**周亚男 2020131062**

## 操作过程

<https://www.bilibili.com/video/BV1WN4111754/?spm_id_from=333.1007.0.0>

## 完善合约代码

003\_CrowdFund.sol

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.17;

interface IERC20 {

    function transfer(address, uint) external returns (bool);

    function transferFrom(address, address, uint) external returns (bool);

}

contract CrowdFund {

    struct Campaign {

        // 活动创建人

        address creator;

        // 目标筹集金额

        uint goal;

        // 已筹集资金

        uint pledged;

        // 开始时间

        uint32 startAt;

        // 结束时间

        uint32 endAt;

        // 是否已领取

        bool claimed;

    }

    IERC20 public immutable token;

    // 活动的id也是根据count来创建

    uint public count;

    // 查看第几个众筹活动

    mapping(uint => Campaign) public campaigns;

    // campaign id => pledger => amount pledged

    // 参与某个活动的人投的钱

    mapping(uint => mapping(address => uint)) public pledgedAmount;

    // 以下事件需要全部被用上！

    // 创建活动事件

    event Launch(

        uint id,

        address creator,

        uint goal,

        uint32 startAt,

        uint32 endAt

    );

    // 投资事件

    event Pledge(uint id, address caller, uint amount);

    // 撤资事件

    event Unpledge(uint id, address caller, uint amount);

    // 提取众筹

    event Claim(uint id);

    // 取回众筹股份

    event Refund(uint id, address caller, uint amount);

    constructor(address \_token) {

        token = IERC20(\_token);

        count = 0;

    }

    function launch(uint \_goal, uint32 \_startAt, uint32 \_endAt) external {

        require(\_startAt >= block.timestamp, "start at < now");

        require(\_endAt >= \_startAt, "end at < start at");

        require(\_endAt <= \_startAt + 20 minutes, "end at > max duration"); // 最长活动时间为20分钟

        // 补全

        Campaign memory camplaunch = Campaign(

            msg.sender,

            \_goal,

            0,

            \_startAt,

            \_endAt,

            false

        );

        count++;

        campaigns[count] = camplaunch;

        emit Launch(count, msg.sender, \_goal, \_startAt, \_endAt);

    }

    function pledge(uint \_id, uint \_amount) external {

        require(\_id <= count, "no this activity");

        Campaign storage campaign = campaigns[\_id];

        require(!campaign.claimed, "claimed");

        // require(campaign.pledged + \_amount <= campaign.goal, "over");

        require(block.timestamp >= campaign.startAt, "not started");

        require(block.timestamp <= campaign.endAt, "ended");

        // 补全

        token.transferFrom(msg.sender, address(this), \_amount);

        campaign.pledged += \_amount;

        campaigns[\_id] = campaign;

        pledgedAmount[\_id][msg.sender] += \_amount;

        emit Pledge(\_id, msg.sender, \_amount);

    }

    function unpledge(uint \_id, uint \_amount) external {

        // 补全

        require(\_id <= count, "no this activity");

        Campaign storage campaign = campaigns[\_id];

        require(!campaign.claimed, "claimed");

        require(\_amount <= pledgedAmount[\_id][msg.sender], "over");

        require(\_amount <= campaign.pledged, "over");

        require(block.timestamp >= campaign.startAt, "not started");

        require(block.timestamp <= campaign.endAt, "ended");

        // 补全

        token.transfer(msg.sender, \_amount);

        campaign.pledged -= \_amount;

        campaigns[\_id] = campaign;

        pledgedAmount[\_id][msg.sender] -= \_amount;

        emit Unpledge(\_id, msg.sender, \_amount);

    }

    function claim(uint \_id) external {

        // 补全

        require(\_id <= count, "no this activity");

        Campaign storage campaign = campaigns[\_id];

        require(!campaign.claimed, "claimed");

        require(campaign.creator == msg.sender, "not creator");

        require(block.timestamp > campaign.endAt, "not ended");

        require(campaign.pledged >= campaign.goal, "pledged < goal");

        // 补全

        token.transfer(msg.sender, campaign.pledged);

        campaign.claimed = true;

        campaigns[\_id] = campaign;

        emit Claim(\_id);

    }

    function refund(uint \_id) external {

        // 补全

        require(\_id <= count, "no this activity");

        Campaign storage campaign = campaigns[\_id];

        require(!campaign.claimed, "claimed");

        require(pledgedAmount[\_id][msg.sender] > 0);

        require(block.timestamp > campaign.endAt, "not ended");

        require(campaign.pledged < campaign.goal, "pledged >= goal");

        // 补全

        token.transfer(msg.sender, pledgedAmount[\_id][msg.sender]);

        emit Refund(\_id, msg.sender, pledgedAmount[\_id][msg.sender]);

    }

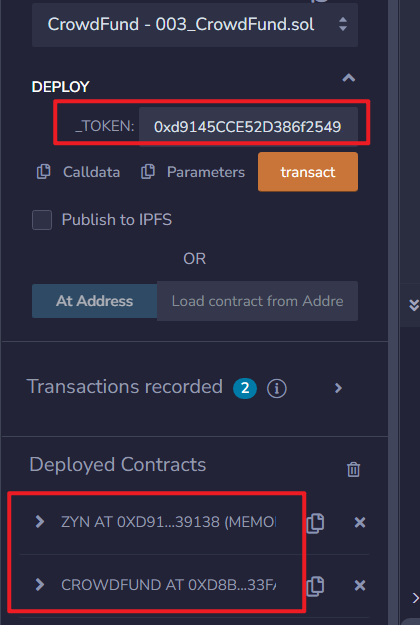
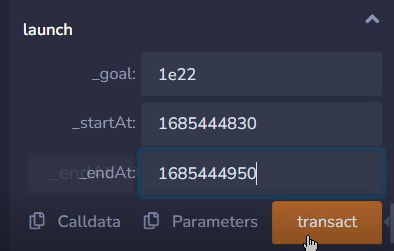
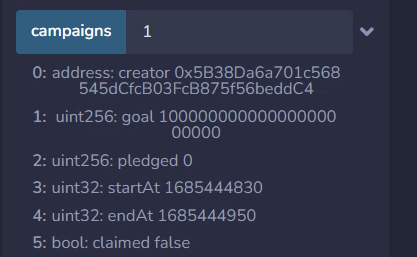
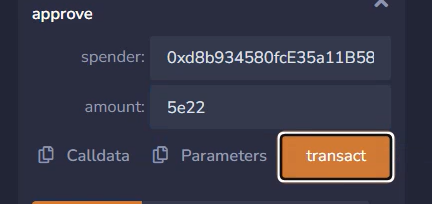
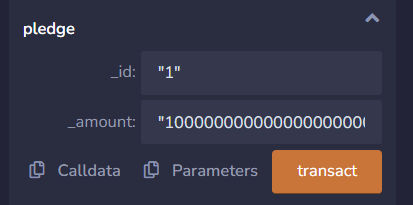
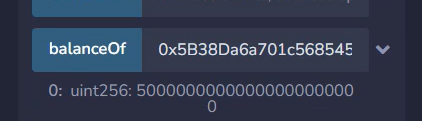
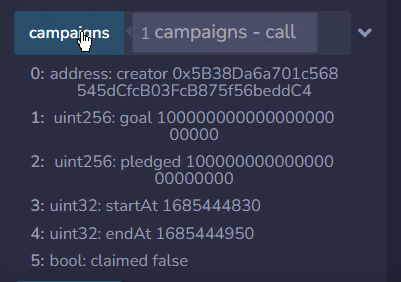
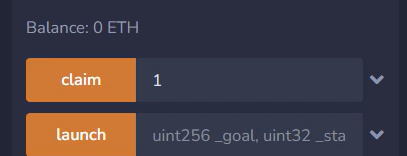
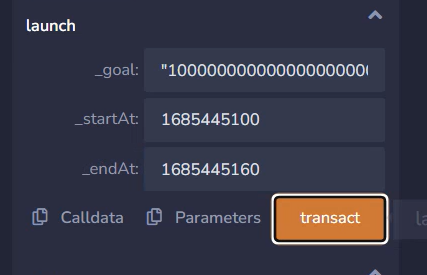
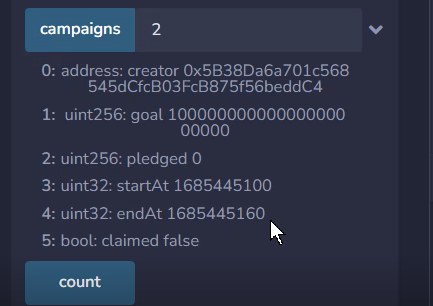
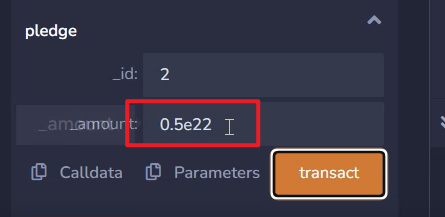
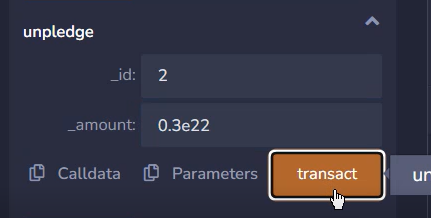
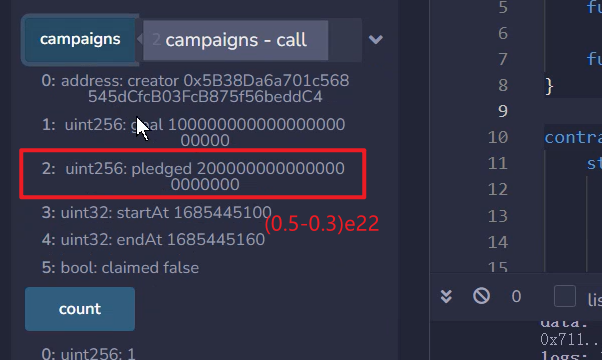
    function getTimeStamp() public view returns (uint256) {

        return block.timestamp;

    }

}

## 实验过程

1. 部署  
   
2. 创建一个众筹活动 目标一万 十分钟(一分钟=60)  
   这里为了简单 我用的120秒  
   
3. 查询创建的活动, 这里为了简单 我用的120秒  
   
4. 等待时间 准备参与  
     
   
5. 查看状态  
   
6. 等待结束时间之后，创建人可以通过claim取出所有的币  
     
   
7. 创建一个新的众筹活动（没按照老师的数据，以自己仅有的token数据做的）  
   
8. 查询状态  
   
9. 开始后参与投资  
   
10. 取回部分资金  
    
11. 查询活动状态  
    
12. Refund取回  
    