (ETH): Actions

## Test

Newly created campaign displayed in the campaign list with initial funding status. User contributing test ETH to the campaign.

The contribution is sent directly to the smart contract. Reward tokens automatically minted and credited to the contributor after a successful ETH contribution. Campaign finalized successfully after reaching the funding goal.

Collected ETH is transferred to the campaign creator.

## Refund

Unsuccessful campaign finalized.

Contributors are allowed to claim refunds if the funding goal is not reached. Refund successfully claimed.

Test ETH is returned to the contributor, demonstrating user fund protection.

## Create Campaign

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

### Транзакция

Одноразовый код 29

Сумма -0 SepoliaETH

Лимит Газа (Единицы) 144594

Использовано Газа (Единицы) 143442

Базовая комиссия (Гвей) 1.041828806

Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.000365 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого 0.0003646 SepoliaETH

+ Журнал активности

## Contribute

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

### Транзакция

Одноразовый код 30

Сумма **-0.002 SepoliaETH**

Лимит Газа (Единицы) 129513

Использовано Газа (Единицы) 128421

Базовая комиссия (Гвей) 0.945867663

Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.000314 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого **0.0023141 SepoliaETH**

+ **Журнал активности**

## Contribute

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

### Транзакция

Одноразовый код 31

Сумма **-0.008 SepoliaETH**

Лимит Газа (Единицы) 60842

Использовано Газа (Единицы) 60021

Базовая комиссия (Гвей) 0.975536213

Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.000149 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого **0.00814858 SepoliaETH**

+ Журнал активности

# Finalize

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

## Транзакция

Одноразовый код 32

Сумма -0 SepoliaETH

Лимит Газа (Единицы) 65618

Использовано Газа (Единицы) 64778

Базовая комиссия (Гвей) 0.96552283

Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.00016 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого 0.00015971 SepoliaETH

+ Журнал активности

## Create Campaign

X

### Статус

Просмотр в проводнике блоков

### Подтверждено

Скопировать ID транзакции

### Из



Account 11



0xc92eE...6f2e6

### Место назначения

### Транзакция

Одноразовый код 33

Сумма **-0 SepoliaETH**

Лимит Газа (Единицы) 127378

Использовано Газа (Единицы) 126294

Базовая комиссия (Гвей) **1.045741073**

Плата за приоритет (Гвей) 1.5

Итого платы за газ **0.000322 SepoliaETH**

Макс. комиссия на газ **0.000000003 SepoliaETH**

Итого **0.00032151 SepoliaETH**

+ Журнал активности

# Contribute

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

## Транзакция

Одноразовый код 34

Сумма **-0.001 SepoliaETH**

Лимит Газа (Единицы) 95190

Использовано Газа (Единицы) 94233

Базовая комиссия (Гвей) **0.940216536**

Плата за приоритет (Гвей) 1.5

Итого платы за газ **0.00023 SepoliaETH**

Макс. комиссия на газ **0.000000003 SepoliaETH**

Итого **0.00122995 SepoliaETH**

+ **Журнал активности**

## Finalize

X

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**



Account 11



0xc92eE...6f2e6

### Транзакция

Одноразовый код 35

Сумма -0 SepoliaETH

Лимит Газа (Единицы) 56005

Использовано Газа (Единицы) 55203

Базовая комиссия (Гвей) 0.955424918

Плата за приоритет (Гвей) 1.5

Итого платы за газ 0.000136 SepoliaETH

Макс. комиссия на газ 0.000000003 SepoliaETH

Итого 0.00013555 SepoliaETH

+ Журнал активности

**Meta**

## Claim Refund

**Статус** Просмотр в проводнике блоков

**Подтверждено** Скопировать ID транзакции

**Из** **Место назначения**

 Account 11 →  0xc92eE...6f2e6

### Транзакция

Одноразовый код	36
Сумма	<b>-0 SepoliaETH</b>
Лимит Газа (Единицы)	41280
Использовано Газа (Единицы)	35930
Базовая комиссия (Гвей)	<b>1.118149906</b>
Плата за приоритет (Гвей)	1.5
Итого платы за газ	<b>0.000094 SepoliaETH</b>
Макс. комиссия на газ	<b>0.000000003 SepoliaETH</b>
Итого	<b>0.00009407 SepoliaETH</b>

+ Журнал активности

## Conclusion

In this project, a tokenized crowdfunding platform was successfully designed and implemented using blockchain technology. The system demonstrates how decentralized applications can be used to transparently collect funds and automatically reward contributors with ERC-20 tokens.

The project includes two smart contracts deployed on the Ethereum Sepolia test network: a reward token contract and a crowdfunding contract. The crowdfunding

contract allows users to create campaigns, contribute test ETH, finalize campaigns based on predefined conditions, and claim refunds in case the funding goal is not reached. The reward token contract ensures that contributors receive tokens proportionally to their contributions.

The integration with MetaMask enables secure user authentication and transaction signing, while the frontend application provides a user-friendly interface for interacting with the smart contracts. All core crowdfunding scenarios were demonstrated, including successful campaign completion with token distribution and unsuccessful campaigns with refund functionality.

Overall, this project illustrates the practical application of smart contracts in decentralized finance (DeFi) and highlights the advantages of blockchain-based crowdfunding systems, such as transparency, automation, and user fund protection.<sup>1</sup>

---

<sup>1</sup> <https://youtu.be/9o7yCS3V6V0>