Sorbonne université

Faculté des sciences et de l'ingénierie

Master Science et Technologie Logiciel (STL)

Rapport de Projet N°2 DAAR

Tabellout Salim 21307533
Tabellout Yanis 21307532

21/10/2024

TABLE DES MATIÈRES

CHAPITRE 1

ETH PROJECT

1. Introduction

Le développement de la technologie blockchain a ouvert de nouvelles possibilités dans de nombreux domaines, y compris le monde du divertissement et des jeux. Ce projet a pour objectif de créer une plateforme innovante pour un jeu de cartes à collectionner numériques, basé sur la blockchain, avec une attention particulière portée aux cartes Pokémon. L'objectif principal de cette initiative est de permettre aux utilisateurs de mint (créer), acheter et échanger des cartes de manière sécurisée et transparente. En utilisant des contrats intelligents, ce projet vise à gérer efficacement la propriété et l'unicité des cartes tout en offrant une expérience utilisateur fluide.

La popularité croissante des NFT (tokens non fongibles) a été un moteur clé derrière ce projet. Les NFT permettent de représenter des actifs numériques uniques sur la blockchain, rendant ainsi possible la collection, l'achat et la vente d'objets numériques de manière vérifiable et sécurisée. Avec ce contexte, nous allons explorer la conception et l'architecture de notre application.

2. Architecture

Dans le système décrit, le backend et le frontend collaborent de manière fluide pour gérer les cartes et les smart contracts. Voici un résumé de chaque composant :

— **Backend** : Il est responsable de la gestion des cartes. C'est à dire récupérer les cartes de l'api pokémon et de l'intégrer dans la blockchain en créant des collections.

— **Smart Contracts** : Les collections de cartes sont créées et manipulées à travers le contrat textbfMain, qui permet de créer de nouvelles collections et de minter des cartes. Pour celà nous avons défini deux smart contracts **Main** qui est deployé par la personne qui s'authentifie en premier, ainsi que **Collection** qui permet de répresente une collection de cartes.

— **Frontend** : La personne connectée via l'interface frontend peut déployer le contrat **Main** et interagir avec celui-ci, notamment pour minter de nouvelles cartes.

3. Frontend

Le frontend a été implémenté grâce à **React** et **TailWindCss** permettant d'avoir une interface fluide et intuitive et réactive. Les utilisateurs peuvent facilement naviguer dans les différentes sections de la plateforme, pour visualiser les collections, minter des cartes. Le design vise à rendre l'expérience utilisateur aussi fluide que possible, avec une attention particulière portée à l'esthétique et à l'ergonomie. Les principales fonctionnalités du frontend comprennent :

- **Visualiser les collections**
- **Minter les cartes**
- **Visualiser les owners** : Permet de voir les collections et les cartes mintés de chaque personne dans la blockchain.

Le frontend est connecté au backend via des requêtes API, ce qui permet un échange de données en temps réel. Cela signifie que lorsque l'utilisateur interagit avec le frontend, toutes les actions sont immédiatement reflétées sur la blockchain via le backend.

4. Backend

Le backend est écrit avec du pur ExpressJs et TypeScript. Il interagit directement avec les contrats intelligents ethers.js pour effectuer des transactions et permettre au frontend d'intéragir avec les smart contracts pour récupérer certaines informations. Lors du démarrage du serveur, le backend est responsable de la récupération des données des cartes depuis l'API de Pokémon et puis les ajouter dans les collections de la blockchain. Les principales responsabilités du backend incluent :

- **Récupération des Cartes** ou collections
- **Gestion des Transactions** : Suivi et gestion des transactions effectuées par les utilisateurs.
- **Interfaçage avec le Frontend**

L'interaction entre le frontend et le backend est essentielle, car elle garantit que toutes les actions des utilisateurs sont traitées et enregistrées de manière sécurisée sur la blockchain.

5. Smart Contracts

Nous avons deux fichiers principales. Le **Main** qui est responsable de la création des collections ainsi que de minter les cartes dans une collections. Le contract **Collection** permet de gérer les cartes et ses owners. L'utilisation de la bibliothèque OpenZeppelin pour les contrats ERC721 facilite le développement en offrant des fonctionnalités prêtes à l'emploi pour la gestion des tokens non fongibles (NFT).

5.1. Contrat Main

Le contrat Main est conçu pour gérer les collections de cartes. Il comprend plusieurs attributs clés :

- **count** : Un compteur d'entiers qui suit le nombre de collections créées, facilitant la gestion des collections et la création de nouvelles.

- **collections** : Un mapping qui associe un identifiant numérique à une instance de la collection correspondante. Cela permet une récupération rapide des collections à partir de leur identifiant.
- **collectionsById** : Un mapping reliant un identifiant de chaîne (string) à une instance de collection. Cela permet de retrouver les collections via des chaînes de caractères, ce qui est particulièrement utile pour les utilisateurs qui cherchent à accéder à une collection spécifique.
- **owners** : Un tableau d'adresses qui stocke les propriétaires uniques des cartes. Cela garantit que chaque utilisateur puisse être identifié de manière unique dans le système, facilitant ainsi la gestion des droits de propriété.
- **ownerExists** : Un mapping qui vérifie si une adresse est déjà enregistrée comme propriétaire, évitant ainsi les doublons et les erreurs lors de l'attribution des cartes.

Le contrat Main a aussi des méthodes permettant de :

- Créer des collections
- Minter une carte d'une collection : abstraction sur la fonction mintNft de collection
- Récupérer les owners dans la blockahin
- Récupérer les cartes mintés d'un owner

5.1.1. Héritage et Importations

— Le contrat **Main** hérite de **Ownable**, ce qui permet de restreindre certaines fonctions au propriétaire du contrat.

— Il importe les bibliothèques **Ownable** de **OpenZeppelin** pour la gestion des droits d'accès et **Collection** pour les opérations sur les collections.

5.1.2. Initialisation

Le contrat est initialisé avec un propriétaire qui est la personne ayant déployé le contract. Ce dernier détient les droits exclusifs pour créer des collections et minter des cartes. Pour des raisons de simplicité nous avons défini un propriétaire par défaut qui sera le owner du contrat Main.

5.2. Contrat Collection

Le contrat Collection est conçu pour gérer les cartes individuelles et comprend les attributs suivants :

- **collectionName** : Le nom de la collection, permettant de l'identifier facilement.
- **cardCount** :Indique le nombre total de cartes que la collection peut contenir, définissant ainsi la capacité de la collection.
- **__tokenIds** : Un compteur privé qui garde une trace des identifiants des tokens mintés, garantissant que chaque carte a un identifiant unique.
- **accountsTokens** : Un mapping qui associe une adresse à un tableau d'identifiants de tokens, permettant de suivre quels tokens sont possédés par quel utilisateur.

Le contract Collection permet de :

— **Minter des nfts**

— d'autres fonctionnalités qui permettent de surcharger les fonctions de base.

5.2.1. Héritage et Importations

Le contrat **Collection** hérite de **Ownable, ERC721URIStorage, ERC721Enumerable**, ce qui permet de restreindre certaines fonctions au propriétaire du contrat et de mieux gérer les NFTS.

Conclusion

Ce projet de jeu de cartes à collectionner sur blockchain représente une avancée significative dans le domaine des actifs numériques. En intégrant des fonctionnalités de minting, de gestion des collections et de propriété des cartes, nous avons créé un système robuste et flexible. Le backend, en s'occupant de l'initialisation des données sur la blockchain, joue un rôle clé dans la fonctionnalité du système. Les choix technologiques adoptés, notamment l'utilisation de la blockchain et des contrats intelligents, garantissent la sécurité et la transparence nécessaires pour inspirer la confiance des utilisateurs. Ce projet ouvre la voie à de futures innovations dans le domaine des jeux et des collections numériques.

- **Lien github** : `https://github.com/Yanis540/collectible-card-game-daar`