Sorbonne University

Faculté des Sciences et Ingénierie

Développement des Algorithmes d'Application Réticulaire

Projet 2 – Collectible Card Game.

Étudiante :

GUIASSA Rayene

Date du rendu :

05/11/2023

Table des matières

Introduction

Dans ce projet, nous visons à créer un jeu de cartes à collectionner décentralisé sur Ethereum, transformant chaque carte en un NFT unique selon la norme ERC-721. Le but est de simuler l'expérience des TCG classiques, permettant aux joueurs de collectionner, échanger et gérer des cartes numériques. Alors que la mise en place d'un gameplay complet est hors de portée, l'accent est mis sur la partie collection du jeu.

Objectif

Dans le monde des jeux, les jeux de cartes à collectionner (TCG/CCG) occupent une place de choix, offrant aux joueurs l'excitation de découvrir des cartes aléatoires dans des paquets, ou "boosters", pour ensuite construire un "deck" personnalisé. Ces jeux, comme Magic : The Gathering ou Pokémon TCG, ont conquis des fans partout, en version physique ou numérique. Le projet que nous abordons ici repose sur une idée : transposer ce concept dans l'univers numérique d'Ethereum, la célèbre blockchain. L'objectif est ambitieux : créer un jeu de cartes à collectionner décentralisé où chaque carte serait unique et échangeable sous forme de jeton non fongible (NFT).

Pour les non-initiés, les NFT sont des actifs numériques qui garantissent l'unicité et la propriété d'objets virtuels, idéaux pour répliquer la rareté et l'authenticité des cartes à collectionner. En utilisant la norme ERC-721, chaque carte devient un objet de collection numérique, échangeable et traçable sur la blockchain Ethereum. Ce rapport décrit ce que j'ai fais pour la création de ce TCG décentralisé : de la conception des cartes comme NFT jusqu'à la mise en place d'un marketplace, et le développement d'une interface utilisateur pour gérer et visualiser sa collection.

Les Contrats Onchain en Solidity

Présentation de Solidity et son rôle dans le développement de smart contracts

Solidity est le langage de programmation utilisé pour écrire des smart contracts sur des plateformes comme Ethereum. Ces contrats intelligents sont des programmes qui s'exécutent sur la blockchain et peuvent automatiser l'échange d'argent, de contenu, de droits, d'actions, ou de tout autre valeur. Solidity permet de créer des contrats avec des règles complexes, garantissant une grande sécurité et une fiabilité dans leur exécution.

Description du smart contract ERC-721 pour les NFT

Le smart contract ERC-721 est une norme pour la création de tokens non fongibles (NFT) sur la blockchain Ethereum. Ce type de smart contract permet à chaque token d'avoir des caractéristiques uniques et de ne pas être interchangeable, ce qui est parfait pour la représentation de cartes à collectionner numériques. Chaque NFT peut ainsi être vu comme une carte digitale unique avec sa propre identité et propriété vérifiables.

Gestion de collection, carte, de la propriété et transférabilité des NFT avec les contrats développés

J'ai élaboré le contrat principal Main qui orchestre la création de collections de cartes et le minting des NFT. Il est conçu pour maintenir une liaison entre les collections et leurs cartes respectives, tout en permettant la création et la distribution des NFT de manière contrôlée.

Le contrat CardNFT gère les NFT individuels. Chaque carte est mintée avec un identifiant unique et des métadonnées spécifiques, ce qui assure l'unicité et la propriété des NFT. De plus, il conserve un registre des tokens possédés par chaque utilisateur, permettant une gestion transparente et accessible des propriétés des joueurs.

Enfin, le contrat Collection permet de regrouper les NFT par collection, en limitant le nombre de cartes et en offrant la possibilité d'ajouter ou de retirer des cartes de la collection d'un utilisateur.

Le script de déploiement utilise Hardhat, une suite de développement Ethereum, pour automatiser le déploiement des contrats. Ce processus met en place l'infrastructure nécessaire pour que les utilisateurs interagissent avec les NFT via le contrat Main, qui est le point d'entrée de l'ensemble du système.

Chaque élément du système est conçu pour fonctionner en harmonie, offrant une expérience utilisateur fluide et intuitive pour la gestion de NFT dans le jeu de cartes à collectionner décentralisé.

Backend Offchain et API

Structure et Fonctionnalité du Serveur server.js

Initialisation et Configuration

- Le serveur utilise express pour faciliter la création d'APIs web avec Node.js.
- Le module ethers est utilisé pour interagir avec la blockchain Ethereum, notamment pour la gestion des contrats intelligents et des portefeuilles (wallets).
- Les variables d'environnement sont chargées grâce à dotenv.
- Le module axios est utilisé pour effectuer des requêtes HTTP à des services externes, dans ce cas, l'API du jeu de cartes Pokémon.
- Le cors permet de gérer les questions de sécurité liées aux requêtes cross-origin.
- L'adresse du contrat déployé et le fournisseur (provider) de services blockchain sont définis et initialisés.

Middleware pour la gestion des erreurs

Un middleware Express capture les erreurs asynchrones sur les routes et renvoie une réponse d'erreur générique.

API Routes

- /pokemon/sets : Récupère tous les ensembles (sets) de cartes Pokémon.
- /pokemon/sets/ :setId : Récupère les cartes d'un ensemble spécifique.
- /sets/ :setId : Obtient les détails d'un ensemble spécifique.
- /pokemon/sets/ :setId/cards/ :cardId : Récupère une carte spécifique d'un ensemble.
- /select-sets : Sélectionne certains ensembles en fonction de critères définis.

— /set-card-count/ :setId : Compte le nombre de cartes dans un ensemble.

— /nft-info/ :tokenId : Récupère les informations d'un NFT spécifique.

— /nfts/ :address : Récupère les cartes NFT associées à une adresse Ethereum spécifique.

— /pokemon/boosters/random-cards : Génère un ensemble aléatoire de cartes.

— /mint-nft/ :userAddress/ :setId/ :cardId : Mint une carte NFT pour une adresse utilisateur spécifique.

— /transfer-nft/ :oldOwnerAddress/ :newOwnerAddress/ :setId/ :cardId/ :setName : Transfère une carte NFT d'un utilisateur à un autre.

— /nfts/ :address/collection/ :collectionId : Récupère tous les NFT d'une collection pour un utilisateur.

Autres Fonctions

Des fonctions helper pour vérifier "l'actualité" des fichiers JSON et écrire des données dans ces fichiers.

Importance du Backend Offchain

Ce backend est essentiel pour plusieurs raisons :
Interactivité : Il permet aux utilisateurs d'interagir avec le système sans devoir exécuter des transactions sur la blockchain, ce qui peut être lent et coûteux.
Performance : Il améliore la vitesse et l'efficacité des requêtes en fournissant un accès rapide aux données nécessaires sans les frais de gaz associés aux appels de la blockchain.
Fonctionnalités Avancées : Il offre des fonctionnalités qui ne sont pas possibles directement sur la blockchain, telles que la création de boosters de cartes aléatoires et la gestion des NFT.
Sécurité : Il agit comme une couche de sécurité supplémentaire, masquant les détails de la blockchain et réduisant les risques d'attaques directes sur les contrats intelligents par les utilisateurs finaux.

Interaction Backend-Smart Contracts Ethereum

— Connexion : Le backend se connecte à Ethereum avec ethers.js via un fournisseur comme Infura.
— Lecture : Lecture des données sans frais avec des appels de type "view".
— Écriture : Envoi de transactions signées pour actions comme le minting de NFT.
— Smart Contracts : Utilisation de l'ABI pour interagir avec les contrats via le backend.
— Minting et Transfert : Transactions pour créer ou échanger des NFTs via les fonctions des contrats.
— Événements : Écoute des événements des contrats pour les mises à jour ou notifications.
— Coûts de Gaz : Gestion intelligente du gaz pour réduire les coûts.
— Confirmation : Suivi et gestion des statuts des transactions envoyées.

Technologies Utilisées :
Node.js : Environnement d'exécution pour JavaScript côté serveur.
Express.js : Framework pour Node.js qui simplifie la création de serveurs web et API.
Ethers.js : Bibliothèque pour interagir avec Ethereum et ses contrats intelligents.
Axios : Bibliothèque HTTP pour effectuer des requêtes HTTP.
CORS (Cross-Origin Resource Sharing) : Package pour gérer les politiques de sécurité entre différentes origines.
File System (fs) : Module de Node.js pour travailler avec le système de fichiers.
Le serveur off-chain agit comme un pont entre le front-end et la blockchain, gérant les interactions complexes et potentiellement coûteuses avec la blockchain.

Intégration de l'API Pokémon TCG

Présentation de l'API Pokémon TCG et son utilité

L'API Pokémon TCG permet d'accéder à une vaste base de données de cartes à collectionner Pokémon. Cette API fournit des informations détaillées sur chaque carte, y compris les images, les statistiques et la rareté. L'utilité principale de cette API dans le cadre de mon projet est de récupérer et d'utiliser ces données pour enrichir une application backend offchain, créant une expérience interactive et mise à jour pour les utilisateurs.

Méthodologie d'intégration de l'API avec le backend offchain

L'intégration a débuté par l'inscription pour obtenir une clé API auprès de Pokémon TCG API. Cette clé a été sécurisée dans un fichier .env pour faciliter les interactions avec l'API. Dans le serveur Node.js, des routes spécifiques ont été mises en place pour interroger l'API et récupérer les données nécessaires. Par exemple, la route '/pokemon/sets' récupère tous les ensembles de cartes Pokémon disponibles.

Gestion des appels d'API pour récupérer les données des cartes

Pour gérer les appels API, j'ai utilisé la bibliothèque axios pour effectuer des requêtes HTTP. Les routes définies dans server.js, telles que '/pokemon/sets/ :setId' et '/pokemon/sets/ :setId/cards/ :cardId', permettent de récupérer les données spécifiques d'un set ou d'une carte. Ces données sont ensuite envoyées au client en tant que réponse JSON.

Stockage et mise à jour des informations de cartes

Les informations des cartes sont gérées de manière dynamique, avec des fonctions pour vérifier la fraîcheur des données stockées localement. Si les informations des cartes sur le serveur sont obsolètes, un nouvel ensemble de données est récupéré de l'API et stocké. Cela garantit que les utilisateurs ont accès aux informations les plus récentes sans devoir systématiquement solliciter l'API, optimisant ainsi la performance et la réactivité de l'application.

La méthodologie adoptée pour l'intégration assure un accès efficace aux données nécessaires pour les utilisateurs finaux, tout en respectant les bonnes pratiques de développement et la sécurité des informations d'authentification.

Développement Frontend

Conception de l'interface utilisateur pour l'affichage des cartes

- ***Home page :*** La page d'accueil est une invitation engageante au monde du jeu de cartes à collectionner Pokémon. Elle présente un message de bienvenue centré qui motive les visiteurs à explorer les cartes disponibles et à s'immerger dans l'entraînement pour devenir un dresseur émérite. Le design visuel utilise un fond d'écran thématique qui est adouci par un panneau semi-transparent, ajoutant une touche d'élégance et de focus au texte d'accueil.
- ***Login page :*** La page de connexion offre aux utilisateurs la possibilité de se connecter facilement à l'application en utilisant leur portefeuille MetaMask. En cliquant sur un bouton, les utilisateurs peuvent relier leur compte Ethereum à l'application, ce qui leur permet de gérer leur collection de cartes Pokémon TCG. La navigation est automatiquement redirigée vers la page d'accueil une fois la connexion établie, offrant une expérience utilisateur fluide et sans interruption.

— ***User page :*** La page utilisateur affiche la collection de cartes Pokémon de l'utilisateur, offrant la possibilité de filtrer les cartes par nom, set ou force pour faciliter la recherche. Les cartes sont présentées avec des images et des détails pertinents, et peuvent être triées selon différents critères. Cette fonctionnalité rend la gestion de la collection intuitive et efficace, améliorant l'expérience de l'utilisateur dans la navigation et l'organisation de ses cartes Pokémon TCG.
— ***Marketplace page :*** La page Marketplace offre une interface intuitive pour explorer et acquérir des cartes Pokémon. Elle permet aux utilisateurs de parcourir des ensembles de cartes triés par série, affichant chaque ensemble avec une image et une date de sortie pour une identification rapide. En sélectionnant un ensemble, on peut visualiser les cartes individuelles et, en cliquant sur une carte, accéder à ses détails spécifiques tels que le type, le niveau et les capacités, ainsi que les options d'achat. L'expérience visuelle est renforcée par des réactions au survol et des transitions fluides pour une navigation agréable. Avec une mise en page claire et une interaction simplifiée, la page Marketplace rend l'acte d'achat de cartes Pokémon à la fois direct et agréable.
— ***Booster page :*** La page Booster permet aux utilisateurs de découvrir aléatoirement des cartes Pokémon à ajouter à leur collection. Dès l'ouverture de la page, un appel est fait pour récupérer un ensemble de cartes aléatoires que l'utilisateur peut explorer. En cliquant sur une carte, l'utilisateur obtient une vue détaillée avec des informations telles que le nom de l'attaque, les dégâts et la rareté de la carte, ainsi qu'une image plus grande et le prix actuel. Cette interface immersive facilite l'expérience d'ouverture des boosters virtuels et enrichit l'expérience globale du collectionneur.
— ***Achat page :*** La page Achat guide les utilisateurs à travers le processus d'acquisition d'une carte Pokémon NFT. Après avoir sélectionné une carte, l'utilisateur est amené à une interface où les détails de la transaction sont clairement affichés, incluant le prix en euros et le solde de l'utilisateur. Si le solde est insuffisant, l'achat ne peut pas être effectué. Une fois l'achat confirmé, un message de succès informe l'utilisateur que le NFT a été ajouté à son portefeuille, renforçant ainsi l'aspect ludique et interactif de la collection.

Choix des technologies frontales

Dans le développement de l'interface utilisateur, React.js a été choisi pour sa capacité à créer une expérience utilisateur dynamique et réactive *(et aussi parce que le squelette nous a été donné en react...)*. L'utilisation de composants fonctionnels React et des hooks tels que useState et useEffect permet une gestion efficace de l'état et du cycle de vie des composants, facilitant ainsi la création d'une interface riche et interactive pour la gestion et l'achat de cartes Pokémon NFT.

Intégration avec le backend et les contrats intelligents

L'intégration avec le backend et les contrats intelligents a été réalisée en utilisant Axios pour les appels API REST, permettant une interaction fluide entre l'interface utilisateur React et le serveur, ainsi que l'interaction avec les contrats intelligents pour les opérations de blockchain, telles que l'achat et le minting de NFTs. Cette synergie assure que les actions sur le frontend se reflètent de manière sécurisée et cohérente dans la logique métier et l'état du blockchain.

Conclusion

En concluant ce projet, j'ai mis en place une interface interactive pour les amateurs de Pokémon TCG, articulant avec succès la collection, l'achat et la visualisation de cartes. Ce travail a exigé une compréhension approfondie de React pour le front-end et une intégration délicate avec les contrats intelligents pour une expérience utilisateur sécurisée et fluide.

Face aux obstacles techniques, j'ai adopté des solutions innovantes qui ont consolidé mes compétences en développement et affiné mon approche de la programmation.

Envisageant l'avenir, je suis déterminé à introduire des fonctionnalités permettant aux utilisateurs de revendre leurs cartes, de jouer en duel et d'échanger, projetant ainsi la plateforme vers une simulation fidèle de l'expérience Pokémon TCG. Ces améliorations potentielles promettent non seulement d'enrichir l'engagement des utilisateurs mais aussi de s'aligner sur l'évolution dynamique du jeu de cartes Pokémon.