BlockChain

Masse Thibault
Félis Nicolas
Fernandez Bastien
Avril 2025 A2MSI

Présentation de l’équipe

Contexte et choix du sujet

Vote Electronique en Assemblée Générale

Problématique : Manque de confiance dans les systèmes électoraux

Objectif : Offrir une solution transparente, rendre les votes infalsifiable

Cas D’utilisation

Cas concret : éléction présidentielle

Problème actuelle : Difficulté à rassembler les participants, votes non vérifiables, manque de transparence

Solution : Application web connectée à Metamask et un smart contract avec Etherum

Fonctionnalités principales

Attribution des rôles

Vote avec horodatage

Résultat affiché automatiquement après clôture du vote

Liste blanche pour limiter les votants

Historique des votes consultable

Avantages

Transparence

Vote infalsifiable

Horodatage

Décentralisation

Inconvéniants

Coût des transactions

Compléxité technique pour les non-initiés

interface utilisateur

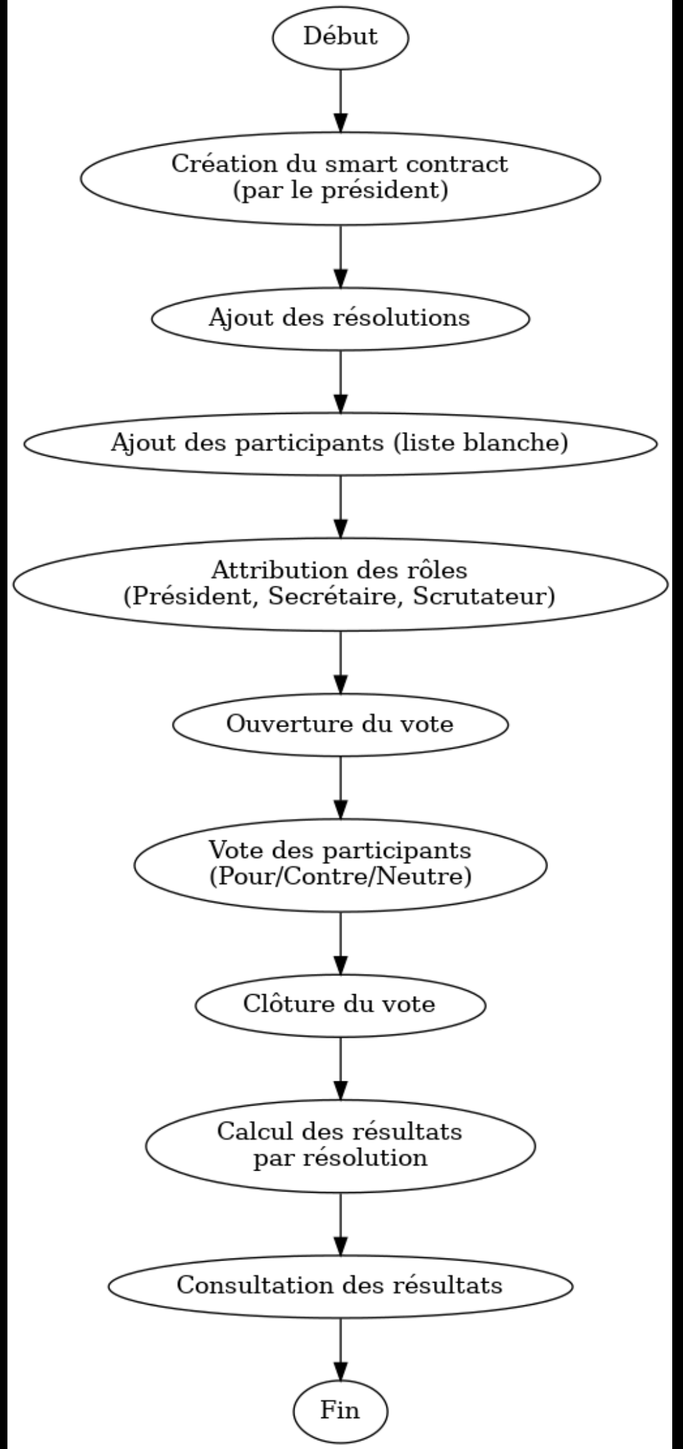
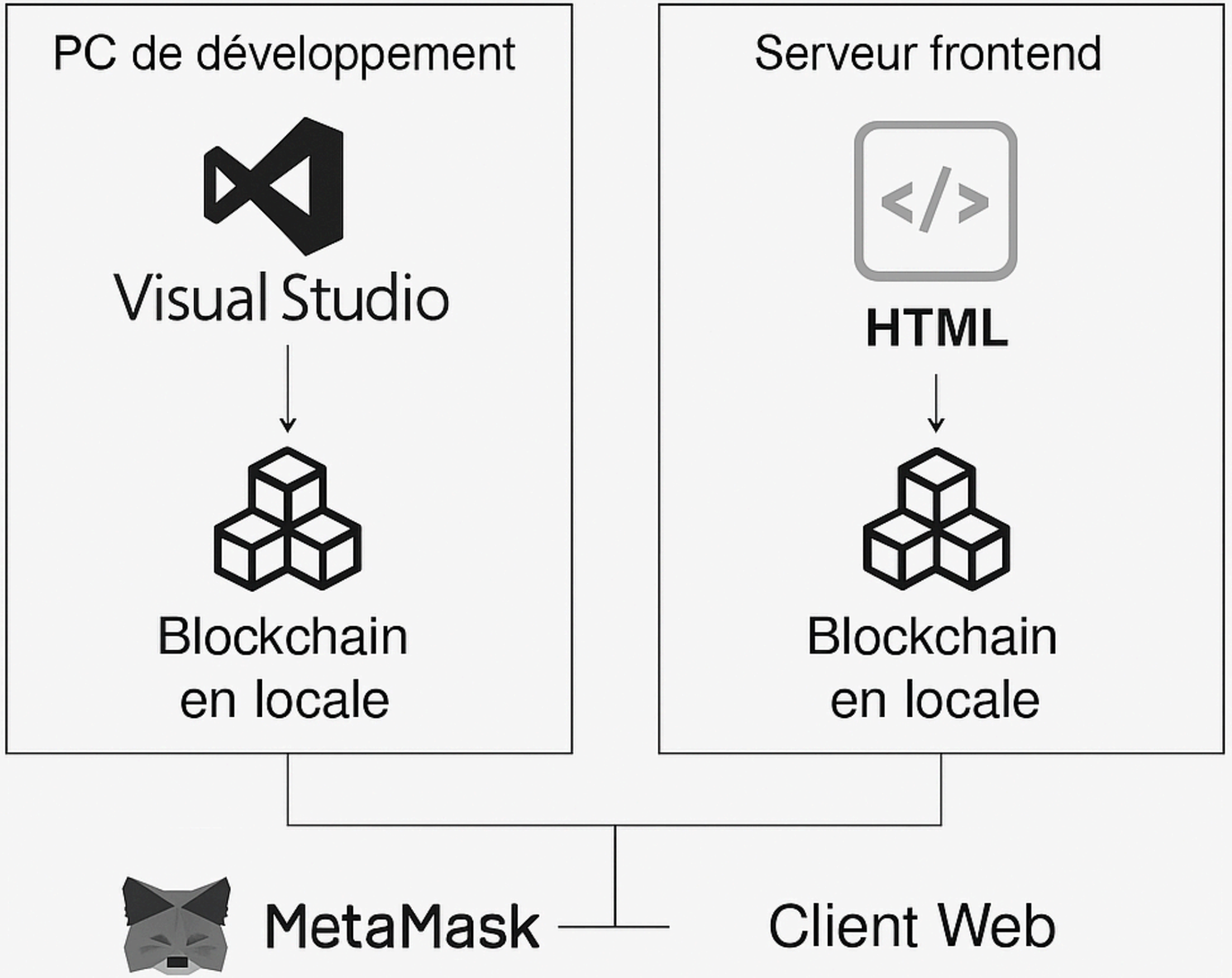


Diagramme UML d'Activité

Architecture



Smart Contract

Fonctions essentielles développées :

- addResolution(string title) → création d'une résolution (président uniquement)
- vote(uint resolutionId, VoteOption) → permet de voter (participants whitelistés)
- getVoteCounts(uint resolutionId) → lecture des résultats

Sécurité intégrée :

- Whitelist des votants (mapping)
- Vote unique par utilisateur et par résolution
- Clôture manuelle possible d'une résolution par le scrutateur

Smart Contract

```
// Comptabiliser les votants pour chaque résolution
mapping(uint => mapping(address => bool)) public hasVoted;
```

hasVoted permet d'empécher le double vote

```
// Modifier pour vérifier si l'adresse est dans la liste blanche
modifier onlyWhiteListed() {
    require(whiteList[msg.sender], "Vous n'etes pas autorise a voter.");
    _;
}
```

onlyWhiteListed filtre l'accès aux votants autorisés

```
// Événement de vote
event VotedEvent(uint indexed _resolutionId, VoteOption _voteOption);
event WhiteListUpdated(address _voter, bool _status);
event ResolutionAdded(uint _resolutionId, string _title);
event RolesUpdated(address _president, address _scrutateur);
```

les events assurent la traçabilité sur la blockchain

Smart Contract

```
// Voter pour une résolution avec un type de vote spécifique
function vote(uint _resolutionId, VoteOption _voteOption) public onlyWhiteListed {
    // Vérifier que l'ID de la résolution est valide
    require(_resolutionId > 0 && _resolutionId <= resolutionsCount, "ID de resolution invalide");
    require(resolutions[_resolutionId].isActive, "Cette resolution n'est plus active");

    // Vérifier que l'utilisateur n'a pas déjà voté pour cette résolution
    require(!hasVoted[_resolutionId][msg.sender], "Vous avez deja vote pour cette resolution.");

    // Enregistrer que l'électeur a voté pour cette résolution
    hasVoted[_resolutionId][msg.sender] = true;

    // Mettre à jour le compteur de votes en fonction du type de vote
    if (_voteOption == VoteOption.POUR) {
        resolutions[_resolutionId].voteCountPour++;
    } else if (_voteOption == VoteOption.CONTRE) {
        resolutions[_resolutionId].voteCountContre++;
    } else if (_voteOption == VoteOption.NEUTRE) {
        resolutions[_resolutionId].voteCountNeutre++;
    }

    // Déclencher l'événement de vote
    emit VotedEvent(_resolutionId, _voteOption);
}
```

La fonction vote() permet à un participant autorisé(présent dans la liste blanche) de voter pour une résolution parmi trois option possible :POUR/CONTRE/NEUTRE

lien GitHub

Démonstration