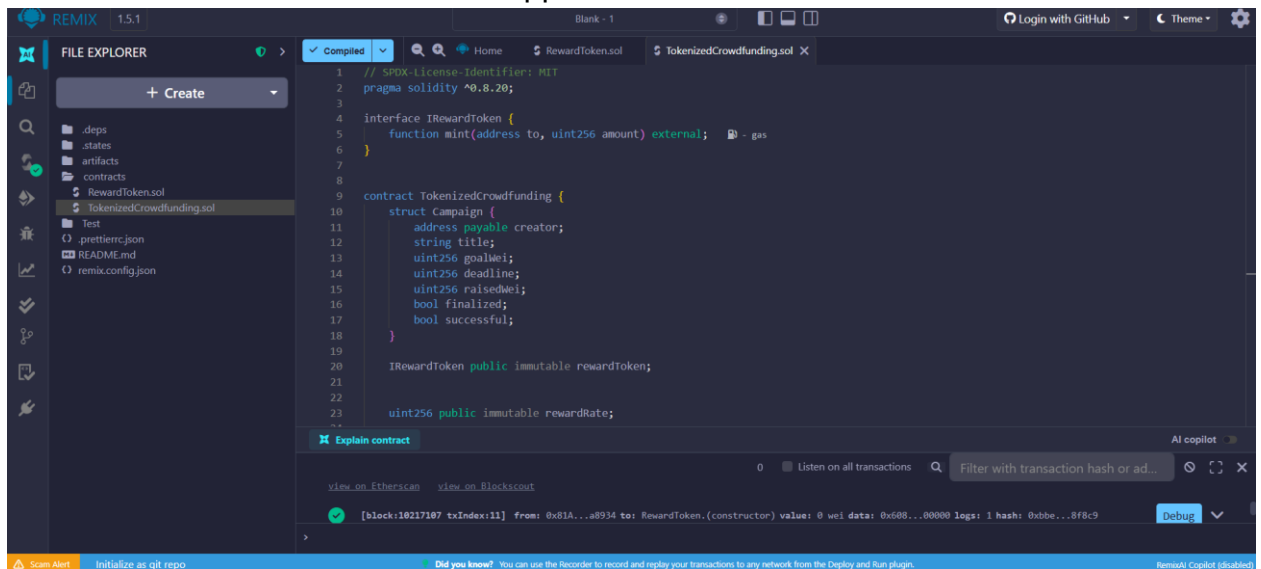
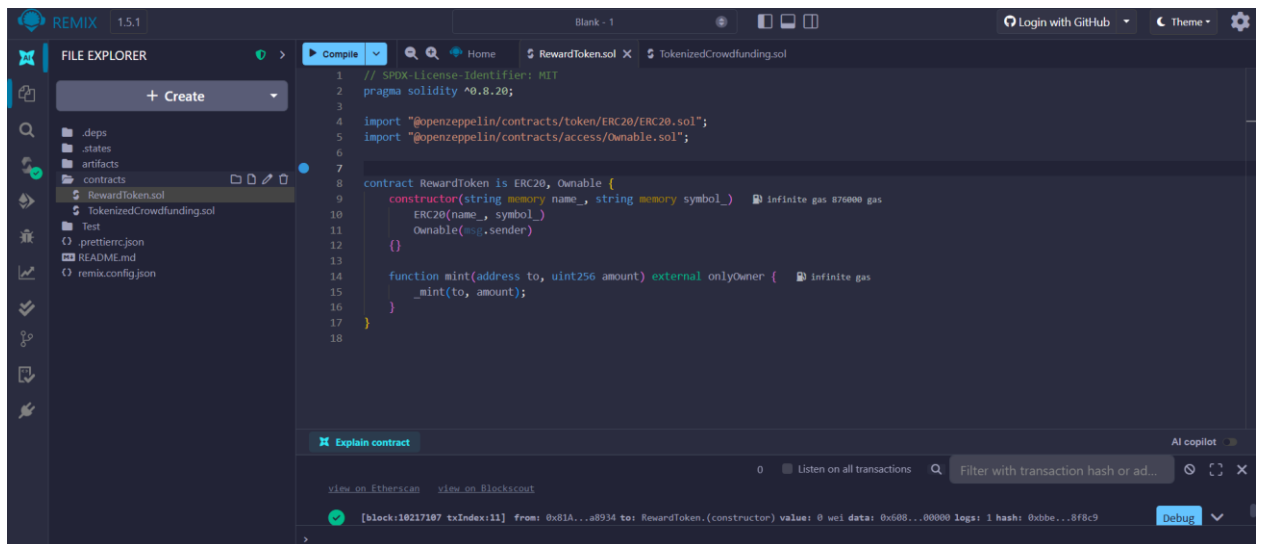


MetaMask wallet connected to the Sepolia test network with test ETH balance. Testnet is used to demonstrate the application without real funds.



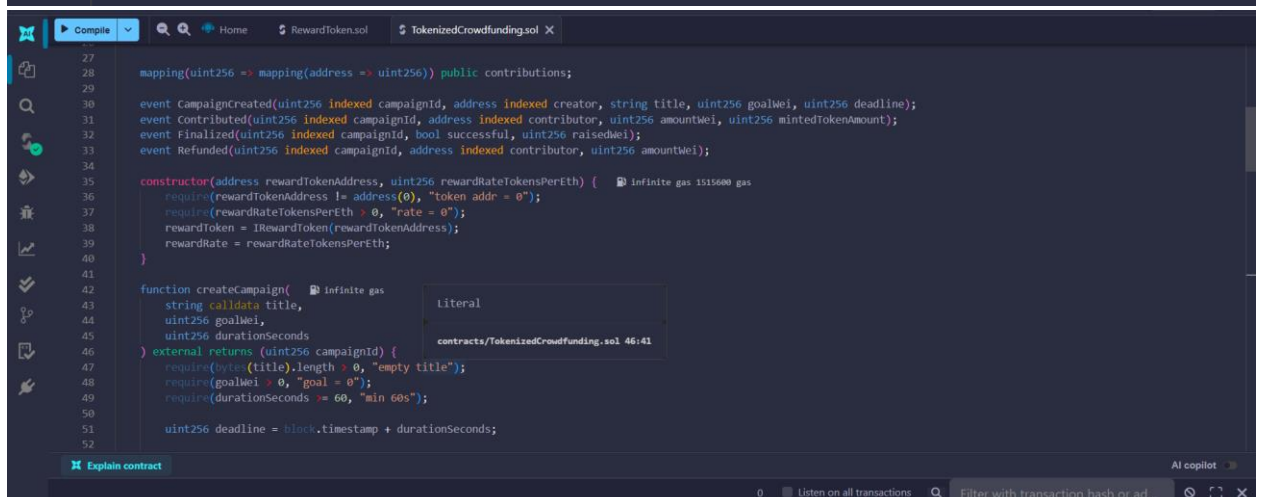
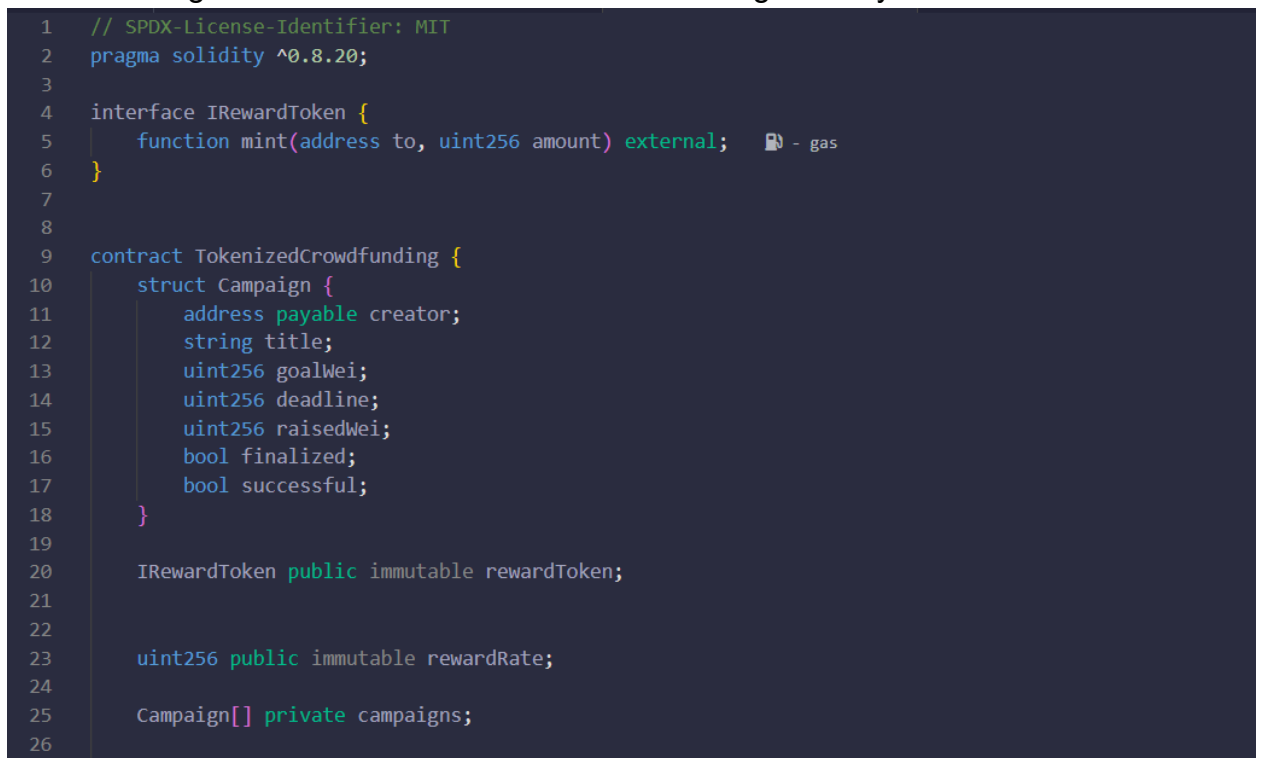
Project structure in Remix IDE.

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



ERC-20 reward token contract.

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



The screenshot shows the Remix IDE interface. At the top, there's a header bar with the 'REMX' logo, version '1.5.1', and buttons for 'Login with GitHub' and 'Theme'. Below the header, a toolbar contains icons for Explorer, Search, and Run and Debug. The main editor area displays the 'TokenizedCrowdfunding.sol' file. The code defines a 'Campaign' struct with fields: creator, payable, title, goalWei, deadline, raisedWei, finalized, and successful. It also defines a 'campaigns' array and functions for creating campaigns and getting campaign counts. The IDE includes a sidebar with icons for Explorer, Search, and Run and Debug, and a bottom bar with 'Explain contract' and 'AI copilot' buttons.

```

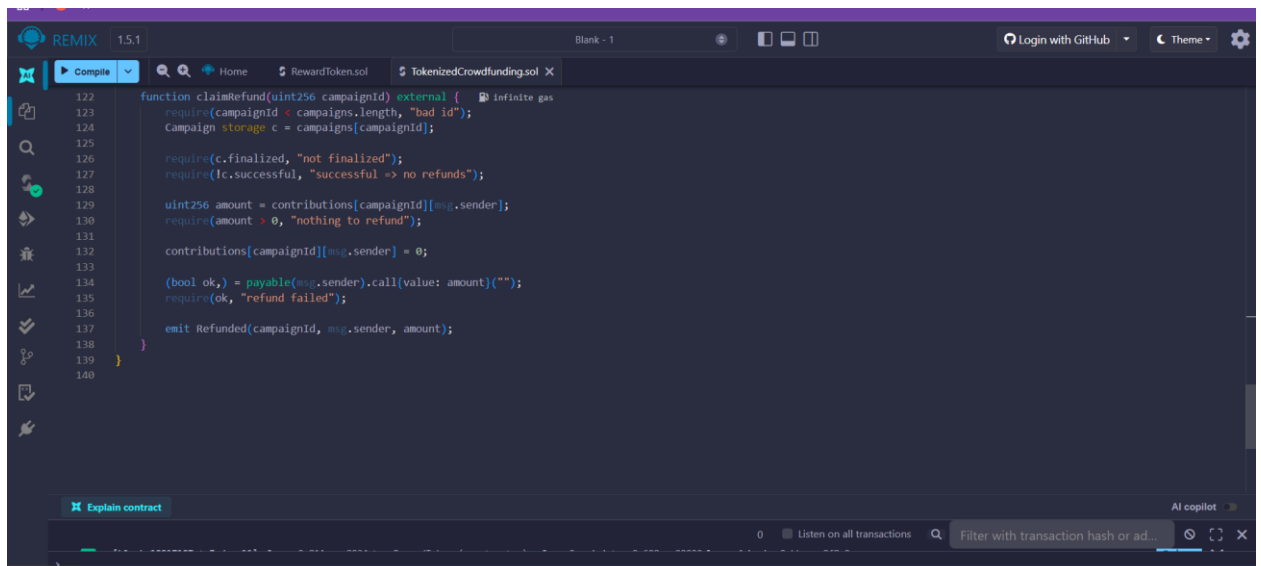
52 campaigns.push(Campaign({
53   creator: payable(msg.sender),
54   title: title,
55   goalWei: goalWei,
56   deadline: deadline,
57   raisedWei: 0,
58   finalized: false,
59   successful: false
60 }));
61
62 campaignId = campaigns.length - 1;
63 emit CampaignCreated(campaignId, msg.sender, title, goalWei, deadline);
64 }
65
66
67 function getCampaignCount() external view returns (uint256) {
68   return campaigns.length;
69 }
70
71 function getCampaignId(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

```

```
REMIX 1.5.1  
Blank - 1  
Login with GitHub Theme  
Compile  
RewardToken.sol TokenizedCrowdfunding.sol  
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];
```

```
REMIX 1.5.1
Blank - 1
Login with GitHub
Theme
File Explorer
Search
Home
RewardToken.sol
TokenizedCrowdfunding.sol X

105
106
107 require(!c.finalized, "already finalized");
108 require(block.timestamp >= c.deadline || c.raisedWei >= c.goalWei, "too early");
109
110 c.finalized = true;
111
112 if (c.raisedWei >= c.goalWei) {
113     c.successful = true;
114     (bool ok,) = c.creator.call([value: c.raisedWei])( "");
115     require(ok, "transfer failed");
116 } else {
117     c.successful = false;
118 }
119
120 emit Finalized(campaignId, c.successful, c.raisedWei);
121
122 function claimRefund(uint256 campaignId) external {
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 }
```



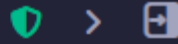
```
122 function claimRefund(uint256 campaignId) external {
123     require(campaignId < campaigns.length, "bad id");
124     Campaign storage c = campaigns[campaignId];
125
126     require(c.finalized, "not finalized");
127     require(!c.successful, "successful => no refunds");
128
129     uint256 amount = contributions[campaignId][msg.sender];
130     require(amount > 0, "nothing to refund");
131
132     contributions[campaignId][msg.sender] = 0;
133
134     (bool ok,) = payable(msg.sender).call{value: amount}("");
135     require(ok, "refund failed");
136
137     emit Refunded(campaignId, msg.sender, amount);
138 }
139
140
```


The screenshot shows the Remix IDE interface. The top bar includes the Remix logo, version 1.5.1, and buttons for 'Login with GitHub' and 'Theme'. The left sidebar contains icons for file explorer, search, and other IDE features. The main editor displays the Solidity code for the 'claimRefund' function. The function is marked as 'external' and 'infinite gas'. It includes several 'require' statements to validate the campaign ID, campaign state, and contribution amount. The function then updates the contribution to zero and sends the refund amount back to the sender via a call. An event 'Refunded' is emitted with the campaign ID, sender address, and amount. At the bottom, there is a bar with 'Explain contract' and 'AI copilot' buttons, and a transaction monitoring section with a 'Listen on all transactions' checkbox and a search filter.

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 (0.024833442326



+ Create Smart Account

GAS LIMIT

☒ Estimated Gas

☐ Custom

3000000

VALUE

0

Wei

CONTRACT


IRewardToken - contracts/TokenizedC


evm version: osaka

☐ Verify Contract on Explorers

DEPLOY

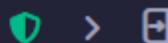





 Calldata

 Parameters

transact

DEPLOY & RUN TRANSACTIONS



✓ REWARDTOKEN AT 0X9E2...0C2   

Balance: 0 ETH



approve

address spender, uint256 va



mint

address to, uint256 amount



renounceOwne...

transfer

address to, uint256 value



transferFrom

address from, address to, ui



TRANSFER OWNERSHIP



newOwner: 0xc92eEb53Dd4e4eA11286cD0cf6f6



Calldata



Parameters

transact

allowance

address owner, address spe



balanceOf

address account



decimals

name

owner

Transfer Ownership



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

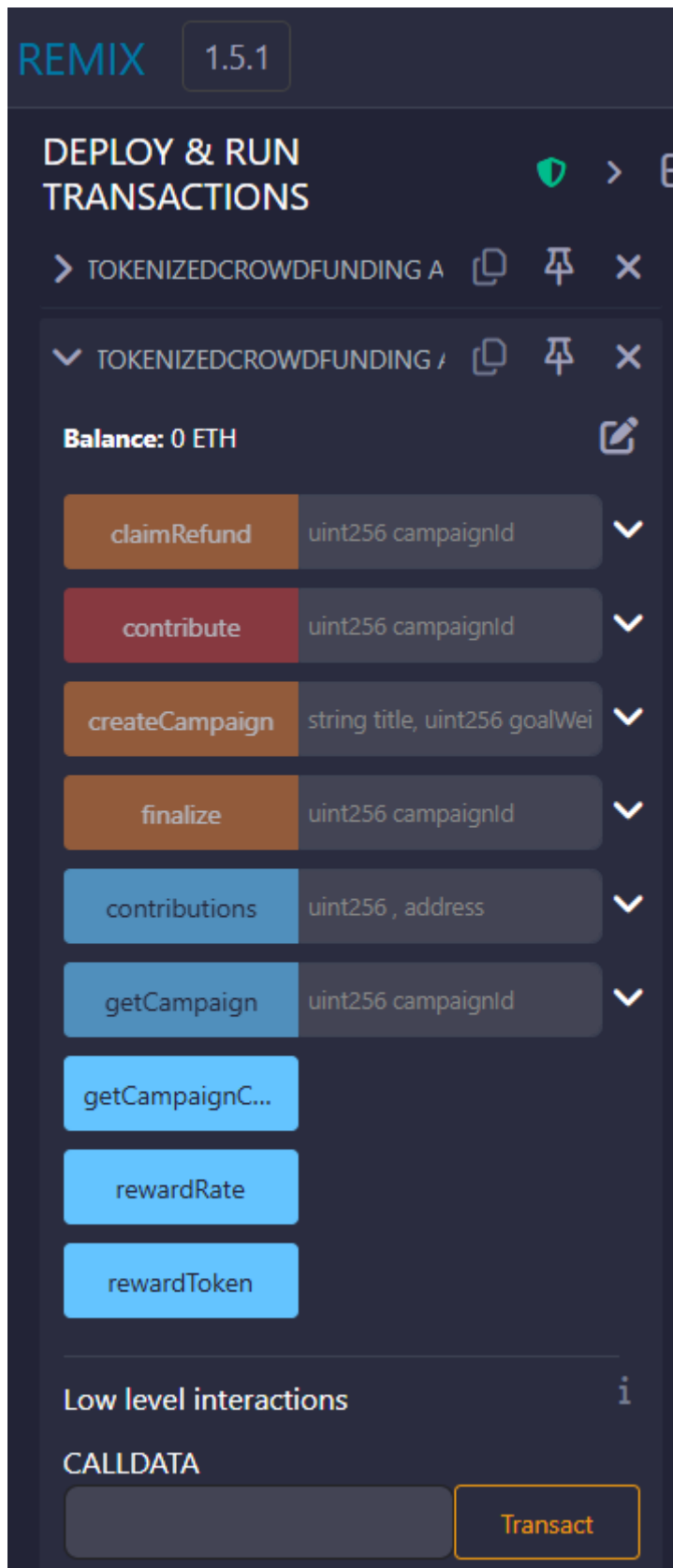
Плата за приоритет (Гвей) 1.5

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

Debug

⌵

status

1 Transaction mined and execution succeed

transaction hash

0x481f24d44d19949e4a901c3c3fa8d085cdf65ebac781b76deb8f3bbbf2a563

🔗

block hash

0xbbea4085130a603d2bd32bac404393e87c4915cafca9ae10a3119bf1e9a8f8c9

🔗

block number

10217107

🔗

contract address

0x9e2124f845233f6a965d68f75e98e57071c0c2f6

🔗

from

0x81Ad508F00aF9971989A88116625C809CCAa8934

🔗

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...

🔕

🔗

✕

✓

[block:10217120 txIndex:39] from: 0x81A...a8934 to: TokenizedCrowdfunding.(constructor) value: 0 wei data: 0x60c...003e8 logs: 0 hash: 0x808...2dcc6

Debug

⌵

status

1 Transaction mined and execution succeed

transaction hash

0xc958ed7f3e90e45255233ac0df3f0e413a5deecb1c87383417e39149598cf7f3

🔗

block hash

0x80837b2757fa58bf8b8a03169d0064d016f34f4d4224675993d90850a492dcc6

🔗

block number

10217120

🔗

contract address

0xc92eEb53Dd4e4eA11286cD0cf6fe313e5076f2e6

🔗

from

0x81Ad508F00aF9971989A88116625C809CCAa8934

🔗

to

TokenizedCrowdfunding.(constructor)

🔗

transaction cost

1691772 gas

🔗

decoded input

{
 "address rewardTokenAddress": "0x9e2124f845233f6a965d68f75e98e57071c0c2f6",
 "uint256 rewardRateTokensPerEth": "1000"
}

🔗

decoded output

-

🔗

>

0

☐ Listen on all transactions

🔍 Filter with transaction hash or ad...

🔕

🔗

✕

✓

[block:10217126 txIndex:16] from: 0x81A...a8934 to: RewardToken.transferOwnership(address) 0x9e2...0C2f6 value: 0 wei data: 0xf2f...6f2e6 logs: 1 hash: 0xd93...25b6a

Debug

⌵

status

1 Transaction mined and execution succeed

transaction hash

0xa9302d79c18614133a4628b3a174a7a71f1dfa50cecd77b49dd2faa12d010c66

🔗

block hash

0xd931e267504e63f76b0ac4353c52c2ee34615780b8c51282a00e0520db525b6a

🔗

block number

10217126

🔗

from

0x81Ad508F00aF9971989A88116625C809CCAa8934

🔗

to

RewardToken.transferOwnership(address) 0x9e2124f845233f6a965d68f75e98e57071c0c2f6

🔗

transaction cost

29051 gas

🔗

decoded input

{
 "address newOwner": "0xc92eEb53Dd4e4eA11286cD0cf6fe313e5076f2e6"
}

🔗

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

Debug

⌵

status

1 Transaction mined and execution succeed

transaction hash

0xa9302d79c18614133a4628b3a174a7a71f1dfa50cecd77b49dd2faa12d010c66

🔗

block hash

0xd931e267504e63f76b0ac4353c52c2ee34615780b8c51282a00e0520db525b6a

🔗

block number

10217126

🔗

from

0x81Ad508F00aF9971989A88116625C809CCAa8934

🔗

to

RewardToken.transferOwnership(address) 0x9e2124f845233f6a965d68f75e98e57071c0c2f6

🔗

transaction cost

29051 gas

🔗

decoded input

{
 "address newOwner": "0xc92eEb53Dd4e4eA11286cD0cf6fe313e5076f2e6"
}

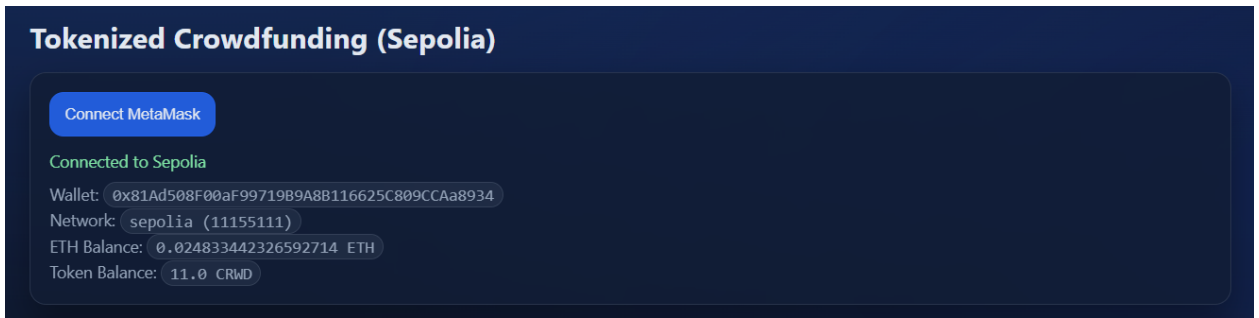
🔗

decoded output

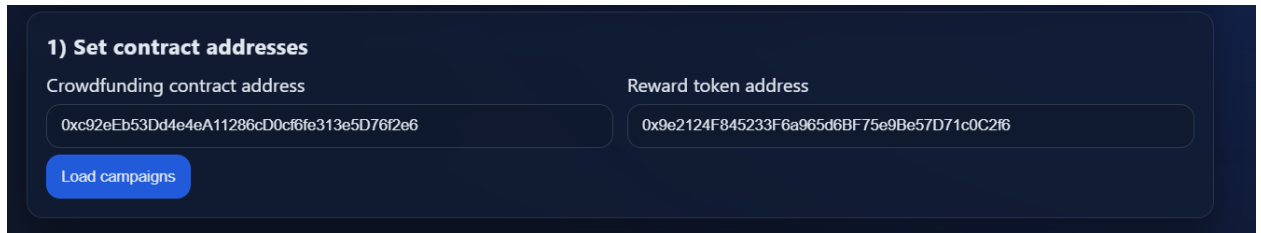
-

🔗

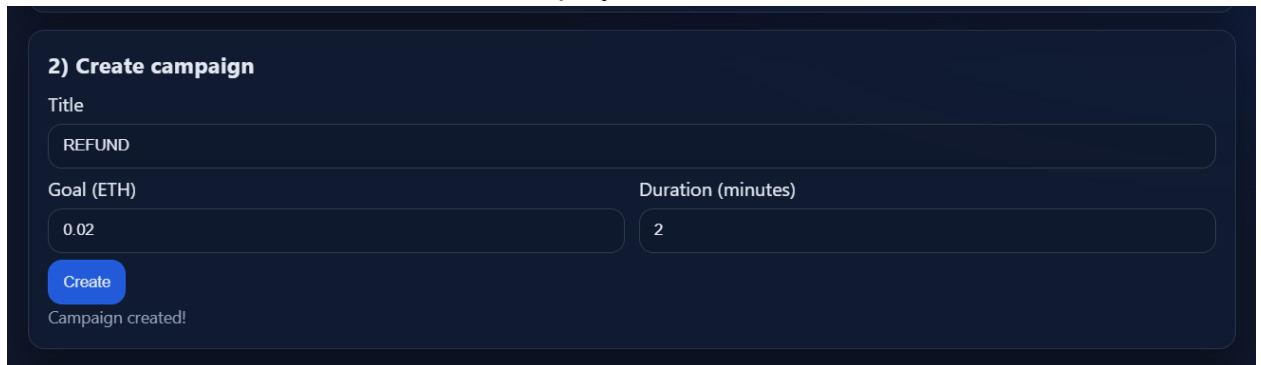
>



Frontend application connected to MetaMask wallet.
User address, network, ETH balance, and token balance are displayed.



Configuration of deployed smart contract addresses in the frontend application.
These addresses are obtained after deployment in Remix.



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

campaign duration.

3) Campaigns

Found 2 campaigns

#0 — Arlan_777

Creator: 0x81Ad508F00aF99719B9A8B116625C809CCAa8934

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

Contribute

Actions Finalize Claim Refund

#1 — REFUND

Creator: 0x81Ad508F00aF99719B9A8B116625C809CCAa8934

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 Finalize Claim Refund

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



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



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



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



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



Статус

Просмотр в проводнике блоков

Подтверждено

Скопировать ID транзакции

Из

Место назначения

 Account 11



 0xc92eE...6f2e6

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



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



Статус

[Просмотр в проводнике блоков](#)

Подтверждено

[Скопировать ID транзакции](#)

Из

Место назначения



Account 11

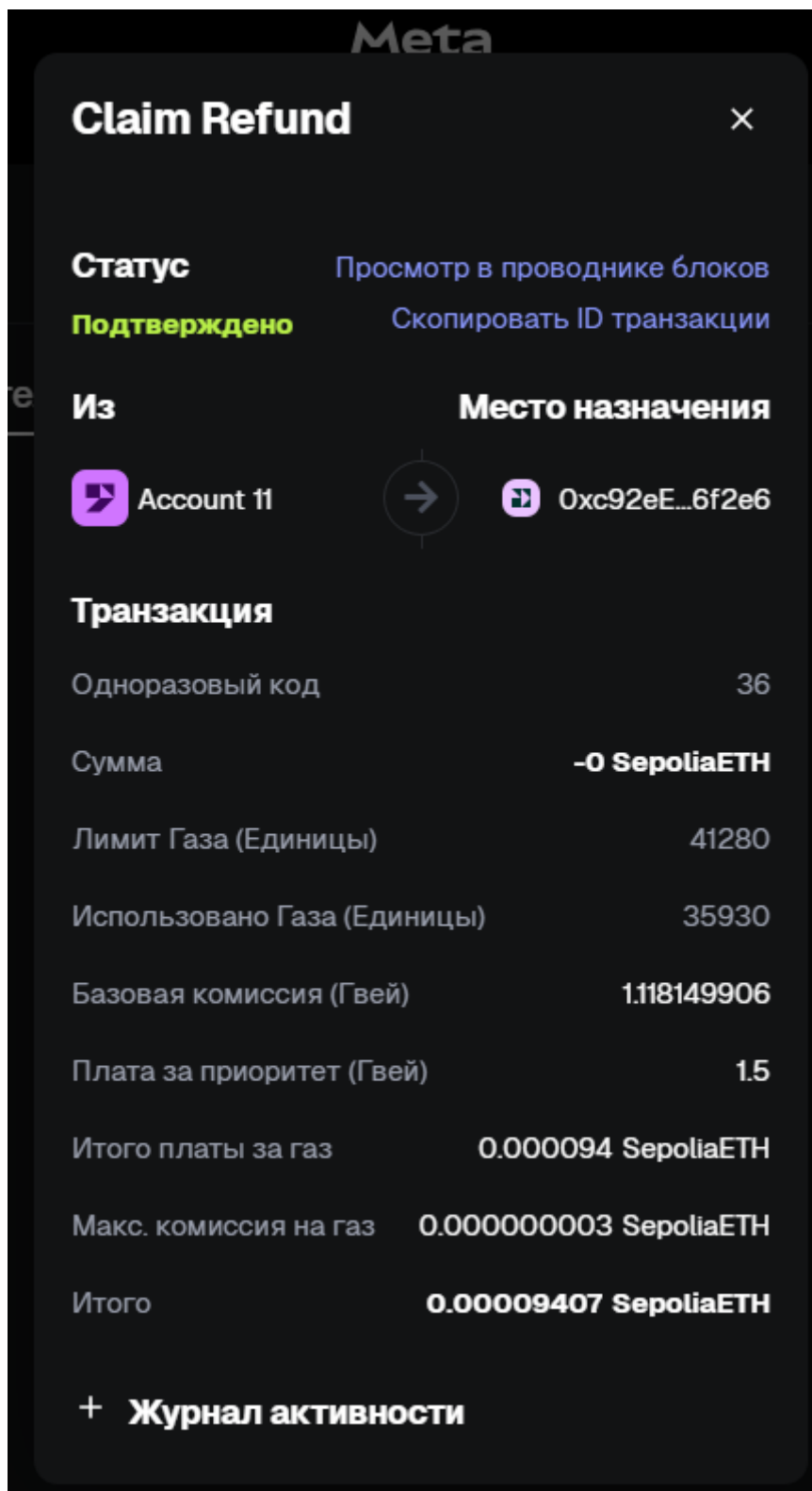


0xc92eE...6f2e6

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](#)



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.¹

¹ <https://youtu.be/9o7yCS3V6V0>