

# Rendu du devoir pratique – Projet Blockchain : Système de vote

Lien vers le repo GitHub : <https://github.com/bbombardella/smart-contract-voting>

## Composition de l'équipe :

- Bastien BOMBARDELLA (Estiam E5 WMD Lyon)
- Bartholomé GILI (Estiam E5 WMD Lyon)
- Noé FAVIER (Estiam E5 WMD Lyon)
- Kilian VEST (Estiam E5 WMD Lyon)
- Florian HAVARD (Estiam E5 WMD Lyon)

## Résumé expliquant qui a fait quoi dans l'équipe

### Bartholomé GILI

- Mise en place du smart contract principal en Solidity.
- Implémentation de la logique de gestion des candidats (ajout, stockage, accès).
- Vérification de la bonne structuration des fonctions d'administration.

### Noé FAVIER

Développement de la partie sécurité du vote :

- Gestion de l'autorisation des électeurs.
- Vérification qu'un électeur ne peut voter qu'une seule fois.
- Réalisation des tests d'intégrité des votes (tentatives de double vote, vote non autorisé).

### Kilian VEST

- Conception et gestion des différents états du vote (ouverture, fermeture).
- Ajout des restrictions pour empêcher certaines actions au mauvais moment (ex. pas d'ajout de candidat après ouverture du vote).
- Contribution aux tests de gestion des états.

### Bastien BOMBARDELLA

- Développement des fonctions de consultation publique :
  - `getResults()`

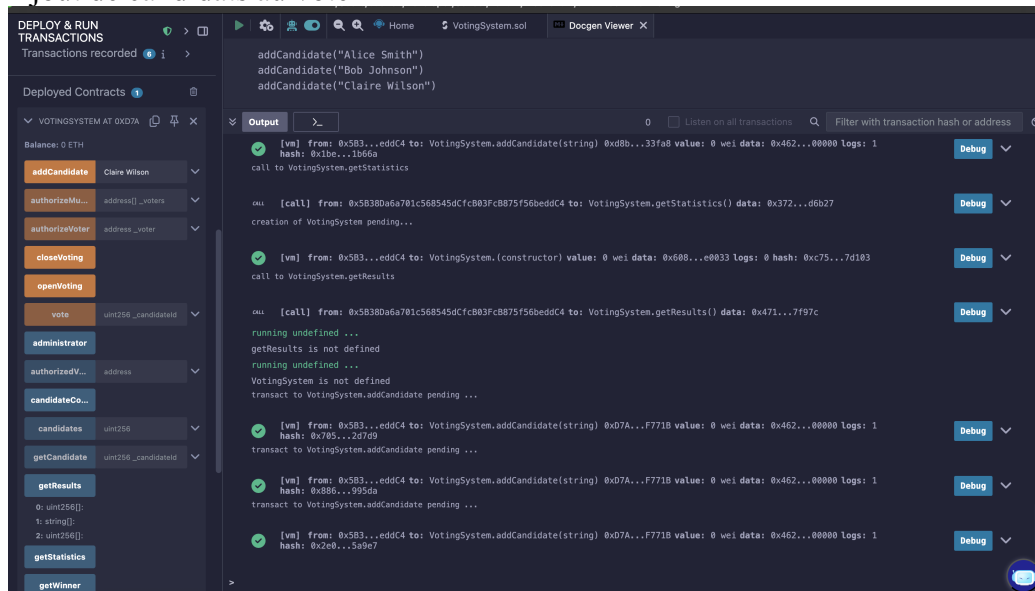
- `getWinner()`
- `getStatistics()`
- Mise en place des tests fonctionnels pour vérifier l'exactitude des résultats et l'identification correcte du gagnant.

## Florian HAVARD

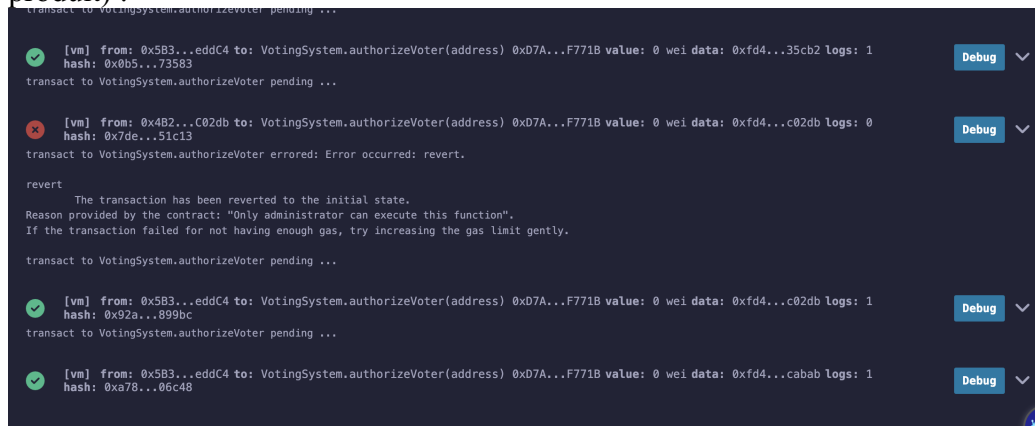
- Rédaction du guide d'utilisation (README) avec scénarios de test détaillés.
- Organisation des cas de simulation (ajout de candidats, votes simulés, résultats attendus).
- Coordination générale et validation finale du bon fonctionnement du projet.

## Captures d'écran de l'exécution du smart contract

### 1. Ajout de candidats au vote



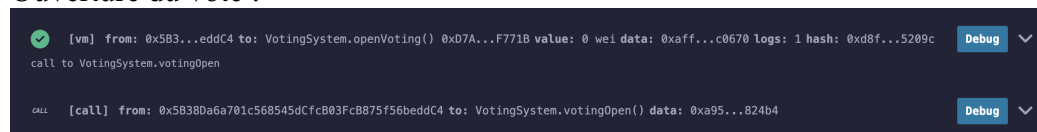
### 2. Ajout des votants éligibles (seulement par le rôle admin, sinon une erreur se produit) :



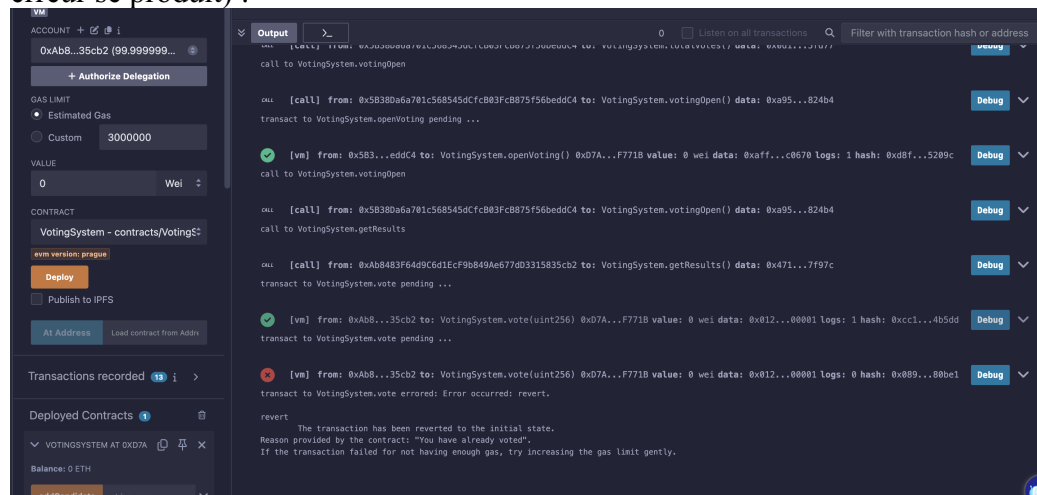
### 3. Récupération des stats du vote :



### 4. Ouverture du vote :



### 5. Vote via différents comptes (possible de voter qu'une seule fois, sinon une erreur se produit) :



## 6. Récupération du gagnant :

The screenshot displays a blockchain explorer interface with a dark theme. On the left, a sidebar lists functions: `getWinner`, `hasAlreadyVoted`, `hasVoted`, `isAuthorized`, `totalVotes`, and `votingOpen`. The `getWinner` function is selected, showing its parameters: `0: uint256: winnerId_1` and `1: string: winnerName_Bob Johnson`. Below this, the `totalVotes` function is shown with parameters `0: uint256: 0` and `1: bool: true`. The main panel displays a transaction `transact to VotingSystem.vote pending ...` with a green checkmark and a `Debug` button. Below the transaction, a list of function calls is shown, each with a green checkmark and a `Debug` button. The calls are: `[vm] from: 0x787...caba8 to: VotingSystem.vote(uint256) 0x07A...F771B value: 0 wei data: 0x012...00000 logs: 1 hash: 0xc4f...5e9f0`, `[vm] from: 0x583...eddC4 to: VotingSystem.closeVoting() 0x07A...F771B value: 0 wei data: 0xc63...1b292 logs: 1 hash: 0x110...425a2`, `[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: VotingSystem.getStatistics() data: 0x372...d6b27`, `[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: VotingSystem.getResults() data: 0x471...7f97c`, and `[call] from: 0x58380a6a701c568545dCfcB03FcB875f56beddC4 to: VotingSystem.getWinner() data: 0x8e7...ea5b2`. The bottom of the sidebar shows `Low level interactions` with a `CALLDATA` tab.