

Compte rendu de projet DAAR
Collectible Card Game Décentralisé



MALEK BOUZARKOUNA — 28706508

YACINE KESSAL — 21311739

27 OCTOBRE 2024

Table des matières

| | | |
|----------|---|----------|
| 1 | Introduction | 1 |
| 2 | Architecture | 1 |
| 2.1 | Front-end | 1 |
| 2.2 | Smart Contracts | 1 |
| 2.2.1 | collection.sol | 1 |
| 2.2.2 | main.sol | 2 |
| 3 | Présentation Du DApp TCG | 3 |
| 3.1 | Collections et Cartes | 3 |
| 3.1.1 | Création et Publication des Cartes Administrateur | 3 |
| 3.1.2 | Visibilité des Collections par les Joueurs | 4 |
| 3.1.3 | Restrictions de l'Administrateur | 5 |
| 3.2 | Page Profil | 5 |
| 3.3 | Market | 6 |
| 3.4 | Booster | 7 |
| 4 | Conclusion | 8 |

1 Introduction

Les **TCG** (Trading Card Games), ou en français **Jeux de cartes à collectionner**, sont des jeux dans lesquels le joueur collectionne des cartes pour constituer un deck et jouer contre d'autres joueurs. Ces cartes peuvent s'obtenir en achetant des **packs** de collection, appelés aussi **boosters**, ou en les achetant auprès d'autres joueurs. Les TCG existent aujourd'hui sous plusieurs formats : physiques ou numériques. Les plus connus d'entre eux sont *Magic : The Gathering*, *Pokémon TCG*, *Yu-Gi-Oh !*, *Hearthstone*, *Marvel Snap*, ou encore *Legends of Runeterra*.

Dans ce projet, nous allons créer un **TCG** (Trading Card Game) sur le thème de la licence **Yu-Gi-Oh !**, basé sur la blockchain Ethereum autrement dis une **Dapp** (Decentralized applications). Pour ce faire, nous utilisons **React/TypeScript** pour la partie affichage, **Solidity** pour la logique de la blockchain, et **Hardhat** pour le déploiement et le test des smarts contrats.

Les cartes dans notre projet seront des NFT, utilisant la norme **ERC-721** pour assurer leur unicité et leur traçabilité. Bien que notre projet ne couvre pas toutes les fonctionnalités présentes dans un TCG classique, nous y intégrons les éléments principaux, que nous allons démontrer dans ce rapport.

2 Architecture

L'architecture de notre DApp TCG repose sur une structure composée de deux parties principales.

2.1 Front-end

Pour le front-end, nous avons utilisé **React** et **TypeScript** en association avec **Tailwind CSS**, ce qui nous permet de créer une interface utilisateur réactive et élégante, facilitant la navigation au sein de notre DApp. Ce choix technologique assure également une grande flexibilité dans le développement et l'esthétique de l'application.

Pour garantir une utilisation efficace de nos fonctionnalités, nous avons mis en place un **contexte**. Ce dernier facilite la gestion des états et des interactions au sein de l'application. Grâce à ce contexte, nous pouvons accéder facilement à diverses fonctions telles que la connexion au portefeuille, la création et la gestion des collections de cartes, ainsi que l'achat et la vente de cartes sur le marché. Cette approche nous permet d'organiser notre code de manière claire et de simplifier les échanges d'informations entre les différents composants de l'application.

2.2 Smart Contracts

En ce qui concerne les Smart Contracts, nous avons développé deux fichiers principaux : un fichier *main* et un fichier *collection*. Nous avons également intégré la bibliothèque **OpenZeppelin**, y compris le contrat **Ownable**, pour garantir la conformité avec la norme **ERC-721**. Cela nous permet de gérer efficacement la création, la gestion et la traçabilité des cartes uniques en tant que NFT sur la blockchain Ethereum.

2.2.1 collection.sol

Le fichier `Collection.sol` définit un contrat intelligent basé sur la norme ERC-721 pour la création d'une collection de cartes à collectionner en tant que jetons non fongibles (NFT). La structure principale est la suivante :

1. **Structure de la carte** : La structure `Card` contient les attributs essentiels d'une carte, tels que `tokenId`, `name`, `cardType`, `rarity`, `imageUrl`, `effect`, `attack`, `defense`, `quantity`, et `price`.
2. **Propriétés de la collection** : Le contrat maintient un tableau `CollectionCards` pour stocker toutes les cartes, ainsi que des mappings pour lier chaque carte à son propriétaire et à son index dans le tableau.
3. **Constructeur** : Le constructeur initialise le nom de la collection et le nombre maximum de cartes pouvant être créées.
4. **Fonctions principales** :
 - `addCard` : Permet à l'Owner d'ajouter une nouvelle carte à la collection, en s'assurant que le nombre de cartes n'excède pas la limite spécifiée.

- `purchaseCard` : Permet aux utilisateurs d'acheter des cartes, en vérifiant que l'acheteur n'achète pas sa propre carte.
- `openBooster` : Génère un nombre de cartes aléatoires clonées à partir de la collection existante et les attribue à un utilisateur, simulant ainsi l'ouverture d'un booster de cartes.

5. Fonctions d'information : Des fonctions comme `getCollectionCards`, `getCard`, et `getOwnerCard` permettent de récupérer des informations sur les cartes et les propriétaires.

Ce contrat intelligent utilise la bibliothèque OpenZeppelin pour garantir une mise en œuvre sécurisée et standardisée des fonctionnalités des NFT, facilitant ainsi la gestion des cartes et des interactions des utilisateurs.

2.2.2 main.sol

Le contrat `Main` est un contrat de gestion pour notre TCG. Il permet de créer et de gérer plusieurs collections de cartes. Voici une brève description de sa structure et de ses fonctionnalités :

Variables de classe :

- `count` : Compteur pour le nombre de collections créées.
- `cardCount` : Compteur total des cartes à travers toutes les collections.
- `collections` : Un mapping qui stocke chaque collection en fonction de son identifiant.
- `marketCards` : Un tableau qui contient les identifiants des cartes actuellement sur le marché.
- `cardToCollection` : Un mapping qui relie chaque carte à sa collection respective.

Fonctions principales :

1. `createCollection` : Permet à l'Owner de créer une nouvelle collection de cartes en spécifiant son nom et le nombre de cartes initiales.

2. `getOwnerCount` : Retourne le nombre total de cartes détenues par un utilisateur à travers toutes les collections.

3. `purchaseCard` : Permet à un utilisateur d'acheter une carte d'une collection donnée, en vérifiant que la carte est disponible et que le paiement est suffisant.

4. `addCardToCollection` : Fonction réservée à l'Owner pour ajouter une carte à une collection spécifique, en fournissant tous les détails nécessaires.

5. `getCardCollection` : Retourne toutes les cartes d'une collection spécifiée.

6. `getCollectionDetails` : Renvoie des informations sur une collection donnée, y compris son nom et le nombre de cartes.

7. `getMarketCards` : Retourne toutes les cartes qui sont actuellement sur le marché.

8. `buyCardFromMarket` : Permet à un utilisateur d'acheter une carte du marché, en vérifiant que le paiement est suffisant et en retirant la carte du marché après l'achat.

9. `openCollectionBooster` : Permet d'ouvrir un booster de cartes dans une collection, en émettant un événement lorsque de nouvelles cartes sont créées et ajoutées à l'utilisateur.

Événements : - `BoosterOpened` : Émis lors de l'ouverture d'un booster, indiquant à qui les nouvelles cartes sont attribuées et à quelle collection elles appartiennent.

Avant d'aller à la section suivante, il est important de noter que chaque action qui modifie la blockchain, telle que l'achat, la création ou la suppression d'éléments, nécessite l'accord explicite de l'utilisateur, matérialisé par une pop-up `MetaMask`. Cela garantit la sécurité et le contrôle des utilisateurs sur leurs transactions.

3 Présentation Du DApp TCG

Dans cette section, nous présentons un aperçu de notre DApp TCG pour illustrer ses différentes fonctionnalités et les sections principales du site. Le site comporte quatre sections principales : Collection, Market, Profile et Booster, comme montré à la Figure 1.



FIGURE 1 – Vue de la page d'accueil

3.1 Collections et Cartes

Dans ce projet, nous définissons un système de “collections” de cartes uniques, accessibles aux joueurs et administrées par un administrateur unique. Ce document présente les règles de création, de gestion et d'achat de ces cartes, ainsi que leurs interactions dans l'application.

3.1.1 Création et Publication des Cartes Administrateur

Les collections sont des ensembles de cartes uniques que seul l'administrateur, ici le Owner du contrat, peut créer et publier comme illustré à la Figure 2. Chaque collection a une limite de cartes, définie par l'administrateur.

Chaque carte ajoutée à une collection est mintée (créeée) de manière unique et associée directement au compte de l'administrateur, comme illustré à la Figure 3. Ces cartes sont donc non transférables à l'administrateur après création, et leur nombre total ne peut dépasser la limite établie.

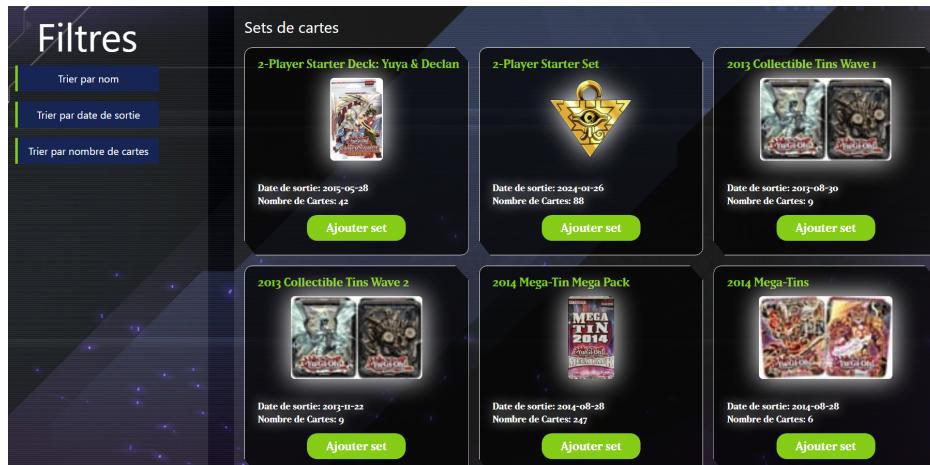


FIGURE 2 – Collection vu admin

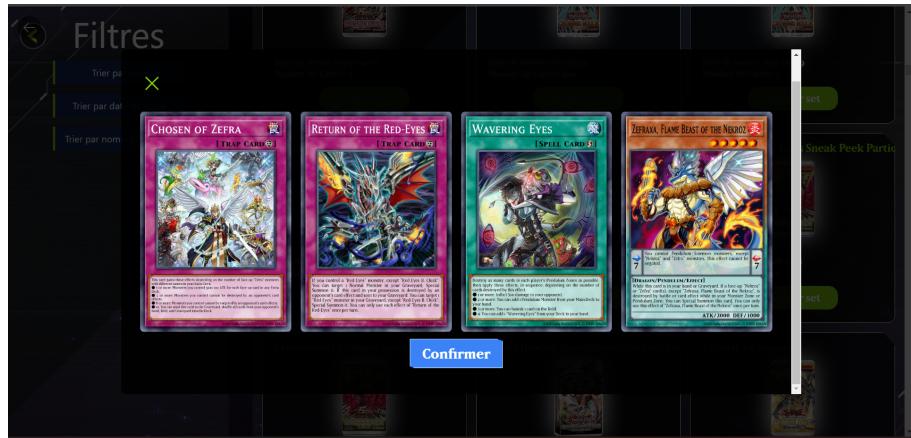


FIGURE 3 – Processus de minting des cartes uniques par l'administrateur

3.1.2 Visibilité des Collections par les Joueurs

Les collections créées sont visibles aux autres joueurs comme illustré à la Figure 4 et qui peuvent les acheter pour les posséder sur leurs comptes comme illustré à la Figure 5. Ce système permet aux joueurs de consulter les cartes disponibles et de décider de leurs achats.

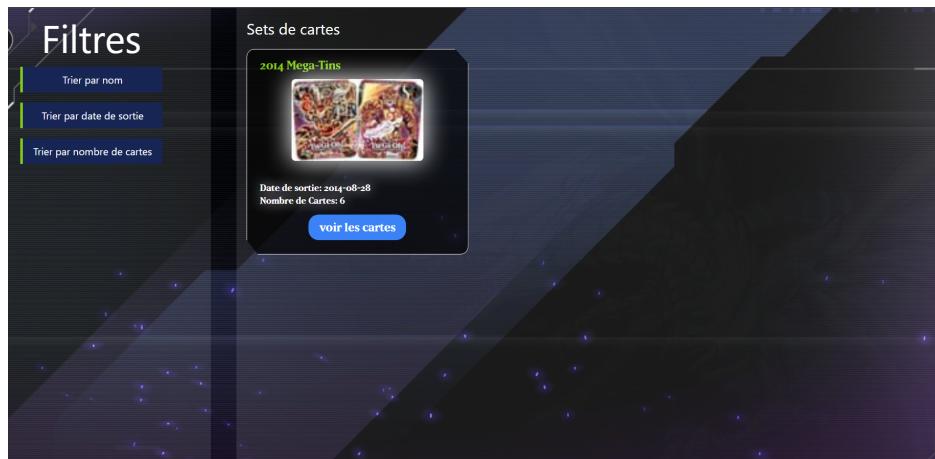


FIGURE 4 – Vue des collections par les joueurs



FIGURE 5 – Processus d'achat des cartes par les joueurs

3.1.3 Restrictions de l'Administrateur

L'administrateur ne peut pas acheter des cartes de sa propre collection et ne peut ajouter un nombre de cartes supérieur à la limite indiquée lors de la création. Ces restrictions garantissent un équilibre dans le contrôle des cartes.

3.2 Page Profil

La page profil de notre DApp fournit aux utilisateurs une interface personnalisée pour gérer leurs informations et leur collection de cartes. Elle affiche des éléments clés, tels que le nombre total de cartes possédées et la liste de leurs cartes, comme illustré à la Figure 6. Les utilisateurs peuvent également visualiser des détails spécifiques de chaque carte (Figure 7) et effectuer des transactions sur le marché (Figure 8). Cette section offre une gestion efficace et intuitive de leur collection, enrichissant ainsi l'expérience utilisateur.



FIGURE 6 – Page de profil : vue d'ensemble des collections de l'utilisateur



FIGURE 7 – Détails d'une carte spécifique

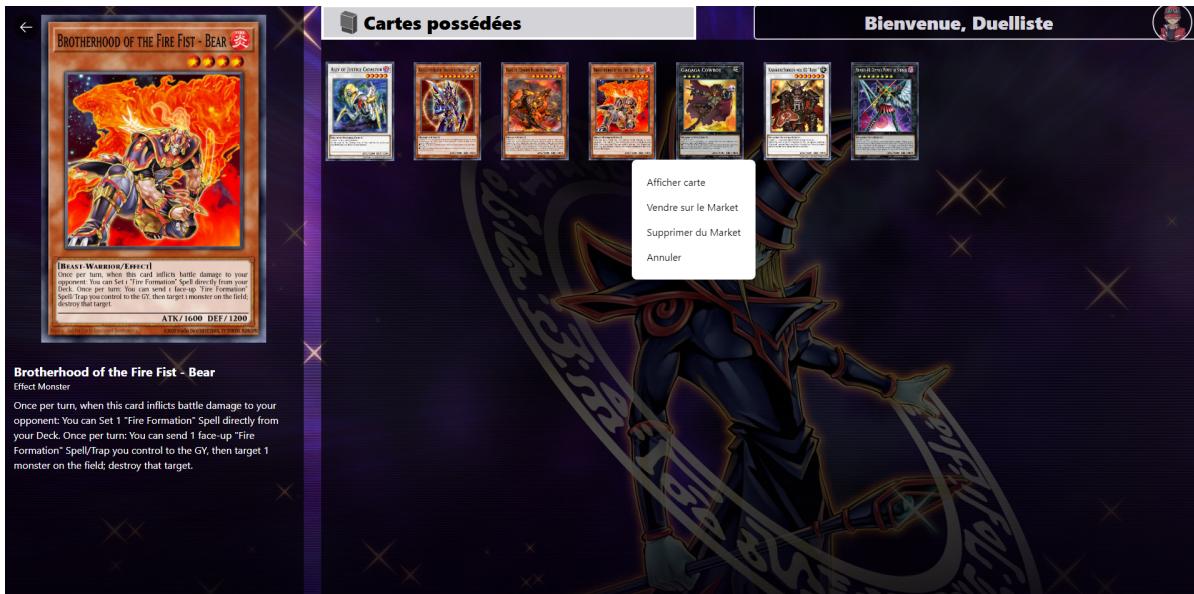


FIGURE 8 – Boutons pour mettre en vente et retirer des cartes

3.3 Market

Les joueurs de notre DApp auront la possibilité de mettre en vente , de retirer et d'acheter des cartes sur le marché (Figure 10). Ce dernier constitue une plateforme où ils peuvent acquérir des cartes qui ne sont pas disponibles dans leur collection personnelle (Figure 9).

Grâce à cette fonctionnalité, les utilisateurs peuvent enrichir leur expérience en accédant à une variété de cartes, ce qui leur permet de personnaliser davantage leur collection. Le marché favorise également l'interaction entre les joueurs, créant ainsi une communauté dynamique et engagée autour de notre DApp.

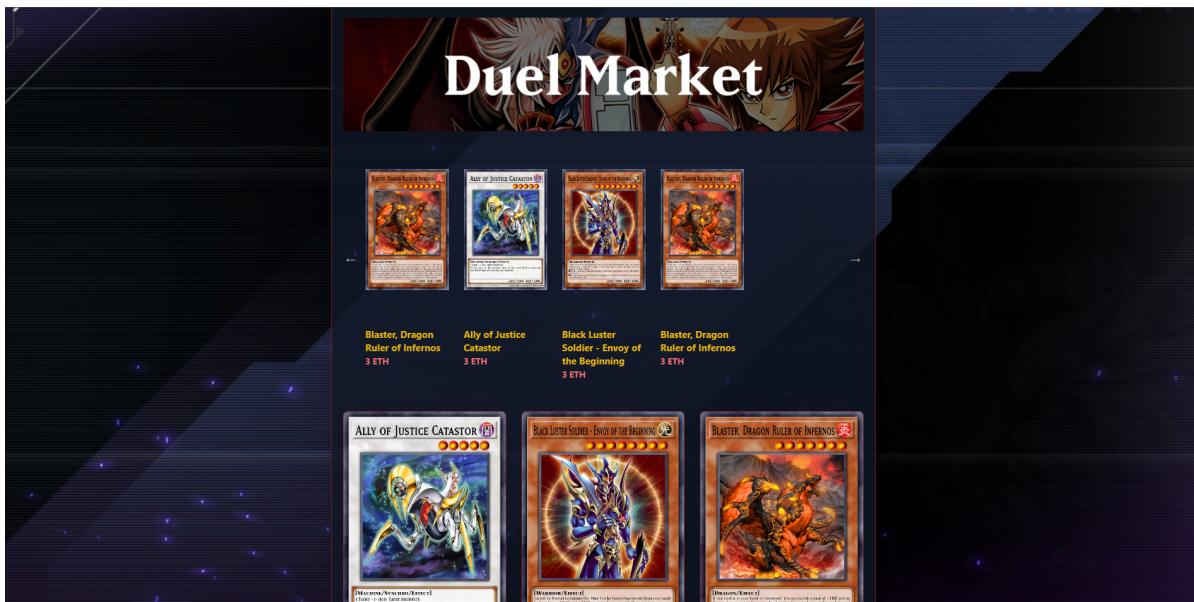


FIGURE 9 – Page market : vue d'ensemble du market

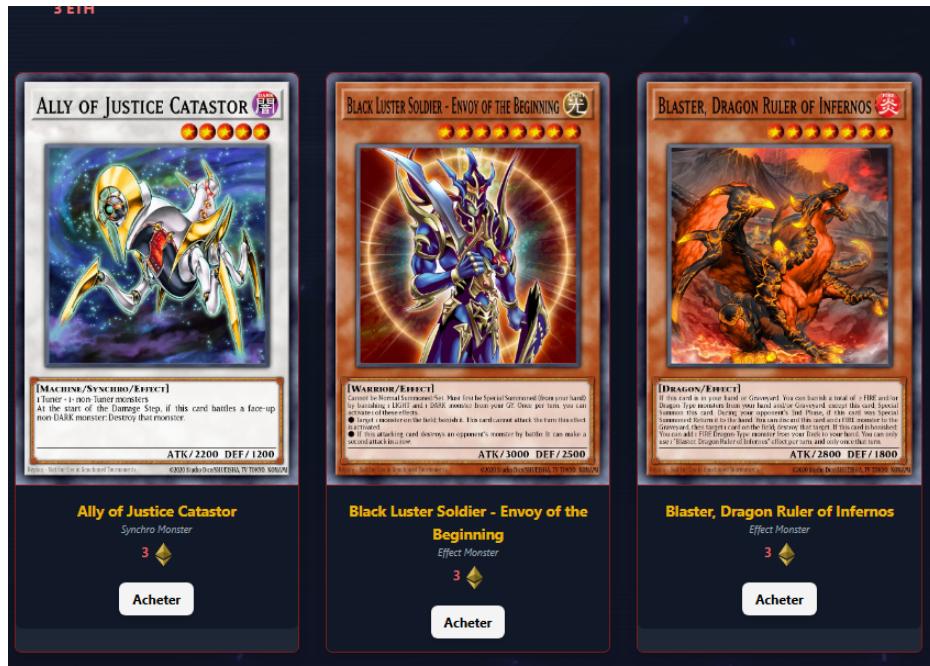


FIGURE 10 – Page market : vue d'ensemble du market

3.4 Booster

L'interface booster, illustrée dans la Figure 11, permet aux utilisateurs de sélectionner les paquets qu'ils souhaitent ouvrir. Grâce à cette fonctionnalité, ils peuvent acquérir des cartes qui ne sont plus disponibles à l'achat dans la collection et qui n'ont pas été mises en vente par d'autres joueurs. Cela crée une dynamique de collection et encourage les utilisateurs à explorer différentes stratégies de jeu.

Après avoir ouvert un paquet, les joueurs découvrent les cartes qu'ils ont obtenues. Comme montré dans la Figure 12.

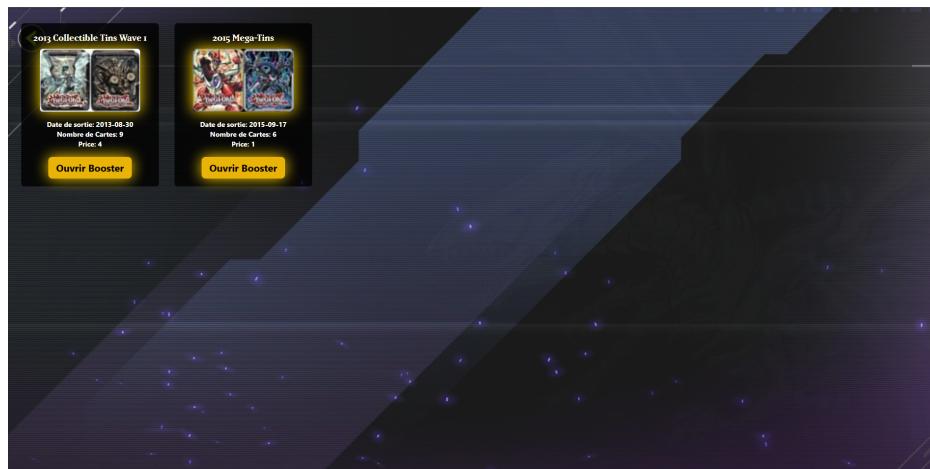


FIGURE 11 – Interface pour ouvrir des paquets de cartes

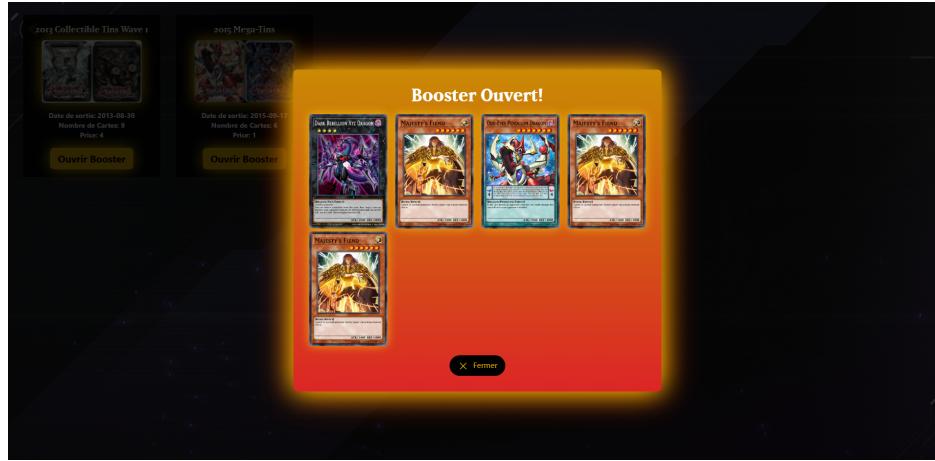


FIGURE 12 – Cartes obtenues après ouverture d'un paquet

4 Conclusion

En conclusion, ce projet de TCG sur la blockchain Ethereum nous a permis de créer un système complet comprenant les éléments essentiels d'un jeu de cartes à collectionner : des collections, un market pour l'échange de cartes, des boosters, ainsi qu'un système de profil pour chaque utilisateur.

Ce projet a été l'occasion d'apprendre et de maîtriser de nouvelles technologies spécifiques au développement blockchain, notamment Solidity pour écrire les smart contracts, ainsi que des normes comme l'ERC-721, qui garantit l'unicité et la traçabilité des cartes en tant que NFT. Nous avons également utilisé Hardhat pour le déploiement et les tests de nos smart contracts, et MetaMask pour gérer les interactions des utilisateurs avec la blockchain.

Ainsi, bien que le projet ne couvre pas toutes les fonctionnalités d'un TCG traditionnel, il représente une base solide pour une DApp innovante et extensible, avec des perspectives intéressantes pour ajouter des éléments de gameplay et de nouvelles options d'interaction blockchain.