Ecole des sciences de l'information

Projet Sécurté

RAPPORT : SYSTÉME DE VOTE DÉCENTRALISÉE

Réalisé par :	**Numéro**
Moad GHALMI	19
Hamza BOUALI	6
Reda ELKATE	13
Kamal SAALAOUI	31
Mouad ENNASIRY	17

Professeur :

Mme.Asmaâ HILMI

Abstract

Le vote décentralisé basé sur la blockchain Ethereum représente une approche moderne, sécurisée et résiliente du vote en ligne. Contrairement aux systèmes traditionnels qui dépendent d'autorités centrales ou d'intermédiaires pour gérer le processus électoral, cette solution repose sur un réseau distribué dans lequel chaque transaction de vote est enregistrée de manière permanente et inviolable.

Cette application décentralisée permet aux électeurs d'exprimer leur choix directement via la blockchain, sans intermédiaires, tout en assurant la transparence du dépouillement et l'intégrité des résultats. Grâce à l'utilisation de contrats intelligents, le processus de vote est automatisé, ce qui réduit considérablement les erreurs humaines, les fraudes et les tentatives de manipulation.
Chaque vote est horodaté et chiffré, garantissant ainsi non seulement l'anonymat du votant, mais aussi l'impossibilité de modifier un vote une fois enregistré. Les électeurs peuvent consulter en temps réel l'évolution du scrutin, ce qui renforce la confiance dans le système et favorise la participation citoyenne.
En outre, cette technologie permet de réaliser des élections à faible coût, ouvertes à une population géographiquement dispersée, et facilement auditées. Le présent rapport examine en détail les fondements technologiques, les choix de conception, les résultats expérimentaux, ainsi que les avantages et les limites d'un tel système, tout en proposant des pistes d'amélioration futures pour une adoption à plus grande échelle.

Table des matières

Table des figures

Liste des abréviations

Abréviation	**Signification**
DApp	Decentralized Application (Application décentralisée)
CIN	Carte d'Identité Nationale
OSCE	Organisation pour la Sécurité et la Coopération en Europe
zk-SNARKs	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge
UML	Unified Modeling Language (Langage de modélisation unifié)
DDoS	Distributed Denial of Service
JS	JavaScript

Table 1 – Table des principales abréviations utilisées dans ce rapport

1 Introduction

1.1 Introduction générale

1.1.1 Le vote électoral : enjeux et limites

Le vote constitue un mécanisme essentiel de représentation citoyenne. Toutefois, il est souvent remis en cause pour sa complexité logistique, son coût et sa vulnérabilité. Des études telles que celles de Alvarez et Hall [1] mettent en lumière les limites des systèmes électoraux classiques, notamment en matière de transparence et d'accessibilité. Dans de nombreuses démocraties, des cas de bourrages d'urnes, de manipulation de résultats, et de contestations post-scrutin sont documentés [2].

En outre, l'accessibilité au vote est entravée pour les électeurs éloignés géographiquement, les personnes en situation de handicap ou encore les expatriés. L'Organisation pour la sécurité et la coopération en Europe (OSCE) [3] souligne régulièrement les déficits d'accessibilité observés dans les processus électoraux des États membres. Ces lacunes structurelles contribuent à la perte de confiance des citoyens envers le système électoral.

1.1.2 La blockchain : une technologie de rupture

La blockchain est une technologie de registre distribué apparue avec le Bitcoin [4]. Elle permet d'enregistrer des transactions de manière sécurisée, transparente et immuable, sans organe central de contrôle. Dans une revue de littérature récente, Zheng et al. [5] décrivent les caractéristiques fondamentales de cette technologie : décentralisation, consensus distribué, transparence, immutabilité et auditabilité.

Son application au vote électronique a été explorée dans plusieurs travaux, notamment ceux de Noizat [6] et de Hardwick et al. [7], qui montrent comment les contrats intelligents (smart contracts) peuvent automatiser et sécuriser le processus électoral. Les électeurs peuvent ainsi voter via Internet, leurs choix étant enregistrés dans des blocs validés par consensus, ce qui réduit considérablement les risques de manipulation.

Cette infrastructure technologique offre une alternative prometteuse aux modèles

centralisés actuels, en garantissant l'intégrité des données, la vérifiabilité du processus et l'inviolabilité des votes. Elle s'impose dès lors comme un socle technique pour refonder la confiance dans les systèmes démocratiques.

1.2 Étude de l'existant

1.2.1 Systèmes traditionnels

Le modèle le plus courant reste le vote papier, accompagné d'une logistique lourde (bureaux de vote, urnes, dépouillement manuel). Ce système est jugé fiable dans certaines démocraties stables, mais il reste vulnérable à :

— la falsification des bulletins,

— le bourrage d'urnes,

— l'erreur humaine lors du comptage,

— les difficultés d'accès pour certaines populations (handicap, éloignement, expatriation).

Parallèlement, les solutions de vote électronique centralisé ont vu le jour, notamment sous forme de machines de vote ou de portails web. Toutefois, leur centralisation implique des risques accrus :

— attaques informatiques (DDoS, intrusions, altérations de base de données),

— manque de transparence du code ou du processus,

— dépendance à un prestataire technique ou à une autorité centrale.

1.2.2 Tentatives de modernisation

Certaines initiatives ont tenté d'améliorer la transparence par l'utilisation de mécanismes de chiffrement, de vérification par code QR ou de doubles bulletins. Des rapports de l'OSCE [3] notent que, malgré les efforts d'innovation, ces solutions n'atteignent pas toujours les critères de vérifiabilité universelle ou de neutralité technologique.

1.2.3 Enjeux observés

À travers l'observation des systèmes existants, plusieurs besoins émergent :

— garantir la sécurité du vote et l'unicité de chaque voix,

— protéger l'anonymat tout en assurant la traçabilité globale,

— permettre un accès élargi au vote,

— restaurer la confiance citoyenne par la transparence.

1.2.4 Problématique

À moins d'un an des prochaines élections gouvernementales au Maroc, le pays se prépare à élire un nouveau gouvernement appelé à jouer un rôle stratégique dans l'organisation de la Coupe du Monde 2030, un événement d'envergure mondiale que le Maroc coorganisera. Ce "gouvernement du Mondial" portera la responsabilité de guider le pays à travers des défis économiques, sociaux et institutionnels majeurs, tout en assurant la crédibilité du processus démocratique. Or, les systèmes de vote traditionnels présentent encore aujourd'hui des faiblesses notables en matière de confidentialité, de transparence et de traçabilité, pouvant compromettre la confiance des citoyens dans les résultats électoraux. Dans ce contexte sensible, il devient impératif de repenser le mécanisme de vote afin de garantir des élections libres, sécurisées et incontestables. L'adoption d'un système de vote électronique basé sur la technologie blockchain représente une piste innovante et crédible pour renforcer l'intégrité du processus électoral, tout en assurant l'anonymat des électeurs et la transparence des résultats.

1.2.5 Objectifs

— Garantir la sécurité et la transparence du processus électoral

— Assurer l'anonymat et l'unicité du vote

— Permettre une accessibilité universelle via Internet

— Réduire les coûts d'organisation des élections

Conclusion

Au regard des enjeux démocratiques contemporains et des limites persistantes des systèmes électoraux traditionnels, il devient indispensable d'adopter des solutions innovantes à la hauteur des attentes citoyennes. L'évolution des technologies numériques, et en particulier l'émergence de la blockchain, offre une opportunité inédite de repenser le vote en ligne autour des principes de transparence, d'équité, de sécurité et d'accessibilité.

Les propriétés fondamentales de cette technologie — telles que l'immuabilité, la décentralisation et la vérifiabilité — répondent directement aux failles structurelles identifiées dans l'analyse de l'existant. Ainsi, le recours à une infrastructure de type blockchain s'inscrit comme une voie sérieuse pour garantir l'intégrité des processus électoraux et renforcer la confiance des citoyens dans leurs institutions démocratiques.

C'est dans cette optique que ce rapport propose une étude approfondie des approches existantes, une conception technique d'un système de vote fondé sur

la blockchain Ethereum, ainsi qu'une évaluation de ses avantages, limites et perspectives d'évolution.

2 État de l'art

Introduction

À travers ce chapitre, nous présentons une revue des travaux scientifiques, techniques et expérimentaux traitant de l'usage de la blockchain dans le cadre du vote électronique. L'objectif est d'identifier les approches existantes, les solutions proposées, ainsi que les limites rencontrées, afin d'orienter la conception de notre propre système.

Les publications analysées couvrent principalement la période 2015–2023 et ont été sélectionnées sur la base de leur pertinence, de leur originalité et de leur reconnaissance dans la communauté académique. Elles traitent des thématiques suivantes : sécurité du vote, anonymat, vérifiabilité, scalabilité, et implémentation de contrats intelligents.

Ce chapitre est structuré comme suit : nous abordons d'abord les architectures générales des systèmes de vote basés sur blockchain, avant d'examiner en détail les aspects liés à la sécurité et à l'anonymat, puis les cas d'usage existants, et enfin les principales limites observées dans la littérature.

2.1 Panorama des systèmes de vote blockchain

L'étude de Jafar et al. [8] propose une analyse systématique de 76 articles publiés entre 2017 et 2022 sur les systèmes de vote basés sur la blockchain. Les auteurs classent les solutions selon plusieurs critères : le niveau de vérifiabilité, la scalabilité, le type de blockchain (publique ou privée), et la compatibilité avec l'anonymat. Cette revue met en avant le manque de standardisation des modèles de vote décentralisé, malgré des résultats prometteurs en termes de sécurité.

2.2 Sécurité, vérifiabilité et confidentialité

Hardwick et al. [7] proposent un protocole de vote électronique sécurisé utilisant Ethereum, garantissant l'intégrité du processus sans compromettre l'anonymat des votants. Le système repose sur des smart contracts, avec des mécanismes de chiffrement asymétrique et des signatures numériques. De son côté, Noizat [6]

explore le potentiel de la blockchain pour assurer la transparence sans nécessiter de vérification centralisée, via la vérifiabilité individuelle et universelle.

2.3 Cas d'usage et implémentations concrètes

Plusieurs prototypes fonctionnels ont été développés :

— L'application *FollowMyVote* (États-Unis), reposant sur la transparence et la vérifiabilité publique.

— Le projet de vote municipal de *Zug* (Suisse) utilisant une blockchain privée.

— Le système de vote en ligne *i-Voting* en Estonie, souvent cité pour ses garanties d'authentification, bien qu'il ne repose pas sur une blockchain.

2.4 Limites identifiées

Zheng et al. [5] relèvent que les blockchains publiques souffrent de problèmes de scalabilité, de consommation énergétique et de latence, ce qui constitue un frein pour des scrutins à grande échelle. Par ailleurs, la protection de la vie privée demeure un défi majeur. Plusieurs auteurs suggèrent l'utilisation de preuves à divulgation nulle de connaissance (zk-SNARKs) ou de mécanismes de tokens anonymes pour protéger l'identité des votants tout en garantissant la vérifiabilité.

2.5 Synthèse

L'état de l'art révèle un intérêt croissant pour l'usage de la blockchain dans le cadre électoral. Les avantages perçus — transparence, sécurité, résilience — sont contrebalancés par des défis techniques encore ouverts : anonymat, performance, compatibilité avec les lois électorales. Ces éléments motivent la conception de solutions hybrides combinant blockchain, cryptographie avancée et identité numérique.

3 Conception et analyse des besoins

Introduction

La phase de conception a pour but de traduire les besoins identifiés en une architecture fonctionnelle du système de vote. Cette étape détermine à la fois les fonctionnalités du système et les contraintes techniques et non techniques qu'il devra respecter. Elle s'appuie sur des outils de modélisation pour clarifier les interactions entre les utilisateurs et le système, et pour prévoir son comportement global.

3.1 Besoins fonctionnels

Les besoins fonctionnels définissent les actions que le système doit permettre aux utilisateurs de réaliser :

— Authentification sécurisée des électeurs

— Consultation de la liste des candidats

— Vote unique par électeur

— Dépouillement automatique et affichage en temps réel des résultats

— Accès administrateur pour gérer les élections et les utilisateurs

3.2 Besoins non fonctionnels

Ces besoins expriment les critères de qualité du système :

— **Sécurité** : Protection contre la falsification et l'intrusion

— **Disponibilité** : Accessibilité du système à tout moment

— **Scalabilité** : Capacité à supporter un grand nombre de votants

— **Anonymat** : Séparation entre identité du votant et contenu de son vote

— **Fiabilité** : Résistance aux pannes et garanties de dépouillement correct

3.3 Modélisation UML

Pour représenter l'architecture fonctionnelle, plusieurs diagrammes UML ont été élaborés :

— **Diagramme de cas d'utilisation** : Le diagramme de cas d'utilisation présente les interactions principales entre les deux types d'acteurs du système : **Admin** et **Voter**. L'administrateur dispose de fonctions critiques telles que l'ajout de candidats, le lancement et la clôture du scrutin, ou encore la validation des inscriptions. L'électeur, quant à lui, interagit avec le système pour demander son inscription, voter et consulter les résultats.

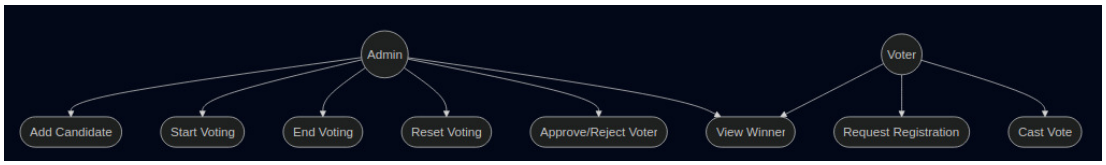

FIGURE 3.1 – Diagramme de cas d'utilisation

— **Diagramme de classes** : Le diagramme de classes structure les éléments fondamentaux du contrat intelligent Ethereum. On retrouve la classe `VotingSystem` contenant à la fois les données globales (états du scrutin, listes de candidats et électeurs) et les méthodes de gestion du vote. Les entités `Candidate` et `Voter` sont modélisées comme des classes simples, reflétant les structures utilisées dans le smart contract.

Les méthodes exposées (`addCandidate`, `vote`, `getWinner`, etc.) couvrent toutes les opérations attendues d'un processus électoral sécurisé. L'usage des mappings représente bien la logique Solidity. Ce diagramme sert de référence pour valider la couverture fonctionnelle de l'implémentation.

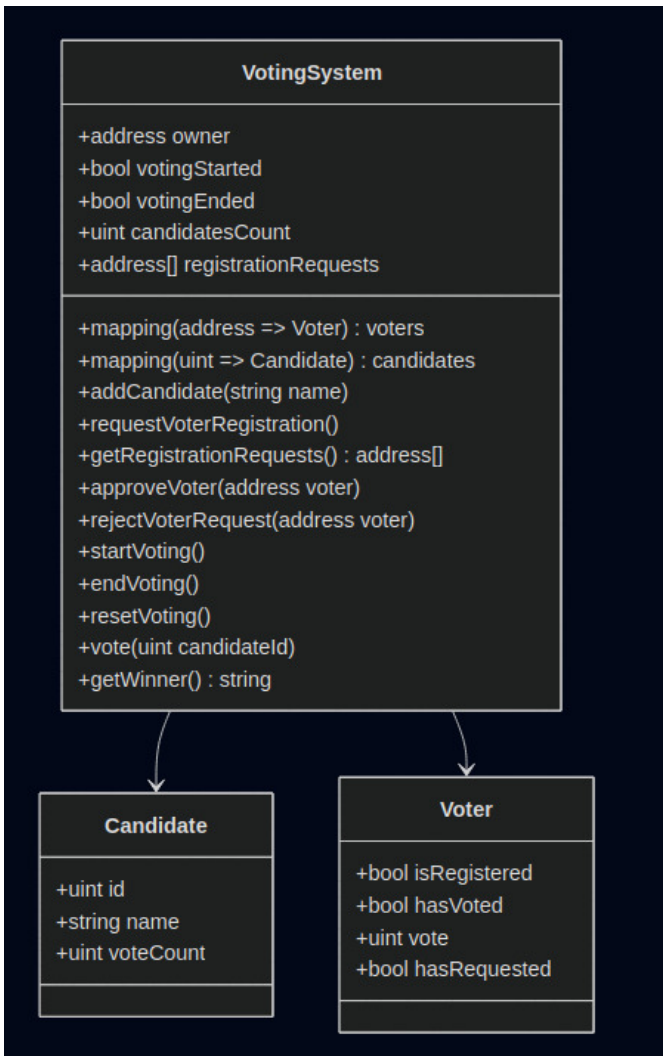

FIGURE 3.2 – Diagramme de classes

— **Diagramme de séquence** : Ce diagramme de séquence modélise le déroulement temporel des échanges entre les entités principales : **Voter**, **DApp**, **Smart Contract** et **Admin**. Il met en évidence le rôle central du contrat intelligent dans le traitement des opérations (enregistrement, vote, dépouillement).

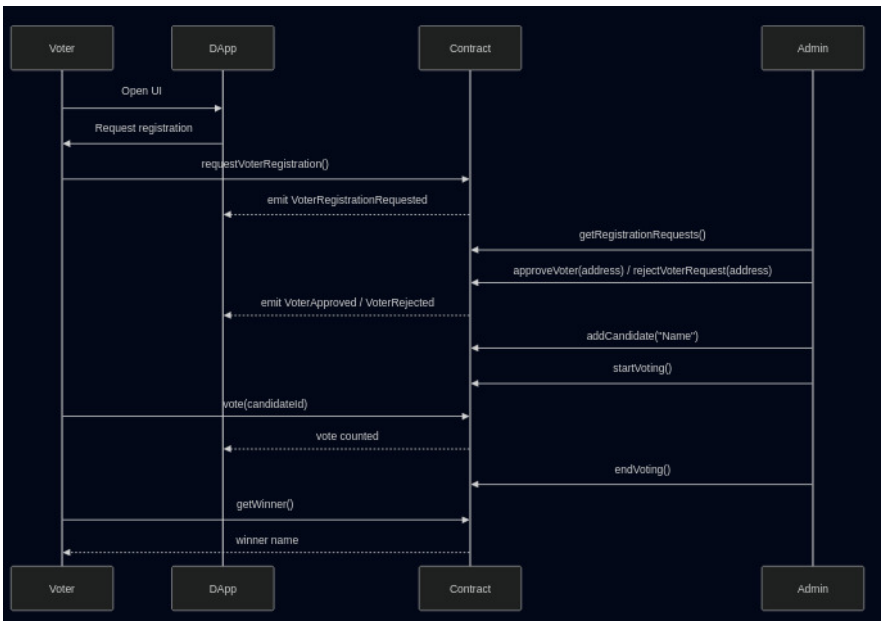

FIGURE 3.3 – Diagramme de séquence

— **Diagramme d'activités** : Le diagramme de cas d'utilisation présente les interactions principales entre les deux types d'acteurs du système : **Admin** et **Voter**. L'administrateur dispose de fonctions critiques telles que l'ajout de candidats, le lancement et la clôture du scrutin, ou encore la validation des inscriptions. L'électeur, quant à lui, interagit avec le système pour demander son inscription, voter et consulter les résultats.

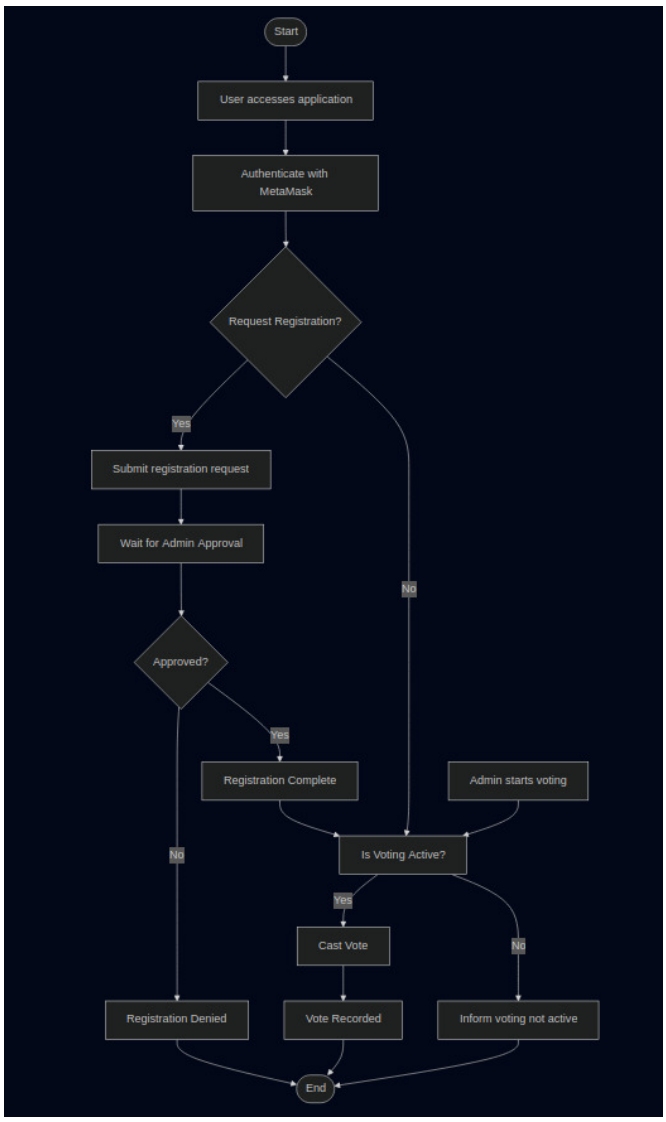

FIGURE 3.4 – Diagramme d'activités

Conclusion

Cette phase de conception permet de cadrer techniquement le projet. Elle traduit les exigences en éléments concrets et prépare la phase de développement en garantissant une vision claire, structurée et vérifiable de l'architecture du système.

4 Méthodologie de travail et conception technique

Introduction

Ce chapitre détaille l'approche méthodologique adoptée pour concevoir, développer et valider le système de vote décentralisé. Il décrit également les choix technologiques, les outils utilisés, les algorithmes clés et l'organisation du projet selon un calendrier structuré.

4.1 Méthodologie adoptée

Le développement du système de vote décentralisé a suivi une approche itérative et incrémentale, inspirée du cycle de vie Agile. Cette méthodologie a été choisie pour sa flexibilité et sa capacité à intégrer progressivement les retours des utilisateurs, tout en garantissant une livraison continue de composants fonctionnels.

4.1.1 Phase 1 : Analyse des besoins

Cette phase a consisté à identifier les exigences fonctionnelles et non fonctionnelles du système à travers :

— l'étude comparative des systèmes de vote existants,

— l'analyse des profils utilisateurs (électeurs, administrateurs),

— la définition des contraintes techniques (blockchain, sécurité, anonymat, intégrité des données).

4.1.2 Phase 2 : Conception UML et architecture

À partir des besoins identifiés, une modélisation UML a été réalisée afin de :

— décrire les cas d'utilisation du système (diagramme de cas d'utilisation),

— structurer les entités et leurs relations (diagramme de classes),

— modéliser les interactions entre composants (diagramme de séquence),

— formaliser le déroulement du processus électoral (diagramme d'activités).

En parallèle, une architecture technique modulaire a été conçue pour intégrer de manière cohérente les composants Solidity, Web3.js .

4.1.3 Phase 3 : Développement technique

Cette étape s'est articulée autour de trois sous-projets principaux :

— le développement du smart contract en Solidity, assurant les règles de vote, l'unicité, et la transparence,

— la création d'une interface web (DApp) interopérable avec MetaMask via Web3.js pour permettre l'interaction utilisateur.

Chaque composant a été développé et testé indépendamment avant l'intégration.

4.1.4 Phase 4 : Intégration, test et validation

Les différents modules ont été intégrés dans un environnement de test simulé avec Ganache, puis testés sur une testnet publique Ethereum. La validation a porté sur :

— le bon déroulement des opérations de vote (authentification, soumission du vote, dépouillement),

— le respect des exigences de sécurité, de confidentialité et d'intégrité,

— la robustesse du système face à des erreurs d'entrée ou des comportements malveillants.

4.1.5 Synthèse

Cette approche incrémentale a permis d'assurer un alignement progressif entre les objectifs fonctionnels et les contraintes techniques du projet. Elle a favorisé une mise en œuvre réaliste, testable et évolutive du système de vote décentralisé, tout en intégrant les bonnes pratiques du développement logiciel sécurisé.

4.2 Outils et technologies utilisés

Les outils suivants ont été sélectionnés en raison de leurs avantages techniques et de leur pertinence pour la mise en œuvre d'un système de vote sécurisé, transparent et décentralisé :

— **Solidity** 

Langage de programmation standard pour le développement de smart contracts sur la blockchain Ethereum. Il permet d'implémenter la logique métier

du vote (enregistrement, vote unique, calcul des résultats) de manière immuable, auditable et transparente.

— **Truffle et Ganache**


Truffle est un environnement complet pour le développement, le test et le déploiement de smart contracts. Ganache simule une blockchain Ethereum en local, ce qui facilite les tests rapides sans frais de transaction ni dépendance à un réseau externe.

— **Web3.js**

Bibliothèque JavaScript utilisée pour connecter l'interface web (DApp) au smart contract Ethereum. Elle permet d'envoyer des transactions, de lire l'état du contrat et d'interagir avec les portefeuilles tels que Metamask.

— **Metamask**

Portefeuille Ethereum sous forme d'extension navigateur. Il permet aux électeurs de s'authentifier et de signer leurs transactions (votes) de manière sécurisée, sans exposer leurs clés privées. Sa compatibilité native avec Web3.js en fait un outil central dans l'écosystème des applications décentralisées.

4.3 Description des Algorithmes

Le contrat `Voting` permet d'organiser une élection simple sur la blockchain Ethereum. Il comporte plusieurs fonctions clés que nous détaillons ci-dessous :

1. Constructeur `Voting`

Le constructeur initialise le contrat avec une liste de candidats transmise lors du déploiement. L'adresse de la personne déployant le contrat devient automatiquement l'administrateur , (`admin`). (Voir Annexe)

2. Fonction `vote`

Cette fonction permet à un utilisateur de voter pour un candidat. Avant d'enregistrer un vote, deux vérifications sont effectuées :

— L'utilisateur n'a pas déjà voté (`hasVoted`).

— Le candidat choisi est bien inscrit dans la liste des candidats (`validCandidate`).

Si les conditions sont remplies, le vote est enregistré et l'adresse de l'électeur est marquée comme ayant voté.(Voir Annexe)

3. Fonction `totalVotesFor`

Cette fonction permet d'interroger le nombre total de votes obtenus par un candidat. Elle effectue une vérification pour s'assurer que le candidat fourni est bien inscrit, puis retourne le nombre de votes associés.(Voir Annexe)

4. Fonction interne `validCandidate`

Cette fonction est utilisée en interne pour vérifier qu'un nom de candidat est bien valide. Elle parcourt la liste des candidats et compare chaque nom avec celui passé en paramètre, à l'aide de la fonction `keccak256` (hachage cryptographique), garantissant une comparaison fiable entre chaînes de caractères.(Voir Annexe)

5. Variables de l'état

(voir annexe) Le contrat utilise plusieurs variables pour conserver l'état de l'élection :

- `admin` : l'adresse Ethereum de l'administrateur du contrat.
- `candidates` : un tableau de chaînes représentant les candidats.
- `votesReceived` : une table de hachage stockant le nombre de votes reçus par chaque candidat.
- `hasVoted` : une table de hachage pour éviter les doubles votes.

Ce contrat est un exemple simple mais fonctionnel d'une application de vote décentralisé. Il garantit l'unicité du vote, la transparence du comptage, et l'intégrité des données grâce à l'utilisation de la blockchain.

4.4 Diagramme de Gantt

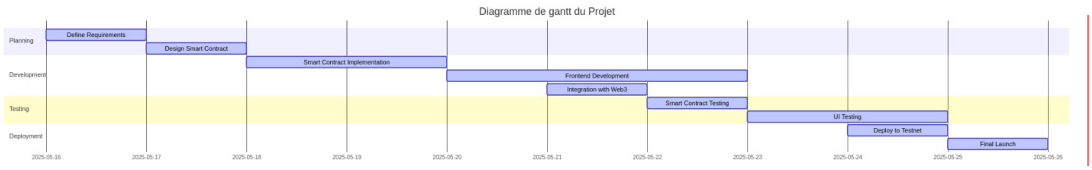

FIGURE 4.1 – Diagramme de Gantt

Conclusion

Cette phase méthodologique a permis de structurer efficacement le projet et d'assurer une répartition temporelle cohérente des tâches. Les outils choisis

sont adaptés aux exigences d'un système sécurisé, distribué et extensible. Le diagramme de Gantt a guidé le développement pas à pas jusqu'à la finalisation du prototype.

5 Réalisation

Introduction

La phase de réalisation technique constitue la concrétisation de la conception préalable du système. Elle a permis de développer les modules prévus, d'intégrer les outils choisis et de valider leur fonctionnement à travers des tests fonctionnels. Ce chapitre présente les composants implémentés ainsi que le fonctionnement global du système, illustré par des captures d'écran de l'interface.

5.1 Interface utilisateur (DApp)

L'interface utilisateur décentralisée a été conçue pour être simple, intuitive et accessible. Elle permet aux électeurs comme à l'administrateur de réaliser leurs actions via le navigateur et MetaMask.

5.1.1 Page d'accueil et connexion à MetaMask

La première interaction commence par la détection automatique du portefeuille MetaMask.

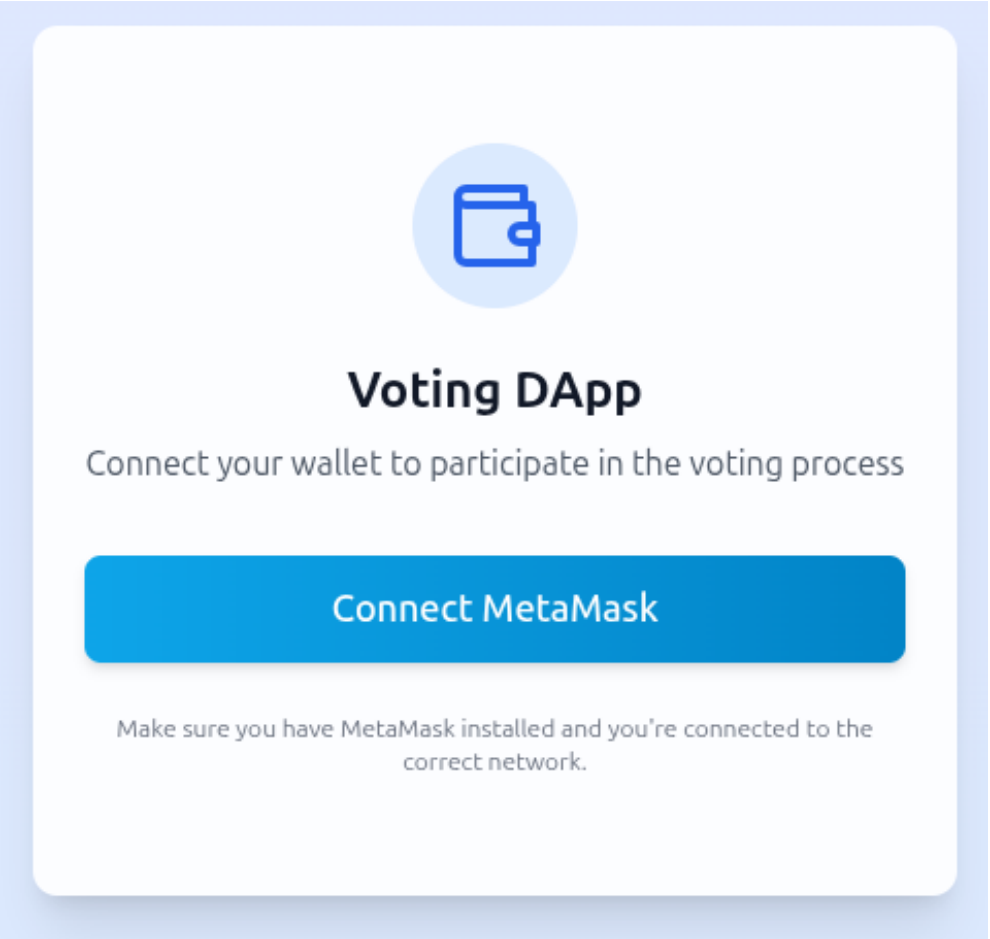

FIGURE 5.1 – Accueil de la DApp et connexion MetaMask

5.1.2 Demande d'inscription d'un électeur

Un électeur peut faire une demande d'inscription en saisissant son identifiant CIN, ce qui permet d'assurer l'unicité des utilisateurs.

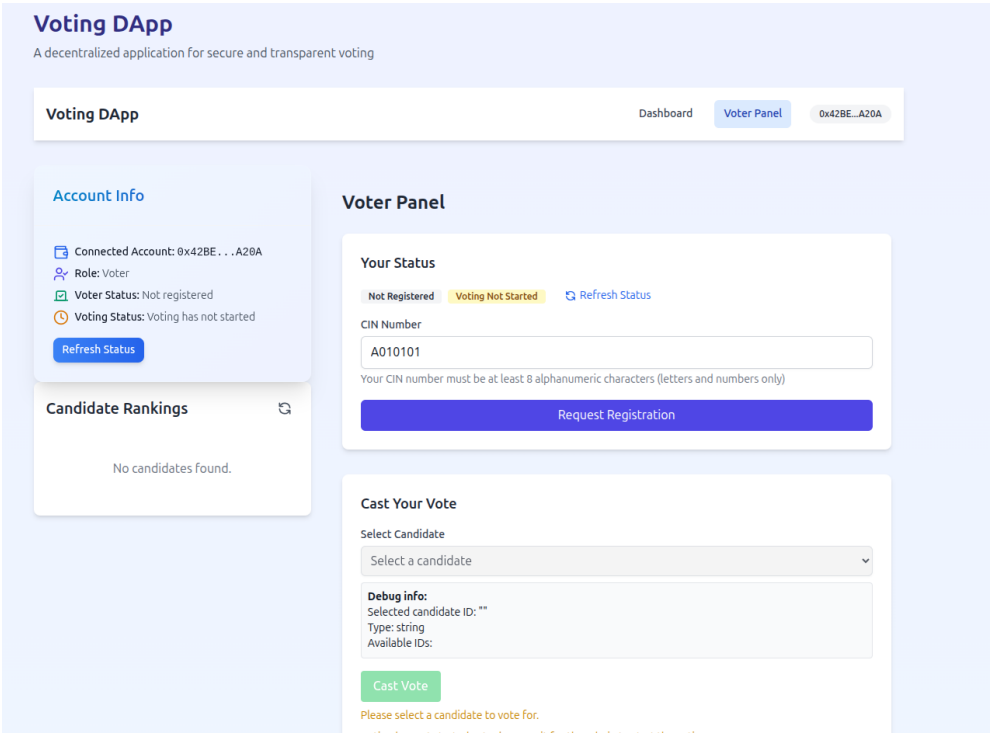

FIGURE 5.2 – Formulaire de demande d'inscription avec CIN

5.1.3 Vote pour un candidat

Une fois inscrit et validé, l'électeur peut voter en choisissant un candidat dans l'interface.

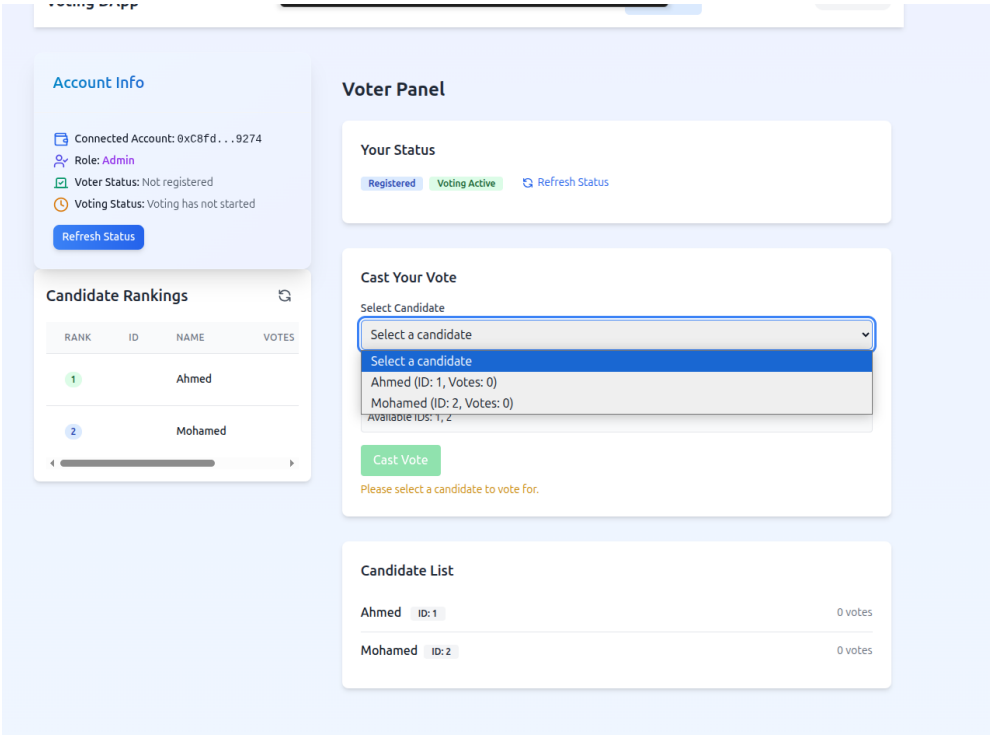

FIGURE 5.3 – Interface de vote pour les électeurs

5.1.4 Vue administrateur : gestion des inscriptions

(voir annexe) L'administrateur a accès à une interface spéciale pour approuver ou rejeter les demandes d'inscription.

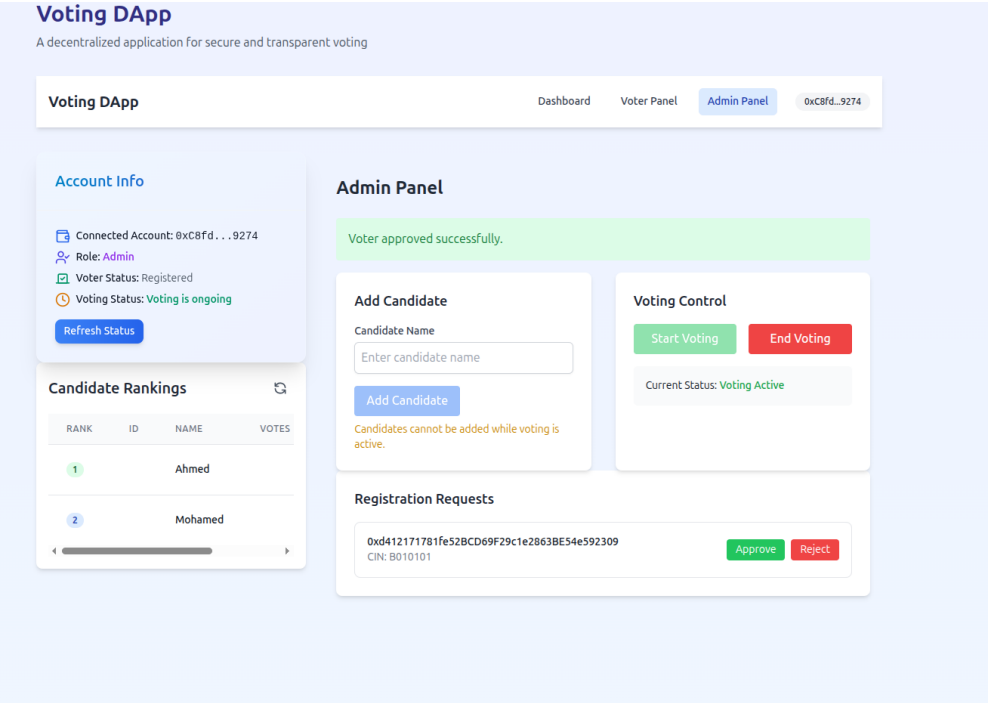

FIGURE 5.4 – Validation des électeurs par l'administrateur

5.1.5 Démarrage du vote et publication des résultats

Le vote peut être lancé, arrêté, et le vainqueur affiché dynamiquement grâce à une interaction directe avec le smart contract.

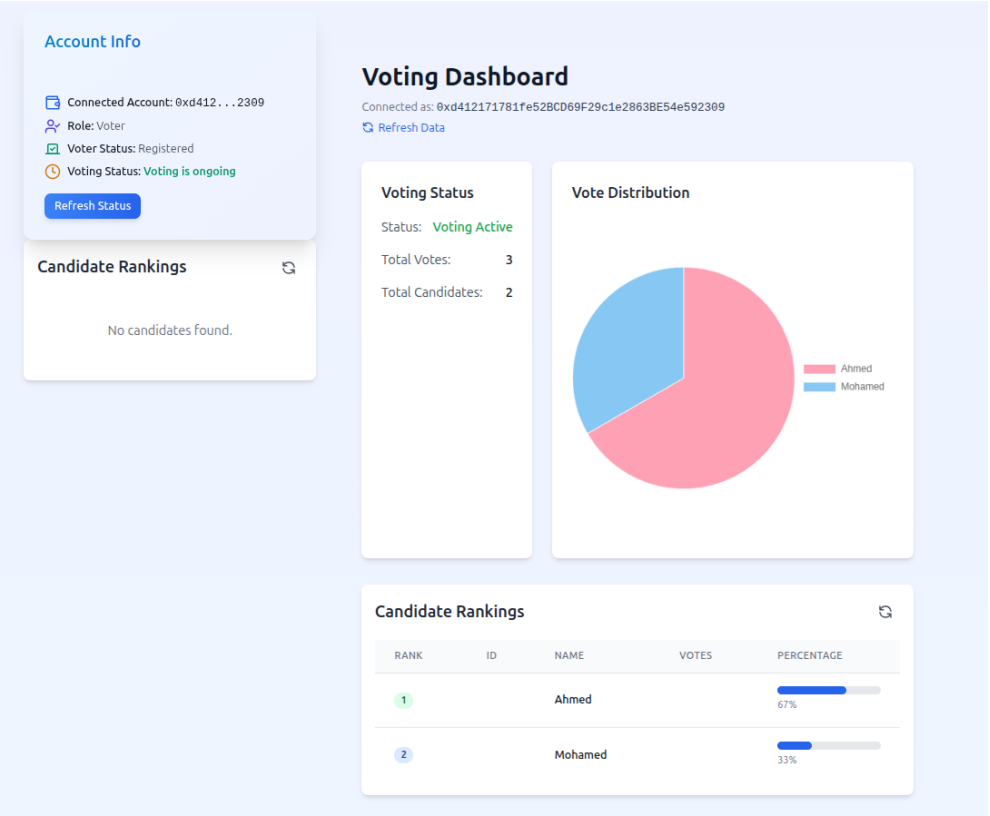

FIGURE 5.5 – Résultat des votes après dépouillement

5.2 Code source et hébergement

Le code source complet du projet est disponible sur GitHub, accompagné d'un guide d'installation :

— **Dépôt GitHub** : `https://github.com/Hamza-Bouali/Decentralized-Voting-System`

— **Site Web déployé** : `https://decentralized-voting-system-six.vercel.app/`

Le site est hébergé sur GitHub Pages et permet une démonstration interactive de l'interface connectée à un contrat Ethereum déployé sur une testnet publique (Sepolia) . L'utilisateur doit disposer de l'extension MetaMask pour interagir avec le système.

Conclusion

Cette phase de réalisation a permis d'aboutir à une application fonctionnelle, testée localement puis sur un réseau de test Ethereum. Les captures d'écran présentées témoignent de la maturité du prototype et de l'aboutissement des exigences techniques et fonctionnelles identifiées dans les phases précédentes.

Synthèse

Ce rapport présente la conception et la mise en œuvre d'un système de vote électronique décentralisé basé sur la blockchain Ethereum, visant à répondre aux failles de transparence, de sécurité et d'accessibilité des systèmes électoraux traditionnels. À travers une analyse critique de l'existant et une revue de l'état de l'art, il démontre que la technologie blockchain, grâce à ses propriétés d'immutabilité, de décentralisation et de vérifiabilité, constitue une solution crédible pour garantir l'intégrité du vote. Le système développé repose sur un contrat intelligent codé en Solidity, intégré à une interface web via Web3.js et MetaMask, permettant à chaque électeur d'exprimer un vote unique, anonyme et traçable. La méthodologie adoptée, de type agile, a permis une conception itérative et rigoureuse, illustrée par des diagrammes UML et validée par des tests fonctionnels sur réseau de test. Le projet aboutit à un prototype opérationnel offrant une alternative moderne, fiable et évolutive pour renforcer la confiance citoyenne dans les processus électoraux, notamment dans des contextes critiques comme les échéances démocratiques à venir.

Bibliographie

[1] R. Michael Alvarez and Thad E. Hall. *Electronic Elections : The Perils and Promises of Digital Democracy*. Princeton University Press, 2008.

[2] Pippa Norris. *Why Electoral Integrity Matters*. Cambridge University Press, 2014.

[3] Organisation pour la sécurité et la coopération en Europe (OSCE). *Rapports d'observation électorale*. `https://www.osce.org/fr/elections`

[4] Satoshi Nakamoto. *Bitcoin : A Peer-to-Peer Electronic Cash System*, 2008. `https://bitcoin.org/bitcoin.pdf`

[5] Zibin Zheng et al. *Blockchain challenges and opportunities : A survey*. International Journal of Web and Grid Services, 14(4), 2018.

[6] Philippe Noizat. *Blockchain Electronic Vote*. In *Handbook of Digital Currency*, Academic Press, 2015.

[7] Fergus W. Hardwick et al. *Electronic Voting with Blockchain : An E-Voting Protocol with Decentralisation and Voter Privacy*. Frontiers in Blockchain, 2018.

[8] Uzma Jafar, Mohd Juzaiddin Ab Aziz, Zarina Shukur, Hafiz Adnan Hussain. *A Systematic Literature Review and Meta-Analysis on Scalable Blockchain-Based Electronic Voting Systems*, 2022.

Annexes

Listing 1 – Déclaration du contrat Voting

```
pragma solidity ^0.8.0;

// Declaration du contrat de vote
contract Voting {
    address public admin; // Adresse de l'administrateur
    string[] public candidates; // Liste des candidats
    mapping(string => uint256) public votesReceived; //
       Nombre de votes par candidat
    mapping(address => bool) public hasVoted; // Suivi des
       electeurs ayant deja vote

    // Constructeur initialisant les candidats et l'admin
    constructor(string[] memory _candidates) {
        admin = msg.sender;
        candidates = _candidates;
    }

    // Les fonctions sont definies ci-dessous
}
```

Listing 2 – Fonction de vote

```
/**
 * Permet a un utilisateur de voter pour un candidat.
 * Verifie que l'utilisateur n'a pas deja vote et que le
    candidat est valide.
 */
function vote(string memory candidate) public {
    require(!hasVoted[msg.sender], "Vous avez d  j   vot
       .");
    require(validCandidate(candidate), "Candidat invalide
       .");
    votesReceived[candidate]++;
    hasVoted[msg.sender] = true;
```

```
}
```

Listing 3 – Fonction pour consulter les votes

```
/**
 * Retourne le nombre total de votes recus par un candidat
    donne.
 * Verifie d'abord que le candidat est valide.
 */
function totalVotesFor(string memory candidate) public view
    returns (uint256) {
    require(validCandidate(candidate), "Candidat invalide
       .");
    return votesReceived[candidate];
}
```

Listing 4 – Fonction de validation d'un candidat

```
/**
 * Fonction interne utilisee pour verifier si un nom
    correspond a un candidat valide.
 * Utilise une comparaison de hachages pour eviter les
    problemes de chaines.
 */
function validCandidate(string memory candidate) view
   internal returns (bool) {
    for (uint i = 0; i < candidates.length; i++) {
        if (keccak256(bytes(candidates[i])) == keccak256(
           bytes(candidate))) {
            return true;
        }
    }
    return false;
}
```