Voting dApp: Participatory Budgeting

Nicolas, Yuren, Maximo & Hortense

M2 Digital Economics - Paris Dauphine-PSL University

March 25, 2024

Elevator pitch

Participatory Budgeting = Democratic process in which citizens are involved in deciding how public money is spent

Our dApp makes it easy for both proposers and voters to participate in the democratic process. By omitting the visibility of the budget allocation, we aim to streamline the voting process and encourage voters to prioritize projects based on their merit rather than financial considerations.

Target audience = local governments, community organizations, and civic-minded individuals who seek a more transparent and inclusive approach to budget allocation

Motivation of the project

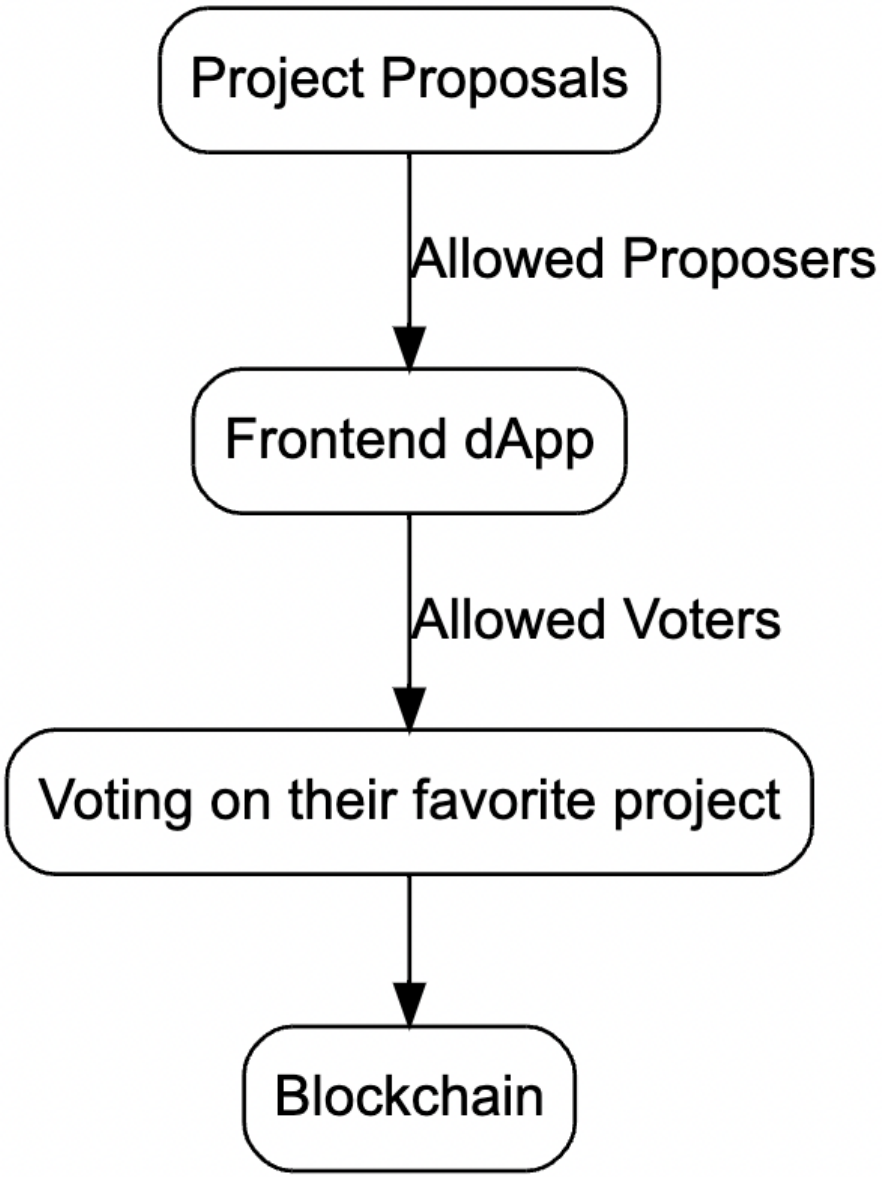
Traditional participatory budgeting process lack of transparency and inclusivity leading to decisions not fully reflecting the priorities of citizens.

The role of our voting dApp is to engage more citizens, ensuring:

- **Security**, relying on blockchain technology
- **Transparency**
- **User-friendly interface**

Technical Overview

General Architecture



- **ParticipatoryBudgeting.sol**: manages participatory budgeting by allowing users to propose projects, vote on them, and allocate funds accordingly. Administered through access control via the Ownable contract
- Main functions include: **totalProjects, Project** and mapping for **projects, proposers**

General Architecture

totalProjects	Tracks the total number of projects proposed
Project	Defines the structure of each project, including admin, description, votes, project ID, and existence status
projects	Maps project IDs to their respective project details
proposers	Maps addresses to their eligibility to propose projects
ProjectProposed	Triggered when a project is proposed, emitting project ID, admin address, and description
VoteCasted	Triggered when a vote is casted, emitting voter address, project ID, and votes
FundsAllocated	Triggered when funds are allocated to a project, emitting project ID and allocated amount

Table: functions, mappings and events in ParticipatoryBudgeting.sol

General Architecture

- **Proposer.sol**: manages eligibility of users to propose projects within the system, controlled by the contract owner through dynamic addition and removal of proposers, ensuring access control and transparency
- ParticipatoryBudgeting contract interacts internally to manage project proposals, voting, and fund allocation whilst Proposer contract manages the eligibility of users to propose projects.

Proposers	Maps addresses to their eligibility as proposers
ProposerAdded	Triggered when a new proposer is added, emitting the account address
ProposerRemoved	Triggered when a proposer is removed, emitting the account address

Table: description of Proposers and Events

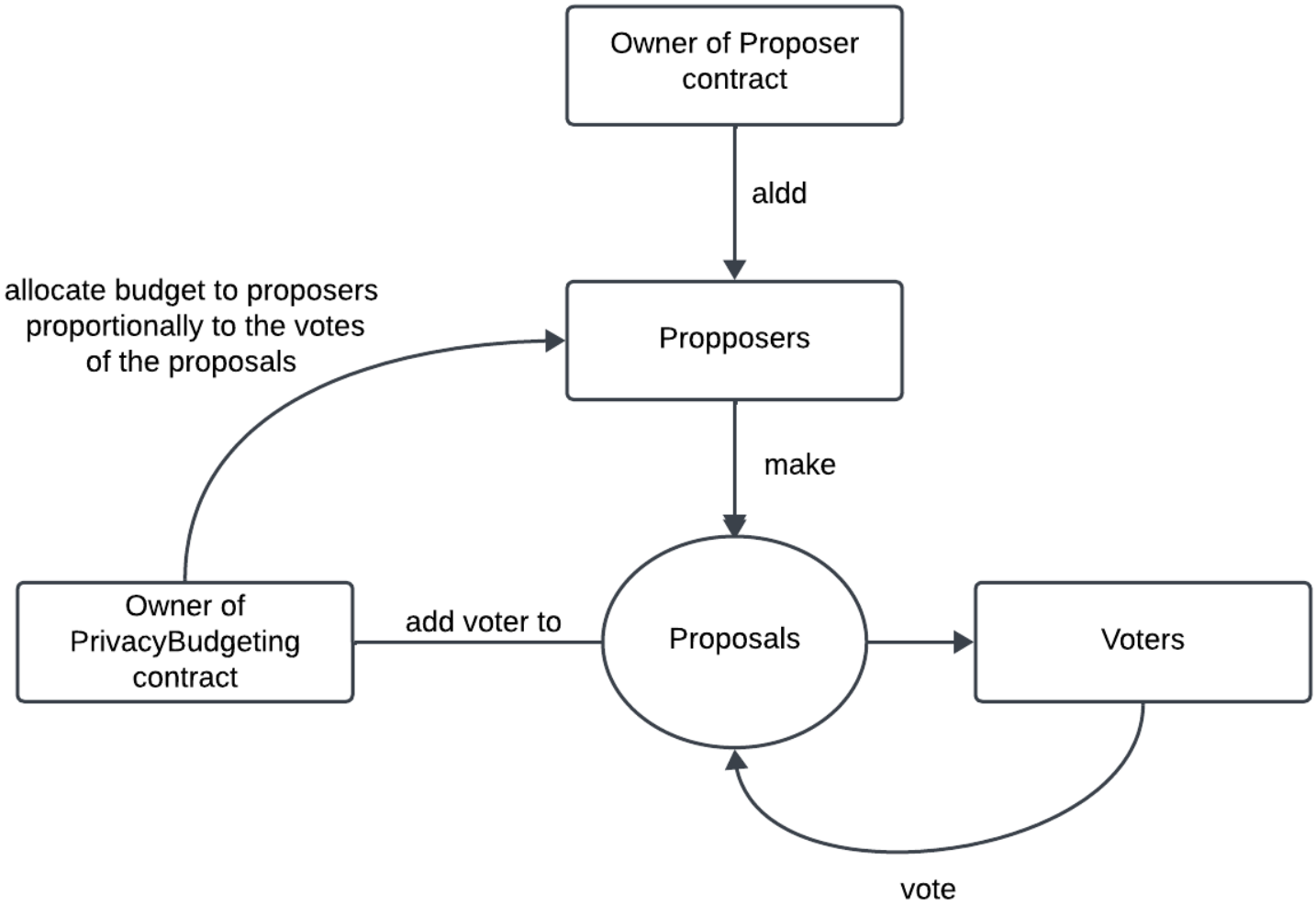
Technical Features & Competitive Analysis

- **Efficiency:** Utilizes SafeMath for secure arithmetic operations, enhancing computational safety
- **Flexibility:** Allows dynamic addition and removal of proposers, providing adaptability to changing requirements
- **Security:** Utilizes the Ownable contract for access control, ensuring only the contract owner can add/remove proposers, i.e. secure management of proposer eligibility
- **User Experience:** Provides clear events for project proposal, voting, and fund allocation, enhancing user transparency and engagement
- **Scalability:** Supports an unlimited number of projects with dynamic project ID assignment
- **Gas Optimization:** Implements gas-efficient code patterns to reduce transaction costs, optimizing gas usage this way

Code quality practice

- **Gas Optimization Techniques:** Utilized gas-efficient patterns such as minimizing storage usage, employing SafeMath for arithmetic operations, and optimizing loops to reduce gas consumption
- **Frequency of Function Calls:** Selected test cases based on the frequency of function calls in real-world scenarios to ensure robustness and efficiency
- **Scalability Testing:** Chose test cases for functions anticipated to handle high transaction volumes to assess scalability and optimize gas usage accordingly
- **Modularization:** Modularized code into separate contracts and functions to promote code reuse and maintainability
- **Standardized Documentation:** Ensured standardized documentation for contracts, functions, and modifiers using appropriate comments and NatSpec format

Final Overview



Product Demo !

Conclusion

- In conclusion, our participatory budgeting system offers a robust and efficient solution for democratic allocation of funds
- By leveraging gas optimization techniques and scalability testing, we ensure that our system can handle high transaction
- Clean and readable code, following Solidity style guides and conventions, ensures maintainability and ease of understanding for developers
- Our frontend proposal interface complements the backend functionality with its simplicity, allowing users to easily propose projects without any technical barriers