RAPPORT DE PROJET BLOCKCHAIN

SMART CONTRACT : RecompenseEcologique

NIVEAU: Master II

SPECIALITE: Data Engineer

PAR:

- Franklin
- Edson
- Ormar

Supervisé par :

M.YANN FORNIER

ANNEE ACADEMIQUE : 2024 / 2025

Table des matières

INTRODUCTION

Le smart contract **RecompenseEcologique** a été conçu pour encourager les initiatives écologiques en permettant aux utilisateurs de soumettre des actions écoresponsables et d'être récompensés sous forme de tokens écologiques après un processus de validation basé sur le vote de la communauté. Ce système vise à promouvoir la participation collective à la protection de l'environnement, tout en offrant une transparence totale grâce à la technologie de la blockchain.

Ce rapport détaille l'architecture du smart contract, les choix techniques effectués ainsi que les instructions nécessaires pour le déployer et l'utiliser.

.

PARTIE I : ANALYSE DES BESOINS

1. Objectifs du système

Le système vise à encourager et récompenser les actions écologiques via un mécanisme de vote. Chaque utilisateur peut proposer une action et les autres membres peuvent voter pour valider l'action. Les actions validées sont récompensées par des points ou des jetons, et les votants ainsi que les proposeurs sont rémunérés en fonction de ces points.

Ces actions favorisent :

- Le recyclage de déchets électroniques, plastiques ou métalliques.
- La plantation d'arbres.
- L'utilisation d'énergies renouvelables dans leur quotidien (ex. utilisation de panneaux solaires).
- La réduction de leur empreinte carbone (par exemple, l'utilisation de moyens de transport verts comme le vélo ou la voiture électrique).
- La participation à des initiatives locales de nettoyage.

2. Analyse Fonctionnelle

a. Acteurs du système

• Administrateur: Le créateur du contrat, responsable de la gestion globale, incluant la distribution de récompenses

• Proposeur: Utilisateur qui soumet une action écologique à valider.

• Votant: Utilisateur qui vote pour valider une action écologique.

b. Description Textuelle de quelques cas d'utilisation :

Nous avons plusieurs cas d'utilisation dans notre projet notamment :

- **Proposer une action écologique**
- **Voter pour une action écologique**

- **Validation automatique d'une action écologique**
- **Rémunérer les votants et le proposeur après validation**
- **Lister toutes les actions proposées**
- **Consulter les détails d'une action spécifique**
- **Obtenir la liste des votants pour une action**
- **Obtenir la liste des actions votées par une adresse spécifique**

Mais nous nous contenterons de décrire les quatre premier

Cas d'utilisation 1 : Proposer une action écologique

- **Objectif** : Permettre à un utilisateur de soumettre une action écologique pour qu'elle soit évaluée et éventuellement validée par la communauté.
- **Acteur concerné** : Proposeur
- **Précondition** : L'utilisateur doit avoir une adresse Ethereum valide.
- **Scénario minimal** :
    1. Le proposeur appelle la fonction `proposeAction` avec une description valide, une adresse Ethereum et un nombre de points de récompense.
    2. Le contrat crée une nouvelle action avec un identifiant unique, et initialise le compteur de votes à zéro.
    3. L'action est enregistrée dans le mapping `actions`.
    4. Le proposeur devient le propriétaire de l'action.
- **Scénario alternatif** :
    - Si le proposeur fournit des informations incomplete ou invalides, l'action ne sera pas enregistrée.

Cas d'utilisation 2 : Voter pour une action écologique

- **Objectif** : Permettre à un utilisateur de voter pour une action écologique proposée pour contribuer à sa validation.
- **Acteur concerné** : Votant
- **Précondition** : L'action doit exister et ne doit pas encore être validée. L'utilisateur ne doit pas être l'administrateur et ne doit pas avoir déjà voté pour cette action.
- **Scénario minimal** :
    1. Le votant appelle la fonction `voteForAction` avec l'identifiant de l'action.
    2. Le contrat vérifie que l'action existe, que l'utilisateur n'a pas encore voté, et que l'action n'est pas encore validée.
    3. Si toutes les conditions sont remplies, le vote est enregistré.
    4. Le compteur de votes de l'action est incrémenté.
    5. Si le nombre de votes atteint ou dépasse le seuil (`VOTE_THRESHOLD`), l'action est validée.
- **Scénario alternatif** :
    - Si l'utilisateur a déjà voté pour cette action ou si l'action est déjà validée, le vote est rejeté.
    - Si l'utilisateur est l'administrateur, il ne peut pas voter.

Cas d'utilisation 3 : Validation d'une action écologique (automatique)

- **Objectif** : Valider automatiquement une action une fois qu'elle a reçu suffisamment de votes.

- **Acteur concerné** : Système (automatiquement déclenché par le vote)
- **Précondition** : L'action doit avoir reçu un nombre suffisant de votes (3 votes ou plus).
- **Scénario minimal** :

    1. Lorsqu'un votant soumet son vote, le contrat vérifie si le nombre de votes pour l'action atteint le seuil défini (`VOTE_THRESHOLD`).
    2. Si le seuil est atteint, l'action est marquée comme validée (`actions[actionId].validated = true`).
    3. L'événement `ActionValidated` est déclenché, notifiant que l'action est validée.

- **Scénario alternatif** :

    - Si le nombre de votes est insuffisant, l'action reste non validée et peut encore recevoir des votes.

Cas d'utilisation 4 : Rémunérer les votants et le proposeur après validation

- **Objectif** : Récompenser les utilisateurs (proposeur et votants) après la validation d'une action écologique.
- **Acteur concerné** : Administrateur
- **Précondition** : L'action doit être validée et des fonds doivent être disponibles pour la distribution des récompenses.
- **Scénario minimal** :

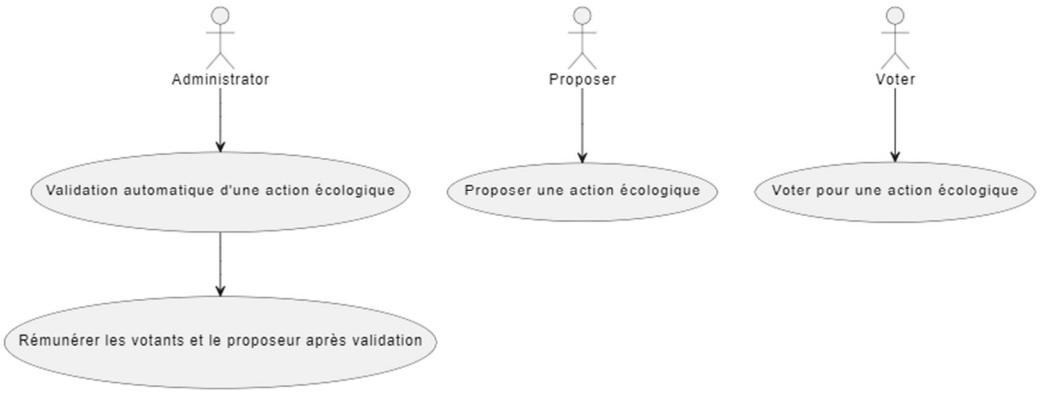
    1. L'administrateur appelle la fonction `payVotersAndProposer` en fournissant l'identifiant de l'action, l'adresse de l'utilisateur et le montant à transférer.
    2. Le contrat vérifie que l'action est validée.
    3. Les fonds sont transférés du solde de l'administrateur vers les adresses des votants et du proposeur.
    4. Les balances de récompenses sont mises à jour.
    5. L'événement `VoterPaid` est déclenché pour notifier le transfert.

- **Scénario alternatif** :

    - Si l'action n'est pas encore validée, le paiement ne peut pas être effectué.

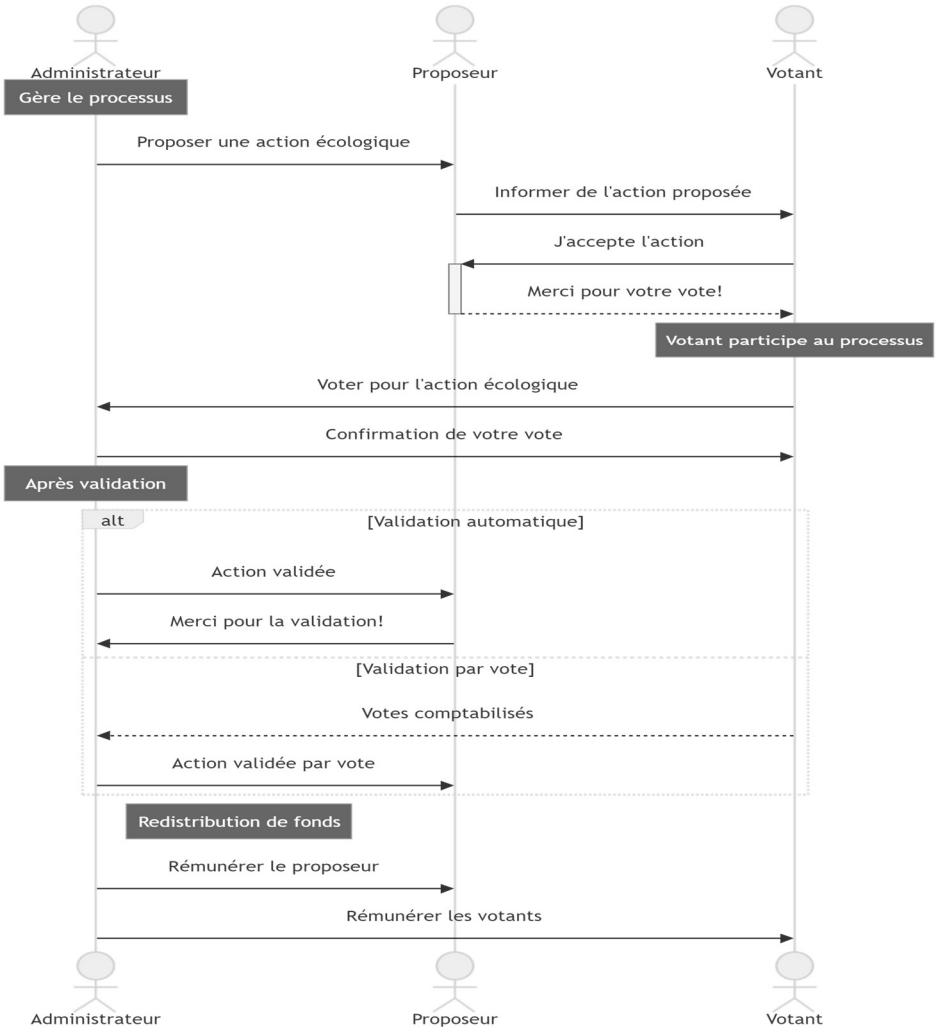
c. Diagramme de cas d'utilisation

Dans notre cas, le diagramme de cas d'utilisation aide à visualiser clairement les interactions entre les acteurs (Administrateur, Proposeur, Votant) et les fonctionnalités du système de récompense écologique. Il permet aussi de clarifier les responsabilités de chaque acteur et les fonctionnalités attendues, facilitant ainsi la conception et le développement du contrat intelligent.



d. Diagramme de Séquence Système

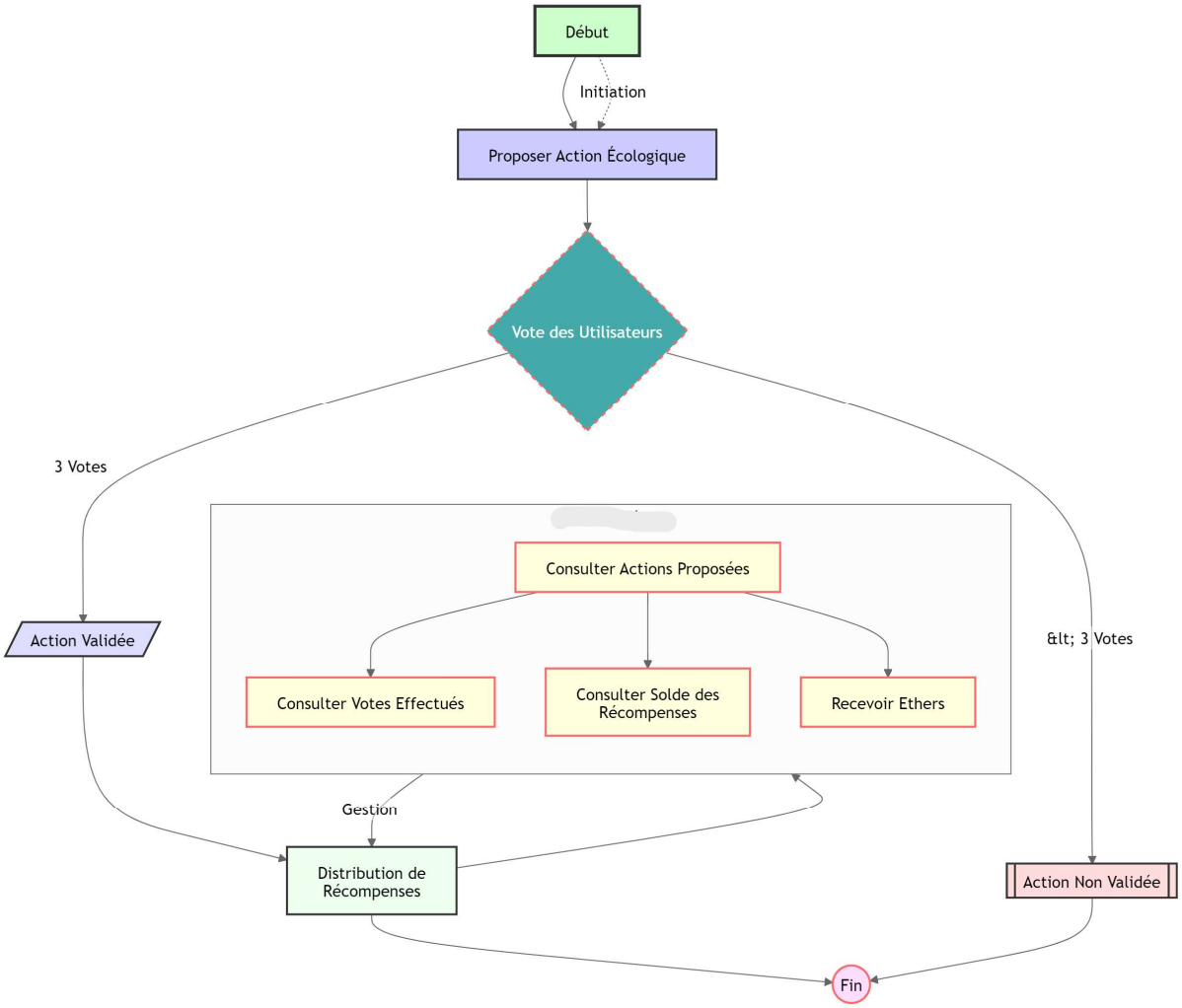
Les diagrammes de cas d'utilisation illustrent les interactions entre les acteurs (utilisateurs, administrateurs, partenaires) et le système. Voici quelques cas d'utilisation pour le système de récompense écologique :



PARTIE II : CONCEPTION ET ARCHITECTURE

1. Flowchart

Il nous aider à visualiser les différentes étapes du processus d'interaction entre les acteurs et le système de récompense écologique. Il montre le **flux des actions**, comme la soumission d'une action écologique, le vote, la validation, et la distribution des récompenses, en mettant en évidence l'ordre d'exécution et les décisions. Cela aide à comprendre la logique du contrat et à identifier d'éventuelles améliorations ou corrections dans le déroulement des opérations



Flux du processus :

a. Création d'une action écologique

b. Vote sur une action par la communauté

c. Validation automatique d'une action (si seuil atteint)

d. Attribution des récompenses (tokens écologiques)

2. Description du schéma

- **Smart Contract** : Le cœur du système qui gère la soumission des actions, les votes et la distribution des tokens.
- **Utilisateur** : Toute personne qui peut soumettre une action écologique ou voter sur une action existante.
- **Propriétaire (Auditeur)** : L'administrateur ou gestionnaire du contrat qui possède des privilèges supplémentaires (comme la validation manuelle des actions)

3. Architecture Fonctionnelle

L'architecture fonctionnelle de cette application décrite par l'image semble être la suivante :

Front-end (React JS) :

- Le navigateur web envoie une requête HTTP à un serveur pour obtenir l'application web, qui est construite avec React JS.
- Cette application web interagit avec le backend via des appels API (Web3.js/Ethers.js) pour effectuer des actions sur la blockchain Ethereum.

Backend (Web3 Provider) :

- Comme précédemment, le serveur web transmet les requêtes du front-end au Web3 Provider, qui sert d'intermédiaire avec la blockchain Ethereum.
- Le Web3 Provider communique également avec MetaMask pour récupérer les informations d'authentification et d'autorisation des utilisateurs.

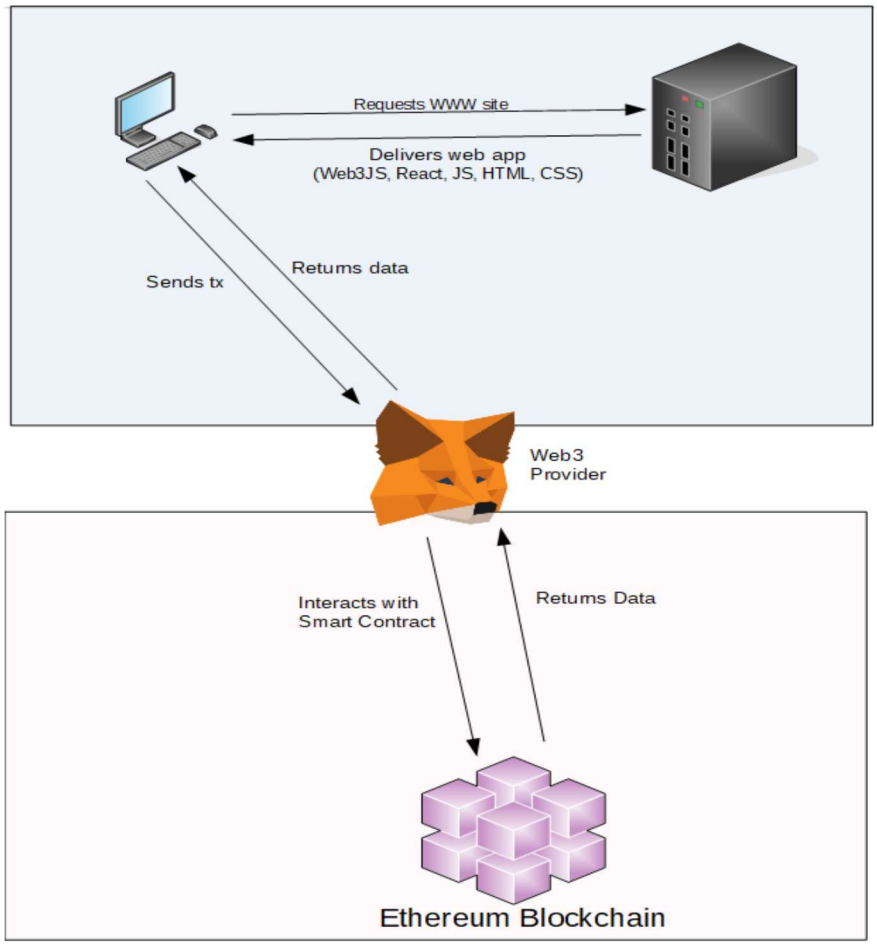
Blockchain (Ethereum) :

- L'application web interagit avec des contrats intelligents déployés sur la blockchain Ethereum.
- Les données et les transactions sont enregistrées sur la blockchain Ethereum de manière décentralisée.

Outils de développement (Hardhat) :

- Hardhat est un environnement de développement Ethereum qui facilite le déploiement, les tests et la compilation des contrats intelligents.
- Il permet aux développeurs de construire, tester et déployer leurs applications Ethereum de manière plus efficace.

Cette architecture combine les technologies front-end (React JS), back-end (Web3 Provider) et blockchain (Ethereum) pour créer une application décentralisée (DApp) fonctionnelle. L'utilisation d'outils de développement comme Hardhat facilite le processus de construction et de déploiement de l'application sur la blockchain Ethereum.


Requests WWW site
Delivers web app
(Web3JS, React, JS, HTML, CSS)
Sends tx
Returns data
Web3
Provider
Interacts with
Smart Contract
Returns Data
Ethereum Blockchain

PARTIE III : Choix techniques

1. Langage de programmation

Le smart contract est écrit en **Solidity**, la langue de programmation standard pour le développement de contrats intelligents sur la blockchain **Ethereum**. La version du compilateur utilisée est ^0.8.0, qui apporte des améliorations de sécurité et des optimisations par rapport aux versions antérieures.

2. Fonctions principale

- **creerAction** : Permet à un utilisateur de soumettre une nouvelle action écologique avec une description. Cette action est stockée dans un mapping.
- **voterSurAction** : Les utilisateurs peuvent voter sur les actions soumises. Les votes positifs ou négatifs sont comptabilisés, et si une action atteint un certain seuil de votes positifs, elle est automatiquement validée.
- **validerAction** : Fonction interne qui valide une action dès que le seuil de votes est atteint et attribue une récompense fixe sous forme de tokens écologiques à l'utilisateur qui a soumis l'action.
- **consulterTokens** : Permet à tout utilisateur de vérifier son solde de tokens écologiques.

3. Structure des données

- **Mapping** : Les actions soumises sont stockées dans un mapping, ce qui permet une gestion efficace des actions par leur identifiant unique.
- **Tokens** : Les tokens écologiques sont attribués et suivis via un autre mapping qui stocke les balances de chaque utilisateur.

4. Sécurité

- **aDejaVote** : Un mécanisme de protection contre les votes multiples est mis en place grâce à un mapping qui enregistre si un utilisateur a déjà voté sur une action spécifique.
- **Modificateur seulementProprietaire** : Ce modificateur permet de restreindre certaines fonctions sensibles (comme la validation manuelle d'une action) à l'administrateur (propriétaire) du contrat.

PARTIE III : Instructions pour déployer et utiliser le smart contract

Instructions pour déployer et utiliser le smart contract

1. Prérequis

Avant de déployer le smart contract, assurez-vous d'avoir les éléments suivants :

- Un environnement de développement Solidity, tel que **Remix** (https://remix.ethereum.org) ,ou VScode
- Un compte Ethereum (Reseau Hardhat local) et un portefeuille comme **MetaMask** pour gérer les transactions.
- Des ETH sur un testnet ou sur le mainnet pour payer les frais de gaz lors du déploiement du contrat.

2. Étapes de déploiement

- **Configuration de l'environnement de développement**
  - Installer Node.js et npm : `node -v` et `npm -v`
  - Installer les dépendances React JS : `npm create-react-app my-dapp`
  - Installer les dependances Hardhat et ethers : npm install hardhat ethers
  - Initialiser un nouveau projet Hardhat : `npx hardhat init`
  - Compiler le projet Hardhat : npx hardhat compile
  - Lancer le reseau hardhat pour avoir les comptes de test : npx hardhat node (le lancer dans un autre shell et ne pas l'éteindre durant tout le développement)
- **Développement du contrat intelligent (Smart Contract)**
  - Créer un nouveau contrat Solidity dans Hardhat : `npx hardhat create-contract`
  - Compiler le contrat : `npx hardhat compile`
  - Écrire les tests unitaires : `npx hardhat test`
- **Déploiement du contrat intelligent**
  - Configurer le script de déploiement Hardhat et déployer le contrat : `npx hardhat ignition deploy ./ignition/modules/EcoRewardSystem.js --network localhost`
  - Récupérer les informations du contrat déployé (adresse, ABI)
- **Intégration du Front-end (React JS)**
  - Importer les bibliothèques Ethers.js ou Web3.js : `npm install ethers` ou

- Intégrer les interactions avec le contrat dans les composants React
- Lancer le serveur de développement React : `npm start`

3. Utilisation

A. Création d'une action écologique

- Un utilisateur peut soumettre une action en appelant la fonction creerAction et en fournissant une description.

B. Voter sur une action

- Les utilisateurs peuvent voter sur une action spécifique en appelant la fonction voterSurAction avec l'ID de l'action et leur vote (positif ou négatif).

C. Validation et récompenses

- Une action est validée automatiquement si elle reçoit suffisamment de votes positifs (3 par défaut). L'utilisateur qui a soumis l'action reçoit des tokens écologiques en récompense.

D. Consulter les tokens

- Les utilisateurs peuvent consulter leur solde de tokens écologiques via la fonction consulterTokens.

CONCLUSION

Le smart contract **RecompenseEcologique** propose un système simple mais efficace pour encourager les actions écologiques via la blockchain. Grâce à un système transparent de soumission d'actions et de votes communautaires, il incite les utilisateurs à contribuer à la protection de l'environnement. Le processus de validation automatique garantit une répartition équitable des récompenses, et les mécanismes de sécurité intégrés assurent une utilisation fiable du système.

Avec cette documentation, les développeurs et utilisateurs intéressés ont à leur disposition toutes les informations nécessaires pour comprendre, déployer et interagir avec le contrat. Cela marque un pas vers l'utilisation de la blockchain pour des causes écologiques, renforçant ainsi la transparence et la responsabilité collective.