



DApp Web3 de trading de ressources tokenisées (NFT)

- ResourceSwap -



Hauteurs :

Yousfi Mohamed Yassine
Mohamed Laachir

Table des matières

1. Introduction	1
1.1 Contexte du projet	1
1.2 Cas d'usage choisi	1
1.3 Acteurs du système	1
1.4 Scénario d'utilisation typique	1
2. Justification de l'utilisation de la blockchain.....	2
3. Architecture générale du projet.....	2
1.1 Stack technique	2
1.2 Structure du dépôt	2
4. Smart Contract – Conception et choix techniques.....	3
4.1 Standard NFT et sécurité.....	3
4.2 Métadonnées on-chain	3
4.3 Contraintes métiers implémentées.....	3
4.4 Système d'échange	3
5. IPFS – Stockage décentralisé.....	4
6. Tests unitaires avec Hardhat	4
6.1 Organisation des tests	4
6.2 Cas testés	4
6.3 Résultats	5
7. Captures d'écran — Démonstration des règles métiers & gestion des erreurs	6
8. Frontend DApp.....	8
9. Respect des exigences du sujet	10
10. Conclusion	10

1. Introduction

1.1 Contexte du projet

Le projet **ResourceSwap** est une application décentralisée (DApp) basée sur la blockchain Ethereum.

Son objectif est de tokeniser des ressources numériques sous forme de NFT et de permettre leur échange entre utilisateurs via un système d'offres, tout en appliquant des contraintes métiers stricts.

Le projet s'inscrit dans un contexte pédagogique visant à démontrer :

- La compréhension des concepts Web3,
- La capacité à concevoir des smart contracts sécurisés,
- La mise en place de règles métiers non contournables,
- L'utilisation de tests unitaires et de stockage décentralisé.

1.2 Cas d'usage choisi

Le cas d'usage retenu est un système d'échange de ressources numériques, inspiré de mécanismes de jeux de collection ou de type *Monopoly*.

Chaque ressource:

- Est unique,
- Possède un niveau (*tier*),
- Appartient à un utilisateur identifié par son adresse blockchain,
- Peut-être échangée contre une autre ressource via un smart contract.

Contrairement à un simple transfert, l'échange est bilatéral et contrôlé, ce qui garantit :

- La propriété effective des actifs,
- La transparence,
- L'impossibilité de fraude.

1.3 Acteurs du système

Utilisateur :

Peut mint des ressources, consulter ses NFT, créer des offres et accepter des échanges.

Smart Contract ResourceSwap :

Garantit l'application des règles métiers et la sécurité des échanges.

Réseau Blockchain (Ethereum local – Hardhat) :

Assure l'exécution et l'immuabilité des transactions.

IPFS :

Stocke les métadonnées et images des ressources de manière décentralisée.

1.4 Scénario d'utilisation typique

1. Un utilisateur connecte son (MetaMask).
2. Il crée (*mint*) une ressource NFT avec :

- Un nom,
 - Un type,
 - Un niveau (tier),
 - Une valeur,
 - Un lien IPFS vers les métadonnées.
3. Il crée une offre d'échange :
Je propose mon NFT « A » contre ton NFT « B ».
 4. Un autre utilisateur accepte l'offre.
 5. Le smart contract:
 - Vérifie les règles,
 - Echange les propriétaires des tokens,
 - Applique un verrou temporel (*lock*).

2. Justification de l'utilisation de la blockchain

La blockchain est pertinente dans ce projet pour plusieurs raisons :

- **Propriété traçable** : chaque ressource appartient à une adresse unique.
- **Échanges « trustless »** : aucun tiers de confiance n'est nécessaire.
- **Historique immuable** : les transferts sont enregistrés on-chain.
- **Règles métiers non contournables**:appliquées directement dans le « smart contract ».
- **Décentralisation des données** : métadonnées stockées sur IPFS.

3. Architecture générale du projet

1.1 Stack technique

Blockchain / Smart Contract

- Ethereum (réseau local Hardhat)
- Solidity
- OpenZeppelin ERC721URIStorage
- ReentrancyGuard

Développement & tests

- Hardhat v3
- Mocha / Chai
- TypeScript

Stockage décentralisé

- IPFS (URI ipfs://)

Frontend

- React
- TypeScript
- Vite
- ethers.js v6
- MetaMask

1.2 Structure du dépôt

BLOCKCHAIN:

frontend/
resourceswap/
docs/

4. Smart Contract – Conception et choix techniques

4.1 Standard NFT et sécurité

Le contrat principal **ResourceSwap.sol** est basé sur :

- **ERC721URIStorage** : permet de lier chaque token à un URI IPFS.
- **ReentrancyGuard** : protège contre les attaques de réentrance.

4.2 Métadonnées on-chain

Chaque ressource NFT possède les champs suivants :

- name: nom de la ressource
- rtype: type de ressource
- tier: niveau (1 à 4)
- value: valeur libre
- ipfsUri: lien vers IPFS
- previousOwners[]: historique simplifié
- createdAt: timestamp de création
- lastTransferAt: timestamp du dernier transfert

4.3 Contraintes métiers implémentées

Contrainte	Description
MAX OWNED = 4	Un utilisateur ne peut pas posséder plus de 4 ressources
COOLDOWN = 5 min	Délai minimum entre deux actions
LOCK DURATION = 10 min	Verrou après action critique

4.4 Système d'échange

Le mécanisme d'échange fonctionne comme un mini-marketplace :

- CreateOffer (offeredTokenId, requestedTokenId)
- AcceptOffer(offerId)
- CancelOffer(offerId)

Avant toute action, le contrat vérifie :

- L'existence des tokens,
- La propriété réelle,
- Les approvals ERC721,
- Les contraintes temporelles.

5. IPFS – Stockage décentralisé

Les métadonnées des NFT sont stockées sur IPFS sous forme de JSON :

- Le tokenURI pointe vers ipfs://...
- Le frontend convertit l'URI via une gateway HTTP

Cela garantit:

- La décentralisation,
- L'intégrité des données,
- La conformité aux exigences du sujet.

6. Tests unitaires avec Hardhat

6.1 Organisation des tests

Deux fichiers principaux:

- ResourceSwap.mint.time.test.ts
- ResourceSwap.offers.cancel.test.ts

6.2 Cas testés

Les tests couvrent:

- Mint valide
- Dépassement de MAX_OWNED
- Cooldown non respecté
- Lock actif
- Création d'offre invalide
- Acceptation invalide
- Échange réussi
- Annulation d'offre
- Mise à jour des timestamps
- Sécurité des permissions

```

● PS C:\Users\Laachir\Desktop\ECOLE\MASTER2\5BLOC\EVAL_V3\blockchain\resourceswap> npx hardhat test

Compiled 1 Solidity file with solc 0.8.28 (evm target: cancan)
No Solidity tests to compile

Running Solidity tests

Running Mocha tests

ResourceSwap - Mint, Time rules and Limits
✓ mintResource: mint succeeds with correct tokenURI and metadata (987ms)
✓ Cooldown: createOffer twice without delay reverts with cooldown error (104ms)
✓ Lock: createOffer immediately after mint reverts with lock error (70ms)
✓ After lock duration, mint becomes possible again (71ms)
✓ MAX_OWNED: allows 4 mints and reverts on the 5th (94ms)
✓ tokensOfOwner returns all owned token IDs (84ms)
✓ Metadata timestamps: createdAt is less than or equal to lastTransferAt on mint (62ms)

ResourceSwap - Offers / Accept / Cancel
✓ 1) createOffer succeeds after lock and creates an active offer with correct fields (89ms)
✓ 2) createOffer fails if offered token does not exist (78ms)
✓ 3) createOffer fails if requested token does not exist (75ms)
✓ 4) createOffer fails if caller is not owner of offered token (72ms)
✓ 5) createOffer fails if offered token is not approved (77ms)
✓ 6) acceptOffer fails if offer is inactive (92ms)
✓ 7) acceptOffer fails if acceptor is not owner of requested token (81ms)
✓ 8) acceptOffer fails if requested token is not approved by acceptor (87ms)
✓ 9) acceptOffer succeeds when both sides approved and swaps owners; offer becomes inactive (99ms)
✓ 10) cancelOffer can only be called by the offerer (86ms)
✓ 11) cooldown: createOffer then immediate createOffer fails (81ms)
✓ 12) lock: after acceptOffer, the user is locked and immediate action fails (97ms)

19 passing (2s)

19 passing (19 mocha)

```

Saved html report to C:\Users\Laachir\Desktop\ECOLE\MASTER2\5BLOC\EVAL_V3\BLOCKCHAIN\resourceswap\coverage\html			
Coverage Report			
File Path	Line %	Statement %	Uncovered Lines
contracts\ResourceSwap.sol	98.75	96.05	156
Total	98.75	96.05	

6.3 Résultats

19 tests passants

Coverage

- Lines ≈ 98.75 %
- Statements ≈ 96.05 %

7. Captures d'écran — Démonstration des règles métiers & gestion des erreurs

Remarque importante : les clés privées affichées par hardhat node sont uniquement destinées au réseau local de développement. Pour un dépôt public GitHub, il est recommandé de flouter/couper la partie "Private key" dans les screenshots.

Capture 01 — Hardhat Clean + Local Node

```
PS C:\Users\Laahir\Desktop\ECOLE\MASTER2\5BLOC\EVAL_V3\BLOCKCHAIN\resourceswap> npx hardhat clean
1
2 PS C:\Users\Laahir\Desktop\ECOLE\MASTER2\5BLOC\EVAL_V3\BLOCKCHAIN\resourceswap> npx hardhat node
3 Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
4
5 Accounts
6 =====
7
8 WARNING: Funds sent on live network to accounts with publicly known private keys WILL BE LOST.
9
10 Account #0: 0xf39Fd6e51aad88f6F4c6a8027279cfffb2266 (10000 ETH)
11 Private Key: 0xaca97abec39a17e35ba1a0c7d81b5a0ed17d7c98
12
13 Account #1: 0x78057970c5181dc3a01c7d81b5a0ed17d7c98 (10000 ETH)
14 Private Key: 0x59c609e599f97a5100a44966f109a5390d986ca88c7a41f4683b67869d
15
16 Account #2: 0x3c44cdd5a5e98fa2b585d299e05d1d12f8a2930c (10000 ETH)
17 Private Key: 0x5de411fa1a4b049088f83103ebf1706367c2e68ca870f37b9a084cdab365a
18
19 Account #3: 0x0f07f9f6eeb2c4f870365e78592aef1d01e93b06 (10000 ETH)
20 Private Key: 0x7a5211829451c653712a810590f419314751b58f695c71c15141b007a6
21
22 Account #4: 0x15034af542579b7c7367359aa721000262605 (10000 ETH)
23 Private Key: 0x47e179ec397488593b1b780a000bbd91f1990b01f8773639f59c30a3492a
24
25 Account #5: 0x9965587d5a5bc2695c58ba16f17d81b9baadc (10000 ETH)
26 Private Key: 0xb8b3a390c5c34c9194ca85629a2df0e3c153bae03185e2d33a8e872092edffba
27
28 Account #6: 0x976ea74026e726554db57fa5473abdec3a8aa9 (10000 ETH)
29 Private Key: 0x2d02b1e4043b3fe3d4f233f83df3a3d7096f21ca900d6d688d8b2b4ec1564e
30
31 Account #7: 0x46d7964d2e09230903d4c7e9223099955 (10000 ETH)
32 Private Key: 0xabb0f95c53377467f5d6f804722181b2b0087f24d91560f1td07cbf94356
33
34 Account #8: 0x23618081c3f5c67754c3d557fc90fb5h216f (10000 ETH)
35 Private Key: 0xbd8a1121080551c9659393292598aa3472b22Fee921c0cf5d20bea7b97
36
37 Account #9: 0xaeae7a142d27c1f16714e4ab4f7612f0a972b (10000 ETH)
38 Private Key: 0x2a971d0798f97d79948a013d429367a7bf4cc922c825d33c1c17073dff6d409c6
39
40 Account #10: 0xbcd0d4d2a49d1d4f35081cbb24a51f3f05409 (10000 ETH)
41 Private Key: 0x71214f2b2d398c06e784c172540f9880106433832709722248c01628897
42
43 Account #11: 0x71b063ff3384f5f0b9995808a00d2f02426e5788 (10000 ETH)
44 Private Key: 0x701b635b0df90e65240c28hd21bbad99645a3d5d7e013b2bdff6f192c82
45
46 Account #12: 0xfab0ba5c6d6b0b445f7357272f2b20c25651694a (10000 ETH)
47 Private Key: 0xa267530f49f828208e0ef313e7a6f6827f2a80ce2897751d68a43f649467b1
48
49 Account #13: 0x1cb3b2770994a6e8f0157ab0847c37264073c96c (10000 ETH)
50 Private Key: 0x47c99abe3324a2797c28ffff126745918cc3f37208a892e08065d2942dd
51
52 Account #14: 0xdf3e18d64bc6a983f673ab319cca4ef1a57c7097 (10000 ETH)
53 Private Key: 0x526e9595f44d8f495158b0884d9123899f6f12e5f13606bb0789009aaa
54
55 Account #15: 0x6cd3b766c56d5a72141f452c558c0353964cc71 (100000 ETH)
56 Private Key: 0xb166f246dabd4a521a360cab06c5d2b0e46670292d85c875ee20e84Ffb61
57
58 Account #16: 0x2546bc1d84621e976d181a591a022aa77eccc (100000 ETH)
59 Private Key: 0xaea44ac03b1f858b476bba4e71648203e1b8a97e276d1beec7c37d2484a0
60
61 Account #17: 0xbda5747b1d6f808de54cb465e807d40e1b197e (100000 ETH)
62 Private Key: 0x8959f6fca8c651a91d28702527f3zf2f9f601074a0861667b5a93c037fd
63
64 Account #18: 0xdd1fd6581271c39360230f93765c04130f144c9 (100000 ETH)
65 Private Key: 0xdea9e5858a4475276426320d9e928ecFcba480ffa5c360fa6c4c28beeee
66
67 Account #19: 0xb802f694ab2cb8938ef4942d1f5c5e1199 (100000 ETH)
68 Private Key: 0xdf57089f6bbacf7b80b8c227dfbf8a9fc08a93fdc688e1e2411a4efc723656e
69
70 WARNING: Funds sent on live network to accounts with publicly known private keys WILL BE LOST.
```

Cette étape permet de repartir sur une base propre et de lancer une blockchain locale : - npx hardhat clean supprime les anciens artefacts (cache / builds) pour garantir un environnement propre. - npx hardhat node démarre un serveur JSON-RPC local et affiche une liste de comptes de test (wallets) avec leur solde. Ces comptes servent à simuler plusieurs utilisateurs pour tester mint, approve, offers et swaps.

Capture 02 — Déploiement du contrat (deploy.ts)

```
PS C:\Users\Laahir\Desktop\ECOLE\MASTER2\5BLOC\EVAL_V3\BLOCKCHAIN\resourceswap> npx hardhat run scripts/deploy.ts --network localhost
1 Deployer: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb2266
2 ResourceSwap deployed at: 0x5FbDB2315678afebfc367f032d93F642f64180aa3
```

Capture 03 — Mint d'un NFT ressource (mint.ts)

```
PS C:\Users\Laachir\Desktop\ECOLE\MASTER2\SBLOC\EVAL_V3\BLOCKCHAIN\resourceswap> $env:ACC="0"; $env:ANIMAL="Lapin"; npx hardhat run scripts/mint.ts --network localhost
1 Using account: 0x0f39fd6e51aad88f6f4ce6a88827279cfffb92266
2 Minting: Lapin URI: ipfs://bafybeidxmkohxp4mdrcg7iv6z37fxp7z15fg6zuqgutv6jjmozd2lztq/Lapin.json
3 tx hash: 0xecd82549a66d4ac6d4064f512867d21c63c1a91d45f73a1e84de67524da881e
4 Mint OK - block: 2
5 Minted tokenId: 1
6 ownerOf(token): 0xf39fd6e51aad88f6f4ce6a88827279cfffb92266
7 tokenURI(token): ipfs://bafybeidxmkohxp4mdrcg7iv6z37fxp7z15fg6zuqgutv6jjmozd2lztq/Lapin.json
```

Cette étape montre la tokenisation d'une ressource : - Le script mint un NFT (ex: ANIMAL="Lapin"). - Il affiche le tokenURI stocké sur IPFS (ipfs://.../Lapin.json). - On voit le hash de transaction, le block, le tokenId créé et le propriétaire. Cela démontre : **tokenisation + métadonnées IPFS**.

Capture 04 — Approve d'un token avant échange (approve.ts)

```
PS C:\Users\Laachir\Desktop\ECOLE\MASTER2\SBLOC\EVAL_V3\BLOCKCHAIN\resourceswap> $env:ACC="0"; $env:TOKEN="1"; npx hardhat run scripts/approve.ts --network localhost
1 Using account: 0x0f39fd6e51aad88f6f4ce6a88827279cfffb92266
2 Token: 1
3 Owner: 0xf39fd6e51aad88f6f4ce6a88827279cfffb92266
4 Approve tx: 0x119726f578f51496d45e8cc7ab64cf459ba233182fb29637a8a427b7ce74bf
5 Approved contract for token: 1
```

Avant qu'un token puisse être échangé, le propriétaire doit donner l'autorisation au contrat : - Le script affiche l'owner et l'id du token. - Une transaction approve est envoyée, permettant au contrat de transférer le token lors du swap. C'est un mécanisme clé : **aucun transfert possible sans consentement explicite**.

Capture 05 — CreateOffer bloqué par lock, puis résolu via warp (createOffer.ts + warp.ts)

Cette capture illustre la contrainte temporelle **LOCK_DURATION** : - Une première tentative de createOffer échoue : l'utilisateur est encore verrouillé (lock active). - Pour tester ce comportement en local, on exécute warp.ts afin d'avancer le temps (ex: 620 secondes). - Après l'avance du temps, la même commande createOffer réussit et retourne un offerId. Cela prouve : **lock bien appliqué + outil de test du temps**.

Capture 06 — AcceptOffer échoue sans approve, puis réussit après approve

```
PS C:\Users\Laachir\Desktop\ECOLE\MASTER2\SBLOC\EVAL_V3\BLOCKCHAIN\resourceswap> npx hardhat run scripts/readstate.ts --network localhost
1 CONTACT: ENS-082315678afecb367fe032d93f642f6418aa28
2 name: ENS-082315678afecb367fe032d93f642f6418aa28
3 symbol: RSWAP
4 maxSupply: 1
5 COOLDOWN: 5m 0s
6 LOCK_DURATION: 10m 0s
7 NOM: 0x69975132
8
9 Accepted: (0xf39fd6e51aad88f6f4ce6a88827279cfffb92266) ===
10 balanceOf: 1
11 ownedCount: 1
12 lockLeft: 5m 0s (COOLDOWN: 5m 0s)
13 lockLeft: 1m 0s (LOCK_DURATION=10m 0s)
14 tokens: 2
15
16 Token: 2
17 ipfs://bafybeidxmkohxp4mdrcg7iv6z37fxp7z15fg6zuqgutv6jjmozd2lztq/Singe.json
18 meta.name: Singe
19 meta.type: animal
20 meta.value: 0
21 meta.createdAt: 1769974164
22 meta.lastTransferAt: 1769975132
23
24 Accepted: (0xf39fd6e51aad88f6f4ce6a88827279cfffb92266) ===
25 balanceOf: 1
26 ownedCount: 1
27 lockLeft: 1m 0s (COOLDOWN=5m 0s)
28 lockLeft: 1m 0s (LOCK_DURATION=10m 0s)
29 tokens: 1
30
31 Token: 1
32 ipfs://bafybeidxmkohxp4mdrcg7iv6z37fxp7z15fg6zuqgutv6jjmozd2lztq/Lapin.json
33 meta.name: Lapin
34 meta.type: animal
35 meta.value: 2
36 meta.createdAt: 1769974164
37 meta.lastTransferAt: 1769975132
38
39 meta.value: 700
40 meta.createdAt: 1769974873
41 meta.lastTransferAt: 1769975132
42
43 == OFFERS (3..5) ==
44 (no active offers)
```

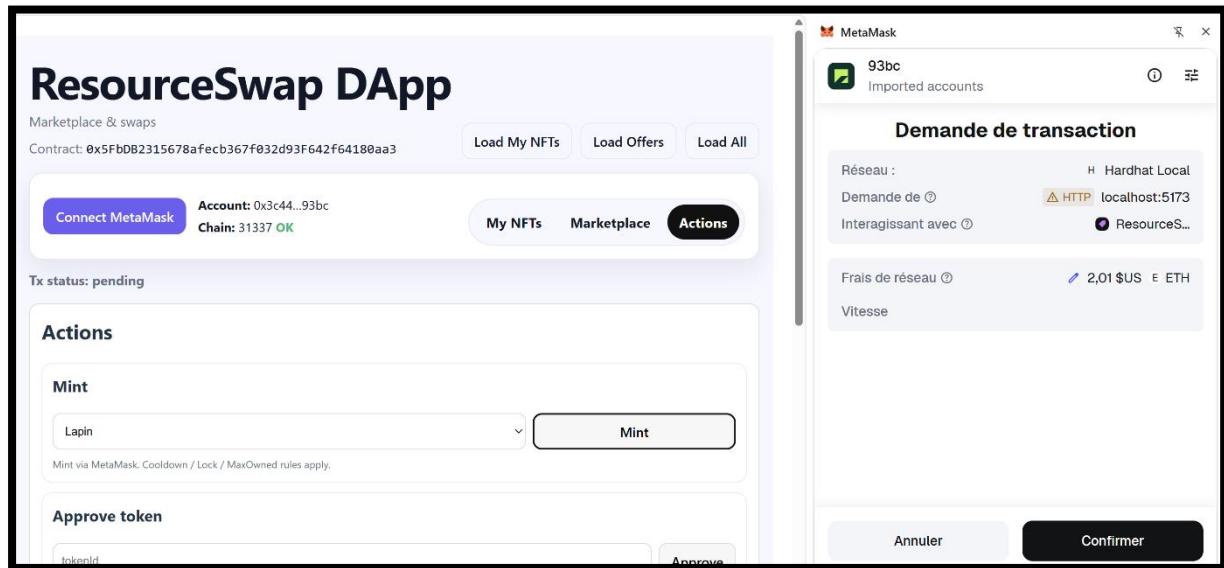
Cette capture montre une validation cruciale dans le swap : - La première acceptation échoue car le token demandé n'est pas approuvé pour le contrat. - Le script indique clairement la correction : exécuter approve.ts pour le token concerné. - Après approve, l'acceptation réussit et affiche les nouveaux propriétaires des deux tokens. Cela démontre : **gestion d'erreur contrôlée + validation stricte avant swap**.

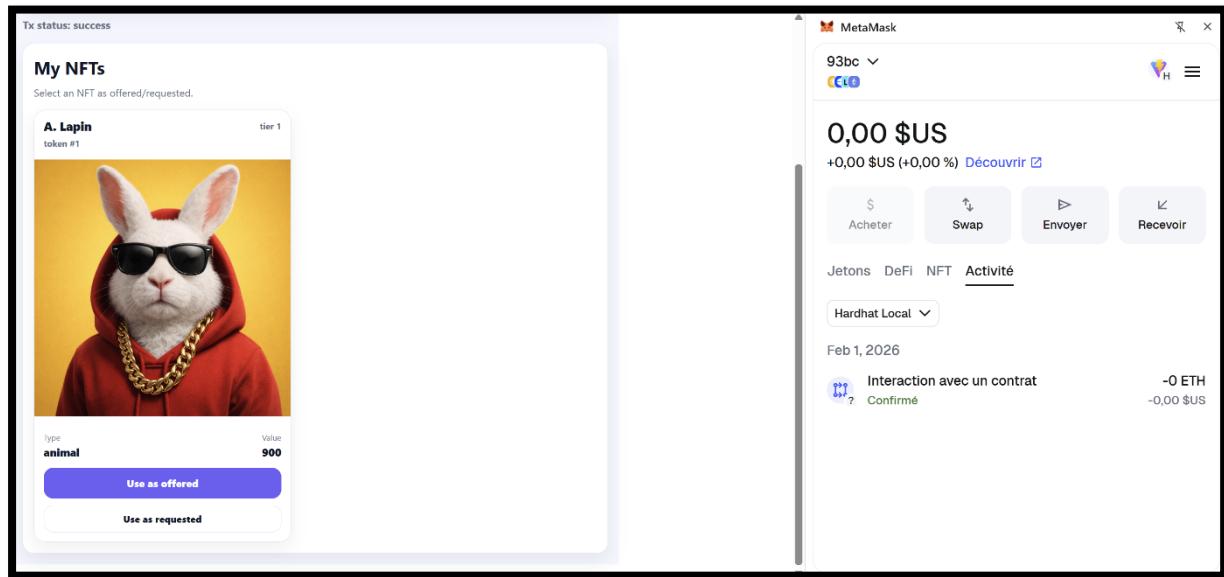
Capture 07 — Lecture de l'état complet (readState.ts)

```
PS C:\Users\Laachir\Desktop\ECOLE\MASTER2\SBLOC\EVAL_V3\BLOCKCHAIN\resourceswap> npx hardhat run scripts/readState.ts --network localhost
1 Contract: 0x5fbDB2315678afecb367f032d93F642f64180aa3
2 name: ResourceSwap
3 symbol: RSAP
4 MAX_OWNED: 4
5 COOLDOWN: 5m 0s
6 LOCK_DURATION: 10m 0s
7 NFT: 1769975132
8
9 *** Account#0 (0xf39fd5e51aad88f674ce6a8827279cffffb2266) ***
10 balanceOf: 2
11 ownedCount: 1
12 cooldown left: 5m 0s (COOLDOWN-5m 0s)
13 lock left: 10m 0s (LOCK_DURATION-10m 0s)
14 tokens: 2
15
16 Token #1
17 url: ipfs://bafybeidxmxkohxpmdncg7iv6z37fxp7sz15fg6zugutv6jmodz1ztq/Singe.json
18 meta.name: Singe
19 meta.type: animal
20 meta.tier: 1
21 meta.value: 900
22 meta.createdAt: 1769974164
23 meta.lastTransferAt: 1769975132
24
25 *** Account#3 (0x90879f6f6824f970365e785982ef101e93b986) ***
26 balanceOf: 1
27 ownedCount: 1
28 cooldown left: 5m 0s (COOLDOWN-5m 0s)
29 lock left: 10m 0s (LOCK_DURATION-10m 0s)
30 tokens: 1
31
32 Token #1
33 url: ipfs://bafybeidxmxkohxpmdncg7iv6z37fxp7sz15fg6zugutv6jmodz1ztq/Lapin.json
34 meta.name: Lapin
35 meta.type: animal
36 meta.tier: 1
37 meta.value: 700
38 meta.createdAt: 1769974073
39 meta.lastTransferAt: 1769975132
40
41 *** OFFERS (1..1) ***
42 (no active offers)
```

Le script readState.ts donne une vue claire de l'état du système : - Informations contrat : name, symbol, constantes (MAX_OWNED, COOLDOWN, LOCK_DURATION), timestamp actuel. - Par compte : nombre de tokens possédés, cooldown restant, lock restant, liste des tokens. - Par token : tokenURI IPFS, champs metadata (name/type/tier/value), timestamps. - Liste des offres actives (ou absence d'offres). Cette capture sert de preuve que **les données on-chain et les contraintes sont bien appliquées**.

8. Frontend DApp





Tx status: error

Actions

Mint

Lapin Mint

Mint via MetaMask. Cooldown / Lock / MaxOwned rules apply.

Approve token

tokenId Approve

Needed before acceptOffer (for the requested token).

Create Offer

2 1 Create

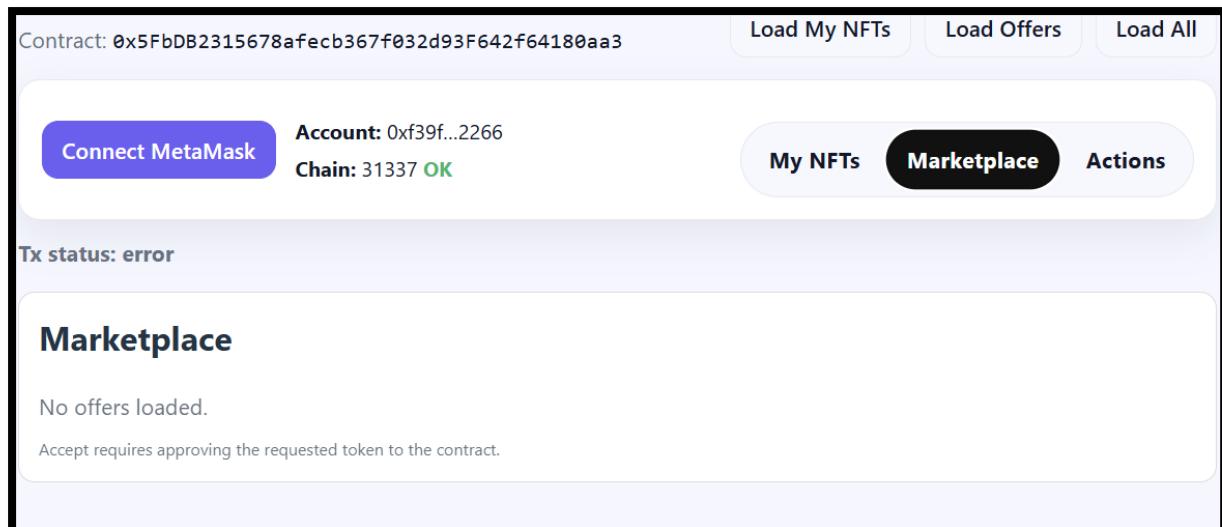
Tip: use "Use as offered/requested" on an NFT card to auto-fill.

Accept Offer

offerId Accept

Cancel Offer

offerId Cancel



9. Respect des exigences du sujet

Exigence	Réalisation
Tokenisation avec niveaux	NFT + tier
Échange de tokens	Offers + swap
Limite de possession	MAX OWNED
Cooldown	COOLDOWN
Lock	LOCK_DURATION
IPFS	tokenURI
Tests unitaires	Hardhat
Couverture significative	>96%

10. Conclusion

Le projet **ResourceSwap** répond intégralement aux exigences du sujet et démontre une maîtrise solide des concepts fondamentaux du Web3.

Il met en évidence une conception rigoureuse de smart contracts sécurisés, une implémentation cohérente et non contournable des règles métiers (limite de

possession, cooldown et verrou temporel), ainsi qu'une architecture claire séparant la logique blockchain, l'interface utilisateur et la documentation. La qualité du projet est également renforcée par une validation approfondie via des tests unitaires couvrant l'ensemble des cas critiques, avec une couverture élevée attestant de la robustesse du contrat.

Enfin, bien que le projet soit fonctionnel et complet dans son périmètre actuel, des améliorations futures pourraient être envisagées, telles que la pagination des offres, l'indexation des données par « The Graph » ou encore un déploiement sur un testnet public afin de renforcer sa scalabilité et son réalisme.