

# Blockchain & Solidity

## White Paper: SimpleVote

### Projekt:

Team: Gruppe 3

**Jonas Buse, Felix Pollok, Michael Meissner, Emir Simsek**

## PROJEKT NAME: SIMPLEVOTE

Ein Projekt, dass kleinere Gruppen/Organisationen ein transparentes, sicheres Wählen bei einer Online-Abstimmung ermöglicht. Hierbei können die Ersteller der Abstimmung in einem privaten (private) oder Öffentlichen (public) Raum für jeden über das Internet per Einladungsverfahren durch eine Adressierung nutzbar machen.

### Was ist das Problem?

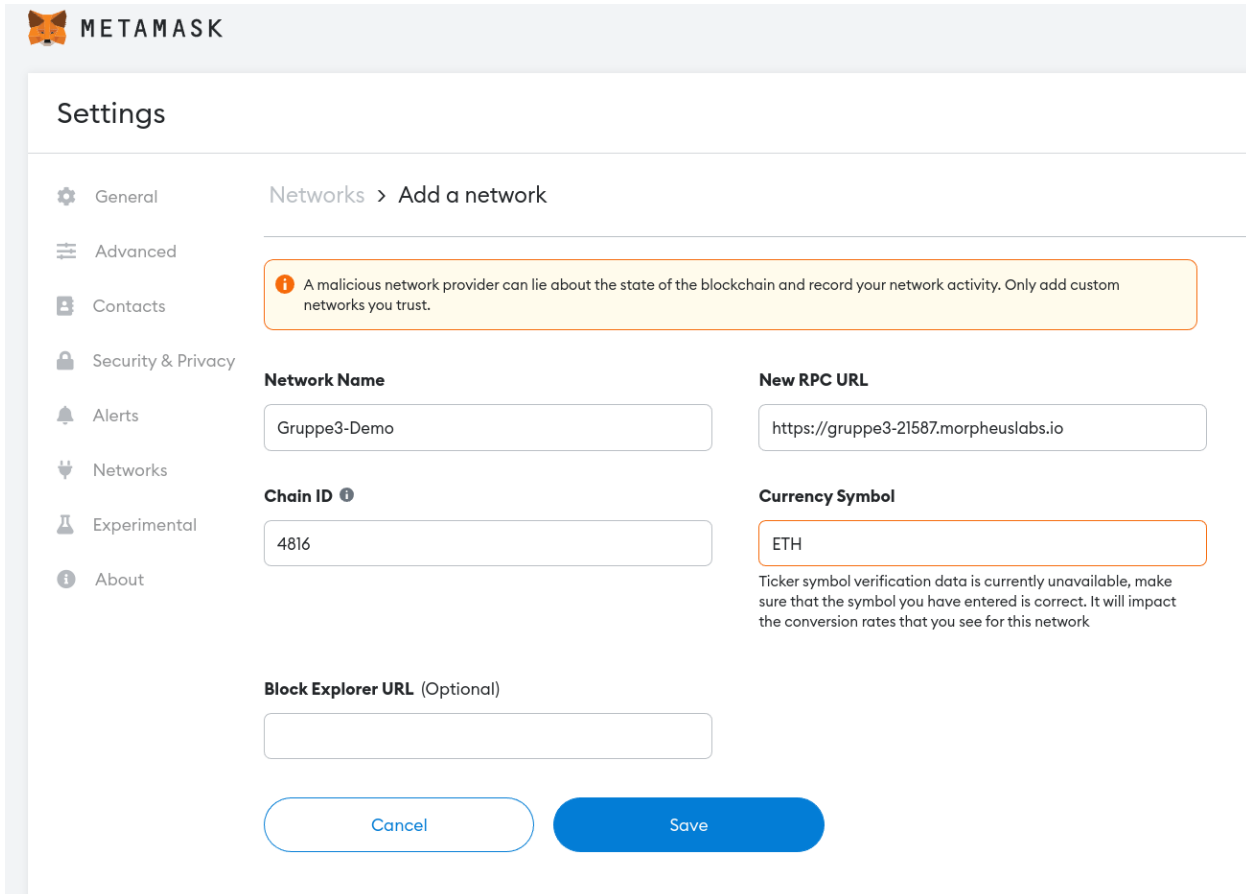
Klassische Wahlen bzw. Abstimmungen sind meist mit hohem Ressourcenverbrauch (Papier, Druck, Versand) verbunden. Je nach Größe der Abstimmung und der Teilnehmer steigt nicht nur der Ressourcenverbrauch, sondern auch der Aufwand. Bei klassischen Abstimmungen sind zum einen oftmals eine physische Anwesenheit und eine zeitintensive Auswertung erforderlich. Da eine Manipulierung in Form von Identitätsklau oder auch Verlust von Wahlstimmen bei papierbasierten Wahlen sehr einfach ist, kann die Sicherheit ebenfalls ein Problem darstellen.

### Was ist die Lösung?

Ist ein Blockchain basiertes Wahlsystem mit einem eigenen Wahlraum („Voteroom“). Jeder Teilnehmer kann dabei nur einmal abstimmen. Wenn der Manager einen Wahlraum erstellt, kann er selbst entscheiden, wen er in diesen Raum hinzufügen möchte. In diesem Raum kann der Ersteller Abstimmungen kreieren, bei denen er die Mindestanzahl der Stimmenabgaben entscheiden kann.

## Vorbereitung zur Nutzung der Applikation:

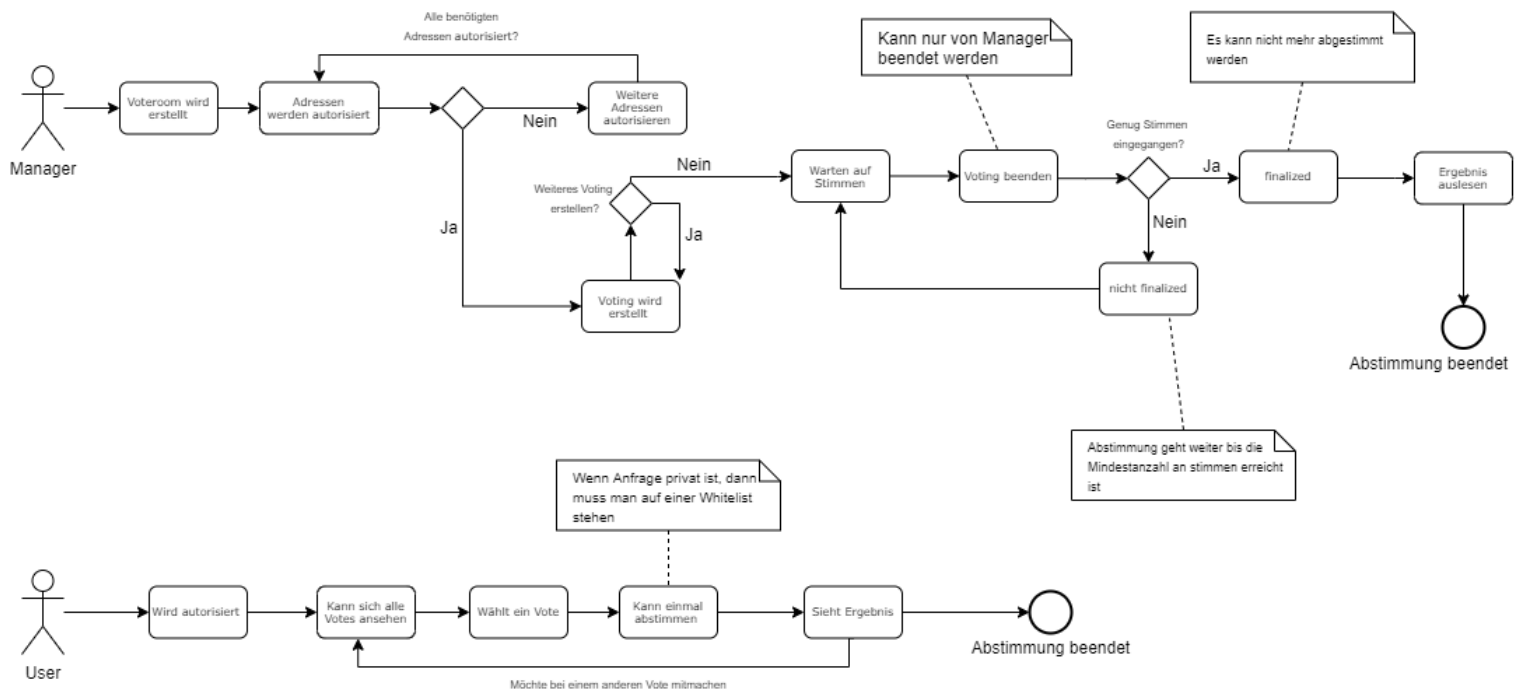
Um die Applikation testen zu können muss das Wallet mit dem Blockchain-Netzwerk verbunden werden. Die Daten können aus der Abbildung entnommen werden oder alternativ aus dem darunter stehenden Text entnommen werden.



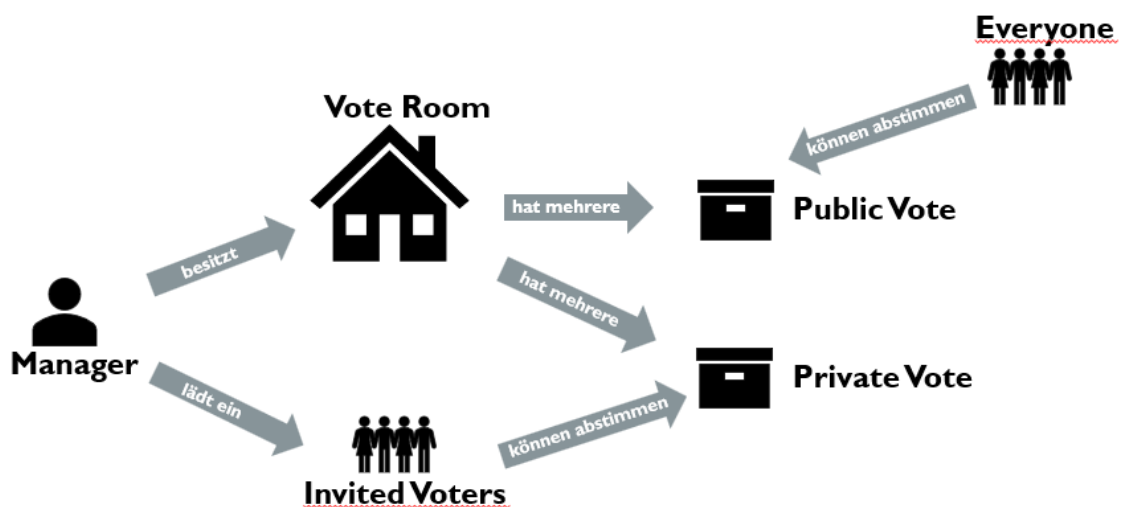
**Name:** Gruppe3-Demo (Name ist egal)  
**Chain ID:** 4816  
**RPC URL:** https://gruppe3-21587.morpheuslabs.io/  
**Private Keys:**  
    **Main:** 774112a126f3fb72456e4887a9c816cd6fd3ffde83ed55cd96fe7da7f4a5d40c  
    **Alt:** f61c08d32986f821ea806fdf860395ca00f3f6bada0969634ec4f5468d246373

## Wie wird es funktionieren?

Der Manager erstellt einen Voteroom. Der Raum wird mit Adressen autorisiert. Sobald alle Adressen autorisiert sind, kann er eine Abstimmung („Voting“) erstellen. Je nachdem ob dieser noch weitere Wahlen erstellen will, kann er dies machen. Nach der gesammelten Mindestanzahl von Stimmen, kann das Voting beendet und anschließend „finalized“ werden.



Der User wird zunächst vom Ersteller autorisiert. Dann kann er sich alle Votings anschauen und Wählen. Der User kann nur auf für ihn berechtigte Abstimmungen teilnehmen und jeweils nur eine Stimme abgeben. Nach dem Abstimmen sieht er das aktuelle Ergebnis.



## ELEMENTE DER APP

## FILES:

Wichtige Files und ihre Bedeutung:

```
SimpleVote/
- components/           // UI elements
- pages/                 // Next/React pages
- routes.js              // Route definition
- server.js              // Main starting point
- ethereum/
  - build/               // Build artifacts
  - contracts/           // Ethereum Contracts
    - VoteRoom.sol
    - RoomCreator.sol

- test/                  // Smart contract tests
- factory.js             // Factory for backend
- truffle.js             // Truffle config
- web3.js                // Web3 connector
```

## TEST:

Die Tests sind im Testordner unter *ethereum/test* und beinhalten 21 Tests für die Smart Contracts. Zum Ausführen *truffle develop* und *truffle test* in Kommandozeile eingeben

## BACKEND (SMART CONTRACT):

RoomCreator.sol:

```
/**
 * @title Vote room Creator smart contract, create vote rooms and manage them
 * @author Group 3 (Jonas Buse, Emir Simsek, Felix Pollok, Michael Meissner)
 * @notice A smart contract that anyone create their own vote rooms and manage them
 */
contract RoomCreator {
  /// List of all vote rooms
  address[] public voteRoomAddresses;

  /// create a new vote room
  function createRoom( description, invitedVoters) public {...}

  /// get all vote rooms
  function getAllVoteRooms() public view returns (address[]) {...}
```

```
}
```

## VoteRoom.sol:

```
/// @title Vote room smart contract, a proof-of-concept voting system with private and public
voting
/// @author Group 3 (Jonas Buse, Emir Simsek, Felix Pollok, Michael Meissner)
/// @notice A smart contract that lets managers create new votes which either designated
adresses can partake in, or which everybody can use
contract VoteRoom {
    /// A single vote which can be public or private
    struct VoteData {...}

    /// Owner and manager who can whitelist people and start votes
    address public manager;

    /// Mapping of all whitelisted voters
    mapping(address => bool) public invitedVoters;

    /// Number of whitelisted voters
    uint256 public voterCount;

    /// Array of votes
    VoteData[] public votes;

    /// Description of the room
    string public voteRoomDescription;

    /// Only the manager can access
    modifier managerGuard() {...}

    /// Only a whitelisted voter can access if the vote is not public on a non finalized vote only
    once
    modifier voterGuard(uint256 voteld) {...}

    /// Only allow if more than the minimum amount of votes are cast
    modifier minimumVotesGuard(uint256 voteld) {...}

    /**
     * @dev constructor that already takes new voters
     * @param author manager of the new vote room
     * @param description description for the whole room
     * @param newVoters a list of addresses of whitelisted voters
     */
    constructor(
        address author,
```

```

    string memory description,
    address[] memory newVoters
) public {...}

```

```

/**
 * @dev Invite new voters after creation
 * @param newVoters array of addresses of newly whitelisted voters
 */
function inviteVoters(address[] memory newVoters) public managerGuard {...}

```

```

/**
 * @dev The manager can create a new Vote with all possible inputs
 * @param description of the singular vote
 * @param minimumVotes the number of the minimum amount of votes
 * @param isPublic boolean if this vote should be restricted to the whitelist of available to the
public
 */
function createVote(
    string memory description,
    uint256 minimumVotes,
    bool isPublic
) public managerGuard returns (uint256 voteId) {...}

```

```

/**
 * @dev The manager can create a new Vote with minimVotes of 0
 * @param description of the singular vote
 * @param isPublic boolean if this vote should be restricted to the whitelist of available to the
public
 */
function createVote(string memory description, bool isPublic)
    public
    managerGuard
    returns (uint256 voteId)
{...}

```

```

/**
 * @param voteId the id of the vote to be voted in favor
 */
function voteInFavor(uint256 voteId) public voterGuard(voteId) {...}

```

```

/**
 * @param voteId the id of the vote to be voted against
 */
function voteAgainst(uint256 voteId) public voterGuard(voteId) {...}

```

```
/**
 * @param voteId the id of the vote to be voted abstain
 */
function voteAbstain(uint256 voteId) public voterGuard(voteId) {...}

/**
 * @dev The manager can finalize a vote
 * @param voteId the id of the vote to be finalized
 */
function finalizeVote(uint256 voteId)
    public
    managerGuard
    minimumVotesGuard(voteId)
    {...}

/**
 * @dev get result of a vote
 * @param voteId the id of the vote to be shown
 */
function getResult(uint256 voteId)
    public
    view
    returns (
        uint256 inFavor,
        uint256 against,
        uint256 abstain,
        bool isFinalized
    )
    {...}

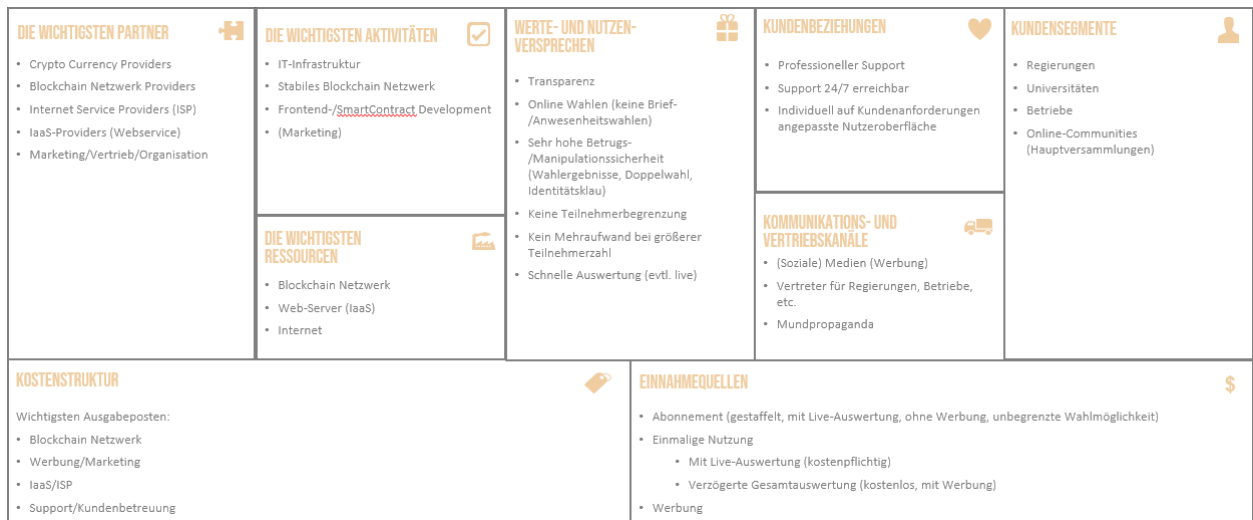
/**
 * @dev get number of votes on a specific vote
 * @param voteId the id of the vote to be shown
 */
function getVoteCount(uint256 voteId) public view returns (uint256 count) {...}

/**
 * @dev get the number of votes
 */
function getNumberOfVotes() public view returns (uint256 count) {...}
}
```

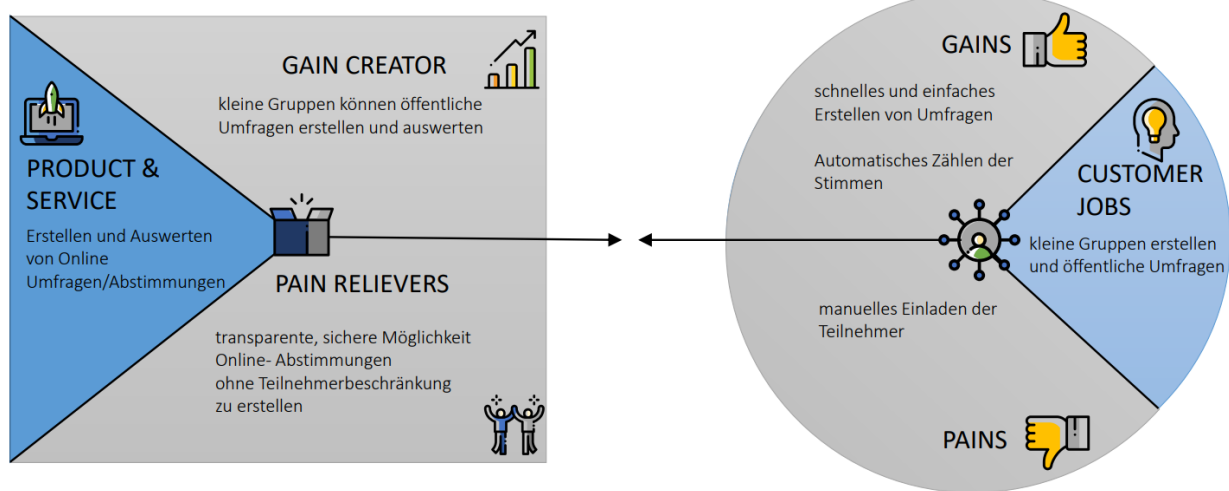


## OPPORTUNITY & VALUE PROPOSITION

### Business Model Canvas:



### Value Proposition:



### DAS TEAM

Jonas Buse:	Software Developer
Felix Pollok:	Web-Designer (Front End)
Michael Meißner:	Modelling
Emir Simsek:	Ressourcemanager

Teilweise wurden Aufgaben von mehreren Gruppenmitglieder bearbeitet.

## ANNEX

- Git Repository: <https://github.com/Pyxels/SimpleVote>
- Code siehe Zip-Datei (kein Zugriff auf die Organisation „HM2022-BC“ in GitHub)

