

DECENTRALIZED AUTONOMOUS ORGANIZATION (DAO)

Yuri DE SENE ALVIZI

Christiane Manuela AYO NDAZO'O

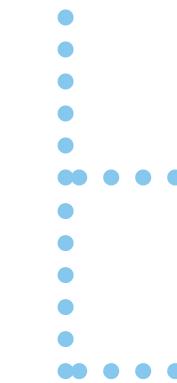
Arthur MARTINS BRAGA



Outline



Presentation of DAOs

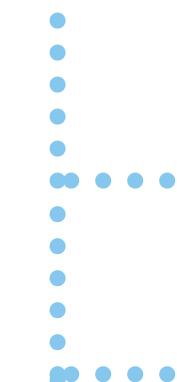


Definition

Some popular DAOs

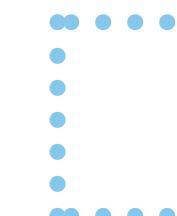


Study Case : a crowdfunding DAO



Definition

How does it work ?



Benefits



Our implementation

Presentation of DAOs



Definition



D ecentralized



Traditional organization



DAO

A utonomous



Self-executing
Self-executing
Self-executing
Self-executing
Self-executing
Self-executing
Self-executing

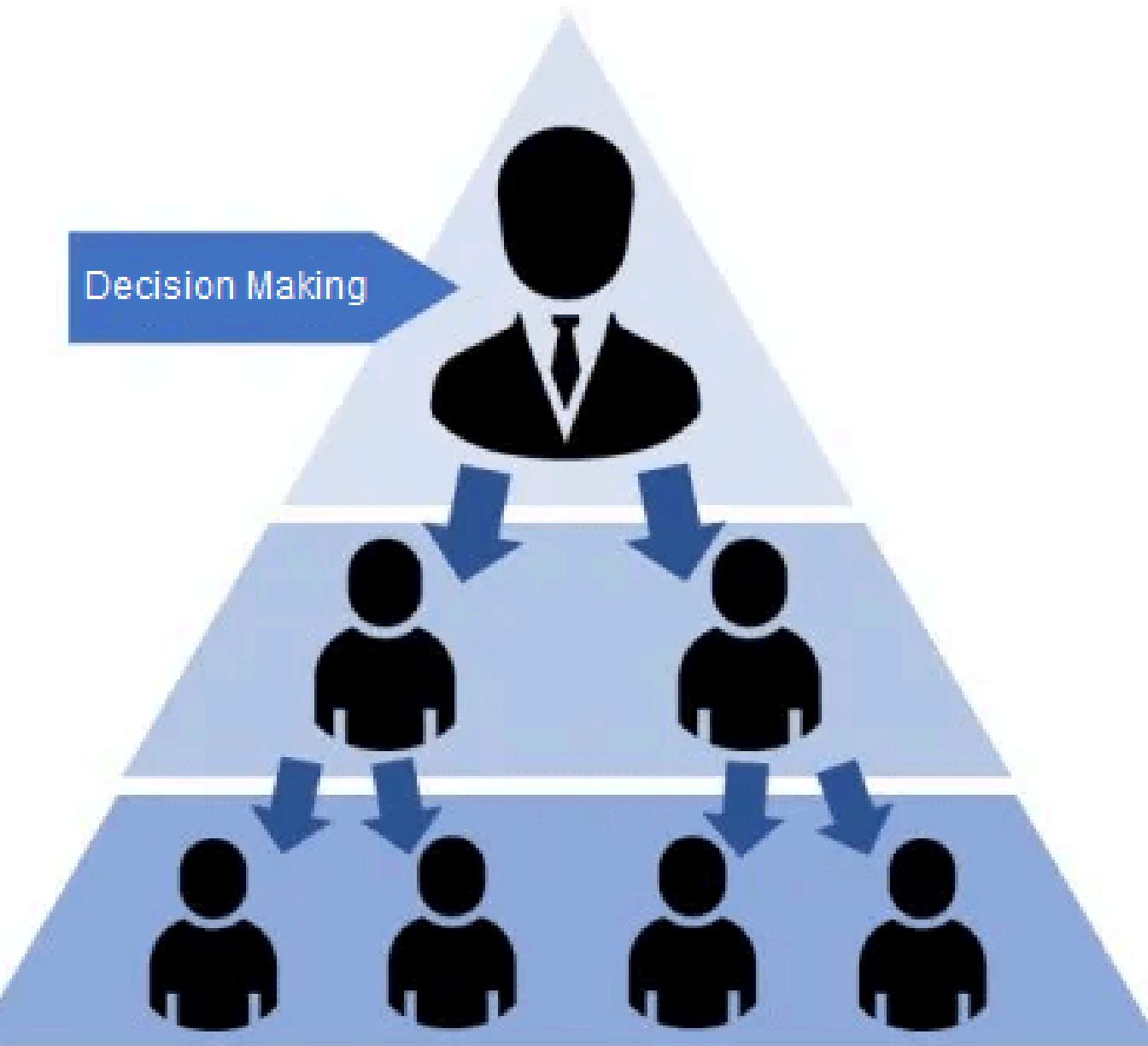


O rganization

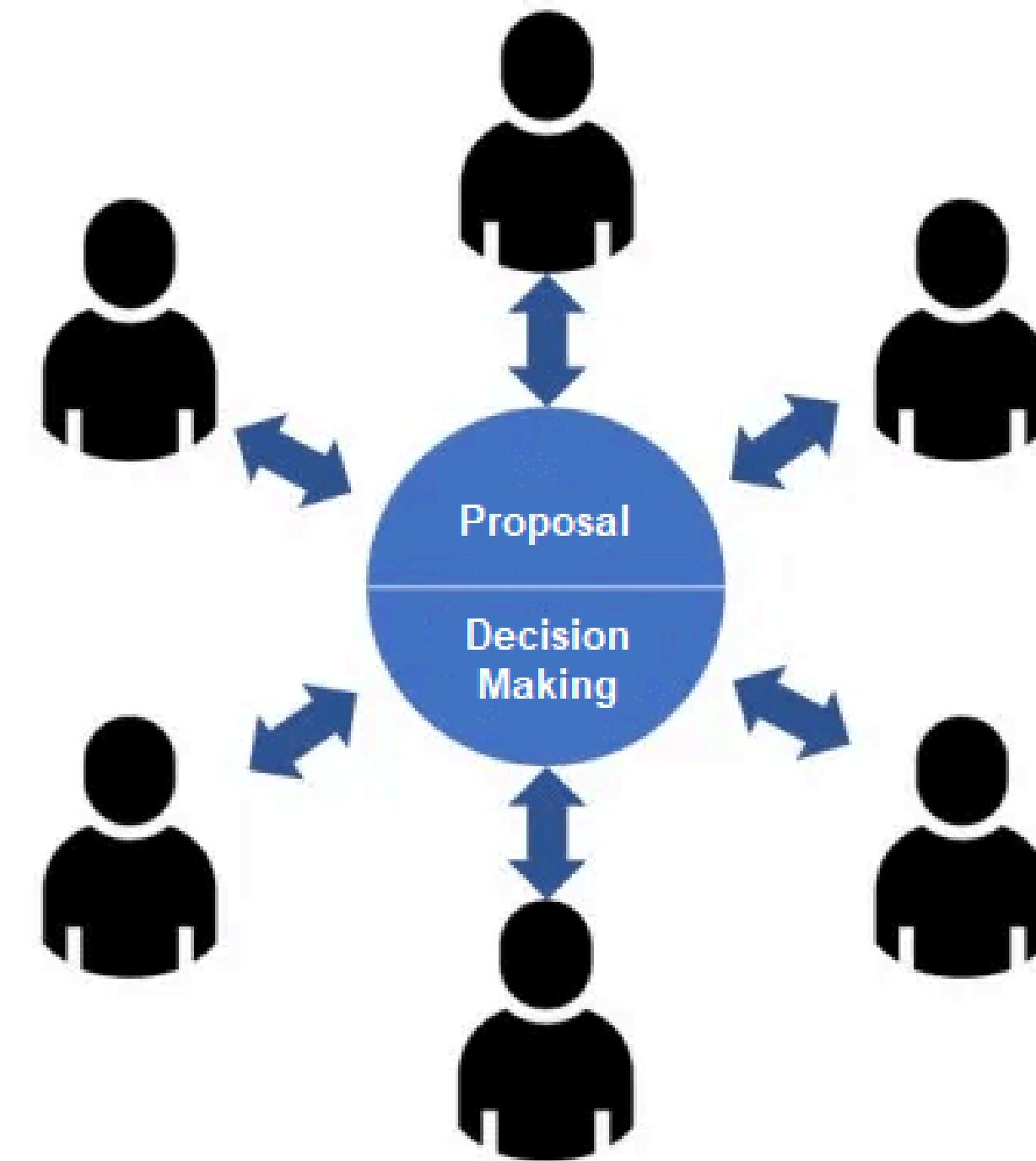


CODE IS LAW

Traditional Organization



DAO



Decentralized Autonomous Organization

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



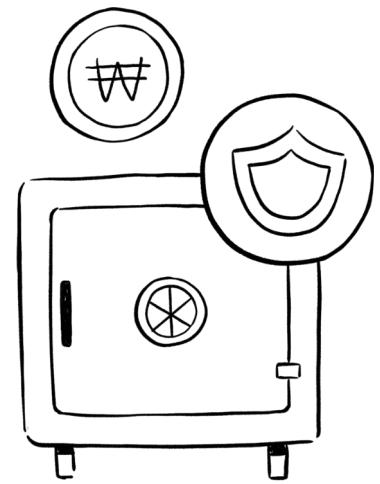
Crowdfunding DAO

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

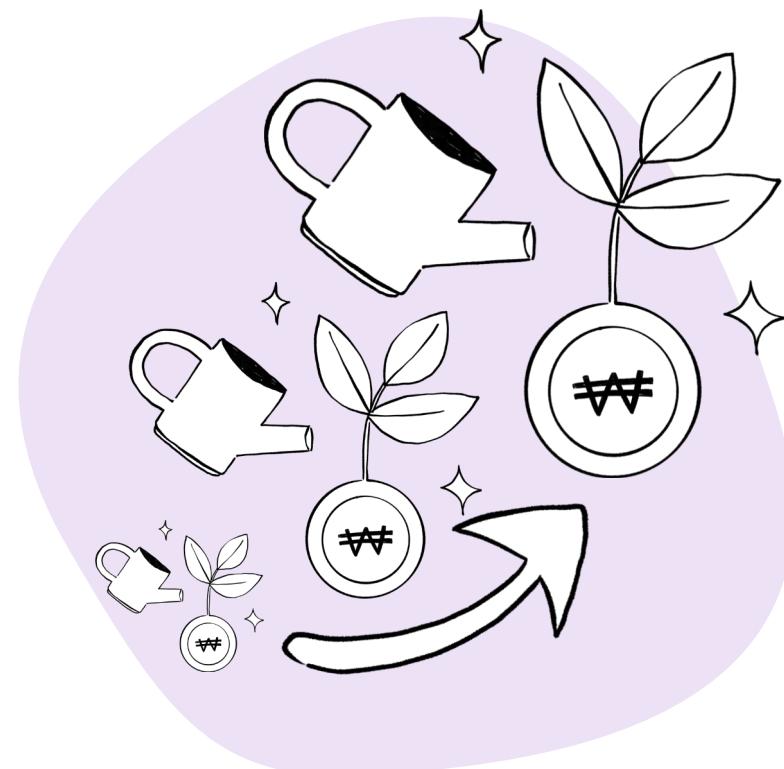
Automated, self-executing contracts with the terms directly encoded on a blockchain

How it works

Project voting



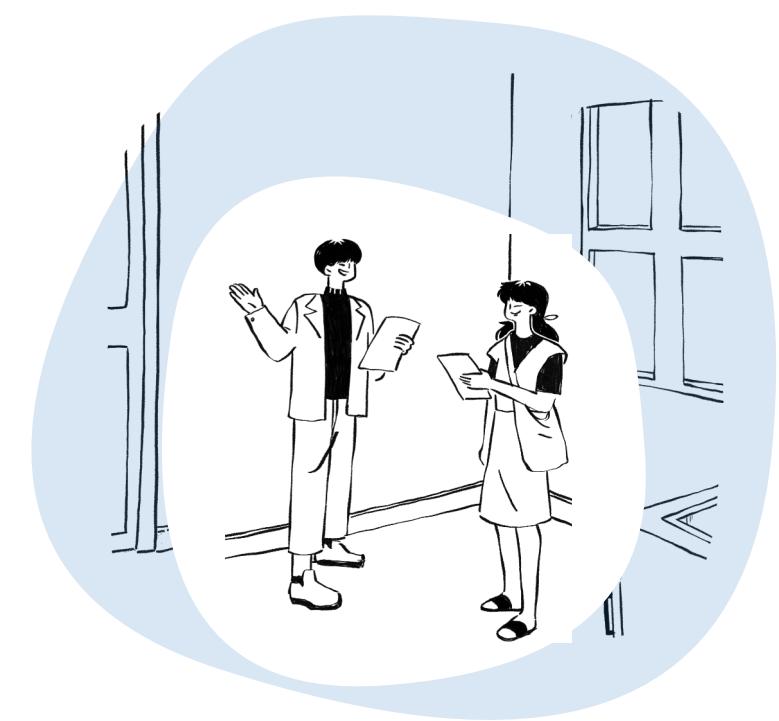
Project idea



Set milestones
and dead lines



Publish the
project



Investors will
vote if your
project should
be crowdfunded



SMART
CONTRACTS



How it works

Project funding



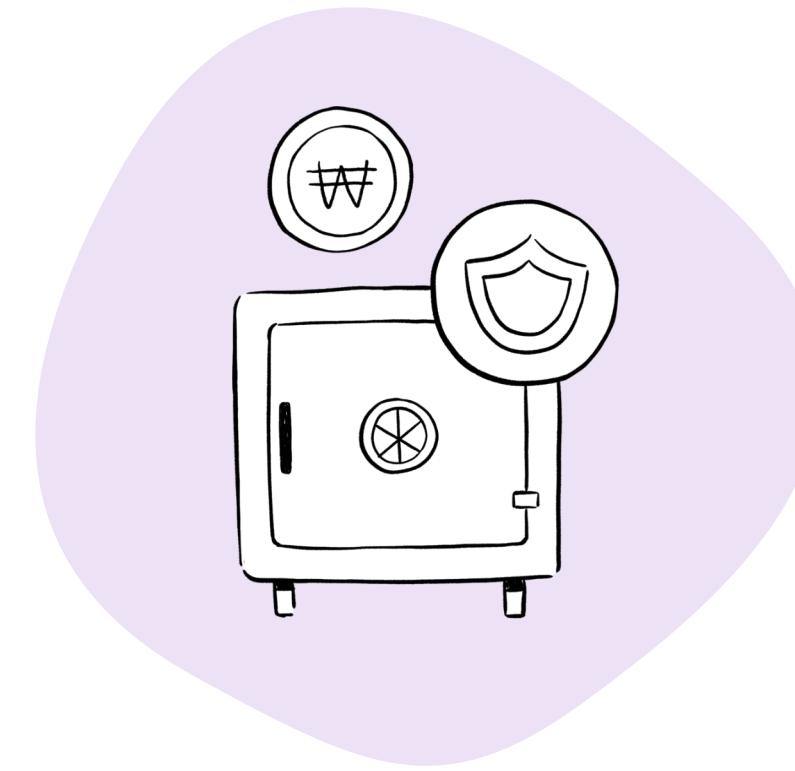
SMART
CONTRACTS



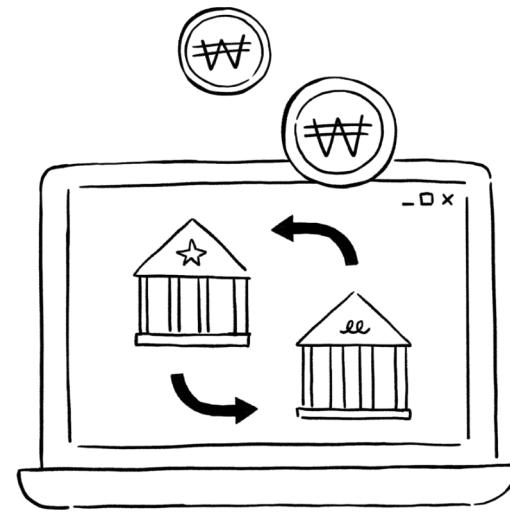
Investors will fund
your project



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.

How it works

Funding realising



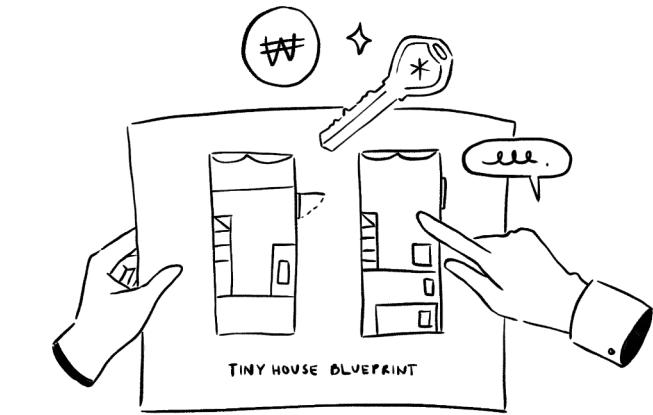
SMART
CONTRACTS



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 funds of the milestone



Why choosing a crowdfunding DAO over a traditional one ?



Interest of a crowdfunding DAO





More reach



More reach



Transparency



More reach

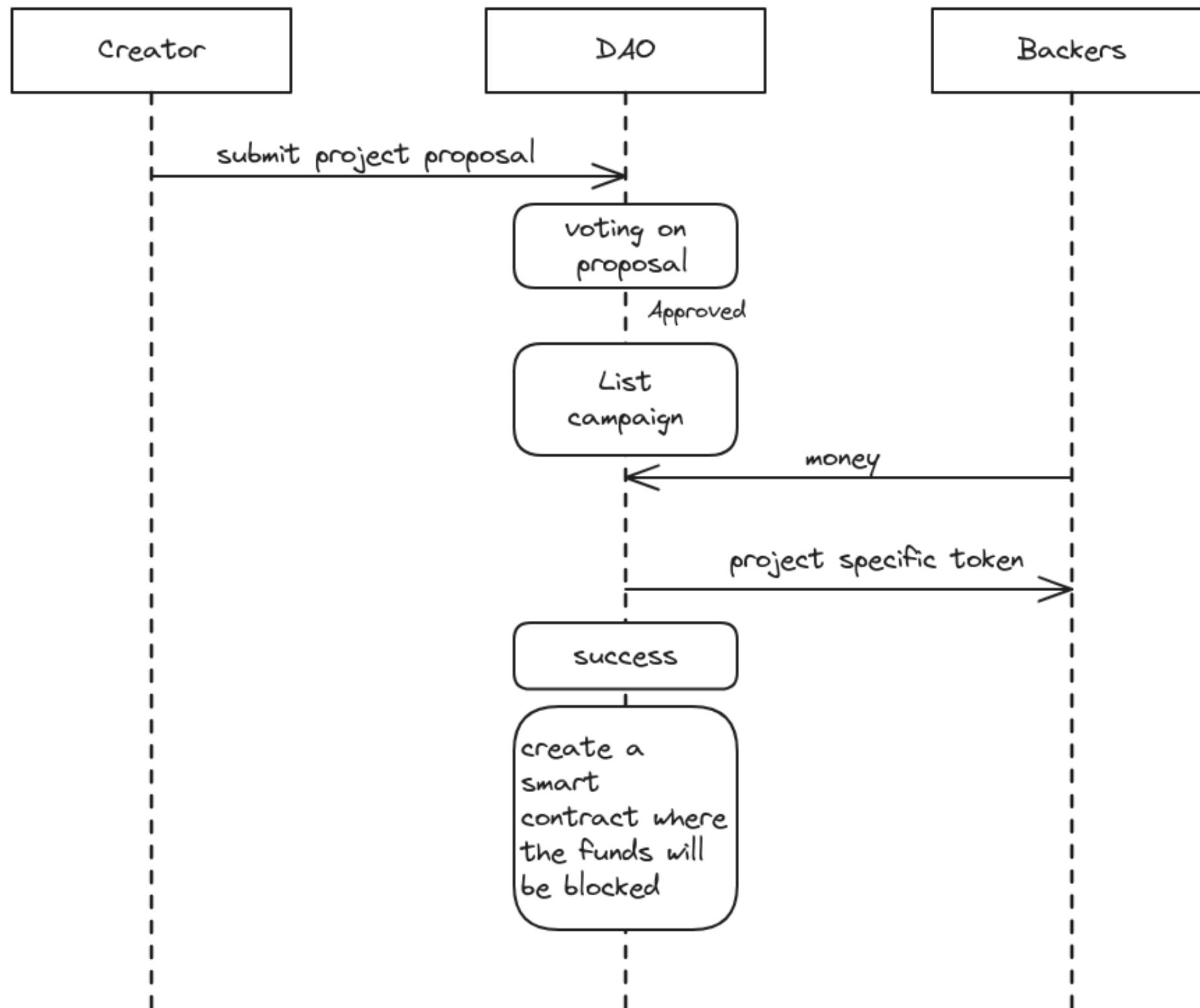


Transparency



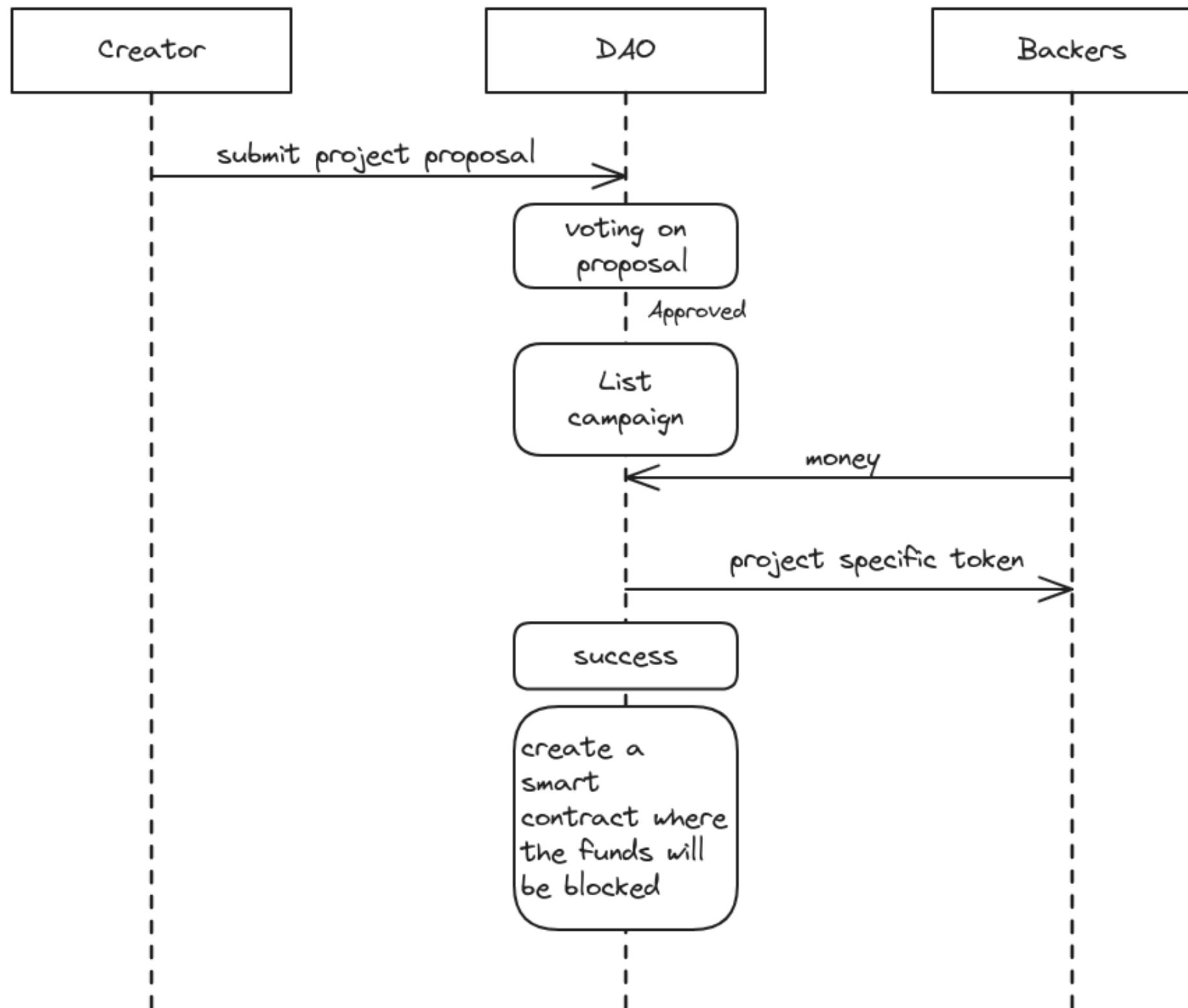
Increase liquidity

Our implementation



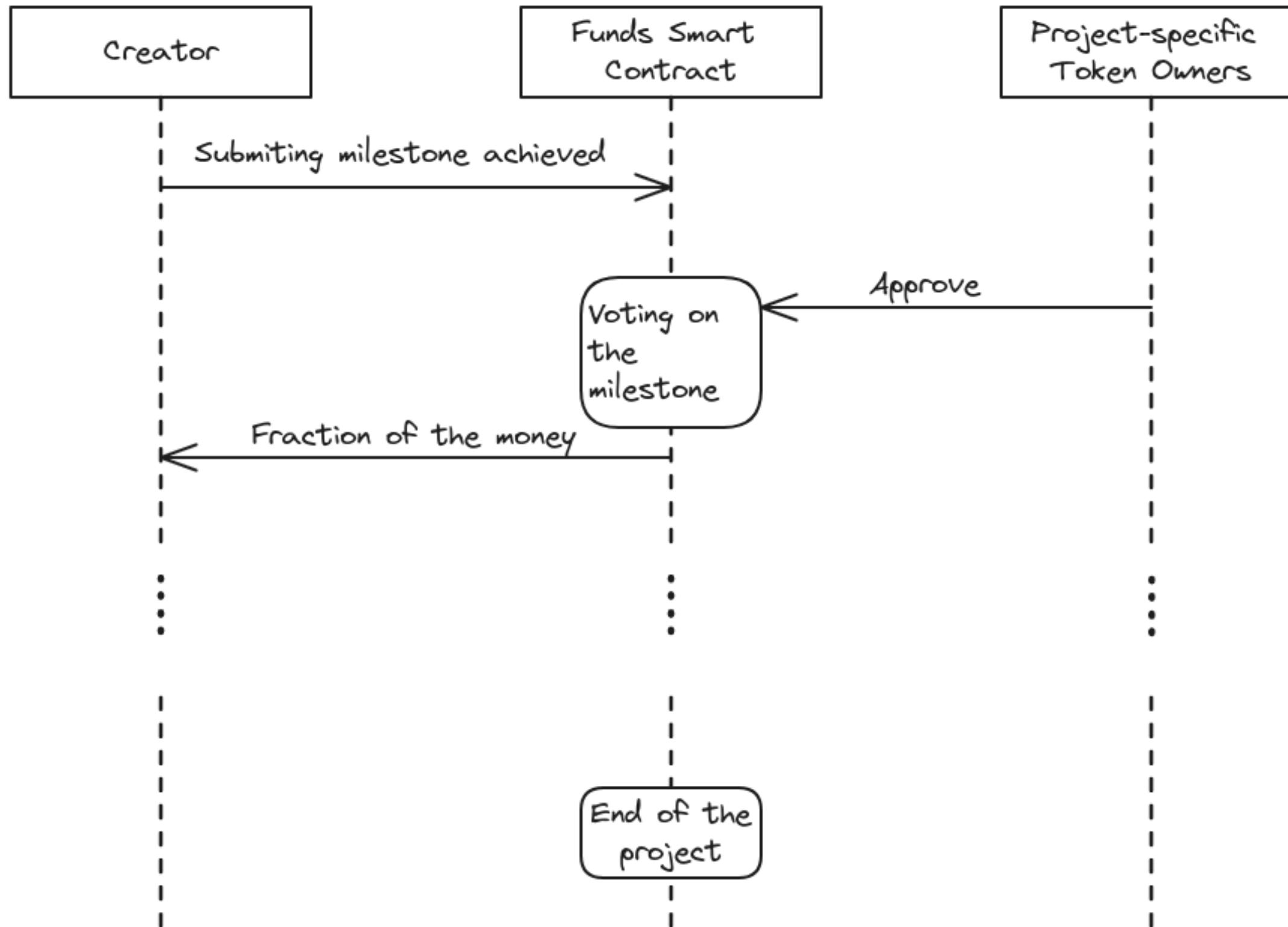
- First, DAO managers need to vote to approve or discard a project proposal
- If approved, the project will be listed in the platform

Our implementation



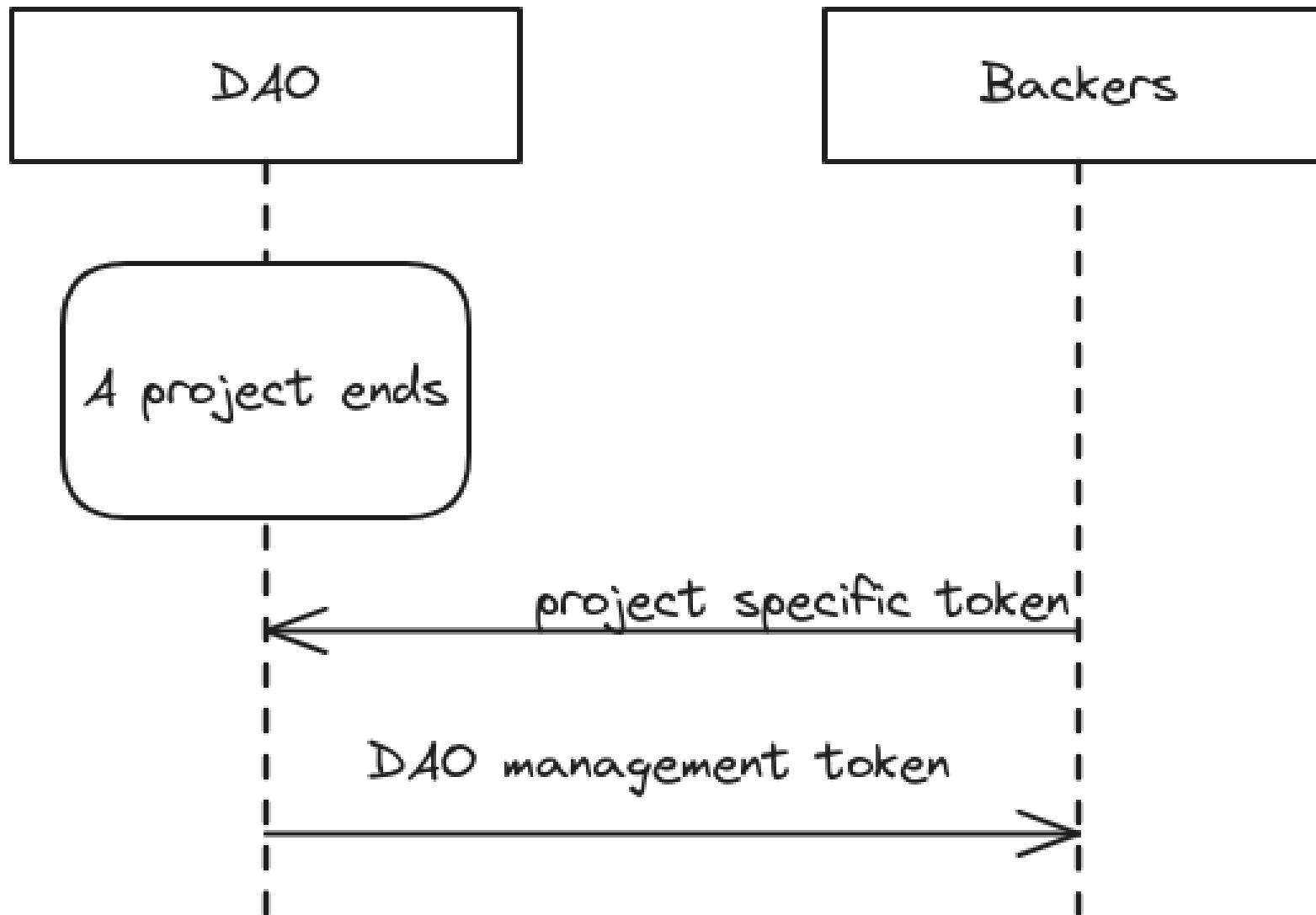
- When listed, a project can receive funds
- Backers receive tokens specific to the project they funded
- If the project reaches the fund target, the funds are blocked in a smart contract

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

The screenshot shows the OpenZeppelin Contracts Library interface. At the top, there is a navigation bar with links for OpenZeppelin, Contracts Library, Defender Platform, Services, Learn, Company, Login, and Sign Up to Defender. Below the navigation bar, there are tabs for ERC20, ERC721, ERC1155, and Governor, with Governor being the active tab. To the right of the tabs are buttons for Deploy with Defender, Copy to Clipboard, Open in Remix, and Download.

The main area is divided into two sections: 'SETTINGS' on the left and the 'Solidity code' on the right. The 'SETTINGS' section contains the following fields:

- Name: MyGovernor
- Voting Delay: 1 day
- Voting Period: 1 week
- Proposal Threshold: 0
- Quorum %: 4
- Token decimals: 18
- Updatable Settings
- Storage

The 'Solidity code' section displays the following contract definition:

```
contract MyGovernor is Governor, GovernorSettings, GovernorCountingSimple {
    constructor(IVotes _token, TimelockController _timelock)
        Governor("MyGovernor")
        GovernorSettings(7200 /* 1 day */, 50400 /* 1 week */, 0)
        GovernorVotes(_token)
        GovernorVotesQuorumFraction(4)
        GovernorTimelockControl(_timelock)
    {}

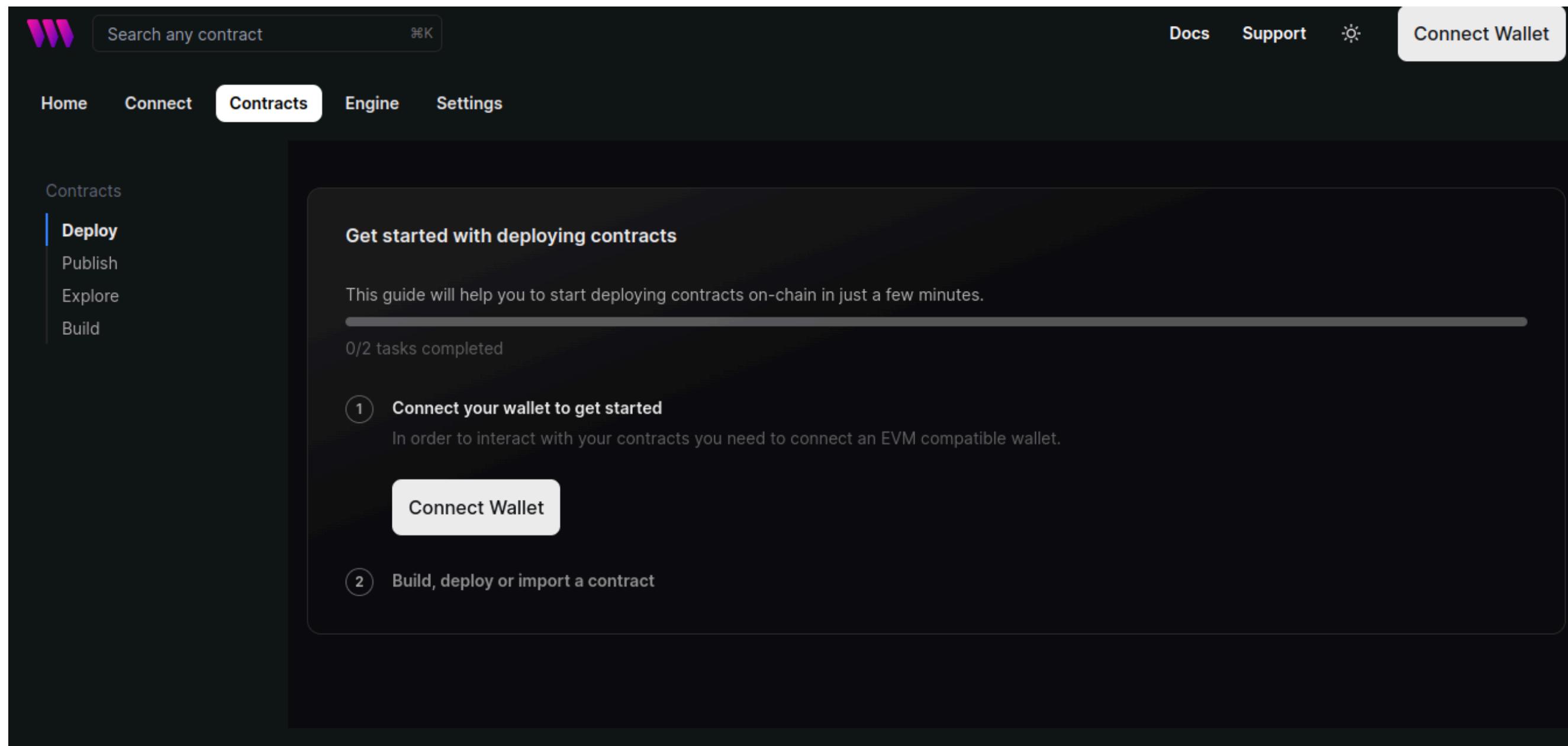
    // The following functions are overrides required by Solidity.

    function votingDelay()
        public
        view
        override(Governor, GovernorSettings)
        returns (uint256)
    {
        return super.votingDelay();
    }

    function votingPeriod()
        public
        view
        override(Governor, GovernorSettings)
        returns (uint256)
    }
```

<https://www.openzeppelin.com/contracts>

Useful tools



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



THANKS!