Projet Blockchain

Smart Contract SafeClub

Rapport final de validation

Réalisé par :
Oumayma Naffeti
Sywar Moussa

Classe : SSIR-G
Module : Blockchain

Année universitaire 2024–2025

Table des matières

1 Introduction

Ce projet a été réalisé dans le cadre du module **Blockchain**. Il consiste à concevoir et implémenter un smart contract Ethereum nommé **SafeClub**, permettant à un groupe de membres de gérer collectivement un fonds commun de manière décentralisée et sécurisée.

Les décisions de dépense sont prises par le biais de propositions soumises au vote des membres, avec application d'un quorum et de règles strictes garantissant la sécurité des fonds et la cohérence des états du contrat.

2 Objectifs du projet

Les objectifs principaux de ce projet sont les suivants :

- concevoir un smart contract sécurisé en Solidity,
- mettre en place un système de gouvernance basé sur des propositions et des votes,
- assurer la protection des fonds contre les abus et les attaques,
- tester et valider le contrat à l'aide de scénarios réalistes.

3 Technologies et outils utilisés

Le projet s'appuie sur les technologies suivantes :

- **Solidity** pour le développement du smart contract,
- **Remix IDE** pour la compilation, le déploiement et les tests,
- **MetaMask** pour la signature des transactions,
- **Ethereum Sepolia Testnet** comme réseau de test,
- **OpenZeppelin** pour les mécanismes de sécurité standards.

4 Description et architecture du smart contract

Le contrat **SafeClub** est structuré autour de plusieurs composants essentiels.

Tout d'abord, une gestion des rôles est mise en place. Le *Owner* est responsable du déploiement du contrat ainsi que de l'ajout et de la suppression des membres. Les membres, quant à eux, peuvent créer des propositions, voter et participer aux décisions collectives.

Ensuite, le contrat gère des propositions de dépense. Chaque proposition contient un destinataire, un montant, une description, une date limite de vote ainsi que des compteurs de votes pour et contre. Un état d'exécution permet d'éviter toute exécution multiple.

Enfin, un système de vote est implémenté. Chaque membre peut voter une seule fois pour ou contre une proposition. L'exécution d'une proposition n'est possible que si le vote est terminé, que le quorum est atteint et que la proposition n'a pas déjà été exécutée.

5 Fonctionnalités implémentées

Les fonctionnalités principales implémentées dans le contrat sont :

- ajout et suppression de membres,
- réception d'ETH dans le contrat (coffre-fort),
- création de propositions de dépenses,
- vote des membres (pour ou contre),
- exécution sécurisée des propositions acceptées.

6 Sécurité du smart contract

Plusieurs mécanismes de sécurité ont été intégrés afin de garantir la fiabilité du contrat.

La protection contre les attaques de type *reentrancy* est assurée par l'utilisation du module `ReentrancyGuard`. Les transferts d'ETH respectent l'ordre recommandé : effets puis interactions.

Le contrôle d'accès est strictement appliqué. Les fonctions sensibles vérifient systématiquement que l'appelant est un membre autorisé ou le propriétaire du contrat.

Enfin, les états et les montants sont validés avant toute action critique. Le contrat empêche la double exécution d'une proposition, bloque les votes après la deadline et vérifie la disponibilité des fonds avant tout transfert.

7 Tests et validation

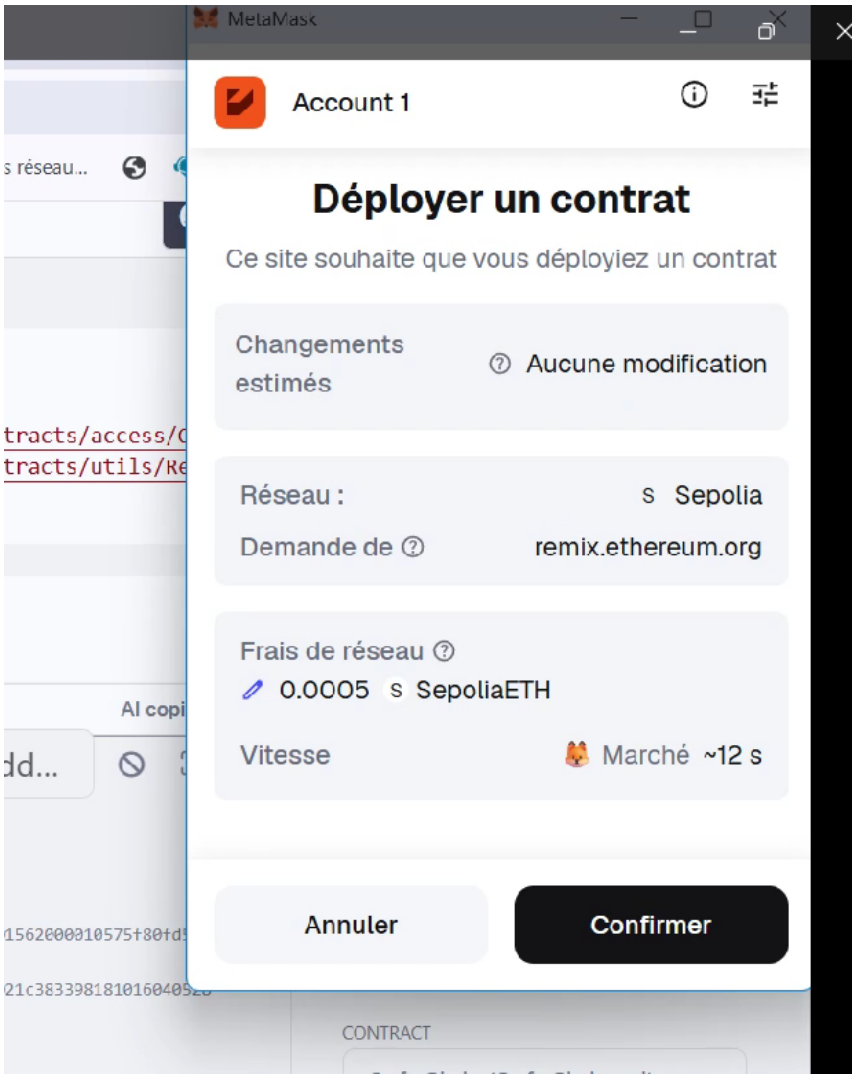
Les tests ont été réalisés manuellement à l'aide de Remix IDE et MetaMask, conformément aux consignes du projet.

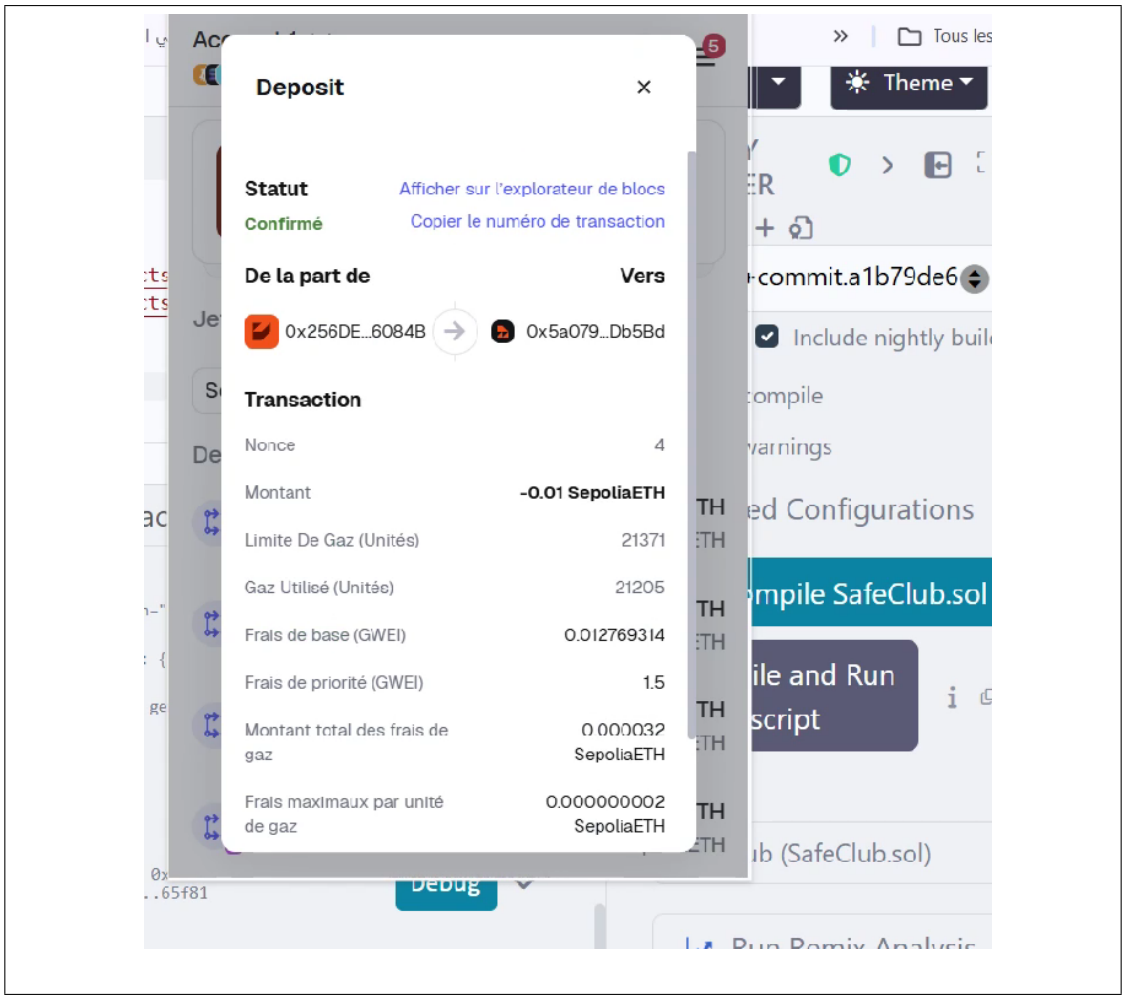
Les scénarios testés incluent le déploiement du contrat, l'ajout de membres, le dépôt d'ETH, la création de propositions, le vote des membres ainsi que l'exécution des propositions acceptées. Des cas d'erreur ont également été testés, notamment les tentatives de vote après expiration, les votes par des non-membres et les exécutions sans quorum atteint.

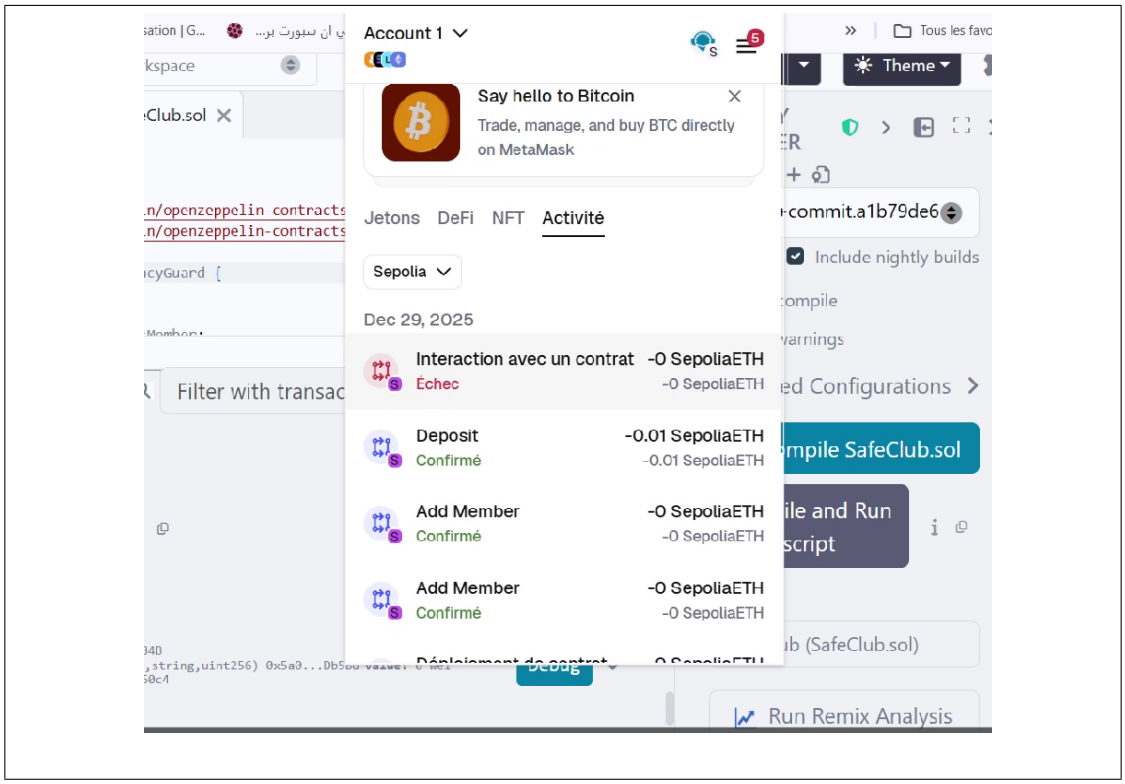
Emplacements des captures d'écran

Les captures d'écran suivantes doivent être insérées dans cette section :

- déploiement du contrat dans Remix,
- confirmation de transaction via MetaMask,
- dépôt d'ETH dans le contrat,
- création, vote et exécution d'une proposition.


MetaMask
Account 1
Déployer un contrat
Ce site souhaite que vous déployiez un contrat
Changements estimés
Aucune modification
Réseau :
Sepolia
Demande de
remix.ethereum.org
Frais de réseau
0.0005 SepoliaETH
Vitesse
Marché ~12 s
Annuler
Confirmer
CONTRACT


Deposit
Statut
Confirmé
Afficher sur l'explorateur de blocs
Copier le numéro de transaction
De la part de
Vers
0x256DE...6084B
0x5a079...Db5Bd
Transaction
Nonce 4
Montant -0.01 SepoliaETH
Limite De Gaz (Unités) 21371
Gaz Utilisé (Unitée) 21205
Frais de base (GWEI) 0.012769314
Frais de priorité (GWEI) 1.5
Montant total des frais de gaz 0.000032 SepoliaETH
Frais maximaux par unité de gaz 0.000000002 SepoliaETH
Theme
Include nightly builds
Configurations
SafeClub.sol
Debug


Account 1
Say hello to Bitcoin
Trade, manage, and buy BTC directly on MetaMask
Jetons DeFi NFT Activité
Sepolia
Dec 29, 2025
Interaction avec un contrat -0 SepoliaETH
Échec -0 SepoliaETH
Deposit -0.01 SepoliaETH
Confirmé -0.01 SepoliaETH
Add Member -0 SepoliaETH
Confirmé -0 SepoliaETH
Add Member -0 SepoliaETH
Confirmé -0 SepoliaETH
Theme
Include nightly builds
Configurations
SafeClub.sol
Debug
Run Remix Analysis

8 Limites et améliorations

Malgré son bon fonctionnement, le projet présente certaines limites. Le système de vote ne prend pas en charge la délégation ou la pondération des votes, et l'interface utilisateur reste volontairement minimale.

Des améliorations possibles incluent l'ajout d'une interface web dédiée, l'implémentation d'une gouvernance avancée, l'intégration d'un mécanisme de multisignature ainsi que l'utilisation de tokens de gouvernance.

9 Documentation technique du smart contract SafeClub

Cette section présente une documentation synthétique du smart contract **SafeClub**. Elle décrit les structures de données utilisées, les fonctions principales ainsi que la règle de décision appliquée pour l'acceptation et l'exécution des propositions.

9.1 Structures de données

Le smart contract repose sur plusieurs structures et mappings permettant de gérer les membres, les propositions et les votes.

Gestion des membres

Les membres du SafeClub sont gérés à l'aide d'un mapping :

```
mapping(address => bool) isMember;
```

Ce mapping permet de vérifier rapidement si une adresse est autorisée à interagir avec les fonctions sensibles du contrat. Le propriétaire du contrat (*Owner*) est responsable de l'ajout et de la suppression des membres.

Structure des propositions

Chaque proposition de dépense est représentée par une structure `Proposal` contenant les informations suivantes :

- **id** : identifiant unique de la proposition,
- **to** : adresse du bénéficiaire,
- **amount** : montant à transférer,
- **description** : description textuelle de la dépense,
- **deadline** : date limite du vote,
- **executed** : état d'exécution de la proposition,
- **forVotes** : nombre de votes favorables,

— **againstVotes** : nombre de votes défavorables.

Les propositions sont stockées dans un mapping indexé par un identifiant numérique :

```
mapping(uint256 => Proposal) proposals;
```

Gestion des votes

Afin de garantir qu'un membre ne puisse voter qu'une seule fois par proposition, un mapping supplémentaire est utilisé :

```
mapping(uint256 => mapping(address => bool)) hasVoted;
```

Ce mécanisme empêche toute tentative de double vote.

9.2 Fonctions principales

Le smart contract SafeClub expose plusieurs fonctions clés permettant d'assurer son fonctionnement.

Ajout et suppression de membres

— `addMember(address)` : permet à l'Owner d'ajouter un nouveau membre.
— `removeMember(address)` : permet à l'Owner de supprimer un membre existant.

Ces fonctions sont protégées par le modificateur `onlyOwner`.

Dépôt de fonds

Le contrat agit comme un coffre-fort collectif. Les fonds peuvent être déposés via :

— la fonction `deposit()`,
— ou la fonction `receive()` lors d'un envoi direct d'ETH.

Les fonds déposés sont stockés dans la balance du contrat.

Création d'une proposition

La fonction `createProposal` permet à un membre de proposer une dépense. Elle prend en paramètres :

— l'adresse du bénéficiaire,
— le montant à transférer,
— une description,
— une durée de vote.

La deadline est calculée automatiquement à partir de l'horodatage courant (`block.timestamp`).

Vote sur une proposition

La fonction `vote(uint256 id, bool support)` permet à un membre de voter pour ou contre une proposition. Chaque membre ne peut voter qu'une seule fois par proposition. Les votes sont comptabilisés séparément en votes favorables et défavorables.

Exécution d'une proposition

La fonction `execute(uint256 id)` permet d'exécuter une proposition acceptée. Elle est protégée par plusieurs vérifications de sécurité :

— la proposition ne doit pas déjà être exécutée,
— la période de vote doit être terminée,
— le quorum doit être atteint,
— le contrat doit disposer d'un solde suffisant.

Le transfert d'ETH est effectué de manière sécurisée via `call`.

9.3 Règle de décision

La règle de décision du SafeClub repose sur un mécanisme de gouvernance simple et transparent.

Une proposition est considérée comme **acceptée** si :

— le nombre de votes favorables est supérieur ou égal au quorum défini lors du déploiement,
— la période de vote est terminée,
— la proposition n'a pas encore été exécutée.

Dans le cas contraire, la proposition est rejetée ou reste inexécutable.

Ce mécanisme garantit que toute dépense est validée collectivement par un nombre minimum de membres, renforçant ainsi la sécurité et la transparence du système.

10 Conclusion

Ce projet a permis de mettre en pratique les concepts fondamentaux des smart contracts Ethereum, notamment la gouvernance décentralisée, la gestion collective de fonds et les mécanismes de sécurité. Les tests réalisés avec Remix et MetaMask confirment le bon fonctionnement du smart contract **SafeClub** et sa conformité aux exigences du projet.