Sorbonne Université

MU5IN552 - DAAR

Année 2024-2025

Projet 2: Collectible Card Game

Stéphanie HUANG & Alex XU
Encadré par BM Bui-Xuan

Sommaire

Introduction

Le projet **Collectible Card Game** a pour but de créer un jeu de cartes à collectionner (**TCG)**, sur la blockchain Ethereum, en utilisant des **NFTs** pour représenter les cartes. Ce type de jeu, qui repose sur l'ouverture de boosters, et la construction de decks pour jouer contre d'autres joueurs, est populaire, tant physiquement que numériquement, avec des jeux célèbres comme **Magic: The Gathering**, **Pokémon TCG**, **Hearthstone**, ou encore **Yu-Gi-Oh!**. L'objectif de ce projet est de se concentrer sur la partie **collection** d'un TCG, où les utilisateurs peuvent **acquérir**, **échanger**, et **organiser** leurs cartes.

Pour ce faire, il s'agira d'implémenter le standard **ERC-721**, qui permet de créer des NFTs sur Ethereum, et de développer une infrastructure capable de gérer les nouvelles collections de cartes. L'ensemble du projet repose sur une approche décentralisée, utilisant **HardHat** pour le développement blockchain local, ainsi que des outils comme **Node.js**, **Yarn**, et **Metamask** pour assurer l'intégration entre les contrats intelligents et les interfaces utilisateurs.

Le projet se décompose en deux parties principales : une partie **onchain**, qui consiste à créer et gérer des collections de cartes sous forme de NFTs, et une partie **offchain**, qui inclut le développement du frontend pour interagir avec les contrats et visualiser les collections. Par ailleurs, des appels à l'API Pokémon TCG sont effectués, afin de récupérer les informations des cartes et afficher les métadonnées associées à chaque NFT. Ainsi, ce rapport présente les étapes d'implémentation et les choix techniques effectués pour créer une plateforme décentralisée de gestion de cartes à collectionner, de la conception des **smart contracts** à la création d'une interface utilisateur, tout en intégrant des éléments réels du **Pokémon TCG** via son API officielle.

1. Technologies et Outils

Le projet a pour objectif de développer un jeu de cartes à collectionner (TCG) en utilisant des technologies décentralisées. Certaines des technologies utilisées, telles que l'Ethereum, Solidity, HardHat, Yarn, et l'API Pokémon TCG, ont été imposées dans le cadre du projet pour répondre aux besoins spécifiques du développement sur la blockchain Ethereum. D'autres technologies, comme React et TypeScript, ont été choisies pour faciliter la création de l'interface utilisateur et la gestion du front-end.

1.1. Ethereum et Solidity

Dans ce projet, **Ethereum** a été utilisé comme plateforme blockchain pour la gestion des **NFTs**, qui sont les cartes à collectionner numériques. La norme **ERC-721** a été employée pour garantir l'unicité et la traçabilité de chaque carte. Ethereum, ainsi que le langage **Solidity** pour développer les smart contracts, ont été imposés dans le cadre du projet afin de garantir une compatibilité avec l'infrastructure blockchain existante. Solidity est largement utilisé pour le développement de smart contracts sur Ethereum et permet de gérer les aspects liés à la création, l'échange, et la gestion des cartes en tant que NFTs.

1.2. HardHat

HardHat a été choisi comme environnement de développement pour les smart contracts. HardHat permet de déployer et tester les contrats intelligents dans un environnement de blockchain local, ce qui est essentiel pour itérer rapidement sans avoir à payer des frais de gas. Ce choix a également été imposé dans le projet pour faciliter le développement des fonctionnalités onchain tout en garantissant que les contrats intelligents respectent les exigences de la blockchain Ethereum.

1.3. Web3.js

Pour interagir avec la blockchain Ethereum depuis l'application web, la bibliothèque **Web3.js** a été utilisée. Cet outil permet à l'application front-end de se connecter aux contrats intelligents déployés et d'envoyer des transactions à la blockchain. Il est également responsable de la gestion des portefeuilles utilisateurs, permettant aux joueurs de visualiser et d'interagir avec leurs cartes NFT. Bien que l'utilisation de Web3.js n'ait pas été imposée, elle constitue le standard de facto pour les interactions avec Ethereum dans les applications décentralisées.

1.4. React et TypeScript

Pour développer le front-end, nous avons opté pour **React** et **TypeScript**. **React** est un framework JavaScript populaire qui permet de créer des interfaces dynamiques et réactives, basées sur des composants. Il a été choisi pour sa modularité et sa capacité à rendre les données blockchain de manière fluide et interactive. **TypeScript** a été utilisé pour ajouter du typage statique, ce qui améliore la robustesse du code en facilitant la détection des erreurs lors du développement.

1.5. React-Bootstrap

Pour concevoir l'interface utilisateur, **React-Bootstrap** a été utilisé pour assurer un design moderne et responsive. **Bootstrap** est un framework CSS pour le design rapide et adaptable, **React-Bootstrap** permet d'utiliser des composants Bootstrap directement dans une application React.

1.6. API Pokémon TCG

L'**API Pokémon TCG** a été intégrée pour fournir des informations sur les cartes Pokémon à collectionner, comme les images et les descriptions des cartes. Cela permet à l'application de s'appuyer sur

des données réelles pour afficher les cartes et enrichir l'expérience utilisateur. Nous avons choisi l'API Pokémon dans le projet pour simuler un scénario réel de collection de cartes avec une source de données externe.

2. Conception Onchain

Dans cette section, nous décrivons la conception des aspects onchain de notre système NFT en lien avec la norme ERC-721, les structures de contrats intelligents, les fonctionnalités clés, ainsi que les avantages et les limites de l'implémentation sur la blockchain.

2.1. NFTs et norme ERC-721

Les NFTs (Tokens Non Fongibles) sont des actifs uniques vérifiables sur la blockchain, utilisés pour représenter des éléments numériques, dans notre cas, des cartes de collection. En utilisant la norme ERC-721, nous garantissons que chaque NFT est unique et que son historique est enregistré de manière immuable. La norme ERC-721 a été choisie car elle permet :

- La gestion de propriétés uniques avec un identifiant pour chaque token.
- Une interopérabilité avec les portefeuilles et les plateformes qui supportent les NFTs de norme ERC-721.

L'ERC-721 standardise les méthodes pour créer, transférer, et vérifier la propriété des NFTs. Dans notre application, nous avons étendu les fonctionnalités de base de l'ERC-721 en implémentant des fonctionnalités personnalisées comme la création de cartes de collection, la vérification de possession et la mise en vente sur un marché.

2.2 Structure des Smart Contracts

L'implémentation repose sur quatre contrats principaux : **Collection, Booster, Trading, et Main**. Chaque contrat a un rôle spécifique dans la gestion, la vente, et le trading des cartes (NFTs) au sein du système. L'objectif est de créer, distribuer, et permettre l'échange de cartes de collection sous forme de NFTs en utilisant la norme ERC-721.

- **Contrat Collection** :

  Le contrat Collection gère la création et la gestion des cartes de collection sous forme de NFTs en conformité avec la norme ERC-721. Il intègre :

  - **Gestion des cartes créées** : Les cartes créées sont stockées dans un tableau de structures Card, qui contient les attributs num, img, name (nom de la collection), et cardCount (nombre de cartes maximales).
  - **Suivi des cartes et propriétaires** : Ce contrat utilise des mappages pour gérer la propriété (cardToOwner) et le nombre de cartes détenues par chaque utilisateur (ownerCardCount).
  - **Fonctionnalités ERC-721** : Il inclut des fonctions de transfert et de récupération de propriété pour gérer les échanges et transferts de cartes, implémentant ainsi les méthodes requises pour être compatible avec ERC-721.
- **Contrat Booster** :

  Le contrat Booster est conçu pour créer des "packs de booster" contenant trois cartes de collection prédéfinies. Ce contrat permet :

    - **Création de cartes de booster** : Seul l'administrateur peut ajouter des cartes spécifiques à un booster.
    - **Ouverture des boosters** : Un utilisateur peut ouvrir un booster pour recevoir les cartes qui lui sont assignées.
- **Contrat Trading** :

  Le contrat Trading gère la vente et l'échange de cartes entre utilisateurs. Il inclut :

    - **Mise en vente des cartes** : Permet aux utilisateurs de mettre des cartes en vente et de les acheter.
    - **Suivi des ventes** : Utilise un mappage pour suivre l'état des cartes mises en vente et facilite la recherche des cartes disponibles.
- **Contrat Main** :

  Le contrat Main est le point d'entrée principal qui orchestre l'interaction entre les utilisateurs et les autres contrats. Il permet :

    - **Création de collections** : Génère une nouvelle collection de carte
    - **Création et ouverture de boosters** : Gère l'appel aux fonctions du contrat Booster pour la création et l'ouverture des packs.
    - **Assignation de cartes** : Transfère la propriété des cartes à un utilisateur spécifique.
    - **Gestion des ventes** : Permet la mise en vente et l'achat de cartes en appelant les fonctions du contrat Trading.

2.3 Avantages de la blockchain

La conception onchain apporte plusieurs avantages au projet :

- **Transparence** : Les cartes et leurs échanges sont enregistrés de manière transparente sur la blockchain. Chaque transaction est visible publiquement, garantissant que les utilisateurs peuvent vérifier à tout moment l'historique et la légitimité de leurs cartes.
- **Propriété vérifiable** : Grâce aux NFTs, les utilisateurs peuvent prouver de manière cryptographique qu'ils sont les propriétaires uniques de leurs cartes.
- **Résilience** : L'utilisation de la blockchain Ethereum garantit que les cartes ne peuvent pas être falsifiées ou supprimées, offrant ainsi une grande sécurité aux utilisateurs.

2.4 Limites et défis

Bien que la blockchain offre de nombreux avantages, elle impose également certaines limites :

- **Coûts de transaction élevés** : Chaque transaction effectuée sur la blockchain Ethereum entraîne des frais, appelés gas fees. Ces frais peuvent devenir élevés, surtout lorsque la blockchain est congestionnée. Cela peut poser problème lors de la création ou du transfert de nombreuses cartes.
- **Complexité de gestion des données** : La gestion des collections, des cartes, des boosters, et des ventes simultanément dans plusieurs contrats nécessite des mappages complexes et un suivi attentif des identifiants. Cette complexité augmente le risque d'erreurs.
- **Problème de taille dans le contrat Main.sol** : Sur Ethereum, la taille maximale d'un contrat intelligent est de 24 KB pour le bytecode déployé. Si le contrat Main devient trop volumineux, il peut dépasser cette limite, empêchant ainsi son déploiement. Cette limite est atteinte lorsqu'ils incluent de nombreux appels à d'autres contrats.

3. Conception Offchain

La conception offchain concerne les éléments de l'application qui ne sont pas directement liés à la blockchain, mais qui interagissent avec elle. Cela concerne principalement le frontend pour l'interface utilisateur et les appels à des APIs externes, comme l'API Pokémon TCG. Cette section explique les choix technologiques et architecturaux de la partie offchain.

3.1 Frontend avec React et TypeScript

Le frontend de l'application a été développé en React, un framework JavaScript qui permet de créer des interfaces utilisateurs réactives et modulaires. L'utilisation de TypeScript en complément permet de renforcer la robustesse du code grâce à la vérification des types, minimisant ainsi les erreurs potentielles liées aux incohérences de type durant le développement.

L'interface utilisateur est organisée en composants, chacun représentant une partie spécifique de l'application. Voici les principales sections du frontend :

- Le composant **PokemonSet** permet aux utilisateurs de visualiser tous les sets de cartes disponibles dans l'application. Chaque set regroupe un ensemble de cartes spécifiques que l'utilisateur peut consulter et éventuellement acheter ou mint.
    - **Affichage des Sets** : PokemonSet affiche les sets disponibles en récupérant les informations depuis l'API Pokémon TCG.
    - **Sélection de Cartes pour Mint** : L'utilisateur peut sélectionner des cartes pour les mint. Lorsque le mint est confirmé, une transaction est initiée sur la blockchain pour générer les NFTs correspondants.
- Le composant **User** gère les informations de base de l'utilisateur et présente une vue de son inventaire de cartes. Il permet à l'utilisateur de visualiser les cartes qu'il possède.
    - **Affichage des Informations Utilisateur** : User affiche l'adresse Ethereum (si connectée).
    - **Gestion de l'Inventaire des Cartes** : User inclut un aperçu rapide des cartes en possession de l'utilisateur. En accédant aux informations via le portefeuille connecté, il peut afficher une liste des cartes en utilisant UserCard.
- Le composant **UserCard** est conçu pour afficher chaque carte détenue par l'utilisateur. Il fournit des informations détaillées sur la carte, telles que son nom, son image, ses caractéristiques et propose des actions spécifiques.
    - **Actions sur les Cartes** : UserCard intègre le bouton "Vendre" qui permet à l'utilisateur de mettre une carte en vente sur le Marketplace.
- Le composant **Marketplace** est le point d'entrée pour les utilisateurs qui souhaitent consulter les cartes disponibles à la vente. Ce composant présente les cartes en vente, avec la possibilité d'acheter.
    - **Récupération et affichage des Cartes en Vente** : Le composant se connecte à la blockchain pour obtenir la liste des cartes en vente et affiche les informations essentielles (image, prix, propriétaire).
    - **Achat de Cartes** : Chaque carte a un bouton "Acheter" qui déclenche une transaction blockchain via MetaMask pour acquérir la carte sélectionnée. Lorsqu'un utilisateur clique sur "Acheter", une transaction est créée, et un événement de confirmation de transaction déclenche la mise à jour des cartes en vente.
    - **Suppression de la carte** : La vente d'une carte peut être annulée seulement par son propriétaire.
- Le composant **Booster** représente la page où les utilisateurs peuvent ouvrir des paquets de cartes. Le concept de booster permet aux utilisateurs d'acheter un ensemble de cartes à un prix fixé.
    - **Affichage des Boosters Disponibles** : Le composant Booster affiche les boosters disponibles en récupérant les informations depuis la blockchain. Chaque booster est affiché avec son nom.

- **Gestion de l'Achat de Booster** : Lorsqu'un utilisateur achète un booster, une transaction est initiée sur la blockchain pour l'achat et l'ouverture du booster. Après une transaction réussie, la liste des cartes du booster est ajoutée à l'inventaire de l'utilisateur.

L'un des principaux avantages de React est sa capacité à gérer l'état de l'application de manière fluide, notamment avec les hooks comme useState et useEffect. Cela permet de réagir rapidement aux changements de l'état, comme la mise à jour des cartes après une transaction réussie.

3.2 Interaction avec la blockchain via Web3.js

L'interface frontend utilise Web3.js pour interagir avec la blockchain Ethereum. Web3.js est une bibliothèque JavaScript qui permet de se connecter à la blockchain, de lire les informations depuis les smart contracts, et d'envoyer des transactions signées par l'utilisateur.

- **Connexion au portefeuille** : Grâce à MetaMask, l'utilisateur peut se connecter à son portefeuille Ethereum directement depuis l'interface. Une fois connecté, l'utilisateur peut effectuer des actions telles que la visualisation des cartes ou le mint de nouvelles cartes.
- **Transactions blockchain** : Les utilisateurs peuvent interagir avec le smart contract en utilisant Web3.js pour envoyer des transactions. Par exemple, lorsqu'un utilisateur décide de mint des cartes, une transaction est signée par son portefeuille et envoyée à la blockchain.

3.3 Appels API dans le frontend

En plus des interactions avec la blockchain, le projet nécessite des interactions avec des APIs externes, notamment l'API Pokémon TCG pour récupérer les données des cartes. Ces appels sont effectués directement depuis le frontend de l'application.

- **Récupération des données Pokémon** : Le frontend récupère les données des cartes Pokémon (images, descriptions, etc.) depuis l'API Pokémon TCG et les affiche dans l'interface utilisateur pour permettre aux utilisateurs de consulter les détails des cartes.
- **Stockage des métadonnées** : Les informations sur les cartes (numéro, image) sont ensuite utilisées pour mint des cartes sur la blockchain, où elles sont représentées sous forme de NFTs.

3.4 Sécurité et gestion des utilisateurs

L'application s'assure que seuls les utilisateurs authentifiés peuvent interagir avec les smart contracts et que les transactions sont sécurisées par la signature cryptographique des portefeuilles Ethereum. MetaMask gère cette authentification via un mécanisme de signature, garantissant que les utilisateurs ne peuvent pas forger de transactions sans la clé privée associée à leur compte Ethereum.

4. Intégration de l'API Pokémon TCG

Une partie clé du projet consiste à intégrer des données réelles provenant de l'API Pokémon TCG pour fournir des informations pertinentes sur les cartes à collectionner. Cette intégration permet d'enrichir l'expérience utilisateur en affichant des cartes authentiques avec des illustrations, des descriptions et des détails tels que le numéro de la carte et la série à laquelle elle appartient. Cette section explique comment l'API a été intégrée dans le projet, ainsi que les défis rencontrés lors de cette intégration.

4.1 Présentation de l'API Pokémon TCG

L'API Pokémon TCG fournit des données complètes sur les cartes à collectionner du jeu de cartes Pokémon. Cela inclut des informations comme :

- **Nom de la carte**
- **Illustration**
- **Numéro de la carte**
- **Série et set d'appartenance**
- **Légalité dans différents formats de jeu**

Ces données sont essentielles pour représenter chaque carte dans l'application et permettre aux utilisateurs de collectionner des cartes numériques basées sur ces informations réelles. Dans le projet, les types PokemonSet et PokemonCard ont été utilisés pour structurer les informations récupérées.

4.2 Fonctionnement de l'intégration

L'intégration de l'API Pokémon TCG s'est faite via des appels HTTP depuis le frontend. L'objectif était de récupérer des informations sur les sets et les cartes, puis de les afficher dans l'interface utilisateur.

- **Récupération des sets** : Dans un premier temps, le frontend fait un appel à l'API pour récupérer la liste des sets disponibles. Ces sets, représentés par le type PokemonSet, sont ensuite affichés sous forme de cartes dans l'interface utilisateur, permettant aux joueurs de naviguer entre les différentes séries.
- **Récupération des cartes d'un set** : Lorsque l'utilisateur clique sur un set particulier, un autre appel est effectué pour récupérer toutes les cartes de ce set. Les informations, structurées via le type PokemonCard, sont ensuite utilisées pour afficher les cartes, permettant à l'utilisateur de sélectionner celles qu'il souhaite mint sous forme de NFT.

4.3 Gestion des métadonnées des cartes

Les informations obtenues via l'API Pokémon TCG sont utilisées pour enrichir les métadonnées des NFTs créés sur la blockchain. Chaque NFT, représentant une carte, contient des informations telles que :

- **L'image de la carte** (fournie par l'API)
- **Le numéro et le nom de la carte**
- **Les détails sur la série et le set**

Ces métadonnées sont intégrées dans le smart contract lorsque l'utilisateur mint une carte. Le type Card est utilisé pour structurer les données des cartes mintées sous forme de NFTs, avec les attributs num et img correspondant aux informations récupérées.

4.4 Défis techniques rencontrés

Comme toute interaction avec une API externe, la latence des requêtes peut affecter la réactivité de l'application. Afin de minimiser l'impact sur l'expérience utilisateur, des indicateurs de chargement ont été ajoutés pour informer l'utilisateur lorsque les données étaient en cours de récupération.

Conclusion

Ce projet de création d'un jeu de cartes à collectionner décentralisé sur la blockchain Ethereum a permis de mettre en œuvre une architecture complexe alliant des technologies onchain et offchain. Le choix d'utiliser la norme ERC-721 pour représenter les cartes sous forme de NFTs garantit la traçabilité des cartes numériques, tout en permettant leur échange sécurisé entre utilisateurs.

Sur le plan technique, la mise en place d'une infrastructure blockchain via HardHat a permis de simuler un environnement de développement local, réduisant ainsi les coûts liés aux transactions sur le réseau principal d'Ethereum. L'intégration de MetaMask et l'utilisation de Web3.js ont permis une interaction fluide entre les utilisateurs et la blockchain, rendant possible le mint et la gestion des cartes via des transactions signées de manière sécurisée.

L'intégration de l'API Pokémon TCG a enrichi l'application en apportant des données réalistes et familières, permettant aux utilisateurs de collectionner des cartes basées sur des jeux de cartes populaires. Malgré les défis rencontrés, comme la gestion des limites d'appels API ou l'optimisation des transactions blockchain, les solutions mises en place ont permis de garantir une expérience utilisateur fluide et sécurisée.

Ce projet met en avant les possibilités offertes par la blockchain pour la gestion d'actifs numériques uniques, tout en soulignant l'importance d'une bonne interaction entre les parties onchain et offchain. L'interface utilisateur développée en React et l'utilisation de TypeScript ont permis d'offrir une application robuste et maintenable, capable de s'adapter à l'évolution des besoins des utilisateurs.

En conclusion, ce projet a permis d’approfondir la compréhension des principes de la décentralisation et des NFTs, tout en mettant en lumière les opportunités uniques qu’offre la blockchain pour la création, la collection et l’échange d’objets numériques.