DECENTRALIZED AUTONOMOUS ORGANIZATION (DAO)

Yuri DE SENE ALVIZI

Christiane Manuela AYO NDAZO'O

Arthur MARTINS BRAGA

Outline

- Presentation of DAOs
  - Definition
  - Some popular DAOs
- Study Case : a crowdfunding DAO
  - Definition
  - How does it work ?
  - Benefits
  - Our implementation

Presentation of DAOs

Definition

Decentralized

Traditional organization

DAO

A utonomous

Self-executing

Self-executing

Self-executing

Self-executing

Self-executing

Self-executing

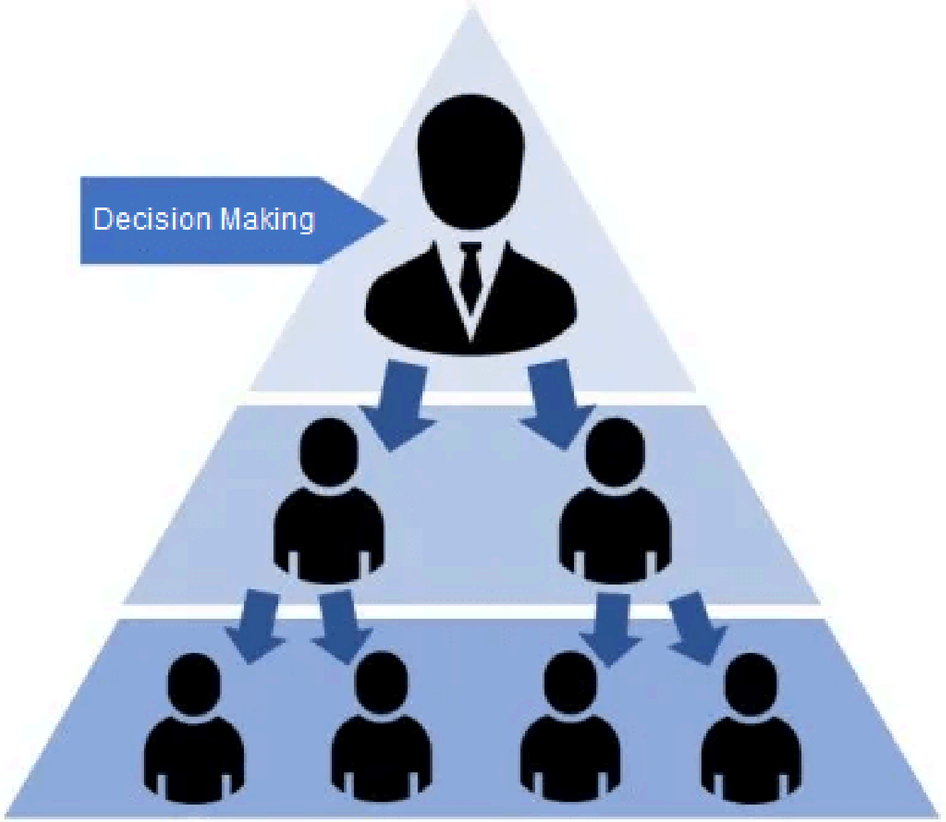
Self-executing

Self-executing

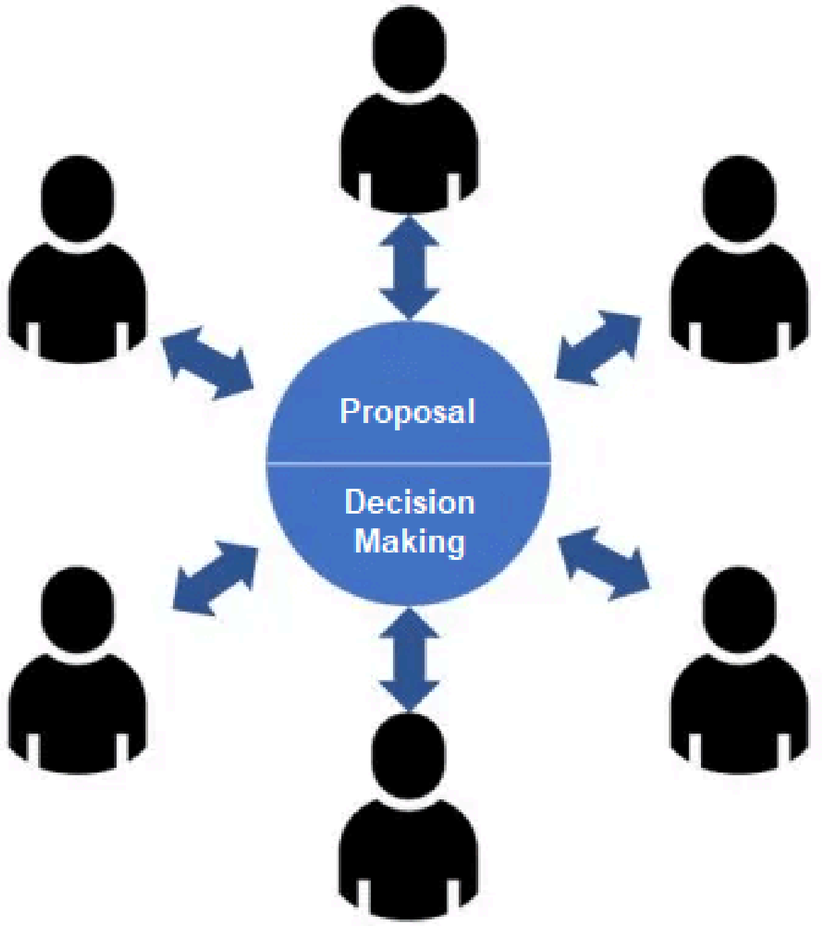
Organization

CODE IS LAW

Traditional Organization



DAO



A blockchain-powered organization that can run on its own without any central authority or management hierarchy

Some popular DAOs

The DAO

The DAO

Founded in **2016**

Instantiated on the **ethereum blockchain**

Token : **DAO**

More than **18000 stakeholders**

11.5 millions eth of assets

MakerDAO

MakerDAO

Founded in **2014**

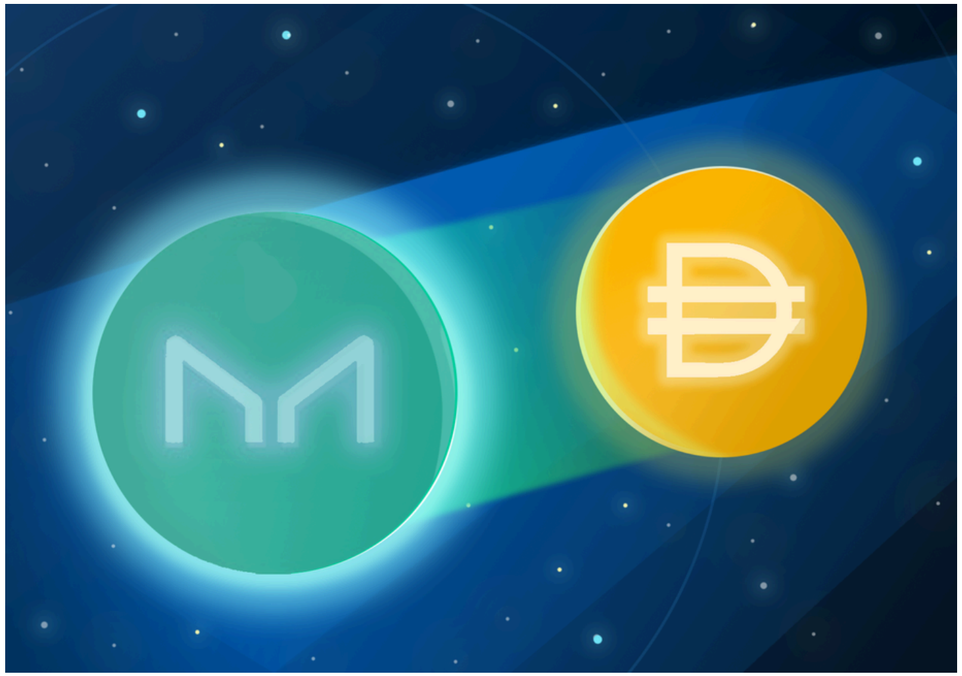
Instantiated on the **ethereum blockchain**

Governance Token : **MKR**

Governs the **Maker Protocol**

DeFi app

Uses and controls the **Dai**



Study case : a crowdfunding DAO

Definition

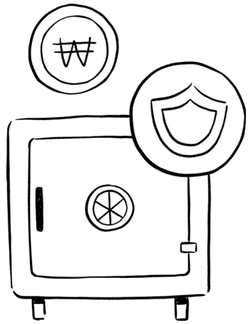
Crowdfunding

Crowdfunding DAO

crowdfunding DAO = spirit of the crowdfunding
but governance and organization made on a DAO

How it works

How it works



BLOCKCHAIN

Provides the infrastructure for secure, transparent, and immutable transactions.

TOKENS

Represents voting power and ownership, typically distributed to members

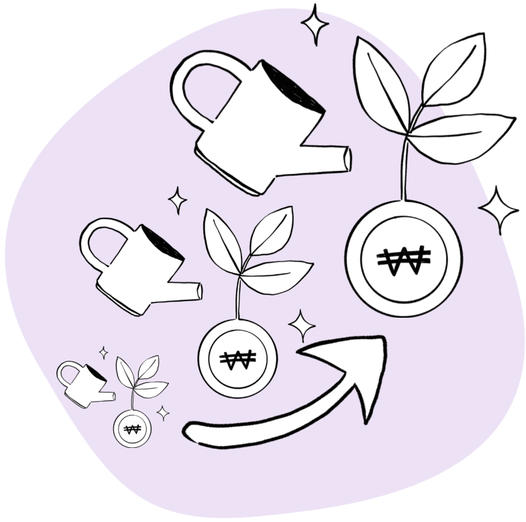
SMART CONTRACTS

Automated, self-executing contracts with the terms directly encoded on a blockchain

How it works

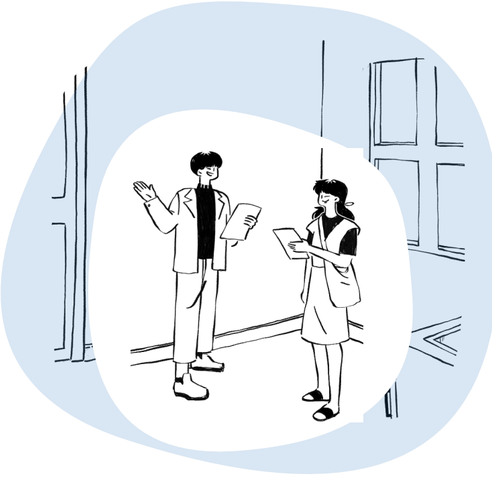
Project voting

Project idea



Set milestones and dead lines

Publish the project



Investors will vote if your project should be crowdfunded

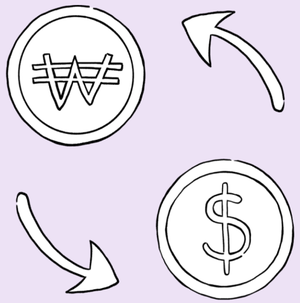
TOKENS

SMART CONTRACTS

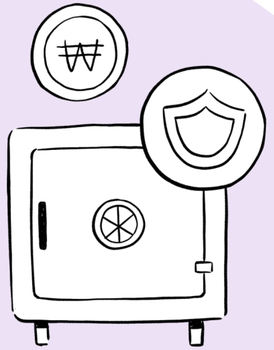
How it works

Project funding

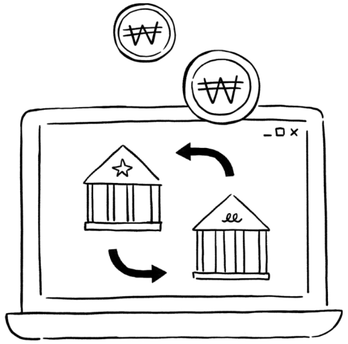
Investors will fund your project



Investors earn tokens proportional to its contribution



The resources are locked until a milestone is achieved



In the meantime, investors are free to retrieve their money.

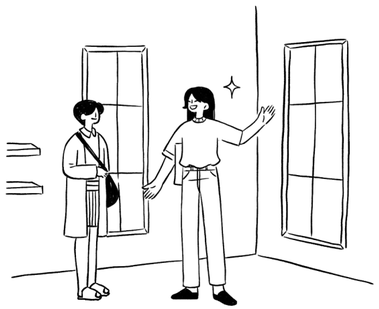
SMART CONTRACTS

How it works

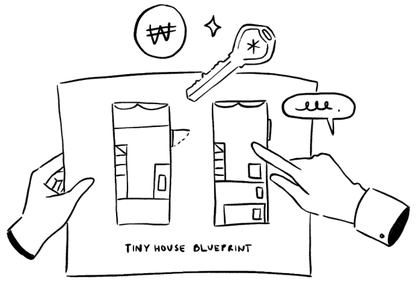
Funding realising

SMART CONTRACTS

TOKENS



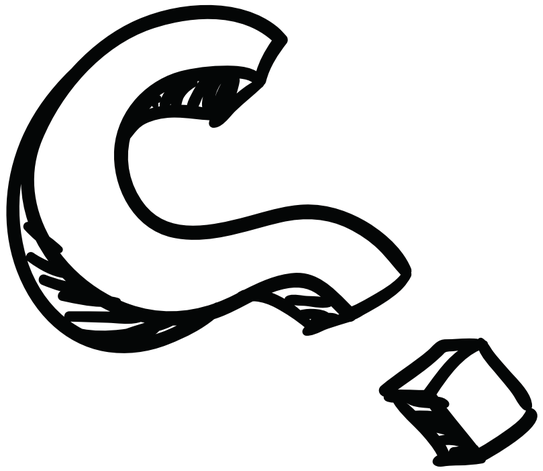
A milestone is achieved before the deadline



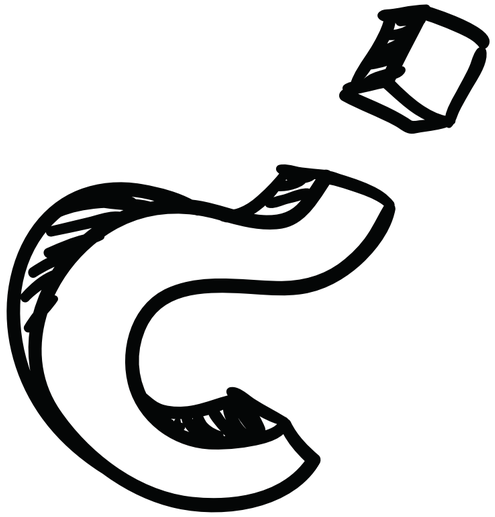
The project owner publishes what was done

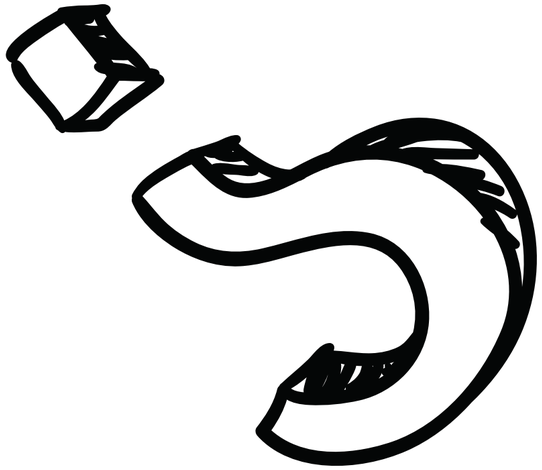
Investors check that what was promised has been delivered

Investors vote for realising the founds of the milestone



Why choosing a crowdfunding DAO over a traditional one ?





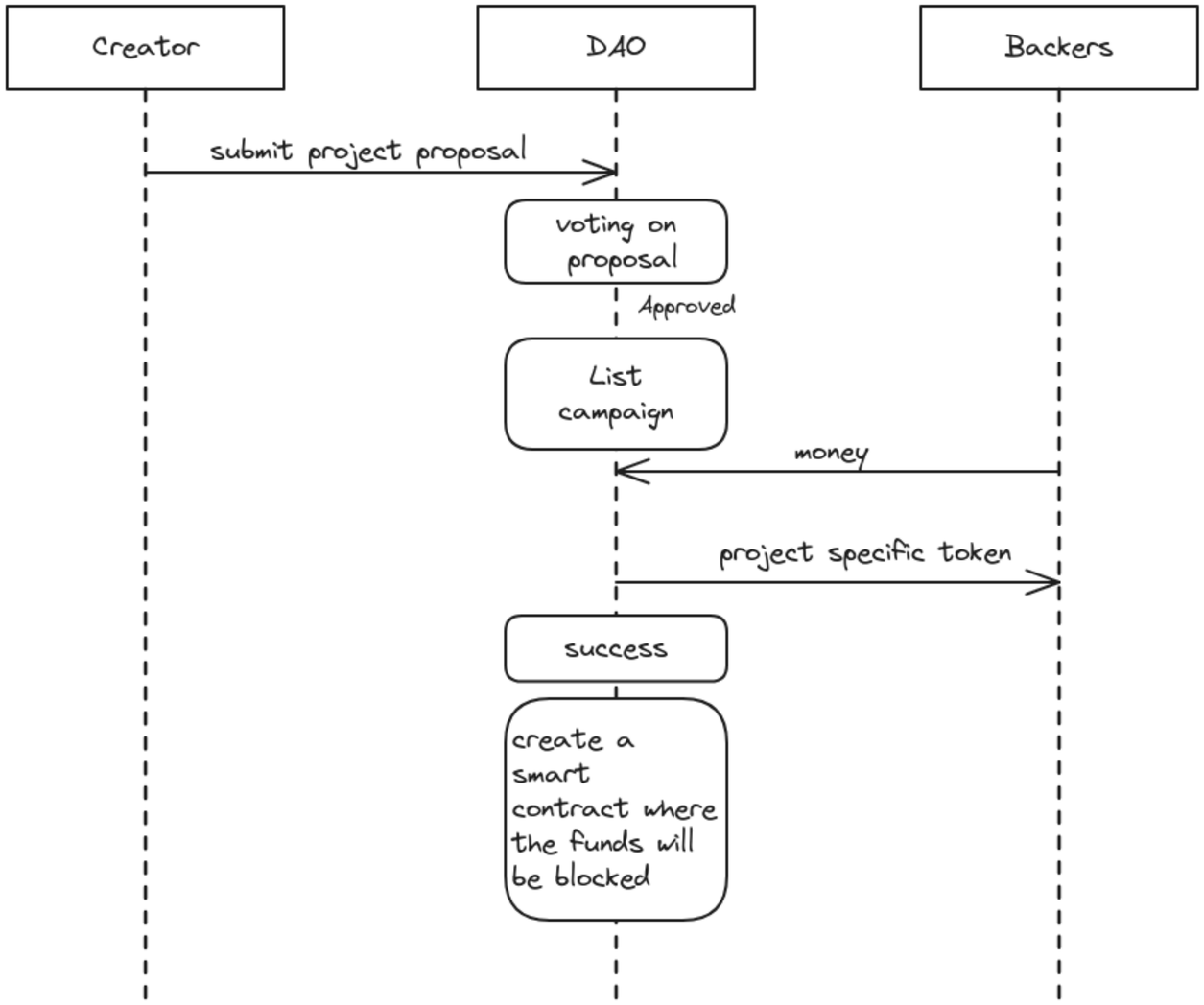
Interest of a crowdfunding DAO

More reach

- More reach
- Transparency

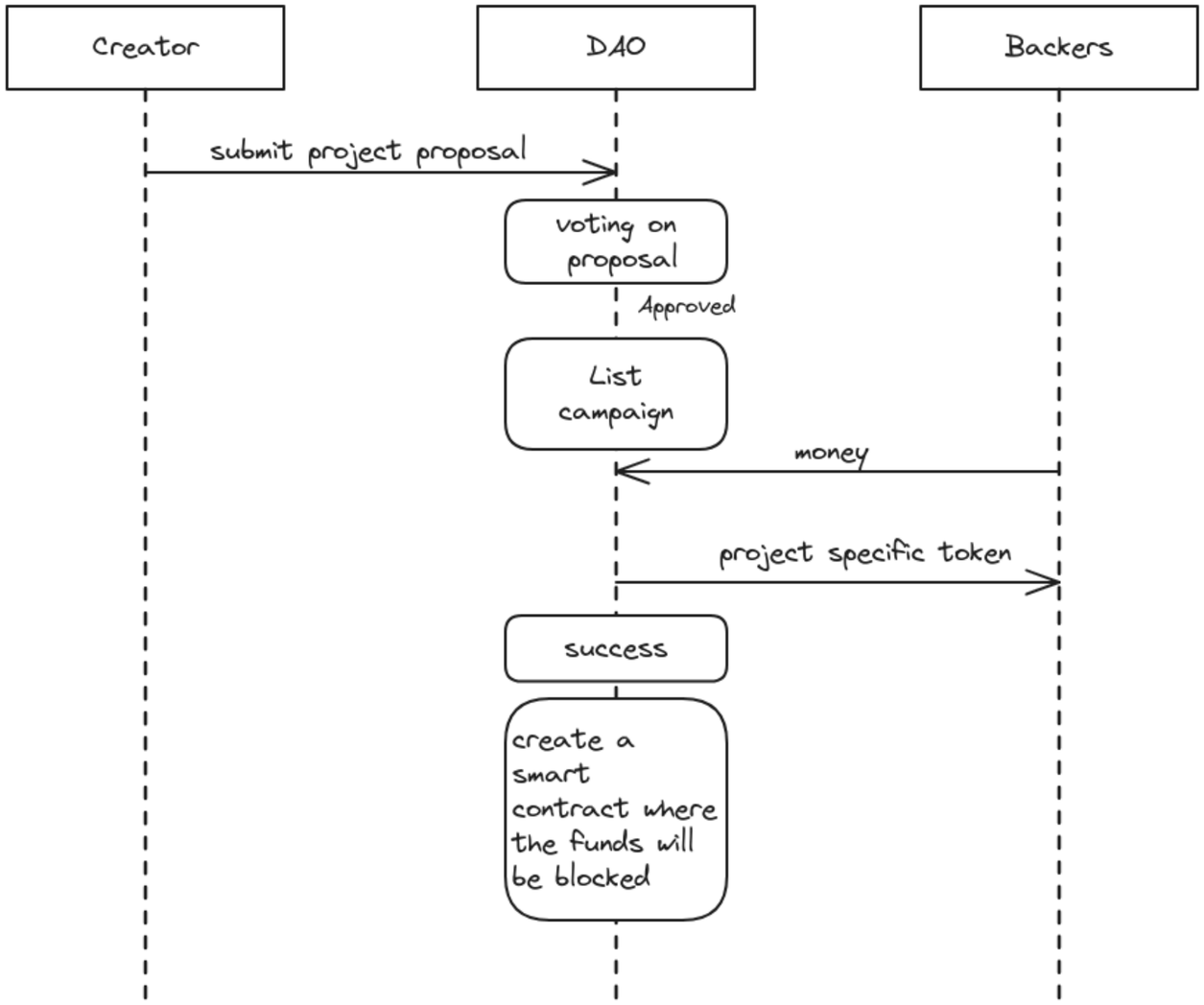
- More reach
- Transparency
- Increase liquidity

Our implementation



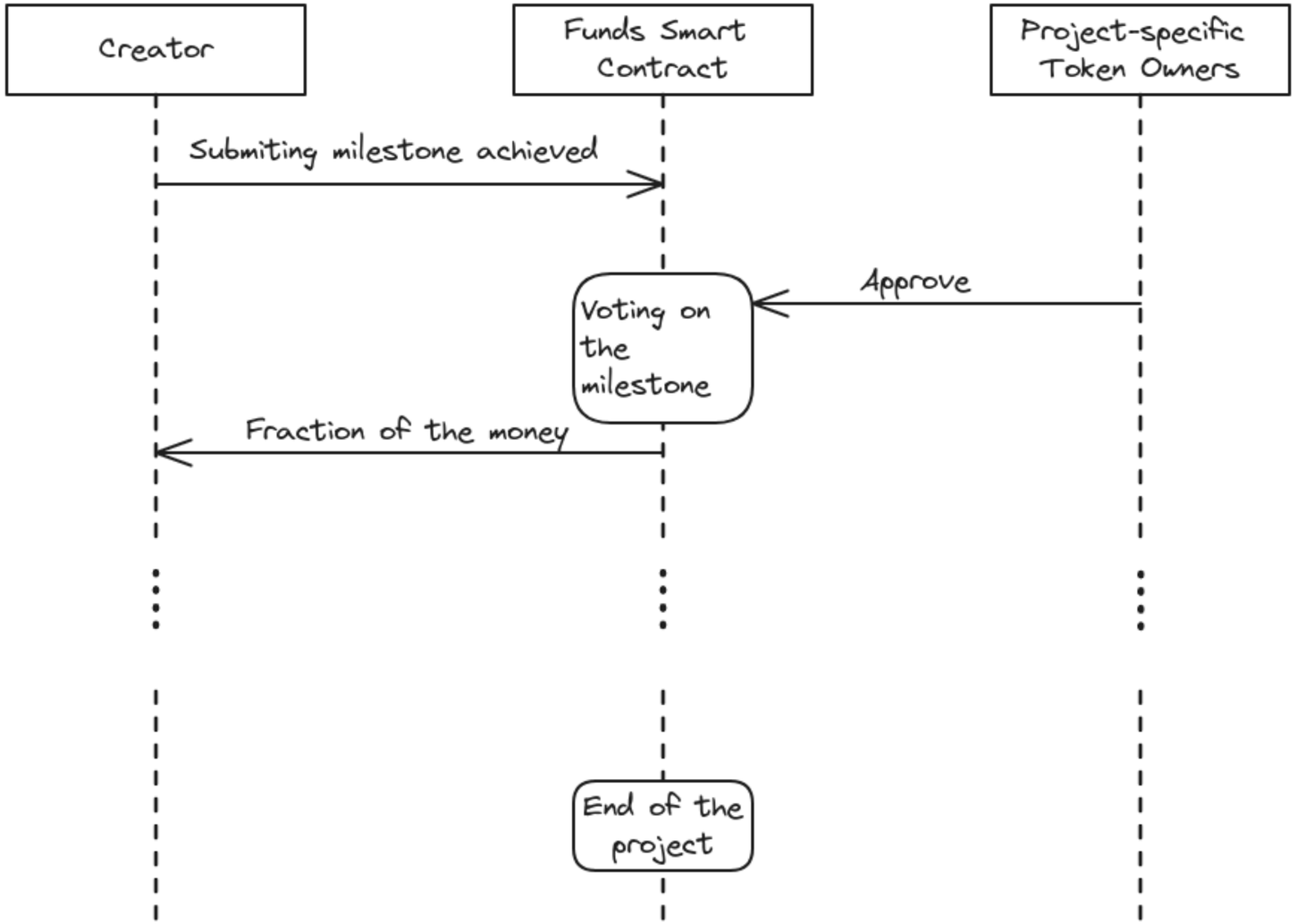
- First, DAO managers need to vote to approve or discard a project proposal
- If approved, the project will be listed in the platform

Our implementation



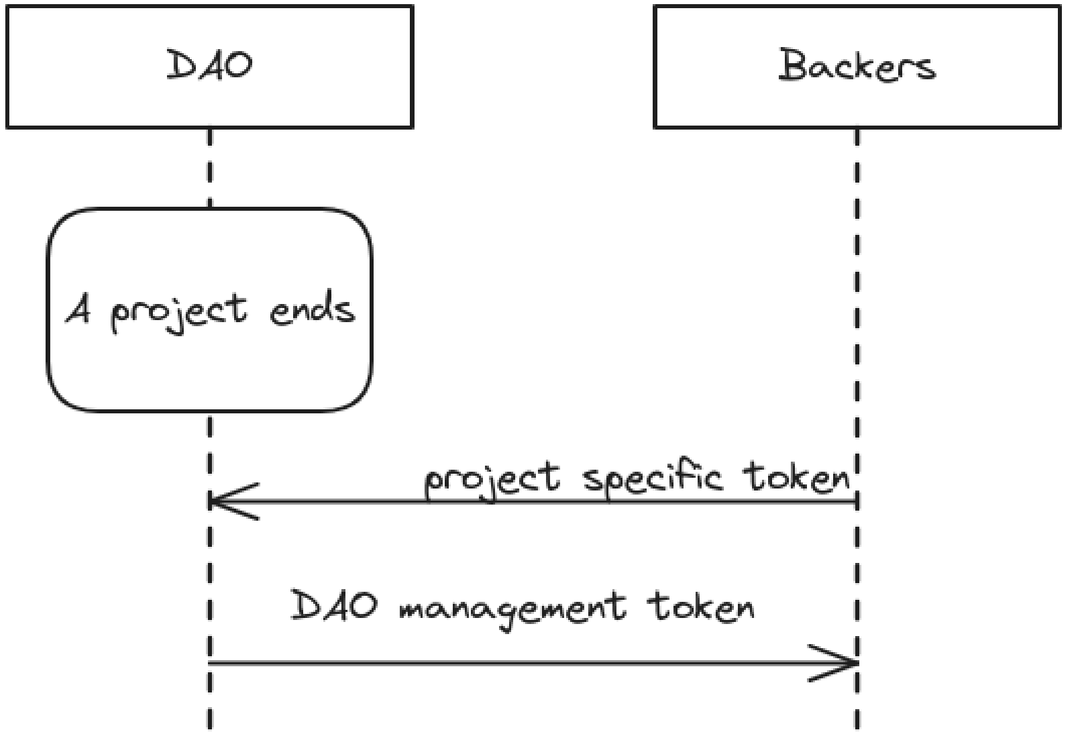
- When listed, a project can receive funds
- Backers receive tokens specific to the project they funded
- If the project reaches the fund target, the funds are blocked in a smart contract

Our implementation



- After the creation of the funds smart contract, creators will submit reports with the milestones
- Funders of that project will vote to approve or not to unlock fraction of the money

Our implementation



- When a project ends, project-specific tokens are exchanged for the DAO management token

Our implementation

```
function approveCampaign(address campaignAddress) external {
    require(managerToken.balanceOf(msg.sender) > 0, "Only managers can approve campaigns");

    require(isCampaignInList(campaignAddress, proposedCampaignsList), "Campaign is not in proposed
            campaigns list");

    ProposedCampaign storage proposal = proposedCampaigns[campaignAddress];
    require(proposal.campaign != address(0), "Campaign does not exist");
    require(!proposal.approvedBy[msg.sender], "Manager already approved this campaign");

    proposal.approvedBy[msg.sender] = true;
    proposal.approvals += managerToken.balanceOf(msg.sender);

    emit CampaignApproved(campaignAddress, msg.sender, proposal.approvals);

    if (isCampaignApproved(campaignAddress)) {
        listedCampaigns.push(campaignAddress);
        removeCampaignFromList(campaignAddress, proposedCampaignsList);
    }
}
```

Our implementation

```
function isCampaignApproved(address campaignAddress) public view returns (bool) {
    ProposedCampaign storage proposal = proposedCampaigns[campaignAddress];
    return proposal.approvals >= (totalManagerTokens * 50) / 100;
}
```

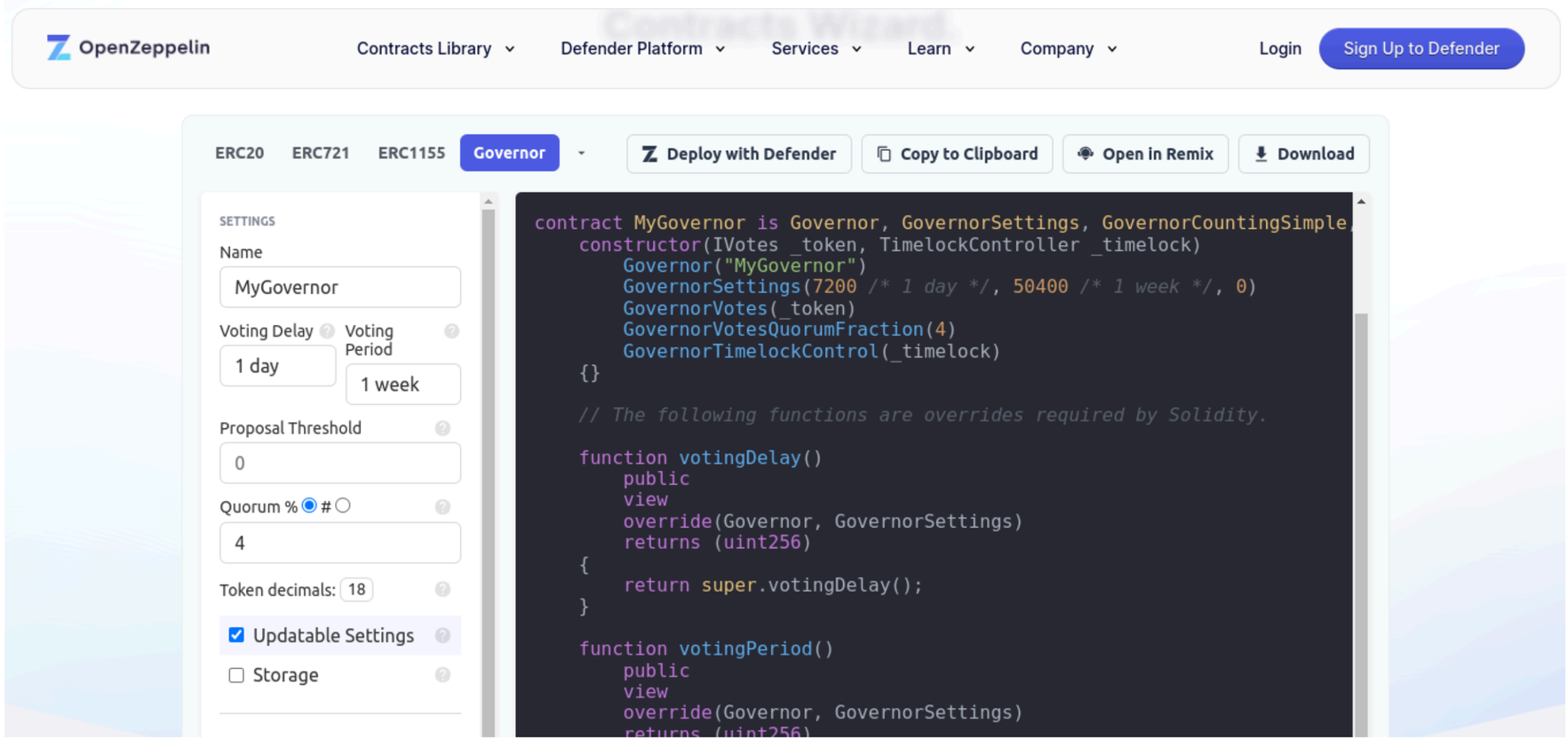
Our implementation

```
function vote(address campaignAddress, bool fundsReleased) public {
    require(isCampaignInList(campaignAddress, successfulCampaigns), "Campaign is not successful");
    Campaign campaign = Campaign(campaignAddress);
    campaign.vote(fundsReleased, msg.sender);
}
```

Our implementation

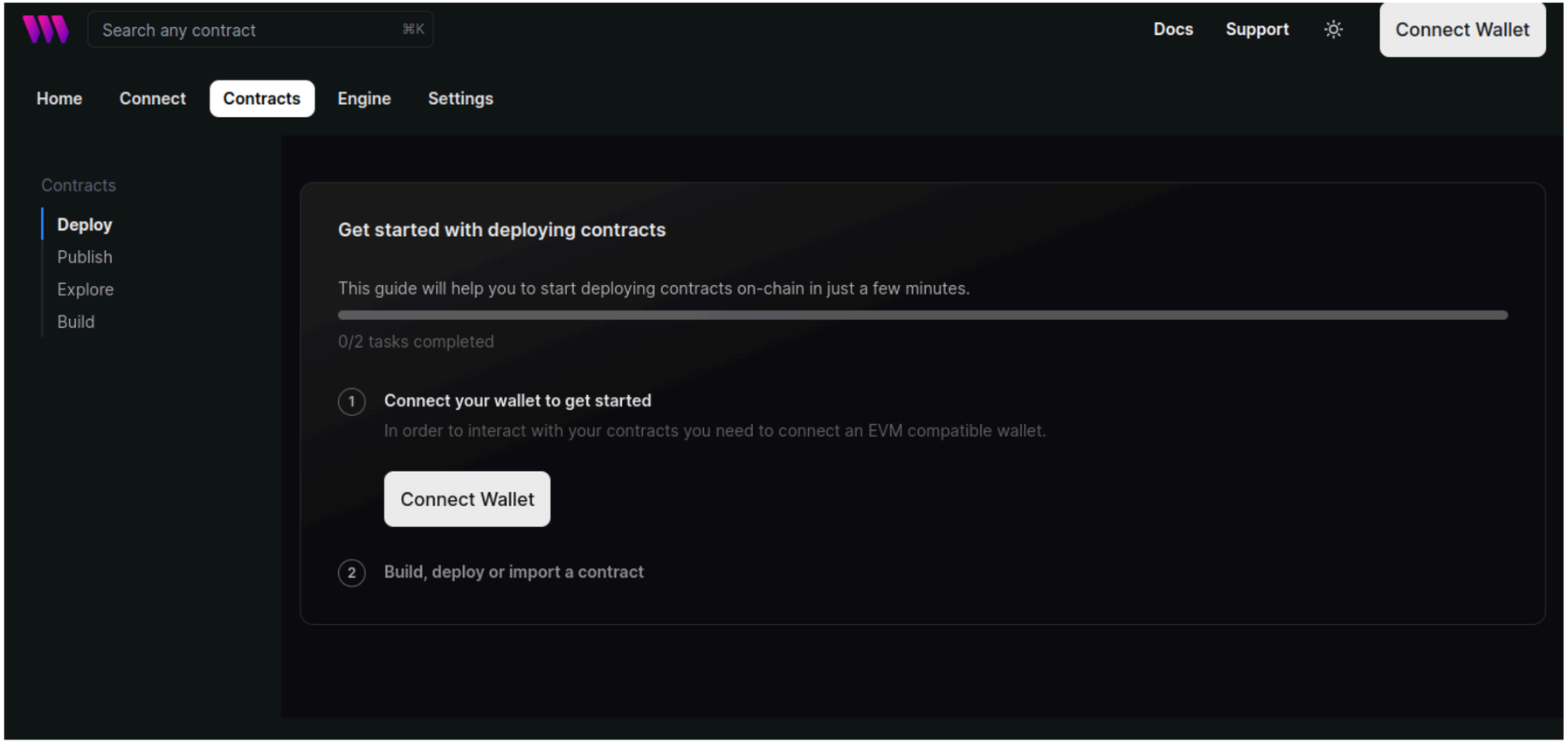
```
function vote(bool releaseFunds, address voter) external {
    require(votingActive, "No active voting");
    require(campaignToken.balanceOf(voter) > 0, "No tokens to vote");
    require(!votes[voter], "Already voted");

    votes[voter] = true;

    if (releaseFunds) {
        votesForRelease += campaignToken.balanceOf(voter);
    } else {
        votesAgainstRelease += campaignToken.balanceOf(voter);
    }

    isVoteEnded();

    emit VoteCast(voter, releaseFunds);
}
```

Useful tools



https://www.openzeppelin.com/contracts

Useful tools



https://thirdweb.com/dashboard/contracts/deploy

THANKS!