ALYRA

Rapport de projet

William SIMON--VEZO

Session Turing

GitHub:
https://github.com/Varadiell/alyra-carbonthink

Certification

Développer une application décentralisée avec les technologies blockchain
RS6515

Introduction

Quel est votre background précédant la formation, et comment en êtes vous, seul ou collectivement, arrivé à cette idée de projet?

Développeur Full-Stack, dans le domaine depuis 2014.

10 ans d'expérience :

- Algo Data, hébergement de sites web, emailing, événementiel.
- Wiztivi, interfaces pour box TV (SFR & Vodafone).
- TheFork, widget de réservation pour restaurants, TFManager.

Je suis curieux par nature, j'aime les choses bien faites et maîtriser ce que je fais de bout en bout. Globalement, mes expériences en entreprise et mes projets personnels m'ont beaucoup appris sur le plan technique.

Après un plan de licenciement à TheFork qui a touché les bureaux nantais principalement, et désireux depuis fort longtemps d'effectuer des formations, me voici à Alyra pour deux sessions (Web 3 & IA).

J'ai mis le doigt dans l'engrenage du monde de la crypto après l'affaire GameStop de 2021, très remonté contre un système qui "triche" sans trop se cacher. Je me suis donc intéressé à la crypto et à la décentralisation depuis lors, et au fonctionnement global du web 3.

Je suis convaincu, sans doute naïvement, que l'on peut rendre le monde meilleur avec plus de transparence, moins de censure, et avec la suppression d'intermédiaires aujourd'hui quasi obsolètes grâce à la décentralisation.

—

J'ai rejoint le projet de tokenisation de Crédits Carbone de 3 autres consultants, qui avaient chacun un projet similaire déjà bien avancé sur le plan théorique. Nous voilà donc en équipe à créer une dapp de suivi, d' émission et de burn de crédits carbone sous forme de NFTs, et dont la gouvernance sera assurée par une seule entité (entreprise), donc centralisée. Les NFTs, eux, appartiendront à leurs propriétaires respectifs.

Cahier des charges

Décrivez l'application que vous imaginez, ses fonctionnalités…

Avant tout, un peu de contexte…

La genèse de ce projet vient d'un problème du monde réel : un cousin d'un des consultants gère une entreprise qui accompagne des agriculteurs (que nous appellerons plus tard "Porteurs de Projet") dans des projets de plantation de bambou sur leurs parcelles, afin de générer des Crédits Carbone, qui pourront ensuite être revendus à des entreprises soit désireuses de "compenser" leurs émissions de CO2, soit par obligation légale (exemple des compagnies aériennes).
Aujourd'hui, cette entreprise a des demandes de ses clients pour une plus grande traçabilité et plus de confiance dans le suivi de leurs projets, spécifiquement grâce à la blockchain.
Autre problème que l'entreprise veut éliminer : les registres tiers. Il en existe plusieurs, qui font office d'autorité (validateur) tierce dans l'émission de Crédits Carbone, ce qui permet la confiance de la part des acheteurs. Ces validations coûtent une certaine somme, dont l'entreprise veut s'affranchir.

Nous avons donc imaginé une application dont les fonctionnalités sont (voir PDF "spécifications" dans le GitHub pour plus de détails) :

- La création et la gestion de projets de décarbonation dont le fruit du travail pourra justifier de l'émission de tokens TCO2, chacun correspondant à 1 tonne de CO2 capturée, équivalent à 1 Crédit Carbone.
- L'émission de tokens NFTs TCO2 par projet, avec insertion de metadata pour faciliter la lisibilité.
- Le suivi de ces tokens TCO2 et de ces projets via la blockchain.
- Le "burn" de tokens pour permettre aux entreprises de "compenser" leurs émissions.

Pour les échanges, achats et ventes de ces tokens, nous avons choisi par manque de temps de déléguer cela aux exchanges de NFT, tels qu'OpenSea ou Rarible par exemple.

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

Expliquez en quoi votre projet nécessite l'usage d'une blockchain, ainsi que la pertinence de votre projet dans l'écosystème :

Le projet CarbonThink ne révolutionne rien, de nombreux concurrents existent déjà et sont très bien implantés (pour ne citer qu'eux) :

- Regen Network
- KlimaDAO
- CarbonMark
- Carboneutre
- Carbonable
- …

Comme expliqué précédemment pendant la mise en contexte, notre entreprise dans le besoin a des demandes de ses clients pour plus de transparence dans la gestion de leurs projets, la disponibilité des informations et l'assurance de données immuables.

Tous ces facteurs créent de la confiance envers les clients, qui seront alors plus à-même d'investir dans les projets de plantation de bambou.

Des structures concurrentes ont créé des tokens dédiés à leur plateforme, nous restons sur quelque chose de plus simple et direct avec uniquement de l'émission de tokens correspondant aux Crédits Carbone.

Ce que le projet CarbonThink tente d'apporter de plus est une sécurité aux projets qui se lancent avec les agriculteurs, porteurs de projet. Pour chaque projet, nous prélevons 20% des Crédits Carbone générés dans un Fond de Sécurité, qui servira d'assurance dans le cas où des catastrophes arrivent et qui mettent à mal la bonne réalisation des projets. C'est donc l'assurance pour les agriculteurs partenaires de ne pas travailler pour rien en cas d' échec.

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

Précisez le périmètre précis de votre MVP qui, bien entendu, n'englobe pas nécessairement l'intégralité d'un projet final imaginé :

Pour notre projet, nous avons fonctionné par User Stories. Les US listées ici ont été toutes réalisées :

"As CarbonThink" :

- je veux qu'une partie des tokens mint aille dans le fonds de garantie.
- je veux créer un nouveau projet de décarbonation.
- je veux mint et transférer des tokens au porteur de projet.
- je veux recevoir des royalties des échanges de tokens.
- je veux des coûts de transaction réduits sur mes actions blockchain.
- je veux pouvoir montrer les documents relatifs aux projets.
- je veux pouvoir montrer les photos relatives aux projets.
- je veux montrer combien de tokens ont été générés sur la plateforme.
- je veux gérer mon fonds de sécurité.

"As a User" :

- je veux avoir un dashboard de mes tokens (burn ou non).
- je veux pouvoir brûler mes tokens TCO2 et "compenser" (offset).
- je veux acheter / vendre / envoyer des tokens TCO2.
- je veux pouvoir consulter toutes les données des projets et tokens.

"As a Project Holder" :

- je veux vendre mes TCO2 tokens sur une marketplace.

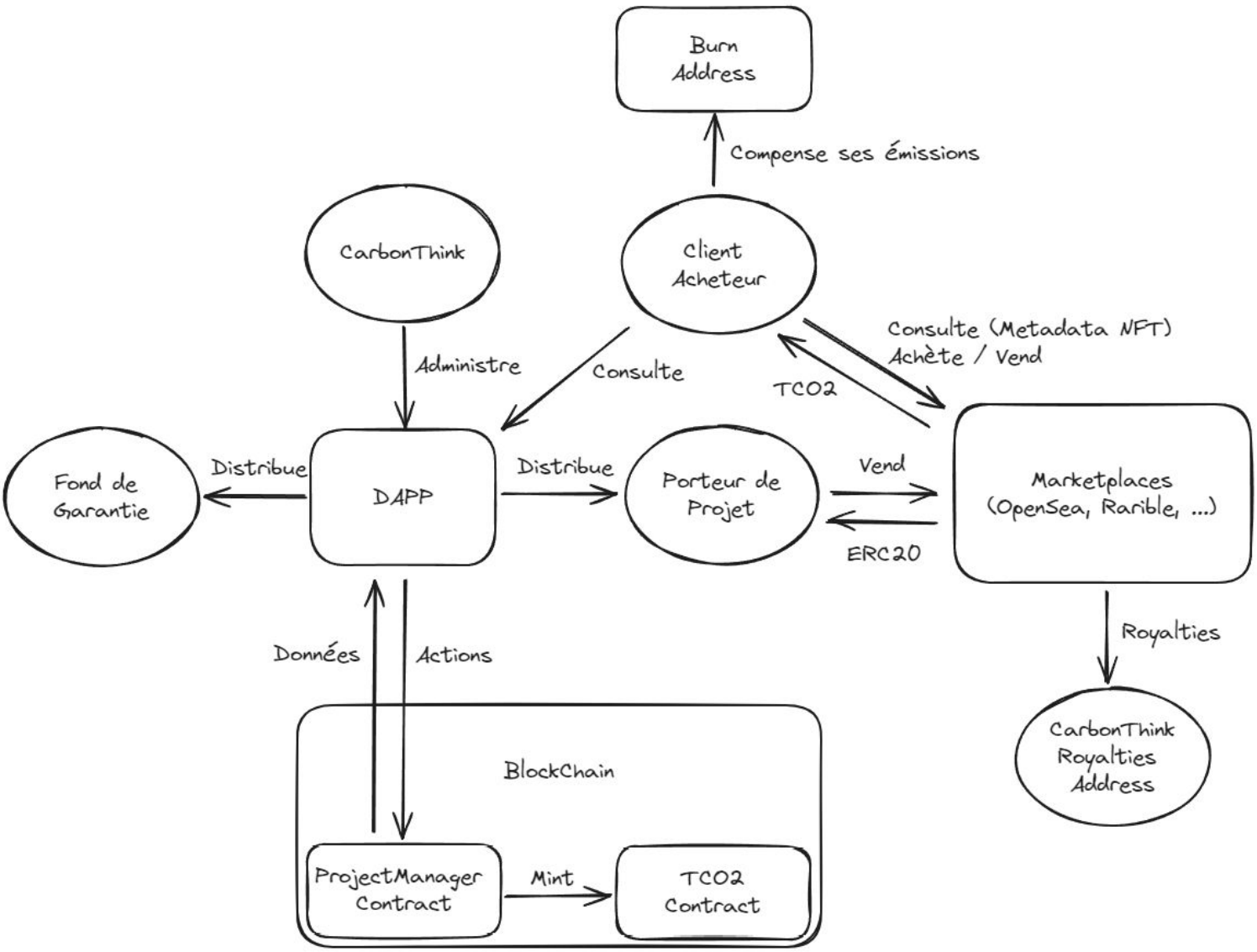
"As an authority" :

- je veux pouvoir vérifier les preuves de réalisation des projets.

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

Réalisez un schéma fonctionnel de votre application. Vous pouvez le réaliser sur Excalidraw.com pour plus de simplicité :



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

Expliquez ici vos smarts contracts : liens entre eux, librairies utilisées, fonctions principales…

Les smart contracts sont au nombre de deux.

- “TCO2”, chargé de la gestion des tokens TCO2.
- “ProjectManager”, chargé de la gestion des projets, owner de TCO2 pour le mint de tokens.

Contrat TCO2 :

Héritages / utils :

- @openzeppelin “Arrays.sol”.
- @openzeppelin “ERC1155.sol”.
- @openzeppelin “ERC1155Burnable.sol”.
- @openzeppelin “ERC1155Supply.so”l.
- @openzeppelin “ERC2981.sol” (royalties).
- @openzeppelin “Ownable.sol”.

Fonctions principales :

- mint (orchestré par ProjectManager).

Autres fonctions :

- Getters divers pour compter les tokens en circulation & burn.

Contrat ProjectManager :

Héritages / contrats:

- @openzeppelin “Ownable.sol”.
- Référence au contrat TCO2.

Fonctions principales :

- addDocument.
- addPhoto.
- create (projet).
- mintTokens (appel vers TCO2).
- setStatus.

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

Expliquez l'usage d'un jeton numérique (fongible ou pas) dans votre application, ainsi que le choix de la norme utilisée :

Token TCO2, collection de NFTs, de norme ERC-1155 :

- ERC1155Burnable: "burn" & "burnBatch".
- ERC1155Supply: "totalSupply" & "exists".
- ERC2981: royalties sur marketplace (au bon vouloir) avec "royaltyInfo".
- Ownable: afin de définir le contrat ProjectManager en tant qu'Owner.

Dans le cas de notre application, nous avons à imiter le fonctionnement des Crédits Carbone dans la vie réelle. Cela implique que nous allons avoir besoin de générer des NFT en batch, parfois à l'unité, parfois par milliers. Nous ne pouvons pas nous permettre d'effectuer ces transferts un à un.

La norme ERC-721 ne convient pas à ce cas d'usage. De plus, nous n'avons pas besoin de dupliquer les données de metadata au niveau de chaque type de token, mais uniquement au niveau de chaque collection. Enfin, pour réaliser des actions par batch (mint, burn, transfer), nous allons pouvoir plutôt nous tourner vers la norme ERC-1155, qui contiendra à un seul endroit nos jetons, plutôt que de passer par exemple par un factory.

Cette norme étant en quelque sorte une "collection de collection" de tokens, les metadata à conserver pour les tokens de chacun de nos projets seront à indiquer au niveau de la collection de tokens, indiquée par un tokenId, grâce à la fonction "uri". Les informations relatives à l'ensemble de la collection de collection seront récupérables grâce à la fonction "contractURI".

Afin de ne pas être dépendant d'autres acteurs et d'avoir des données persistantes, le choix a été fait de conserver toutes les données de metadata on-chain.

Note : les photos et documents sont stockés sur IPFS, donc seul le lien IPFS est stocké sur la blockchain.

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

Décrivez votre méthode d'auto audit, en nommant les failles de sécurité potentielles, et leurs résolutions :

Afin d'assurer la confiance envers nos tokens TCO2, nous avons réutilisé des standards de l'industrie, tels que la norme ERC-1155, Ownable, etc.

La vérification de faille de sécurité a été effectuée en suivant notamment la liste des failles les plus classiques :

- Reentrancy : les deux contrats étant directement ou indirectement gérés par un unique wallet, pas de risque.
- DoS : pas de boucle "for" ni d'Array (sauf "documents" & "photos" dans les struct "Project", mais directement gérés par l'owner).
- Force feeding : pas de condition sur la balance d'ETH dans le contrats.
- Integer underflow / overflow : pas d'utilisation de "int", uint256 utilisé quand nécessaire, optimisations quand ça a été possible.
- Mapping data persistence : pas de "delete" effectué dans les mappings.
- BlockTimestamp manipulation : pas d'utilisation.
- tx.origin : pas d'utilisation.
- etc…

Pour maintenir à un haut niveau la qualité du code des contrats, un système d'audit automatique de sécurité a été ajouté dans la CI/CD : Slither.

Optimisation

Une optimisation du code a été effectuée au niveau des variables de type uint, afin de limiter les coûts en gaz.

L'utilisation d'Array a été limitée au possible : des mappings ont été préférés, avec des variables servant de compteur là où c'était nécessaire (par exemple : quantité de projets au total).

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

Ici nous attendons un descriptif de l'ensemble des outils utilisés pour votre intégration continue.

GitHooks
A chaque commit, un script pre-commit est effectué en local qui exécute :

- lint (backend & front end)
- code spell (backend & front end)
- unit tests (smart contracts)

GitHub Actions
Quand un commit arrive sur Github, il déclenche les Github actions suivant :

- backend-lint
- frontend-lint
- backend-test-hardhat
- slither
- codespell

En simultanée, Vercel watch le repository Github et effectue un déploiement automatique en "production".

Globalement, le repo Github d'OpenZeppelin a été une excellente source d'inspiration pour la mise en place des règles de la CI/CD.

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

Indiquez ici la logique ainsi qu'un résumé des tests réalisés dans le cadre de votre développement.

Les tests ont été réalisés avec Hardhat & Chai sur les deux contrats de notre application, avec une couverture de 100%.

Pour tester nos contrats, j'ai procédé fonction par fonction, en vérifiant chaque variable altérée, chaque event, chaque erreur, et en réalisant à chaque fois :

- Le "happy path".
- Tous les autres embranchements possibles.
- Tous les cas d'erreur.

TCO2:

- constructor
- burnBalanceOf
- burnBalanceOfBatch
- mint
- contractURI
- supportInterface
- totalBalanceOf
- totalBurnBalanceOf
- totalBurnSupply(uint256)
- totalBurnSupply
- uri

ProjectManager:

- constructor
- addDocument
- addPhoto
- create
- get
- mintTokens
- setStatus

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

Indiquez quelles sont les technologies utilisées pour mener à bien une interaction avec votre smart contract depuis une app web, un descriptif des pages réalisées ainsi que des captures d'écran

Technologies :
Typescript, NextJS, React, ConnectKit, Wagmi.
(Visuels : Shadcn, Tailwindcss, Recharts, Confetti, …).

Pages réalisées :

- Page de présentation du projet
- Dashboard : contient les informations sur le total de tokens mint, burn, en circulation, dans le fonds de garantie de la plateforme, mais également des informations sur les tokens de l'utilisateur, et enfin les 5 derniers projets créés.
- Projects : liste avec pagination 10 par 10 de tous les projets de la plateforme.
- Project (page d'info) : affiche toutes les informations relatives au projet, cette page permet également aux administrateurs d'effectuer directement des actions sur les projets en cours (changement de statut, mint, ajout de documents, ajout de photos).
- Marketplace : affiche les liens directs dans les marketplaces OpenSea et Rarible vers les collections de tokens TCO2, ainsi que les derniers échanges effectués (events).
- Création : page uniquement visible aux administrateurs permettant de créer de nouveaux projets.

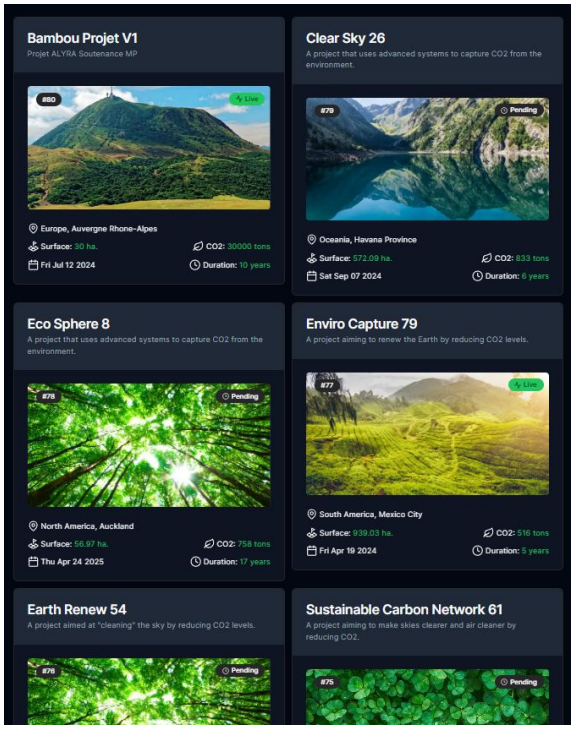
Cette partie du rapport permet d'évaluer la compétence suivante :
C7. Développer un code informatique en utilisant un langage de programmation adapté aux technologies web afin de rendre effective la communication entre une interface web et/ou mobile et un ou plusieurs contrat(s) intelligent(s) (smart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

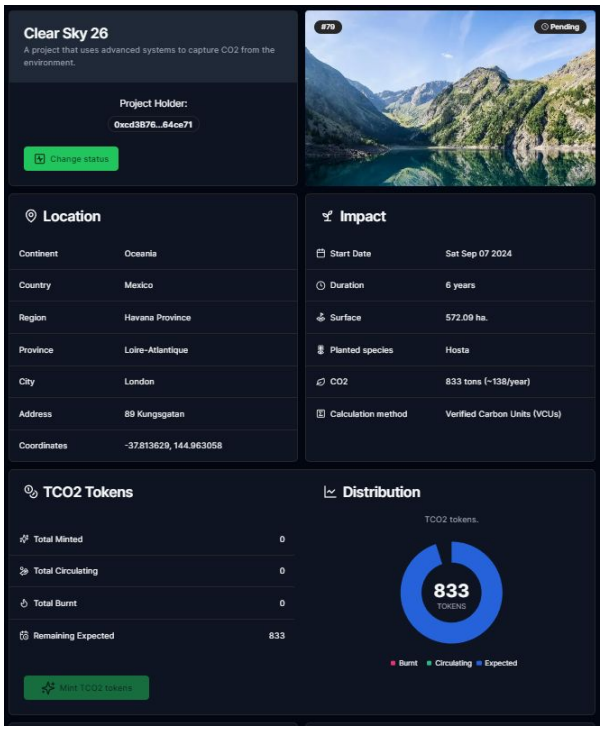
Faites une démo de votre front ici

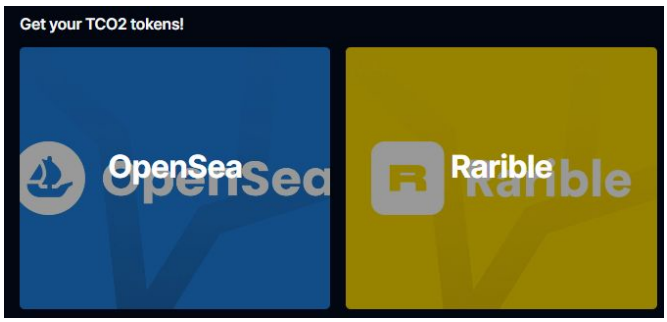
Front 2/2

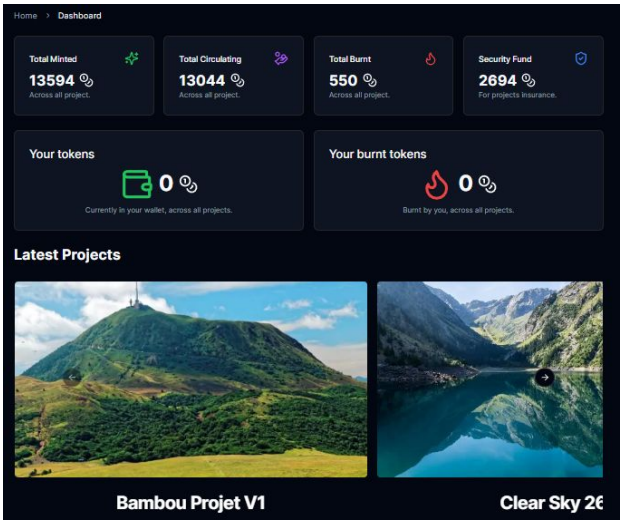
Indiquez quelles sont les technologies utilisées pour mener à bien une interaction avec votre smart contract depuis une app web, un descriptif des pages réalisées ainsi que des captures d'écran

Captures d'écran :









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

Indiquez ici le tooling utilisé pour travailler sur vos smarts contracts, et d'éventuelles procédures de déploiement particulières

Le déploiement du projet a été effectué avec Hardhat Ignition.

Afin d'avoir facilement des données à montrer pour la démo, un script complet de randomization a été développé. Cela permet d'avoir des données diverses et cohérentes à montrer rapidement, sans avoir à passer du temps à les ajouter manuellement.

Pseudo code de la génération :

- Déploiement du contrat TCO2
- Déploiement du contrat ProjectManager
- Donne le "owner" de TCO2 au ProjectManager
- [DEBUT BOUCLE x80]
- Crée un projet, données aléatoires
- Change le statut du projet (aléatoire)
- Mint des tokens pour le projet (nombre aléatoire, si statut "actif")
- Second mint des tokens pour le projet (optionnel et aléatoire).
- [FIN BOUCLE]

Déploiement :
Pour éviter d'éventuelles erreurs d'étourderie, deux fichiers .json ont été créés : un pour le déploiement en localhost, un second pour le déploiement vers Base-Sepolia.

Exemple de commande :

```
pnpm hardhat ignition deploy ignition/modules/ProjectManager.ts --network baseSepolia --verify --parameters ignition/parameters-base-sepolia.json
```

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

Suite à la conception et la réalisation de votre projet, vous expliquerez l'expérience de développement, vos satisfactions ainsi que vos axes d'amélioration pour votre application. Vous devez prendre du recul pour mieux aborder votre projet :

Améliorations possibles :

- Contrairement à ce qui a été indiqué dans les premières ébauches de spécifications, le scope a été largement revu à la baisse.
- La partie “achat” et “vente” de NFT a été déléguée aux exchanges tels qu’OpenSea et Rarible. Impossible donc de forcer les royalties (en tout cas avec la norme ERC2981 utilisée).
- La partie AccessManager, prévue initialement pour ajouter une couche de vérification avant d’effectuer des actions sur les projets ou des mints, n’a pas été réalisée. Elle peut cependant être imitée via la création d’un wallet multi-sig.
- L’interface de Burn des tokens est perfectible, car elle demande à l’utilisateur de se souvenir de la provenance de ses tokens dans son wallet, et d’indiquer l’ID de projet d’où ils proviennent. Il aurait été possible d’ajouter dans le smart contract quelque chose pour lister plus facilement les possessions globales et les projets liés, directement. Cela aurait demandé plus de travail, et surtout des vérifications supplémentaires au niveau sécurité, l’usage d’Array étant une option pour ce besoin.

Bilan de projet :

Globalement, je suis très satisfait du travail accompli. J’ai pu sentir que les consultants de mon équipe avaient enfin vu naître le fruit de leur imagination et qu’ils étaient satisfaits du résultat. J’ai donné le meilleur de moi-même pour réaliser une application intuitive, utilisable, pratique dans la courte période de temps disponible. Je suis convaincu que le travail effectué pourra devenir utilisable dans le monde réel, avec encore un peu de travail.

Conclusion

Retour sur 3 mois de formation, votre apprentissage, vos motivations, vos regrets :

Pour un retour plus global sur la formation Alyra Blockchain, je me sens heureux et chanceux d'avoir eu l'opportunité d'y participer, j'y ai pris beaucoup de plaisir et j'ai essayé de le rendre. Cela faisait longtemps que je désirais effectuer des formations pour essayer et découvrir de nouvelles choses, voilà qui est maintenant fait.

Depuis 2021, j'avais toujours eu de la curiosité par rapport au développement blockchain, sans avoir eu le courage de commencer à me former réellement, par manque de temps.

Une des grandes satisfactions que j'ai pu avoir est que, pendant, la formation, j'ai eu du temps pour approfondir chaque sujet. Cela m'a permis de bien préparer le terrain pour le projet final, et de tout reprendre.

On a su nous transmettre les bons cours, les bonnes infos au bon moment et à la bonne fréquence. La promotion de développeurs Turing a également été très soudée et s'est beaucoup entraidée. Grâce à cette formation, j'ai la satisfaction de pouvoir me dire que je suis maintenant autonome sur le développement d'applications que je pourrai avoir pour des projets personnels, et je sais que la formation est également un excellent tremplin pour travailler dans le web3 par la suite.

Évidemment, mes derniers mots seront un immense merci aux formateurs d'Alyra, Ben et Cyril, pour leur travail acharné et pour nous avoir transmis leur passion. Aussi, un grand merci pour la qualité des intervenants pendant les lives de formation, à la fois passionnés et passionnants.