Monopoly Dapp

Documentation Technique Complète

5BLOC Projet Web3 - Développement d'une DApp basée sur la Blockchain

Par Jérôme Philippe et Antonin Montagne

Table des matières

Monopoly Dapp 1
1. Présentation du projet 3
Stack technique 3
2. Cas d'usage et justification 3
3. Règles du jeu 4
Inscription : 4
Les propriétés : 4
Limite de possession 4
Système d'échange 4
4. Architecture technique 5
5. Smarts Contracts 5
MonopolyToken.sol 5
PropertyNFT.sol 5
GameManager.sol 6
6. Stockage IPFS 7
7. Frontend 7
8. Tests 8
9. Déploiement 8
Commandes 9
10. Difficultés rencontrées 9
11. Conclusion 10

1. Présentation du projet

Notre projet est une version décentralisée du célèbre jeu de société Monopoly, déployée sur la blockchain Ethereum (testnet Sepolia). Les joueurs peuvent s'inscrire, acheter des propriétés et les échanger entre eux, le tout sans serveur central.

L'objectif était de créer une DApp fonctionnelle qui démontre les avantages de la blockchain pour la gestion d'actifs numériques et les interactions entre utilisateurs.

Stack technique

Composant	Technologie
Blockchain	Ethereum (Sepolia Testnet)
Smart Contracts	Solidity 0.8.28
Framework	Hardhat
Librairies	OpenZeppelin Contracts 5.0
Frontend	React 18 + ethers.js 6
Stockage	IPFS (Pinata)
Wallet	MetaMask

2. Cas d’usage et justification

Nous nous sommes demandé : qu’apporte réellement la blockchain à un jeu comme le Monopoly ?

Dans un jeu de société numérique classique, tout repose sur un serveur central : c’est lui qui détermine quelles propriétés appartiennent à quel joueur, qui valide les transactions et qui applique les règles. Avec la blockchain, cette logique change complètement.

Pourquoi utiliser la blockchain ?

Propriété vérifiable : au lieu qu’un serveur déclare « tu possèdes cette propriété », c’est la blockchain elle-même qui en atteste. Cette approche garantit que personne ne puisse tricher, modifier ou falsifier les données.

Transparence : toutes les transactions sont publiques et consultables. Par exemple, sur Etherscan, on peut vérifier à tout moment qui détient quelle propriété, qui l’a vendue et à quel prix.

Persistance : même si l’interface du jeu disparaît, les smart contracts restent en ligne sur la blockchain. Ainsi, toute personne peut recréer une nouvelle interface et continuer à interagir avec le système sans que rien ne soit perdu.

3. Règles du jeu

Inscription :

Quand un joueur s'inscrit via la fonction registerPlayer(), il reçoit automatiquement 1500 MONO. Un joueur ne peut s'inscrire qu'une seule fois.

Les propriétés :

Le jeu contient 28 propriétés représentées sous forme de NFTs :

Type	Couleur	Exemple	Valeur
STREET_BROWN	Marron	Boulevard de Belleville	60 MONO
STREET_LIGHTBLUE	Bleu clair	Rue de Vaugirard	100 MONO
STREET_PINK	Rose	Boulevard de la Villette	140 MONO
STREET_ORANGE	Orange	Avenue Mozart	180 MONO
STREET_RED	Rouge	Avenue Matignon	220 MONO
STREET_YELLOW	Jaune	Faubourg Saint-Honoré	260 MONO
STREET_GREEN	Vert	Avenue de Breteuil	300 MONO
STREET_DARKBLUE	Bleu foncé	Rue de la Paix	400 MONO
STATION	Noir	Gare de Lyon	200 MONO
UTILITY	Spécial	Compagnie des eaux	150 MONO

Limite de possession

Chaque joueur peut posséder maximum 4 propriétés. Cette limite encourage les échanges et empêche un joueur de tout monopoliser. L'owner (la banque) est exempté de cette limite pour pouvoir détenir toutes les propriétés au départ.

Cooldown (5 minutes) : Délai obligatoire entre chaque transaction pour éviter le spam.

Lock (10 minutes) : Après un achat de propriété, le joueur est bloqué pendant 10 minutes. Cela empêche la spéculation rapide.

Après un achat, nous avons un cooldown ainsi qu'un lock.

Après avoir créer une offre d'échange, nous avons un cooldown puis un lock jusqu'à la vérification de l'échange.

Après avoir accepté une offre, nous avons les mêmes contraintes que pour un achat.

Système d'échange

Les joueurs peuvent échanger des propriétés entre eux :

Le vendeur crée une offre en spécifiant le destinataire, la propriété et le prix
L'acheteur voit l'offre et peut l'accepter
Le vendeur peut annuler son offre à tout moment

4. Architecture technique

Notre DApp repose sur trois contrats interconnectés :

MonopolyToken : gère la monnaie du jeu (MONO). C'est un token ERC-20 classique qui permet aux joueurs d'acheter des propriétés et de payer des loyers.

PropertyNFT : représente les propriétés sous forme de NFTs (ERC-721). Chaque propriété est unique et possède ses propres caractéristiques.

GameManager orchestre les interactions entre les deux autres contrats. Il implémente les règles du jeu.

Contrats déployés (Sepolia)

Contrat	Adresse
MonopolyToken	0xd657296ad0960d0E87C3305d52D0c3e7b856db28
PropertyNFT	0x2a3CcDEf8994Ec2Fe27625dedAE15095525e893c
GameManager	0x433B688AE86339f364f623696e148cFaE64dfd0B

5. Smarts Contracts

MonopolyToken.sol

Ce contrat hérite d'ERC20, ERC20Burnable et Ownable d'OpenZeppelin.

Quand un joueur s'inscrit via le GameManager, il reçoit automatiquement 1500 MONO. Ce montant est défini dans la constante INITIAL_PLAYER_BALANCE.

Fonctions principales :

- registerPlayer(address) : Inscrit un joueur et lui donne 1500 MONO
- mint(address, uint256) : Crée de nouveaux tokens
- setGameManager(address) : Définit l'adresse du GameManager

PropertyNFT.sol

Ce contrat combine ERC721, ERC721URIStorage et ERC721Enumerable.

Chaque propriété possède une structure qui stocke le nom, le type, la valeur, le loyer, les timestamps et la liste des anciens propriétaires.

À chaque transfert, l'ancien propriétaire est automatiquement ajouté à la liste previousOwners.

Fonctions principales :

- mintProperty(...) : Crée une nouvelle propriété NFT
- getProperty(uint256) : Retourne les infos d'une propriété
- getPreviousOwners(uint256) : Liste des anciens propriétaires
- canReceiveProperty(address) : Vérifie la limite de 4

GameManager.sol

C'est le cœur du jeu. Il définit les contraintes temporelles et orchestre les interactions.

Fonctions principales :

- registerPlayer() : Inscription d'un joueur
- buyPropertyFromBank(uint256, uint256) : Achat à la banque
- createTradeOffer(address, uint256, uint256) : Créer une offre
- acceptTradeOffer(uint256) : Accepter une offre
- cancelTradeOffer(uint256) : Annuler une offre

Les contrats disposent d’erreur personnalisées.

Erreur	Description
PlayerAlreadyRegistered	Joueur déjà inscrit
MaxPropertiesReached	Limite de 4 propriétés atteinte
CooldownActive	Cooldown en cours
LockActive	Lock en cours
InsufficientBalance	Balance MONO insuffisante
PropertyNotAvailable	Propriété non disponible
TradeOfferNotFound	Offre inexistante

6. Stockage IPFS

Les métadonnées des NFTs sont hébergées sur IPFS via Pinata. Le NFT contient juste un lien vers ces données. C'est le standard pour les NFTs car stocker des images on-chain coûterait trop cher.

Pinata est configuré avec un CID (bafybeicuhnu5e3evh6acwzizymp675sfato5mmkoe7d6kq2pkkapem4dji) qui nous permet d'interagir avec.

Les métadonnées ont le format suivant :


```
{
 "name": "Boulevard de Belleville",
 "description": "Propriete Monopoly: Boulevard de Belleville",
 "image": "ipfs://CID/belleville.png",
 "attributes": [
  { "trait_type": "Type", "value": "STREET_BROWN" },
  { "trait_type": "Valeur", "value": 60 },
  { "trait_type": "Loyer de base", "value": 2 }
 ],
 "properties": {
  "type": "STREET_BROWN",
  "value": 60,
  "rent": 2,
  "previousOwners": [],
  "createdAt": 1704067200,
  "lastTransferAt": 1704067200
 }
}
```


7. Frontend

L'interface utilisateur est développée en React avec ethers.js pour interagir avec la blockchain.

Composants principaux

- Header : Connexion MetaMask, affichage balance ETH
- PlayerInfo : Balance MONO, nombre de propriétés, cooldowns
- PropertyList : Liste des 28 propriétés avec filtres
- PropertyCard : Carte individuelle d'une propriété

- TradeOffers : Offres d'échange reçues et envoyées
- TradeModal : Formulaire de création d'offre

Un hook personnalisé centralise toutes les interactions avec les smart contracts : inscription, achat, échange, lecture des données.

8. Tests

Nous avons écrit 37 tests unitaires avec Hardhat qui couvrent les cas nominaux et les cas d'erreur.

Il y a 18 tests pour le contrat MonopolyToken et 19 pour GameManager.

Les catégories testées sont les suivantes :

- Déploiement des contrats
- Inscription des joueurs
- Mint des propriétés
- Achats de propriétés
- Système d'échange
- Cooldowns et locks
- Limite de 4 propriétés
- Cas d'erreur (balance insuffisante, non autorisé, etc.)

Pour lancer tous les tests, il faut utiliser la commande :

Npx hardhat tedt

9. Déploiement

Le déploiement se fait en plusieurs étapes :

1. Déployer les trois contrats avec deploy.js
2. Configurer les permissions (setGameManager sur chaque contrat)
3. Mint les 28 propriétés avec mint-properties.js
4. Approuver le GameManager pour les transferts avec approve-nfts.js

Commandes

Pour compiler les contrats :
npx hardhat compile

Pour déployer les contrats :
npx hardhat run scripts/deploy.js --network sepolia

Pour initialiser les propriétés :
npx hardhat run scripts/mint-properties.js --network sepolia

Pour approuver les NFTs :
npx hardhat run scripts/approve-nfts.js --network sepolia

10. Difficultés rencontrées

Le problème du owner :

Au début, on avait la même limite de 4 propriétés pour tout le monde, y compris l'owner. Résultat : on ne pouvait initialiser que 4 propriétés sur 28. Il a fallu modifier le contrat pour exempter l'owner de cette limite.

L'approbation des NFTs :

Quand un joueur achète une propriété, le GameManager doit transférer le NFT de l'owner vers le joueur. Mais pour ça, l'owner doit d'abord autoriser le GameManager à manipuler ses NFTs. On a créé un script approve-nfts.js pour ça.

La synchronisation frontend/blockchain :

Le frontend ne sait pas instantanément ce qui se passe sur la blockchain. Après un achat, il faut recharger les données. On a ajouté des appels pour rafraîchir l'état après chaque transaction.

RPC timeout :

Le RPC public de Sepolia était trop lent et causait des timeouts. On a dû utiliser Alchemy pour avoir un RPC plus fiable.

11. Conclusion

Ce projet nous a permis de comprendre concrètement le développement Web3. On a vu comment les smart contracts interagissent, comment gérer les tokens et NFTs et comment connecter tout ça à une interface utilisateur.

La blockchain apporte vraiment quelque chose pour ce type d'application : la certitude que personne ne triche, que les règles sont appliquées automatiquement, et que les possessions des joueurs sont incontestables.

Avec plus de temps, on aurait aimé :

- Ajouter un système de loyer automatique
- Implémenter les maisons et hôtels
- Créer un système de partie avec un vrai plateau
- Ajouter des événements temps réel avec des websockets