



# Carnet de projet

Bellengé Nicolas

GitHub:

<https://github.com/Chewbaccuin/ProjetAlyra>

dApp :

<https://projet-alyra-nbellenge.vercel.app/>

**Certification**

Développer une application décentralisée  
avec les technologies blockchain  
RS6515

# Introduction

Je m'appelle Nicolas Bellengé, connu sous le pseudonyme de Chewbaccoin dans le monde de la crypto. J'ai 43 ans et je vis à Bordeaux. Je suis passionné par l'univers des crypto monnaies depuis 2012, une aventure qui a commencé avec le solo mining de bitcoin et qui m'a mené à travers toutes les grandes étapes de l'écosystème : la perte de BTC chez MtGox, la participation à l'ICO d'Ethereum, et la création de projets basés sur des blockchains dédiées à l'époque des masternodes.

Mon parcours m'a également permis de devenir validateur sur différentes blockchains, notamment la BSC (BEP2), et aujourd'hui sur Avalanche et Cosmos.

Professionnellement, je dirige ma propre société, NBTC SAS, un jeu de mots entre mes initiales et le BTC, dédiée à l'investissement en cryptomonnaies. J'ai fait le choix de transférer mon patrimoine crypto à ma société pour des raisons fiscales, et cela me permet de gérer mes activités d'investissement de manière optimale. Depuis trois ans, je vis des rendements offerts par la DeFi.

Avant de plonger pleinement dans la crypto, j'ai travaillé comme développeur web, d'abord en tant que développeur frontend, puis fullstack, de 2004 à 2021. J'ai quitté mon poste en CDI en ingénierie informatique lorsque mes revenus crypto sont devenus suffisants pour en tirer des revenus supérieurs à mon ancien emploi.

J'ai décidé de suivre une formation de développeur Blockchain pour élargir mes compétences techniques et donner vie à des idées de projets Web3 qui me tiennent à cœur. Ce projet de soutenance, Pump Music, est le fruit de cette démarche et reflète mon envie de combiner des concepts novateurs avec mon expérience dans la crypto et le développement.

# Cahier des charges

**Pump Music** est une plateforme décentralisée qui révolutionne la monétisation musicale en permettant aux artistes et producteurs de tokeniser leurs royalties. En émettant des tokens ERC-20 représentant un pourcentage de leurs revenus futurs, ils peuvent lever des fonds tout en conservant leur indépendance.

## Fonctionnalités principales

1. **Émission de tokens**
  - Les artistes créent des tokens via la DApp, représentant une part de leurs royalties.
  - Les smart contracts définissent la distribution automatique et transparente des revenus.
2. **Calcul et redistribution des royalties**
  - Intégration avec des API de plateformes comme Spotify pour collecter et redistribuer les royalties.
3. **Marché primaire et secondaire**
  - Les tokens sont vendus sur la plateforme ou échangés entre investisseurs.
  - Un système de swap permet de convertir les tokens en USDC ou d'autres tokens artistiques.
4. **Suivi des performances**
  - Tableau de bord en temps réel pour suivre ventes, royalties générées et performance des tokens.

## Objectifs

- Aider les artistes à lever des fonds indépendamment des labels.
- Offrir aux investisseurs un nouveau type d'actif basé sur la musique.
- Créer un écosystème transparent et automatisé grâce à la blockchain.

## Technologies principales

- Blockchain (EVM compatible / L2 Ethereum), Smart Contracts (Solidity), API (Spotify, Apple Music etc).

**Cette partie du rapport permet d'évaluer la compétence suivante :**

C1. Rédiger un cahier des charges en décrivant les objectifs, les modalités de réalisation et les fonctionnalités d'une future application décentralisée, afin de la conceptualiser et d'en assurer la réussite

# Compréhension de l'écosystème

**Pump Music** s'appuie sur la blockchain pour garantir :

- **Transparence** : Les transactions et la distribution des royalties sont immuables et vérifiables.
- **Automatisation** : Les smart contracts redistribuent les revenus sans intermédiaires, réduisant coûts et délais.
- **Sécurité** : Les tokens de royalties sont protégés par des mécanismes cryptographiques.
- **Accessibilité globale** : Une infrastructure décentralisée connectant artistes et investisseurs partout dans le monde.

Sans blockchain, il serait impossible d'offrir un tel niveau de confiance, de décentralisation et d'efficacité.

## Pertinence dans l'écosystème

L'industrie musicale souffre de :

- **Centralisation** : Les artistes dépendent des labels pour financer leur carrière. Pump Music leur permet de lever des fonds directement auprès des investisseurs.
- **Manque de transparence** : La blockchain assure une répartition claire et traçable des royalties.
- **Monétisation limitée** : Avec Pump Music, les artistes valorisent leur travail en cédant une partie de leurs revenus futurs.

## État de l'art

Des projets comme **Opulous**, **Audius** ou **Royal.io** explorent la musique et la blockchain, mais se concentrent sur des niches (NFTs ou streaming). Pump Music se distingue par une approche globale : tokenisation des royalties via ERC-20 et marché secondaire intégré.

**Pump Music** est une solution innovante qui combine blockchain et finance participative, alignée avec les tendances actuelles du Web3 et de la DeFi, en créant une nouvelle voie pour monétiser la musique.

**Cette partie du rapport permet d'évaluer la compétence suivante :**

C1. Rédiger un cahier des charges en décrivant les objectifs, les modalités de réalisation et les fonctionnalités d'une future application décentralisée, afin de la conceptualiser et d'en assurer la réussite

# Périmètre de l'application réalisée

Le projet **Pump Music** étant ambitieux dans sa version finale, le MVP se concentre sur les fonctionnalités essentielles permettant de valider les concepts de base.

## Fonctionnalités incluses dans le MVP

1. **Whitelisting des Artiste par le projet via un NFT Soulbound**
  - L'équipe de Pump Music autorise des artistes précis à utiliser la plateforme.
2. **Création de tokens ERC-20**
  - Les artistes peuvent créer des tokens via un formulaire simple.
  - Chaque token représente un pourcentage défini de royalties.
3. **Marché primaire simplifié**
  - Les artistes listent leurs tokens à un prix fixe.
  - Les investisseurs achètent des tokens via un wallet connecté (MetaMask).
4. **Simulation des royalties**
  - Interface affichant un revenu simulé (données manuelles).
  - Redistribution des royalties selon les tokens détenus.
5. **Interface utilisateur minimaliste**
  - Achat de tokens.
  - Suivi des tokens détenus et des revenus simulés.
  - Système de swap de tokens.

## Hors périmètre

- Intégration avec des API musicales.
- Marché secondaire ou swap.
- Distribution réelle des royalties.
- Fonctionnalités avancées comme les analyses de performance.

## Objectif du MVP

Le MVP vise à démontrer :

- La faisabilité de la tokenisation des royalties.
- Une expérience utilisateur simplifiée.
- Une validation initiale du concept.

Ce périmètre limité garantit un prototype fonctionnel et réalisable sur la durée prévue par Alyra.

**Cette partie du rapport permet d'évaluer la compétence suivante :**

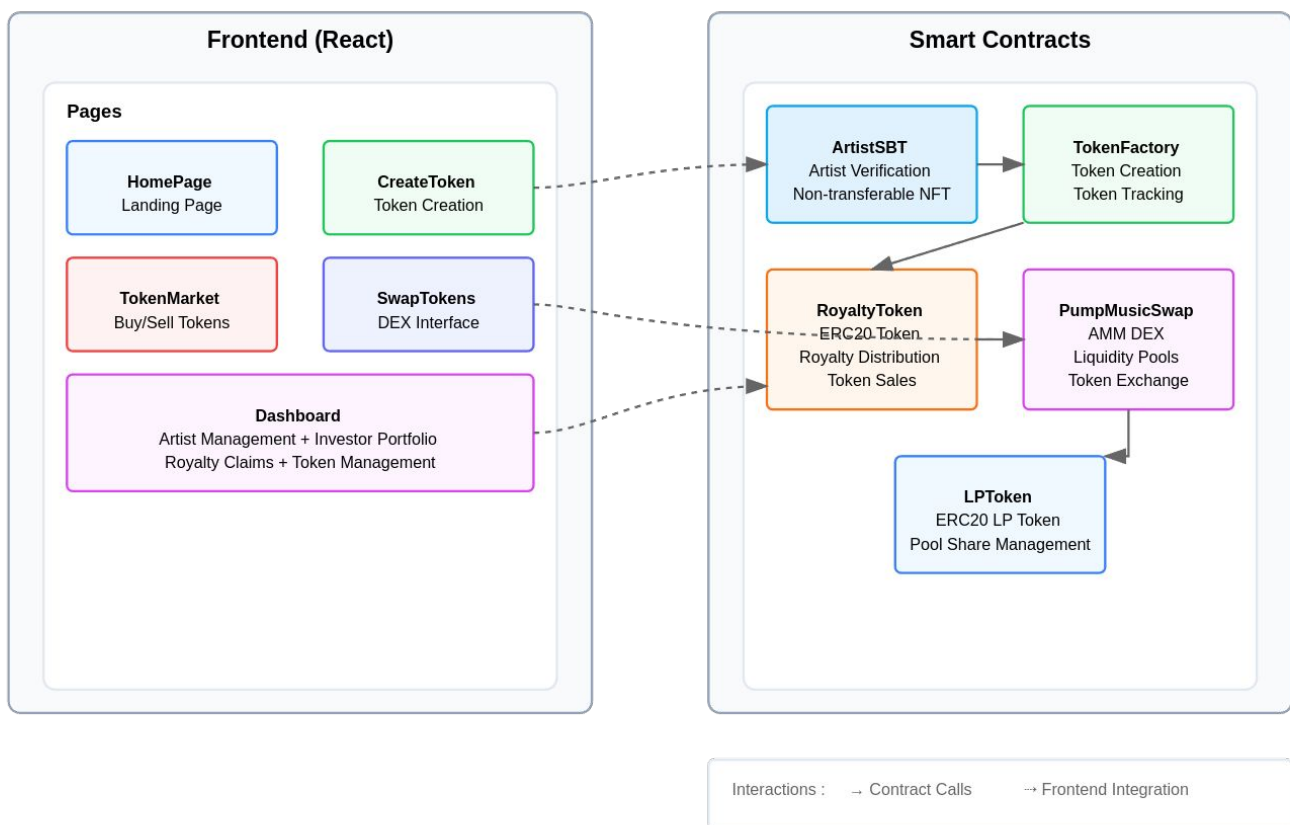
C1. Rédiger un cahier des charges en décrivant les objectifs, les modalités de réalisation et les fonctionnalités d'une future application décentralisée, afin de la conceptualiser et d'en assurer la réussite

# Schéma fonctionnel de l'app

## Flot d'interaction

1. **Admin** : Autorise des artistes via attribution/révocation de SBT.
2. **Artiste** : Crée des tokens -> Liste les tokens sur la marketplace -> Distribue des royalties.
3. **Investisseur** : Achete des tokens -> Perçoit des royalties -> Echange ses tokens par swap.

## Architecture PumpMusic dApp



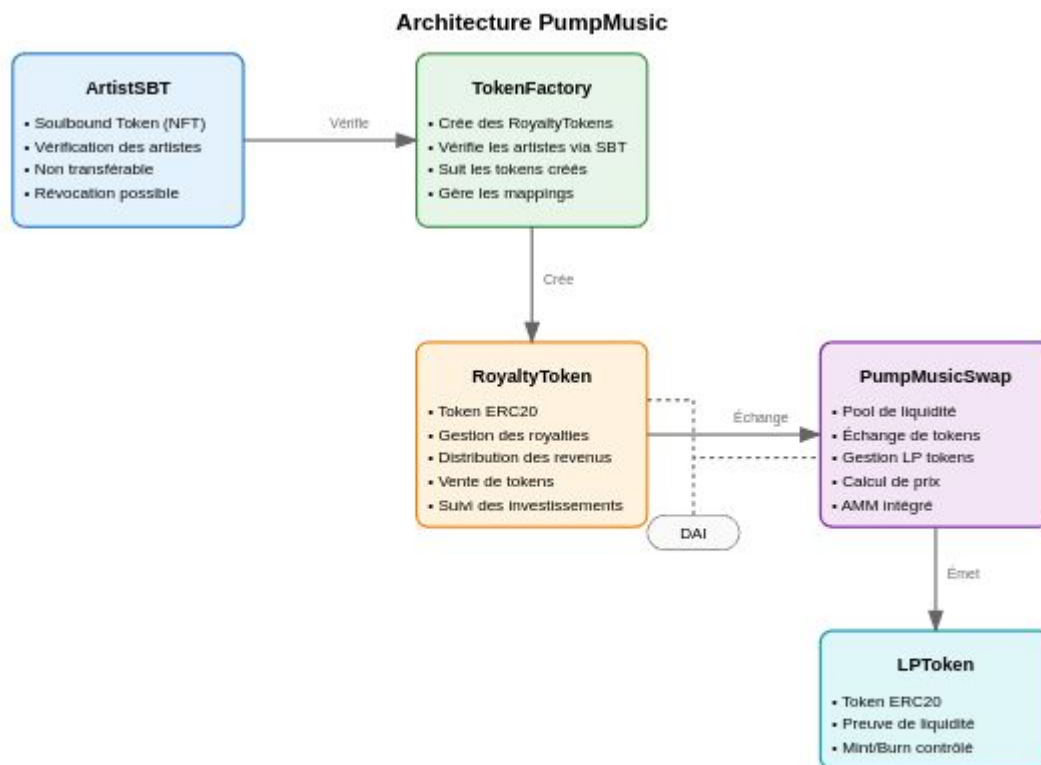
**Cette partie du rapport permet d'évaluer la compétence suivante :**

C1. Rédiger un cahier des charges en décrivant les objectifs, les modalités de réalisation et les fonctionnalités d'une future application décentralisée, afin de la conceptualiser et d'en assurer la réussite

# Smart contract

La dApp Pump.Music repose sur plusieurs smart-contracts :

- ArtistSBT.sol : permet de gérer les autorisations de publications réservées aux artistes.
- PumpMusicTokenFactory : permet de générer des tokens ERC20 pour chaque titre qu'un artiste souhaite monétiser.
- PumpMusicRoyaltyToken : représente le token qui est émis et qui est vendu aux investisseurs contre perception des royalties de l'artiste.
- PumpMusicSwap : gère le système de LP et de Swap inter-tokens.
- LPToken : représente les token de liquidité émis par le système de swap.



**Cette partie du rapport permet d'évaluer la compétence suivante :**  
 C2. Développer un contrat intelligent (smart contract) en utilisant un langage de programmation adapté à une technologie blockchain afin de répondre au besoin de fonctionnalités d'une application décentralisée

Présentez vos  
Smarts contracts  
en même temps!



## Votre jeton

Pump.Music embarque deux types de jetons, chacun avec un usage spécifique :

1. **ArtistSBT** (Non-Fongible, ERC-721 Modifié)
  - Type : Token non-fongible "Soulbound" (SBT)
  - Norme : ERC-721 modifiée pour être non-transférable
  - Usage :
    - Certification et vérification des artistes
    - Sert de système d'accréditation
    - Un seul token par artiste
  - Choix de la norme :
    - ERC-721 choisi pour son unicité (un token = un artiste)
    - Modifié pour être "Soulbound" (non-transférable) afin de garantir que les accréditations d'artistes ne peuvent pas être vendues ou transférées
    - Utilise la version Upgradeable d'OpenZeppelin pour permettre des mises à jour futures
2. **RoyaltyToken** (Fongible, ERC-20)
  - Type : Token fongible
  - Norme : ERC-20 standard
  - Usage :
    - Représente les droits sur les royalties musicales
    - Permet la fonctionnalisation des revenus
    - Facilite les échanges et la liquidité
  - Choix de la norme :
    - ERC-20 choisi pour sa fongibilité, permettant de diviser facilement les droits
    - Standard largement adopté facilitant l'intégration avec les DEX et autres protocoles DeFi
    - Fonctionnalités étendues pour gérer :
      - La distribution des royalties
      - Le suivi des investissements
      - Les périodes de validité des droits

Le système combine donc intelligemment :

- Un token non-fongible pour l'identité/certification (SBT)
- Des tokens fongibles pour les droits économiques (ERC-20)

Cette architecture permet de :

1. Garantir l'intégrité du système de vérification des artistes
2. Faciliter la tokenisation et l'échange des droits musicaux
3. Assurer une distribution équitable et automatisée des royalties

L'utilisation de standards établis (ERC-20, ERC-721) garantit la compatibilité avec l'écosystème Ethereum tout en les adaptant aux besoins spécifiques de la plateforme.

**Cette partie du rapport permet d'évaluer la compétence suivante :**

C3. Exploiter un jeton numérique (fongible\* ou non) en utilisant les librairies et les standards pratiqués dans l'industrie afin de rendre effectif un ensemble de fonctionnalités propres à une application décentralisée

Indiquez son utilisation dans votre code





# Sécurité

Analyse sécurité :

## 1. Contrôle d'accès

Faillles potentielles :

- Accès non autorisé aux fonctions administratives
- Manipulation des vérifications d'artistes
- Usurpation d'identité d'artiste

Résolutions implémentées :

- Utilisation du modificateur `onlyOwner`
- Vérification SBT pour les artistes
- Système de révocation des artistes

```
function verifyArtist(address artist) external onlyOwner {  
    if (isArtist(artist)) revert ArtistAlreadyVerified();  
    // ...  
}
```

## 2. Réentrance

Faillles potentielles :

- Réentrance lors de la distribution des royalties
- Réentrance lors des achats de tokens
- Double dépense de royalties

Résolutions implémentées :

- Utilisation de `ReentrancyGuard`
- Pattern checks-effects-interactions

```
function claimRoyalties() external nonReentrant {  
    // Checks  
    if (userInvestments[msg.sender] == 0) revert NoTokensOwned();  
    // Effects  
    royaltyInfo.totalRoyalties -= share;  
    // Interactions  
    (bool success, ) = payable(msg.sender).call{value: share}("");  
}
```

## 3. Manipulation des prix et des montants

Faillles potentielles :

- Overflow/Underflow dans les calculs
- Manipulation des prix des tokens
- Erreurs de précision dans les calculs de royalties

Résolutions implémentées :

- Utilisation de Solidity 0.8+ (protection automatique overflow/underflow)
- Validations des paramètres

## 4. Gestion des erreurs

Faillles potentielles :

- Échecs silencieux des opérations
- États inconsistants après erreur
- Perte de fonds due aux erreurs

Résolutions implémentées :

- Utilisation d'erreurs personnalisées
- Vérification des retours des appels externes

```
error RoyaltyPeriodExpired();  
error NoTokensOwned();  
error ShareTooSmall();  
// ...  
require(success, "Platform fee transfer failed");
```

**Cette partie du rapport permet d'évaluer la compétence suivante :**

C4. Identifier les points d'attention en matière de sécurité et les optimisations adaptées en analysant les fonctionnalités d'un contrat intelligent (smart contract) afin de prévenir des défaillances éventuelles

Présentez du code si nécessaire



# Description de l'intégration continue

Afin de garantir un développement harmonieux et une qualité de code constante, j'ai instauré un workflow d'intégration continue reposant sur plusieurs outils :

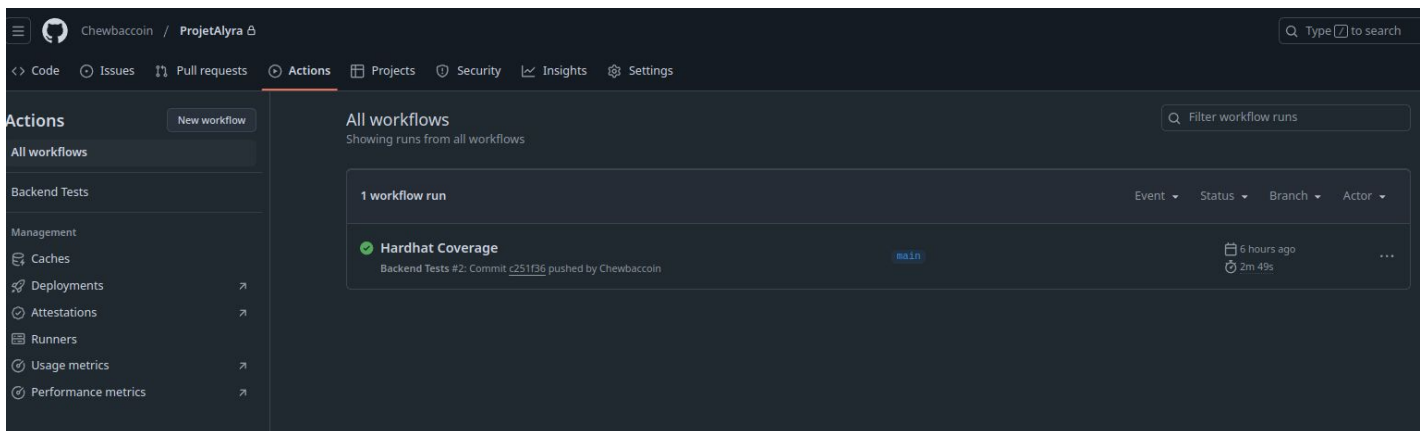
## Gestion des versions avec GitHub :

Même en travaillant en solo sur ce projet, GitHub offre la possibilité de travailler sur différentes fonctionnalités en parallèle tout en les isolant. Cela prépare également le terrain pour une collaboration future si l'équipe venait à s'agrandir.

## Automatisation des tests via GitHub Actions :

J'ai configuré GitHub Actions pour exécuter automatiquement les tests unitaires à chaque mise à jour de la branche principale. Par exemple, la commande `npx hardhat test` est déclenchée à chaque push. Cette approche permet d'identifier rapidement les éventuelles régressions et garantit que toutes les modifications respectent les exigences des tests.

<https://github.com/Chewbaccain/ProjetAlyra/actions/runs/12428501045>



## Déploiement continu avec Vercel :

J'utilise Vercel pour gérer le déploiement automatique de l'application. À chaque mise à jour sur la branche principale, une version de production à jour est générée et mise en ligne, assurant ainsi une disponibilité constante des nouvelles fonctionnalités.

**Cette partie du rapport permet d'évaluer la compétence suivante :**  
C5. Mettre en place la gestion des versions en utilisant des méthodes d'intégration continue, afin de garantir la viabilité du développement de l'application

Présentez du code si nécessaire



# Testing

- Utilisation du framework de test Hardhat avec Chai pour les assertions. Emploi de fixtures pour déployer les contrats et configurer l'environnement de test.
- Tests unitaires pour chaque fonction principale du contrat.
- Vérification des événements émis, des changements d'état, et des révisions attendues.

Exemple :

<https://github.com/Chewbaccuin/ProjetAlyra/blob/main/backend/test/PumpMusicRoyaltyToken.test.js>

106 passing (4s)

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	99.37	75	100	99.51	
ArtistSBT.sol	96.55	87.5	100	97.22	76
LPToken.sol	100	75	100	100	
PumpMusicRoyaltyToken.sol	100	72.92	100	100	
PumpMusicSwap.sol	100	65.22	100	100	
PumpMusicTokenFactory.sol	100	100	100	100	
contracts/mocks/	100	100	100	100	
MockDAI.sol	100	100	100	100	
MockToken.sol	100	100	100	100	
MockUSDC.sol	100	100	100	100	
All files	99.4	76.12	100	99.53	

J'ai veillé à couvrir une large gamme de scénarios, incluant à la fois les cas standards et les cas d'erreur, garantissant ainsi un niveau élevé de qualité et de sécurité. Les tests valident non seulement le fonctionnement des principales fonctionnalités, mais également les mécanismes de sécurité, tels que les contrôles d'accès, la gestion des pools de liquidités et la vérification des soldes des wallets.

**Cette partie du rapport permet d'évaluer la compétence suivante :**  
 C6. Vérifier la résilience d'une application décentralisée, en mettant en place des tests fonctionnels, afin d'en garantir le bon fonctionnement pour les usagers

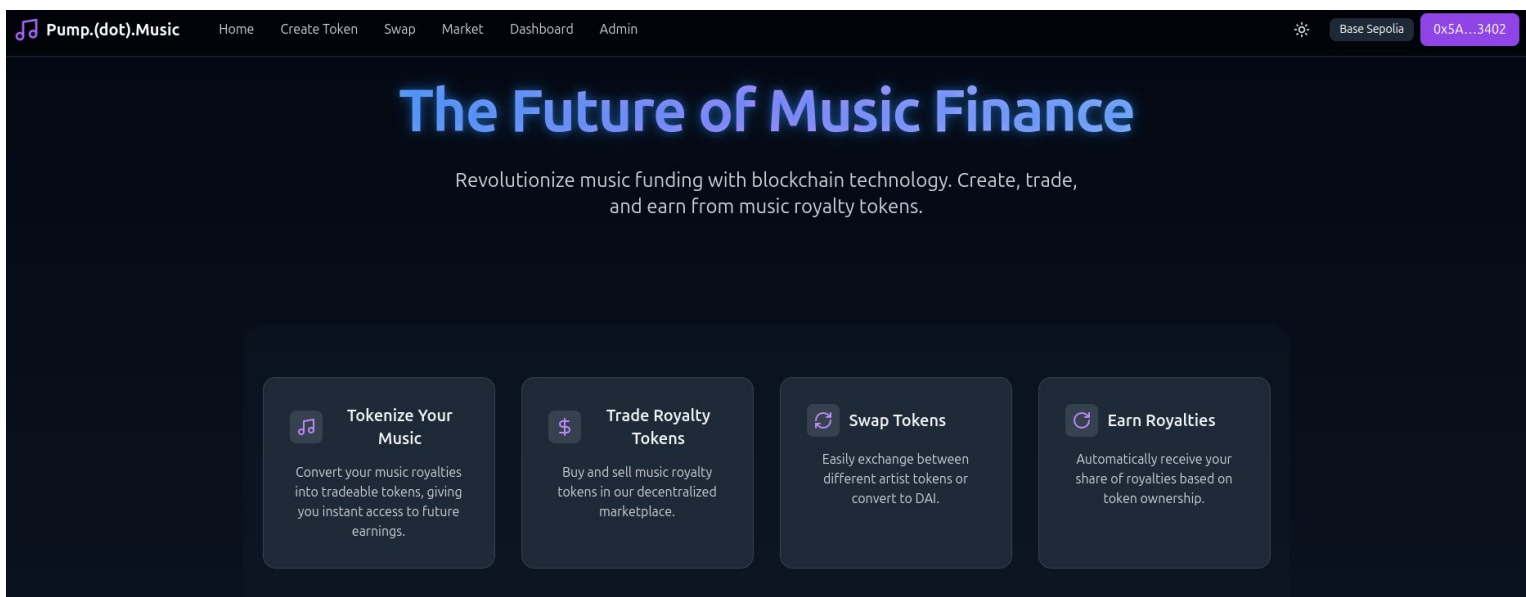
Faite la démonstration d'une série de test en live



## Front 1/2

Technologies employées :

- **Next.js** : Framework de développement basé sur React, offrant le rendu côté serveur (SSR) et la génération de sites statiques (SSG).
- **React** : Utilisé pour la création de l'interface utilisateur.
- **ethers.js** : Permet l'interaction avec les smart contracts sur Ethereum.
- **RainbowKit** : Gestion de l'authentification via wallet.
- **Wagmi** : Outil pour gérer les connexions blockchain et fournir des hooks React adaptés.
- **Tailwind CSS** et **Shadcn-ui** : Utilisés pour le design et les composants.




**Cette partie du rapport permet d'évaluer la compétence suivante :**  
C7. Développer un code informatique en utilisant un langage de programmation adapté aux technologies web afin de rendre effective la communication entre une interface web et/ou mobile et un ou plusieurs contrat(s) intelligent(s) (smart contract)

Faites une démo de votre front ici



## Front 2/2

Dashboard Admin



### Verify Artist

Grant artist verification SoulBound Token

Artist Address


Verify Artist

#### Verified Artists

0x5AAAC7b16096d552a258210737BBE22F39b3402
Revoke

#### Revoked Artists

No revoked artists



### Create Your Music Token

Configure your royalty token parameters

Token Name

Token Symbol

Royalty Percentage

 %

Duration (days)

Token Price (DAI)

 DAI


Create Token


Dashboard


Artist Investor

### Artist

Manage your music tokens and track your earnings


Your Tokens
11


Total Holders
8


Total Amount Raised
226760.00 DAI

#### Your Tokens

Create New Token

##### Example Artist Token

\$EAT

Price

1.00000000 DAI

Royalty Rate

1.00%

Token Holders

1

Amount Raised

200000.00 DAI

Royalties Expiration

Dec 17, 2025

Status

Listed For sale

Manage >

##### Macarena

\$MAK

Price

10.00000000 DAI

Royalty Rate

1.00%

Token Holders

1

Amount Raised

10000.00 DAI

Royalties Expiration

Dec 17, 2025

Status

Listed For sale

Manage >

##### Techno

\$TEK

Price

1.00000000 DAI

Royalty Rate

1.00%

Token Holders

1

Amount Raised

210.00 DAI

Royalties Expiration

Dec 17, 2025

Status

Listed For sale

Manage >

**Cette partie du rapport permet d'évaluer la compétence suivante :**  
**C7. Développer un code informatique en utilisant un langage de programmation adapté aux technologies web afin de rendre effective la communication entre une interface web et/ou mobile et un ou plusieurs contrat(s) intelligent(s) (smart contract)**

Faites une démo de votre front ici



# Déploiement

## Outils utilisés :

- **Hardhat script** : Un framework de développement pour Ethereum, permettant la compilation, le déploiement, les tests et le débogage des smart contracts.
- **Ethers.js** : Une bibliothèque destinée à faciliter les interactions avec la blockchain Ethereum.
- **Basescan** : Un explorateur de blocs utilisé pour vérifier les contrats déployés sur le réseau Base.

## Étapes de déploiement :

1. **Déploiement ordonné** : Les contrats sont installés dans un ordre précis pour permettre l'injection correcte des dépendances nécessaires.
2. **Vérification conditionnelle** : La vérification des contrats via Basescan est effectuée uniquement pour les réseaux de test (testnets), excluant les environnements locaux.
3. **Validation post-déploiement** : Le script effectue une double vérification après le déploiement pour s'assurer que toutes les adresses des contrats sont correctement assignées. Ensuite, ces adresses sont intégrées dans le fichier `frontend/src/contracts/index.js`.
4. **Vérification sur Basescan** : Pour les déploiements hors environnement local, une vérification automatique des contrats est réalisée sur Basescan.

```
Deployment Summary:
=====
MockDAI: 0x6918866e700DD3740041b770FE2Dcf53de482913
MockUSDC: 0xdDe0A44EFd7b7e7537B3920080401868335bEC67
SBT: 0x67b05992574E22d59bdC6b0CBaF84927b12a85e5
Factory: 0x7ff6236Bc59CDd6a53Bde6Cda8896aE5a79b5364
Swap: 0x4B9E0E8675e33B461118e371F13635B5192ba83a
Example Token: 0xF678a1F25989270d7D7fF8f5656A65005a6eed2e
```

## Commandes de déploiement :

- **Sur le réseau Base Sepolia :**

```
npx hardhat run scripts/PumpMusic.js --network baseSepolia
```

- **En local :**

```
npx hardhat run scripts/PumpMusic.js --network localhost
```

Réalisez un déploiement sur Testnet, le jury peut accéder au smart contract sur explorateur

**Cette partie du rapport permet d'évaluer la compétence suivante :**

C8. Déployer un contrat intelligent (smart contract) sur une blockchain en s'appuyant sur un environnement de développement adapté afin de rendre opérationnelle une application décentralisée





# Bilan de projet et améliorations

Le développement de **Pump Music** a été une expérience à la fois complexe et enrichissante, nécessitant l'utilisation de technologies avancées et l'intégration de concepts techniques pointus. Voici un retour détaillé sur les étapes clés du projet :

## Satisfactions techniques :

- **Respect des normes de l'industrie** : Utilisation des standards ERC-20 et ERC-721 modifiés pour aligner les fonctionnalités des tokens avec les besoins du projet.
- **Gestion efficace de la sécurité** : Adoption de mécanismes comme **ReentrancyGuard**, le pattern checks-effects-interactions et des contrôles stricts d'accès avec **onlyOwner** et **SBT verification**.
- **Tests exhaustifs** : Couverture importante des tests unitaires via **Hardhat** et **Chai**, garantissant une validation fonctionnelle des contrats.

## Points à améliorer

### 1. Fonctionnalités avancées

Certaines fonctionnalités initialement prévues restent en attente d'implémentation pour aller au-delà du MVP :

- **Distribution réelle des royalties** : Automatisation via les API des plateformes musicales pour passer de la simulation à des données réelles.
- **Suivi des revenus réels** : Implémenter une interface intégrant des graphiques interactifs via des bibliothèques comme **Chart.js** ou **D3.js**.

### 2. Optimisation des performances

- **Réduction des coûts en gas** : Révision des fonctions dans les smart contracts, en particulier celles liées à la création et à l'échange des tokens, pour réduire les frais d'exécution sur Ethereum.
- **Segmentation des données** : Mise en œuvre d'un mécanisme off-chain pour stocker les informations non critiques, réduisant ainsi la charge sur la blockchain.

### 3. Sécurité renforcée

- Réalisation d'un **audit externe** complet des smart contracts pour identifier les failles potentielles non détectées lors des tests internes.
- Ajout de mécanismes d'alertes sur anomalies via des événements on-chain, surveillés par des scripts externes.

## Enseignements techniques

- L'implémentation d'un système modulaire pour les smart contracts (par exemple, utilisation de **Proxy Patterns** pour gérer les mises à jour des contrats) aurait facilité l'ajout de nouvelles fonctionnalités.
- Les défis rencontrés lors de l'intégration de données off-chain et on-chain mettent en lumière l'importance d'une gestion optimisée des interactions hybrides.

# Conclusion

## Retour sur trois mois de formation

Ces trois mois de formation en développement blockchain ont été intenses et transformateurs. J'ai pu explorer des technologies nouvelles, comme les **smart contracts** et les standards ERC, tout en consolidant mes compétences en programmation et en gestion de projet. La conception et la réalisation de **Pump.Music** ont représenté un défi technique et une opportunité de matérialiser une idée innovante dans le domaine de la musique et de la finance décentralisée.

### Accomplissements marquants :

- La mise en place d'un **MVP fonctionnel** démontrant la faisabilité de la tokenisation des royalties musicales.
- L'intégration réussie d'outils modernes comme **Hardhat**, **Next.js**, **ethers.js**, et **RainbowKit**, garantissant une expérience utilisateur fluide et sécurisée.
- Une compréhension approfondie des enjeux de **sécurité blockchain**, avec des solutions implémentées contre les vulnérabilités critiques comme la réentrance et les surflows.

## Motivations et apprentissages

La blockchain m'a toujours passionné par son potentiel de disruption et de décentralisation. En me lançant dans ce projet, mon objectif était double : approfondir mes connaissances techniques et explorer comment cette technologie peut transformer l'industrie musicale. **Pump.Music** est le fruit de cette ambition et m'a confirmé qu'avec une base solide en blockchain, il est possible de répondre à des besoins concrets de manière novatrice.

En termes d'apprentissage, voici les points les plus marquants :

- L'importance de structurer un projet complexe en étapes réalisables grâce à un **MVP**.
- Les subtilités liées à l'**interopérabilité blockchain**, notamment entre les contrats intelligents et l'interface utilisateur.
- Les bonnes pratiques de **développement sécurisé** dans un environnement où la moindre erreur peut avoir des conséquences financières importantes.

## Regrets et axes d'amélioration

Si je devais revenir sur ces trois mois, voici les aspects que j'aurais abordés différemment :

- **Gestion du temps** : Certaines fonctionnalités, comme l'intégration des API musicales, ont dû être reportées faute de temps. Une planification plus fine aurait permis d'en inclure davantage dans le MVP.
- **Collaborations** : Travailler en solo m'a limité dans certains domaines comme l'UI/UX design et les audits de sécurité. Collaborer avec d'autres experts aurait enrichi le projet.