Rapport de projet : Collectible Card Game in the blockchain

Abdallah HABA (3812460)
Mohammad-Habib JAVAID (21307723)

Master Informatique
Sorbonne Université
27 octobre 2024

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Ce projet porte sur le développement d'un jeu de cartes à collectionner (TCG) implémenté sur la blockchain Ethereum et ses dérivés compatibles avec la Machine Virtuelle Ethereum (EVM).
L'accent est mis sur la dimension collection des cartes numériques, plutôt que sur la mécanique du jeu en elle-même.
L'objectif principal est de concevoir une infrastructure permettant aux utilisateurs de collectionner, d'échanger et de gérer des cartes numériques représentées sous forme de jetons non fongibles (NFTs), conformes à la norme ERC-721. Le système devra assurer la gestion de la publication régulière de nouvelles collections de cartes, à l'instar des TCG traditionnels qui éditent de nouvelles séries tous les trois à quatre mois. Le projet englobe le développement de composantes on-chain (contrats intelligents) et off-chain (interface utilisateur et serveur), incluant notamment l'intégration de l'API Pokémon TCG pour l'utilisation de cartes existantes.

1.2 Réalisations

Nous avons décidé, sur les conseils des professeurs, de réaliser le jeu sur Pokémon. Dans notre jeu, on a accès à une page **Extension** qui permet à l'admin d'ajouter des sets au jeu parmi tous les sets disponibles sur l'API de Pokemon TCG. Les non-admin ont aussi accès à cette page mais ne peuvent pas ajouter d'extensions.

Une page **Collection** permet de visualiser les cartes de l'utilisateur.

Ces cartes peuvent être obtenues de deux manières :

— soit en les achetant sur la market place

— soit en ouvrant des boosters, parmis

Une page **Utilisateur** est également présente pour visualiser les collections des autres joueurs.

Chapitre 2

In-Chain

2.1 Structure In-Chain

Deux contrats sont définis dans notre projet.

Le contrat main contient les fonctions principalement appelées par l'administrateur, le permettant d'effectuer des opérations sur le contrat Collection. C'est ce contrat qui va permettre de générer les NFTs, à partir d'un modèle, choisi aléatoirement ou non.
Le contrat Collection, quant à lui, contient tout le code de gestion des collections. Il permet à la fois de gérer les modèles, les cartes, et les possesseurs de ces cartes. En plus des différentes fonctions de gestion d'une collection, elle implémente également les fonctions de la norme ERC721, permettant, entre autres, le transfert de NFTs.
Notre projet contient également les contrats Ownable et ERC721, dans lesquelles certaines fonctions sont nécessaires au bon fonctionnement de notre projet. Le contrat ownable permet la gestion des droits de possession d'un utilisateur sur un contrat ou autre, et une fonctionnalité supplémentaire y a été définie : l'administration. En plus d'un possesseur de contrat, un administrateur a été défini avec une adresse fixée, et un modifier qui permet d'implémenter des fonctions dont il sera le seul à utiliser.

2.2 Les collections

Une collection est une structure qui enregistre un certain nombre limité de modèles de cartes, et qui à partir de ces modèles, permet la création de NFTs, c'est-à-dire d'exemplaires de ces mêmes cartes. Il est principalement caractérisé par un nom, correspondant au nom d'une extension, et un nombre de cartes maximum, correspondant au nombre de cartes de l'extension.

Le modèle de carte permet de faire le lien entre la collection et l'API du Trading Card Game (TCG) de Pokémon. Il est uniquement composé d'un identifiant sous forme de chaîne de caractères, correspondant à l'identifiant d'une carte dans l'API. Ainsi, via cette chaîne, et en se connectant à l'API, on peut récupérer toutes les informations sur la carte en question, telle que son image ou son nom.

Une carte, dans notre structure, correspond à un NFT. Elle est composée d'un identifiant unique, d'une adresse correspondant à son possesseur, et de l'identifiant du modèle auquel elle correspond. Ainsi, si un même modèle est unique, il peut en exister de nombreux exemplaires.

Les deux structures vues précédemment sont stockées dans des listes respectives, et un mappage permet d'établir le nombre de cartes que possède un utilisateur.

En plus des getters, des fonctions créatrices de cartes et de modèles, et des redéfinitions des fonctions d'ERC721, ce contrat prend en charge la génération d'une carte aléatoirement ainsi que l'affectation d'une carte à un utilisateur donné. La génération aléatoire se fait grâce aux fonctions `keccak256` et `abi.encodePacked` sur le timestamp d'un bloc, l'adresse de l'utilisateur qui appelle cette fonction, et un entier que l'on incrément dès que la fonction de génération aléatoire est appelée. Cela génère donc un entier, que l'on utilise pour obtenir un modèle dans la liste et ensuite générer une carte à partir de son identifiant.

2.3 Le programme principal

Enfin, le main permet une gestion extérieure des collections. En plus d'en créer, il est possible d'y effectuer une ouverture de booster et d'autres fonctions utilitaires pour le front y sont également définies.

Il prend en attribut un mappage, permettant de définir les différentes collections créées avec un identifiant, et un compteur indiquant le nombre courant de Collections. Un autre mappage y a également été défini, permettant un affichage des cartes obtenues sur le front lors de l'ouverture d'un booster.

Les fonctions d'affectations de cartes à un utilisateur permettent depuis un identifiant donné, ou aléatoire dans une collection donnée, de créer des cartes dans la collection en question. Deux fonctions de mint sont définies : une première fonction est définie pour l'ouverture d'un booster et ne coûte rien, et une deuxième, payante, a été définie pour l'achat d'une carte depuis la marketplace.

Un compte à rebours était initialement prévu entre deux ouvertures de booster mais n'a finalement pas pu être implémenté.

Enfin, certaines fonctions ne sont accessibles que par l'administrateur, notamment les fonctions de création de collection et de modèle de carte. Ainsi, dans notre interface, l'administrateur doit rendre une collection (ou extension) visible, et donc la créer, pour que les utilisateurs puissent ouvrir des boosters ou acheter des cartes.

Chapitre 3

Off-chain

Pour la partie off-chain, nous avons significativement utilisé Aider, une application en ligne de commande permettant d'assister dans un projet de développement (elle peut expliquer, déboguer, factoriser et générer du code).

Nous comprenons bien la partie permettant de communiquer avec la blockchain, qui se trouve dans `blockchain.ts`. Le code y est relativement simple, bien que nous ayons également été assistés par Aider.

La partie in-chain est entièrement réalisée manuellement, à l'exception de la fonction `getLastAssignedCards`. Nous avons également utilisé Aider pour déboguer certains aspects ponctuels.

`getLastAssignedCards` nous a été suggérée lors d'une session de débogage pour résoudre le problème lié à l'ouverture d'un booster. En effet, notre fonction d'ouverture de booster, `assignXRandomCardsToOwner`, fonctionne en deux étapes : 1. Elle ajoute les cartes à l'utilisateur. 2. Elle renvoie les cartes ajoutées pour qu'elles puissent être affichées côté front-end.

Le problème est qu'effectuer une écriture dans la blockchain requiert la méthode `send()`, tandis que pour récupérer une valeur, il faut utiliser la méthode `call()`.

Par conséquent, il est impossible de réaliser ces deux opérations dans un même appel à une méthode du contrat.

Aider nous a alors suggéré d'utiliser un mapping comme tampon. Lors de l'appel de `assignXRandomCardsToOwner` (appelé avec la méthode `send()`), toutes les cartes ajoutées sont stockées dans ce tampon. Ensuite, nous récupérons le contenu de ce tampon grâce à la fonction `getLastAssignedCards` (via un appel `call()`).

Toutes les discussions avec Aider, ainsi que tout nos prompts sont disponible dans les fichiers `.aider.chat.history.md` et `.aider.input.history`.

Chapitre 4

Conclusion

En conclusion, ce projet a permis de développer les bases d'un jeu de cartes à collectionner sur la blockchain, avec un accent particulier sur la gestion des collections et l'utilisation de la norme ERC-721 pour représenter les cartes sous forme de NFTs. Les fonctionnalités de base ont été implémentées avec succès, incluant la structure des contrats intelligents pour la gestion des collections et la création de cartes. L'implémentation de l'ouverture de boosters permet d'enrichir l'expérience utilisateur en intégrant un aspect aléatoire semblable à celui des TCG traditionnels.

Toutefois, des améliorations restent possibles pour renforcer l'application, telles que la mise en place du compte à rebours pour les ouvertures de boosters ou l'optimisation des interactions off-chain. Le projet, bien qu'encore en phase de développement, pose ainsi les fondations d'une infrastructure solide pour une gestion efficace des cartes numériques sur la blockchain.