

Royaume du Maroc
UNIVERSITÉ MOHAMED V - RABAT

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE
ET D'ANALYSE DES SYSTÈMES



Rapport de projet fédérateur

Étude et Test de l'Utilisation de la Technologie Blockchain pour la Préservation de la Privacy dans un Contexte Mobile Crowdsensing

Filière : Sécurité des systèmes d'information (SSI)

Réalisé par :

AGOULZI Imane
JOUIJATE Rim

Encadré par :

Pr. Hanan EL BAKKALI

Membre de Jury :

Pr. Rachida AJHOUN

Année universitaire : 2023 - 2024

Remerciements

Louange à ALLAH seul, que ses bénédictions soient sur notre seigneur et maitre Mohamed et sur les siens.

Tout d'abord nous souhaitons témoigner notre immense gratitude et notre profonde reconnaissance au Professeur **Hanan EL BAKKALI** et Madame **Zaina MAQUOR**, nos encadrantes, qui nous a accompagnées durant la réalisation du projet et qui n'ont épargné aucun effort afin d'assurer une bonne qualité de travail. Leurs conseils avisés, leur large expérience et surtout leur encouragement incessant nous ont été d'une aide précieuse.

Nos gratifications sont aussi adressées à tout le corps professoral et à toute l'équipe pédagogique de l'École Nationale Supérieure d'Informatique et d'Analyse des Systèmes de nous avoir passé une meilleure formation qui nous a permis d'exécuter nos missions de manière juste et efficace.

Et pour terminer, nous n'oublions pas d'adresser nos vifs remerciements à nos familles qui nous offrent la motivation et le soutien nécessaires pour mener à bien nos projets.

Résumé

Ce projet vise à proposer une solution pour augmenter le niveau de la protection de la vie privée des participants dans les applications de Mobile Crowdsensing, et cela à travers l'exploitation des avantages offerts par la technologie Blockchain, tels que la décentralisation, l'immuabilité et la transparence.

La solution proposée comprend la conception d'une application mobile de Mobile Crowdsensing basée sur l'utilisation d'une Blockchain privée à travers Ethereum, qui gère le cycle de MCS à grâce aux contrats intelligents.

Cette solution combine donc l'efficacité opérationnelle du MCS avec la sécurité renforcée de la blockchain.

Mots-clés : Blockchain, contrat intelligent, Ethereum, Mobile crowdsensing, Vie privée.

Abstract

This project proposes a solution to increase the level of participant's privacy in Mobile Crowdsensing applications, by exploiting the advantages offered by Blockchain technology, such as decentralization, immutability and transparency.

The proposed solution includes the design of a Mobile Crowdsensing application based on the use of a private Blockchain through Ethereum, which manages the MCS cycle through smart contracts.

This solution combines the operational efficiency of MCS with the enhanced security of blockchain.

Keywords :Blockchain, Ethereum, Mobile Crowdsensing, Privacy, Smart Contract.

Table des matières

Introduction générale	1
1 Contexte général de projet	2
1.1 Introduction	3
1.2 Contexte du projet	3
1.3 Étude de l'existant et Problématique	3
1.4 Objectifs	4
1.5 Planification	4
1.6 Conclusion	5
2 La blockchain et le Mobile Crowdsensing	6
2.1 Introduction	7
2.2 La blockchain	7
2.2.1 Histoire et origine	7
2.2.2 Définition de la blockchain	7
2.2.3 Principes Fondamentaux de la Blockchain	7
2.2.4 Composants de la Blockchain	8
2.2.4.1 Noeud	8
2.2.4.2 Bloc	9
2.2.4.3 Mécanismes de Consensus dans la Blockchain :	11
2.2.4.4 Types de réseaux Blockchain	11
2.2.5 Type de Blockchain	12
2.2.6 Contrats Intelligents	13
2.3 Mobile crowdsensing	13
2.3.1 Qu'est ce que le Mobile crowdsensing ?	13
2.3.2 Processus du Mobile Crowdsensing	14
2.3.3 Catégories des applications de Crowdsensing	14
2.3.4 Les défis du Mobile crowdsensig	15
2.4 Conclusion	16
3 Analyse et conception	17
3.1 Introduction	18
3.2 Intégration de la Blockchain au MCS	18
3.2.1 Qu'apporte la Blockchain au MCS ?	18
3.2.2 Modèle de MCS basé sur la Blockchain	18
3.2.3 L'architecture de solution proposée	19
3.2.4 Les contrats intelligents	20
3.3 Intégration de la solution proposée à une application mobile de MCS	24
3.3.1 Description de l'application	24
3.3.2 Diagramme de cas d'utilisation	24
3.3.3 Les fonctionnalités de l'application	25

3.3.4	Diagramme de séquence	26
3.3.5	Les outils	27
3.4	Conclusion	28
4	Réalisation	29
4.1	Introduction	30
4.2	Déploiement et test des contrats intelligents	30
4.3	Application mobile de MCS basée sur la Blockchain	38
4.3.1	Présentation des défis confrontés	38
4.3.2	Les interfaces de l'application	38
4.4	Conclusion	44
	Conclusion Générale et Perspectives	45

Liste des sigles et acronymes

API	<i>Application Programming Interface</i>
GPS	<i>Global Positioning System</i>
MCS	<i>Mission Control System</i>
OS	<i>Operating System</i>
PoC	<i>Proof of Concept</i>
PoS	<i>Proof of Stake</i>
PoW	<i>Proof of Work</i>
WSNs	<i>Wireless Sensor Networks</i>

Table des figures

1.1	Diagramme de Gantt	4
2.1	Types de nœuds	8
2.2	Structure des blocs	10
3.1	Le modèle de crowdsensing basé sur la Blockchain	19
3.2	L'architecture du système	19
3.3	Diagramme de cas d'utilisation	25
3.4	Diagramme de séquence	27
4.1	Les comptes Ganache	30
4.2	Les contrats deployés sous Ganache	31
4.3	Ajout des travailleurs	32
4.4	Ajout des demandeurs	32
4.5	Les participants ajoutés sous Ganache	33
4.6	Création des tâches	33
4.7	Les tâches créées sous Ganache	34
4.8	Le travailleur sélectionné pour la réalisation de la tâche	35
4.9	Upload des données	35
4.10	Validation des données par le demandeur	35
4.11	Distribution de récompense - cas de données non valides	35
4.12	Distribution de récompense - cas de données valides	36
4.13	Les Transactions effectuées	36
4.14	Les blocs ajoutés à la Blockchain	37
4.15	Les événements déclenchés	37
4.16	Page d'inscription	38
4.17	Page de connexion	39
4.18	Interface principale de demandeur	40
4.19	Formulaire pour créer une nouvelle tâche	40
4.20	Validation de la tâche	41
4.21	Interface de demandeur après la validation des tâches	41
4.22	Les détails de tâche	42
4.23	Interface principale de travailleur	42
4.24	Les détails d'une tâche	43
4.25	Étapes de réalisation d'une tâche	43
4.26	Interface de travailleur	44

List of Algorithms

1	L'ajout d'un nouveau participant	22
2	Création de Tâche	22
3	Gestionnaire MCS	23
4	Sélection du travailleur	24

Liste des tableaux

- 3.1 Tableau de notation 21
- 4.1 Les participants (Demandeur et Travailleur) 32
- 4.2 Les tâches créées 34

Introduction générale

Grâce à l'évolution numérique surtout au niveau de technologies mobiles et des capteurs embarqués, de nouvelles perspectives passionnantes ont été introduites, notamment dans le domaine du Mobile Crowdsensing (MCS). Cette approche collaborative permet la collecte de données en temps réel à partir de capteurs présents sur les appareils mobiles des individus, offrant ainsi un aperçu détaillé de l'environnement qui les entoure.

Cependant, malgré les avantages considérables du MCS, trois défis majeurs se manifestent fréquemment : la gestion efficace de l'énergie sur les appareils mobiles, la qualité et la précision des données collectées dans des environnements dynamiques, ainsi que la protection de la vie privée des utilisateurs en vue de la nature sensible des informations collectées.

Par ailleurs, la Blockchain, en étant une technologie de registre distribué, est connue par sa nature décentralisée et immuable qui garantit la sécurité et l'intégrité des données. Certes, elle est initialement développée pour sécuriser les transactions financières décentralisées, mais de nos jours elle s'est étendue pour couvrir diverses applications dans des différents domaines, y compris le domaine du MCS. En fait, ses avantages résident dans sa capacité à assurer la confidentialité des données par à travers la décentralisation et de l'immuabilité. Ces caractéristiques éliminent le besoin d'une autorité centrale de confiance, ce qui garantit aussi la transparence et la sécurité des données. En ajoutant qu'elle offre une piste de traçabilité, facilitant donc la vérification de l'origine et de l'intégrité des données collectées.

Ce projet se concentrera sur l'intégration de la Blockchain dans le contexte du MCS. L'approche suivie consiste à concevoir une solution qui bénéficie de la décentralisation et de l'immuabilité de la Blockchain pour renforcer la sécurité des données sensibles collectées dans le MCS, et de tester par la suite l'efficacité de cette intégration à travers le développement d'une simple application mobile de MCS, contribuant ainsi à l'avancement de la recherche sur la préservation de la vie privée dans ce domaine en constante évolution.

Ce document sera structuré en quatre chapitres. Le premier abordera le contexte général et les objectifs à atteindre en mettant la lumière sur la problématique de projet. Ensuite, le deuxième sera dédié à la présentation des technologies de la Blockchain et du MCS. Puis, le troisième chapitre, concerne l'analyse et la conception de la solution proposée pour intégrer la technologie Blockchain à MCS . Et finalement, le dernier chapitre se concentrera sur l'évaluation et l'analyse des résultats obtenus après l'implémentation de la solution.

Chapitre 1

Contexte général de projet

1.1 Introduction

Ce chapitre vise à expliquer le contexte général du projet. En commençant par une analyse de l'existant et en mettant en lumière sur les préoccupations en matière de confidentialité et protection de la vie privée qui ont émergé avec l'évolution de MCS. Après, en exposant les objectifs de cette étude, la planification suivie pour les atteindre.

1.2 Contexte du projet

La vie privée, dans le contexte numérique moderne, se réfère au droit fondamental d'un individu de contrôler les informations personnelles le concernant. Il englobe la capacité de décider quelles données sont partagées, avec qui, et dans quelles circonstances. Ce concept complexe va au-delà de la simple confidentialité des informations, englobant également la protection de la sphère personnelle et des activités quotidiennes.[1]

En effet, la protection de la vie privée a toujours une importance majeure dans le monde numérique. C'est là où repose la confiance des individus dans l'utilisation des services en ligne, des applications mobiles et d'autres plateformes numériques. En garantissant la liberté individuelle et le respect de la dignité, elle permet aux personnes de maintenir un contrôle sur leur identité en ligne et de prévenir tout usage abusif de leurs données personnelles.

Donc, les applications reposant sur la collecte des données des individus doivent leur montrer la certitude que leurs informations personnelles sont traitées de manière responsable, et de contribuer non seulement à établir une relation de confiance avec eux mais également à respecter les normes éthiques et légales entourant la confidentialité des données.

1.3 Étude de l'existant et Problématique

Les applications de Mobile Crowdsensing, exploitant les capteurs des appareils mobiles pour recueillir des données en temps réel, ont connu une grande évolution. Elles couvrent de plus en plus divers domaines tels que la santé, l'environnement et la mobilité urbaine. Cette diversité offre une richesse de données précieuses qui peuvent contribuer de manière significative à la compréhension de divers aspects de la société moderne et de différents phénomènes environnementaux et naturels.

Cependant, cette évolution s'accompagne aussi d'une augmentation des défis liés à la vie privée des utilisateurs de ces applications. En effet, la collecte de données peut comprendre des informations sensibles comme la localisation géographique, les habitudes de déplacement et les interactions sociales, augmentant ainsi les inquiétudes des individus ce qui impacte significativement le niveau d'utilisation en décourageant les gens de participer. Cela peut être causé par le manque de transparence quant aux méthodes de collecte, de stockage et de partage de ces données.

Par ailleurs, la centralisation des serveurs dans de nombreuses applications MCS expose ces données à des risques potentiels de sécurité, car la concentration des informations, qui peuvent être sensibles, en un seul endroit expose ces serveurs aux différentes vulnérabilités qui peuvent être exploitées par des utilisateurs malveillants, en mettant en danger alors la confidentialité des données collectées.

Alors, pour garantir le succès de processus de MCS, il faut motiver les utilisateurs à participer et augmenter leur confiance dans ces applications, en leur garantissant un bon niveau de protection de leurs vies privées.

1.4 Objectifs

Ce projet vise à atteindre les objectifs suivants :

- Analyser et comprendre de manière approfondie les problèmes spécifiques liés à vie privée dans le contexte du MCS.
- Acquérir une compréhension approfondie des principes fondamentaux de la technologie Blockchain.
- Proposer une solution qui vise à résoudre les défis de MCS en matière de sécurité et vie privée, en intégrant efficacement la technologie Blockchain.
- Mettre en œuvre de manière pratique la solution conçue, à travers une application mobile de MCS basée sur la Blockchain Ethereum.
- Évaluer et analyser les résultats obtenus.

1.5 Planification

Afin d'atteindre nos objectifs, il fallait d'abord commencer par découvrir la technologie Blockchain et le MCS pour construire des connaissances qui vont nous permettre de comprendre la problématique de projet.

Ensuite, nous avons essayé d'approfondir ces connaissances pour comprendre le comment des choses, et cela à l'aide de deux formations : la première concerne les principes et les pratiques fondamentaux de la blockchain et la création des smart contracts en utilisant le langage de programmation Solidity, et la deuxième fournit une compréhension supplémentaire en matière développement mobile basé sur la blockchain.

Puis, nous avons entamé la conception de l'architecture proposée dans ce projet tout en initiant parallèlement la rédaction du présent rapport.

Le diagramme de Gantt 1.1 résume cette planification en fonction de l'intervalle du temps que nous avons eu pour travailler sur ce projet :

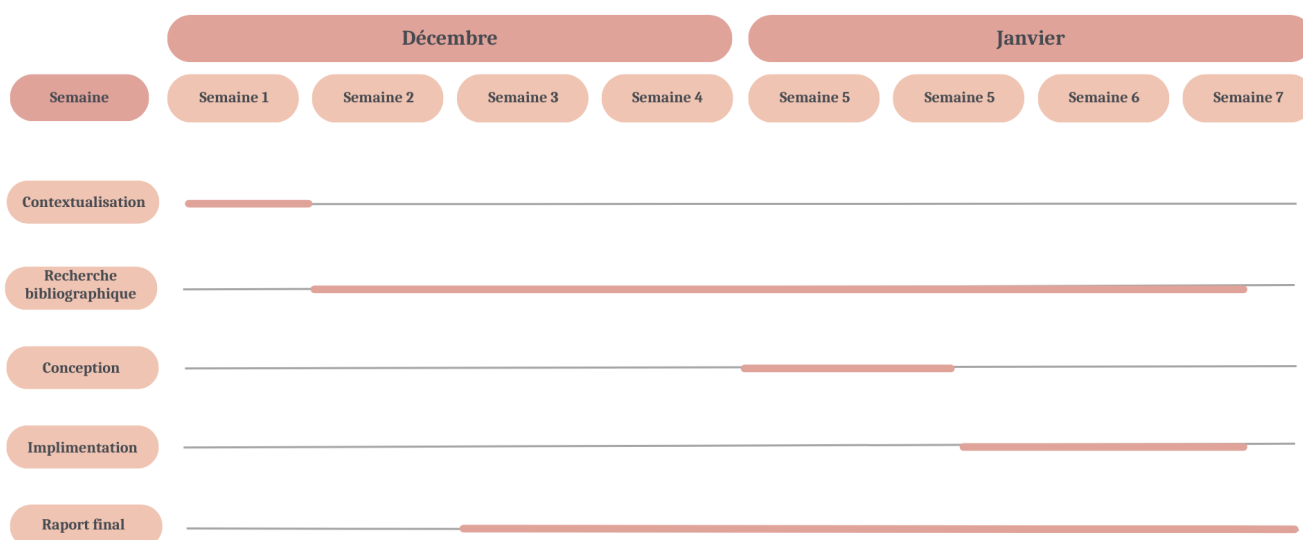


FIGURE 1.1 – Diagramme de Gantt

1.6 Conclusion

Ce chapitre a donné une vue d'ensemble sur ce projet en exposant son contexte, sa problématique, et les objectifs à réaliser. De plus, la planification joue un rôle majeure dans le succès d'un projet, donc dans ce sens une planification détaillée avec un diagramme de Gantt a été présentée.

Chapitre 2

La blockchain et le Mobile Crowdsensing

2.1 Introduction

Ce chapitre offre une connaissance approfondie des deux technologies : la blockchain et le mobile crowdsensing. La blockchain, avec son histoire, ses principes fondamentaux, et ses composants, constitue le premier volet de cette exploration. Nous passerons ensuite au mobile crowdsensing, examinant sa définition, son processus, les catégories d'applications, et les défis associés.

2.2 La blockchain

2.2.1 Histoire et origine

Les débuts de la blockchain remontent à 1991, lorsque qu'un groupe de chercheurs a conceptualisé une idée révolutionnaire : une chaîne de blocs liés ensemble pour stocker des informations de manière sécurisée. Malgré son introduction précoce, ce concept novateur est resté relativement dormant jusqu'en **2009**, date à laquelle une figure anonyme connue sous le nom de **Satoshi Nakamoto** lui a donné vie. Nakamoto a adapté la blockchain pour créer la première cryptomonnaie numérique, le Bitcoin. Cela a marqué le tournant qui a propulsé la blockchain du potentiel théorique à une application concrète dans le monde réel. [2]

2.2.2 Définition de la blockchain

La blockchain est une infrastructure distribuée, utilisée pour enregistrer de manière sécurisée des transactions et assurer la traçabilité des données. Son architecture décentralisée garantit l'intégrité des informations stockées, éliminant ainsi le besoin d'une autorité centrale. Les transactions, une fois enregistrées sur la blockchain, deviennent immuables et accessibles à tous les participants du réseau. Chaque transaction est regroupée dans un bloc, et ces blocs sont liés de manière chronologique, formant ainsi une chaîne de blocs, d'où le nom "blockchain". [3]

2.2.3 Principes Fondamentaux de la Blockchain

La blockchain, en tant que technologie révolutionnaire, repose sur plusieurs principes fondamentaux qui la distinguent et qui sont essentiels à sa compréhension. [3]

- **Décentralisation** : La décentralisation est un principe clé de la blockchain. Contrairement aux systèmes centralisés traditionnels, la blockchain fonctionne sans autorité centrale. Les informations et les transactions sont réparties sur un réseau de nœuds, ce qui élimine le besoin d'une entité de contrôle unique.
- **Cryptographie** : La sécurité de la blockchain repose sur des techniques avancées de cryptographie. Chaque transaction est cryptée, assurant la confidentialité et l'intégrité des données. Les utilisateurs peuvent vérifier la légitimité des transactions grâce à des clés cryptographiques.
- **Consensus** : Les mécanismes de consensus sont utilisés pour valider les transactions et garantir la cohérence du registre distribué. Des méthodes telles que la preuve de travail ou la preuve d'enjeu sont employées pour s'assurer que toutes les parties du réseau sont d'accord sur l'état du système.
- **Immutabilité** : Une fois qu'une transaction est ajoutée à un bloc et que ce bloc est ajouté à la chaîne, il devient extrêmement difficile de modifier les données. Cette immutabilité garantit l'intégrité de l'historique des transactions.

- **Transparence** : Toutes les transactions enregistrées sur la blockchain sont visibles par tous les participants du réseau. Cette transparence favorise la confiance entre les parties prenantes, car chacun peut vérifier les informations.
- **Chronologie et Liens** : Chaque bloc de la blockchain est lié au bloc précédent, créant une chronologie des transactions. Ce lien garantit que l'ordre des événements est clair et vérifiable.
- **Distribution** : Les données de la blockchain sont réparties sur l'ensemble du réseau de nœuds, ce qui réduit les risques de défaillance ou de manipulation. Chaque nœud détient une copie complète du registre, ce qui renforce la résilience du système.

2.2.4 Composants de la Blockchain

2.2.4.1 Nœud

Un nœud est simplement un dispositif qui participe au réseau blockchain. On entend souvent dire que les blockchains sont composées de nœuds pair-à-pair, et nous utilisons ces nœuds pour interagir avec la blockchain. Nous ne communiquons pas directement avec le nœud, mais nous avons la blockchain au niveau 1, l'API au niveau 2 et le nœud au niveau 3.

Il existe deux types de nœuds [4] :

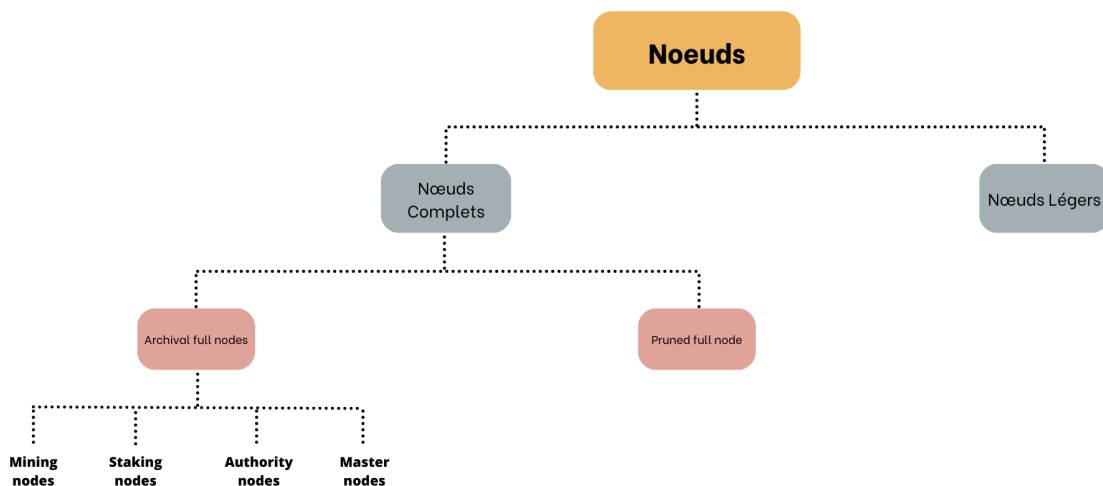


FIGURE 2.1 – Types de nœuds

- Nœuds complets :
 - Archival Full Nodes : Conservent une copie complète de l'historique des transactions d'une blockchain, depuis le bloc génésis jusqu'au plus récent. Ils sont responsables de l'application des règles de l'algorithme de consensus et participent à la validation des transactions.
 - Mining Nodes (Mineurs) : Participent au processus de création de nouveaux blocs dans les blockchains de preuve de travail et reçoivent en retour des récompenses sous forme de cryptomonnaie nouvellement créée.
 - Staking Nodes (Validateurs) : Participent au processus de validation de nouveaux blocs, mais dans les blockchains de preuve d'enjeu. Ces nœuds ont leur crypto-actif

- bloqué, les rendant financièrement peu incitatifs à agir de manière malhonnête dans le réseau.
- Authority Nodes (Nœuds d'autorité) : Utilisés dans un réseau partiellement décentralisé. Les validateurs qui exécutent ces nœuds sont généralement choisis par l'entité derrière le réseau.
 - Master Nodes (Nœuds maîtres) : Contrairement aux précédents, ils ne peuvent pas ajouter de blocs à la blockchain, mais valident et conservent uniquement les enregistrements des transactions.
 - Pruned Full Node (Nœud complet élagué) : Version allégée d'un nœud complet archivé ; les nœuds élagués commencent par télécharger l'intégralité de la blockchain, vérifient qu'elle est correcte, puis stockent uniquement les blocs les plus récents jusqu'à une limite prédéterminée, par exemple, 5 gigaoctets. Cela réduit l'espace disque nécessaire pour stocker la blockchain et l'exécuter.
 - Nœuds légers : Ne nécessitent pas de matériel puissant ni une bande passante élevée, permettant un accès à la blockchain via des appareils tels que des ordinateurs portables et des smartphones. Ils sont utilisés dans les opérations quotidiennes de la blockchain, comme la vérification des soldes ou la diffusion de transactions, mais n'appliquent pas les règles de consensus réseau et stockent uniquement les données pertinentes aux transactions en cours, en comptant sur les nœuds complets pour leur fournir d'autres informations nécessaires.

2.2.4.2 Bloc

Le premier bloc de la chaîne de blocs est appelé le **bloc Genesis**. Chaque bloc de la blockchain contient des livres de compte, une collection de transactions, où chaque bloc peut contenir zéro ou plusieurs transactions. Les transactions sont ajoutées à la blockchain lorsqu'un nœud publie un bloc.

Avant d'explorer les spécificités d'un bloc, revenons aux fondamentaux. Les registres (Ledgers) sont le fondement de tout système financier, et dans le contexte de la blockchain, ce sont une collection décentralisée de **transactions**. Chaque transaction est une interaction numérique, un mouvement de valeur, et ce sont ces transactions qui forment le contenu central d'un bloc. [3]

Dans l'architecture globale de la blockchain, un bloc est bien plus qu'une simple collection de transactions qui contient généralement :

- Information sur les Transactions : Au cœur de chaque bloc se trouve un registre, une collection de transactions. Ces transactions représentent des interactions numériques, des mouvements de valeur entre les participants du réseau blockchain. Chaque bloc encapsule un instantané de ces interactions, créant une chaîne chronologique d'événements.
- Information sur les Participants : Un bloc n'est pas seulement un enregistrement de transactions ; c'est un témoignage des participants dans le réseau. Chaque participant est identifié par ses signatures numériques et adresses uniques, laissant une empreinte indélébile sur la composition du bloc.
- Information sur les Autres Blocs : La beauté de la blockchain réside dans sa connectivité. Les informations sur les autres blocs sont tissées dans la structure de chaque bloc. Les fonctions de hachage cryptographiques opèrent leur magie, créant un identifiant unique pour le bloc, souvent appelé le hachage du bloc. Il est dérivé des données complètes du bloc, assurant ainsi que chaque bloc est intrinsèquement lié à ses prédécesseurs.

La figure 2.2 montre la structure du bloc dans un bloc de la blockchain [3] :

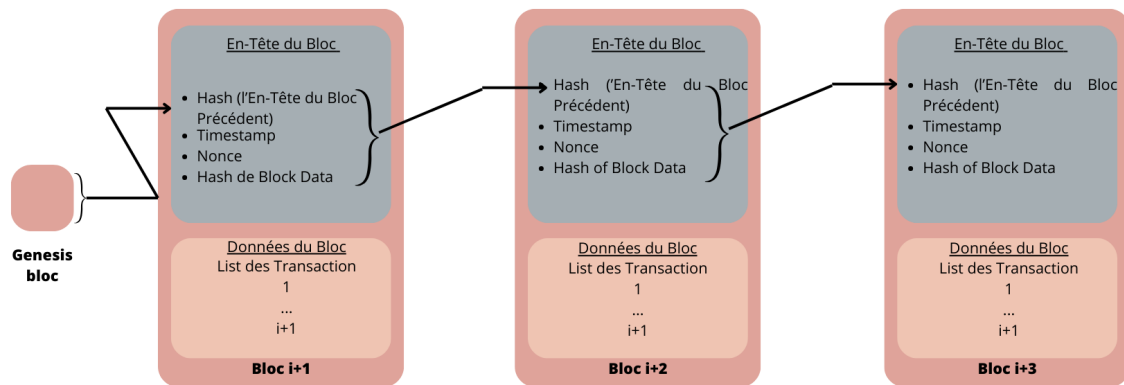


FIGURE 2.2 – Structure des blocs

- **En-Tête du Bloc** : L'en-tête d'un bloc est comparable au plan d'un bâtiment, un élément crucial qui définit sa structure. Voici ce que comprend généralement l'en-tête d'un bloc :
 - **Numéro de Bloc** : L'identifiant unique attribué à chaque bloc dans l'ordre chronologique de la blockchain.
 - **Valeur de Hachage de l'En-Tête du Bloc Précédent** : Reliant les blocs comme une chaîne, cette valeur de hachage lie le bloc actuel à son prédécesseur, établissant un flux continu au sein de la blockchain.
 - **Représentation de Hachage des Données du Bloc** : C'est la signature cryptographique de l'ensemble des données du bloc. Des méthodes comme la génération d'un arbre de Merkle et le stockage du hachage racine, ou l'utilisation d'un hachage de toutes les données combinées du bloc, sont utilisées pour garantir l'intégrité du bloc.
 - **Horodatage** : Un horodatage marque le moment où un bloc est ajouté à la blockchain, ajoutant une dimension temporelle au grand livre.
 - **Taille du Bloc** : Indique la quantité de données contenue dans le bloc, assurant la cohérence et l'efficacité de l'opération de la blockchain.
 - **Valeur de Nonce** : Un nombre arbitraire utilisé dans le processus d'extraction, contribuant à l'unicité du hachage du bloc.
- **Données du Bloc** : Englobe la substance du bloc, abritant une liste de transactions validées et authentiques soumises au réseau blockchain. Cette section inclut :
 - Une liste de transactions et d'événements de registre inclus dans le bloc.
 - D'autres données pouvant être présentes, selon l'implémentation spécifique de la blockchain.

Pour qu'un Bloc Assure sa Place dans la Blockchain, Quatre Étapes Précises se Déroulent :

- **Une Transaction Doit Se Produire** : Le récit commence par une transaction, une conversation numérique entre les participants du réseau blockchain. Que ce soit le transfert de cryptomonnaie ou l'exécution d'un contrat intelligent.
- **Vérification de la Transaction** : Chargés du rôle de vérification, les mineurs scrutinent chaque transaction, s'assurant qu'elle respecte les règles du réseau et possède les ré-

férences nécessaires. Les transactions qui réussissent cet examen rigoureux passent au chapitre suivant.

- Stockage dans un Bloc : Les transactions validées trouvent refuge dans un bloc. Ces blocs agissent comme des vaisseaux numériques, capables d'encapsuler un nombre fini de transactions. À l'intérieur de leurs limites, les transactions attendent leur tour pour être ajoutées dans la blockchain.
- Attribution d'un Hachage : le bloc est doté d'un identifiant unique, un hachage cryptographique. Ce hachage encapsule l'ensemble des données du bloc. Avec ce hachage, le bloc est prêt à prendre sa place dans la séquence majestueuse de la blockchain.

NB : Dans la blockchain, l'idée conventionnelle de créer un nouveau bloc pour chaque transaction cède la place à une approche plus pragmatique. Pour améliorer la scalabilité et l'efficacité opérationnelle, les réseaux blockchain regroupent souvent plusieurs transactions en un seul bloc. Cette pratique optimise l'utilisation des ressources, réduisant les coûts informatiques et favorisant la scalabilité. Plutôt que de créer un bloc pour chaque transaction, les architectes de réseau créent stratégiquement des blocs à des intervalles prédéfinis ou des seuils de transactions, trouvant un équilibre entre la confirmation en temps réel et l'impératif de maintenir une infrastructure scalable et rationalisée. Cette stratégie nuancée garantit que les réseaux blockchain peuvent gérer des volumes croissants de transactions sans compromettre les performances ni surcharger le système avec un excès de petits blocs.

2.2.4.3 Mécanismes de Consensus dans la Blockchain :

Il existe divers mécanismes utilisés dans la blockchain, mais nous allons juste mentionner les plus utilisés dans le monde de la Blockchain. [3]

- La Preuve de Travail (PoW) est un mécanisme de consensus où les mineurs rivalisent pour résoudre des énigmes mathématiques complexes afin de valider les transactions et d'ajouter de nouveaux blocs à la blockchain. Ce processus intensif en énergie requiert un matériel spécialisé. Les mineurs utilisent leur puissance de calcul pour résoudre une énigme cryptographique liée à un bloc, et le premier à réussir obtient le droit d'ajouter le bloc à la blockchain. À mesure que plus de mineurs rejoignent le réseau, les énigmes deviennent plus difficiles pour maintenir un taux constant de création de blocs. Bien que la PoW offre une sécurité éprouvée et une décentralisation, elle est critiquée pour sa consommation énergétique élevée et le risque de centralisation à mesure que le minage devient une opération à grande échelle industrielle.
- La Preuve d'Enjeu (PoS) est un mécanisme de consensus qui sélectionne des validateurs pour créer de nouveaux blocs en fonction de la quantité de cryptomonnaie qu'ils bloquent en guise de garantie. Contrairement à la PoW, la PoS est plus économe en énergie. Les participants bloquent une certaine quantité de cryptomonnaie comme garantie, et plus ils en bloquent, plus ils ont de chances d'être choisis comme validateurs. Les validateurs sont sélectionnés de manière aléatoire et vérifient les transactions avant de créer de nouveaux blocs. La PoS offre une efficacité énergétique, un potentiel de décentralisation et une évolutivité, mais elle peut favoriser les participants riches en raison de la corrélation entre les cryptomonnaies bloquées et le pouvoir de validation. Les validateurs doivent maintenir un fonctionnement impeccable, sinon ils risquent des sanctions, comme la réduction de la quantité de cryptomonnaie bloquée en cas de comportement malveillant.

2.2.4.4 Types de réseaux Blockchain

Dans le monde de la blockchain, différents types de réseaux émergent pour répondre à des besoins spécifiques. Voici quelques catégories de réseaux blockchain basées sur les informations fournies [5] :

- Un réseau blockchain public est accessible à n'importe qui et permet à tous de participer, comme c'est le cas pour Bitcoin. Cependant, des inconvénients peuvent inclure la nécessité d'une puissance de calcul substantielle, peu ou pas de confidentialité pour les transactions et une sécurité relativement faible. Ces considérations sont essentielles pour les cas d'utilisation en entreprise de la blockchain.
- Un réseau blockchain privé, similaire à un réseau public, est un réseau décentralisé de pair à pair. Cependant, un organisme gouverne le réseau, contrôlant qui est autorisé à participer, exécuter un protocole de consensus et maintenir le grand livre partagé. Selon le cas d'utilisation, cela peut renforcer considérablement la confiance entre les participants. Un blockchain privé peut fonctionner derrière un pare-feu d'entreprise et même être hébergé sur site.
- Réseaux Blockchain Permissionnés : Les entreprises qui mettent en place un blockchain privé mettront généralement en place un réseau blockchain permissionné. Il est important de noter que les réseaux blockchain publics peuvent également être permissionnés. Cela impose des restrictions sur qui est autorisé à participer au réseau et dans quelles transactions. Les participants doivent obtenir une invitation ou une autorisation pour rejoindre.
- Consortiums Blockchain : Plusieurs organisations peuvent partager les responsabilités de la maintenance d'une blockchain. Ces organisations présélectionnées déterminent qui peut soumettre des transactions ou accéder aux données. Un consortium blockchain est idéal pour les entreprises lorsque tous les participants doivent être autorisés et partager la responsabilité de la blockchain.

2.2.5 Type de Blockchain

Examinons les types de blockchains à travers des descriptions simples et compréhensibles. Nous avons trois types principaux de blockchains [3] :

- Blockchain Publics : Les réseaux blockchain publics, tels que Bitcoin et Ethereum, opèrent sans restrictions sur la participation et la validation des transactions. Ils offrent une gouvernance décentralisée, où chaque participant a une autorité égale. Ces réseaux garantissent la sécurité des données et contribuent à l'immuabilité des enregistrements grâce à une distribution totale. Cependant, cela peut entraîner des défis tels que des besoins computationnels élevés, une faible confidentialité des transactions et une sécurité parfois moins robuste.
- Blockchain Privés : À l'opposé, les réseaux blockchain privés, à l'instar de Hyperledger, imposent des restrictions sur les participants, nécessitant des invitations pour rejoindre le réseau. Les transactions sont visibles uniquement par les membres autorisés, permettant un contrôle centralisé et une meilleure régulation. Dans ces réseaux, le token interne peut être manipulé selon les préférences du propriétaire, offrant une personnalisation accrue. Cela renforce la confidentialité, mais au détriment de la décentralisation.
- Réseaux Blockchain Hybrides : Les réseaux blockchain hybrides, comme Ripple, combinent des éléments de public et de privé. Certains nœuds sont privés, tandis que d'autres sont publics. Certains participent activement aux transactions, tandis que d'autres contrôlent le processus de consensus. Deux types d'utilisateurs interagissent avec ce type de réseau : ceux qui détiennent le contrôle total sur la blockchain, décidant du niveau de sécurité pour chaque utilisateur, et ceux qui accèdent simplement à la blockchain conformément à leurs fonctions. Cela permet une certaine personnalisation en fonction des besoins spécifiques du projet, cherchant à équilibrer la transparence et le contrôle.

2.2.6 Contrats Intelligents

Un contrat intelligent est similaire à un contrat dans le monde physique, mais il est numérique et représenté par un petit programme informatique stocké à l'intérieur de la blockchain. Plus spécifiquement, c'est un morceau de logiciel qui stocke des règles pour négocier les termes d'un accord, vérifie automatiquement l'accomplissement, puis exécute les termes convenus.

Les contrats intelligents éliminent la dépendance à l'égard d'une tierce partie lors de l'établissement de relations commerciales. Les parties qui concluent un accord peuvent effectuer des transactions directes entre elles.

NB : Les contrats intelligents peuvent remplacer n'importe quelle autre tierce partie de confiance.

La question qui se pose est pourquoi faire confiance à un contrat intelligent ?, la réponse est simple, les contrats intelligents sont conçus et implémentés dans la blockchain, donc ils héritent de certaines propriétés de la blockchain. Les lignes les plus proéminentes incluent :

- Ils sont immuables.
- Ils sont distribués.

2.3 Mobile crowdsensing

2.3.1 Qu'est ce que le Mobile crowdsensing ?

Le Mobile crowdsensing (MCS) est défini comme la participation d'un groupe d'individus, disposant de terminaux mobiles intelligents qui, collectivement, partagent des informations pour la mesure ou la cartographie de phénomènes d'un intérêt commun.

Le Crowdsensing, est connu sous le nom de "Mesures participatives", c'est un paradigme de mesures basé sur la puissance de divers dispositifs et capteurs mobiles. Il est utilisé pour acquérir des connaissances locales à travers des capteurs performants intégrés dans ces dispositifs, au moment où les individus les utilisent pour partager leurs données en public, et d'en extraire des informations pour mesurer et cartographier les phénomènes d'intérêt commun. [7]

La technologie MCS a permis d'étendre avec succès la portée de la détection de l'espace physique unique à l'échelle de la communauté et de la ville, de la reconnaissance des situations environnementales dangereuses à l'information sur le comportement collectif des foules.

Typiquement, le MCS offre de nombreux avantages par rapport aux méthodes classiques de collecte de données telles que les réseaux de capteurs (WSNs), qui peuvent engendrer des coûts considérables liés à l'installation et à la maintenance d'un grand nombre de capteurs statiques. En effet, des millions de dispositifs mobiles sont déjà déployés dans la nature, portés par des utilisateurs dans leur vie quotidienne (Smartphones, tablettes, etc.) . Grâce à la généralisation des magasins d'applications (comme App Store, Google Play, etc.), qu'il s'agisse d'équipes de recherche ou de petites entreprises, ont désormais la possibilité de rapidement diffuser une application mobile auprès de cette vaste communauté d'utilisateurs. Par exemple, au lieu d'installer un ensemble de caméras le long des routes, il est possible de collecter des données du trafic routier et de détecter les embouteillages en utilisant le GPS des dispositifs mobiles des conducteurs.

Par ailleurs, un autre avantages de l'utilisation de ces dispositifs mobiles et qu'ils contiennent un bon nombre de capteurs multimodaux ce qui permet une collecte d'information contextuelle de haut niveau. En fait, le GPS peut fournir la localisation géographique précise d'un utilisateur, l'accéléromètre et le gyroscope permettent de détecter les mouvements et donc de définir les activités journalières, le capteur de température aide à effectuer des surveillances environnementales, le microphone sert à la surveillance acoustique et l'analyse des niveaux sonores, la caméra permet une surveillance visuelle et la reconnaissance d'objets, etc.

Le MCS bénéficie de l'intelligence humaine en transformant les smartphones par exemple, en instruments de collecte de données distribuée. En incitant les utilisateurs à partager activement des informations via leurs capteurs, le MCS exploite la diversité des expériences individuelles pour fournir des données contextuelles riches qui peut dépasser les capacités des capteurs automatisés.

2.3.2 Processus du Mobile Crowdsensing

En règle générale, le cycle de vie du MCS se compose de quatre étapes : la création de tâches, l'attribution de tâches, l'exécution de tâches et le traitement de données. [8] [9]

1. Création de tâches (Task creation) : Après avoir reçu les exigences des demandeurs de tâches (telles que l'heure, le lieu, le type, le budget, etc.), la plateforme MCS construit des tâches MCS. À ce stade, la principale question de recherche est de savoir comment décrire efficacement les tâches sans connaissances spécialisées, afin de maximiser la participation.
2. Attribution des tâches (Task assignment) : Dans cette étape, la plateforme MCS recrute des travailleurs et leur assigne des tâches de détection spécifiques à effectuer sur leurs terminaux. La principale difficulté à ce stade est de trouver suffisamment de personnes qualifiées pour participer aux tâches de détection. En raison de divers facteurs (comme la vie privée), certains travailleurs peuvent refuser les tâches qui leur sont assignées.
3. Exécution des tâches individuelles (Individual Task Execution) : Une fois qu'il a reçu la tâche de détection qui lui a été assignée, un participant tente de la terminer en respectant les critères prédéfinis (par exemple : la durée, l'emplacement, etc.) parallèlement à d'autres tâches. Cette étape peut être divisée en trois sous-étapes : detection, computing et uploading des données.
4. Traitement des données (Data processing) : Cette étape prend en compte et traite les flux de données collectés auprès de tous les participants, y compris le nettoyage des données, l'agrégation et l'évaluation de la qualité. En outre, l'inférence des données manquantes est également une préoccupation importante du MCS. Par ailleurs, pour certaines applications MCS, le traitement des données à ce stade est assez simple : des serveurs centraux stockent les données et fournissent des interfaces aux utilisateurs finaux pour l'interrogation et le partage des données. D'autres applications MCS utilisent des algorithmes complexes pour intégrer les données et extraire une intelligence collective de haut niveau à partir des données brutes de grandes foules.

2.3.3 Catégories des applications de Crowdsensing

Il existe un grand nombre d'applications de collecte de données, mettant en évidence l'utilisation du MCS dans une grande variété de domaines. Ces applications peuvent être classées selon deux critères : [10]

- Le sujet à observer.
- Le mode d'acquisition des données.

Le classement selon le sujet à observer :

Ce classement se scinde, lui encore, en deux dimensions :

- Les applications de collecte **personnelle** : visent à observer des phénomènes liés à un individu et à analyser son comportement. Toutes les données collectées sont généralement couplées avec un identifiant unique, permettant de suivre l'évolution d'un utilisateur spécifique à travers le temps. Ce type de collecte est généralement utilisé dans les domaines de la santé, du sport ou de l'environnement.

- Les applications de collecte **communautaire** : visent à observer des phénomènes à plus grande échelle, sur l'environnement (par ex. qualité réseaux, pollution atmosphérique ou sonore) ou sur une infrastructure publique (e.g., trafic routier, place de parking disponible, dégâts de matériel public), afin d'améliorer la vie d'une communauté (par ex. toutes les personnes habitant dans une ville, les étudiants d'une université).

Le classement selon le mode d'acquisition des données :

Un autre facteur qui détermine le succès d'une application de MCS, Indépendamment du sujet à observer, est le niveau d'implication des utilisateurs dans le processus de collecte. Ce processus peut prendre deux formes :

- Collecte **participative** : , l'utilisateur est directement impliqué dans la prise de décision de la collecte en décidant quand, comment et quelles données doivent être collectées. Le principal avantage de cette approche est qu'elle permet de bénéficier de l'intelligence humaine pour réaliser des opérations plus complexes. Mais, l'inconvénient est que la qualité des données, ainsi que la fréquence à laquelle elles sont collectées, dépend fortement de l'enthousiasme des utilisateurs pour l'application.
- Collecte **opportuniste** : est complètement automatisée par l'application mobile, sans nécessiter une intervention de l'utilisateur. Cela est particulièrement utile dans le cadre d'une collecte communautaire, permettant d'assurer la propagation de données régulièrement sur le serveur (sans que l'utilisateur soit contraint à accomplir des tâches quotidiennes sur son dispositif). Cependant, ces applications sont plus difficiles à développer, et consomment également beaucoup de ressources énergétiques.

2.3.4 Les défis du Mobile crowdsensing

Dans cette section, nous abordons les principaux défis de mobile crowdsensing. Nous identifions cinq défis majeurs [10] :

- Sécurité et Vie privée : Respecter la vie privée des utilisateurs est fondamental pour une plateforme de collecte de données. Les utilisateurs sont sensibles aux données collectées sur leurs dispositifs, surtout si ces informations incluent des localisations, des images sensibles ou des communications audio. Plusieurs approches, telles que l'utilisation d'alias pour les communications ou la possibilité pour les utilisateurs de définir leurs préférences en matière de vie privée, ont été envisagées pour protéger la confidentialité tout en maintenant la qualité des données.
- Gestion des ressources : Malgré l'augmentation des capacités des smartphones, les ressources énergétiques restent limitées. Optimiser la consommation énergétique en utilisant la multimodalité des capteurs, adaptant la fréquence d'échantillonnage en fonction de la batterie de l'utilisateur, est une approche pour réduire la consommation d'énergie. Cependant, ces méthodes peuvent compromettre la qualité des données, nécessitant un équilibre entre qualité et confidentialité.
- Hétérogénéité des équipements et des OS : L'hétérogénéité des systèmes d'exploitation et des dispositifs constitue un défi majeur. Le développement spécifique pour chaque OS, comme Android, iOS et Windows Mobile, demande du temps et des ressources. Concevoir des applications de collecte dans un langage de haut niveau pour supporter diverses plateformes mobiles devient nécessaire pour minimiser la complexité du développement.
- Diffusion et passage à l'échelle des applications : La diffusion efficace des applications est cruciale pour le passage à l'échelle d'une plateforme de collecte. Les mécanismes de déploiement doivent prendre en compte la diversité des utilisateurs, limiter le nombre d'utilisateurs impliqués dans la collecte, assurer la confidentialité, et permettre des mises à jour rapides pour répondre aux exigences évolutives.

- Implication et incitation des usagers : L'implication des utilisateurs est essentielle pour la collecte de données, mais les utilisateurs doivent être motivés. Des modèles économiques, tels que des récompenses financières ou des incitations non monétaires comme des jeux ou des récompenses virtuelles, sont explorés pour encourager la participation. Cependant, la collecte à grande échelle peut nécessiter un budget considérable, suscitant l'exploration de modèles alternatifs basés sur des motivations citoyennes ou scientifiques.
- Équilibre entre la qualité des données collectées et la préservation de la vie privée : En effet, l'amélioration de la qualité des données peut parfois compromettre la confidentialité des utilisateurs. Par exemple, pour garantir la vie privée, il peut être nécessaire de masquer certaines données, ce qui peut impacter négativement la qualité des informations disponibles. À l'inverse, si l'accent est mis sur la qualité des données en les présentant de manière détaillée, cela peut potentiellement compromettre la vie privée des personnes concernées.
- Incitation Équitable : Un autre défi crucial réside dans la mise en place d'un système d'incitation équitable pour motiver la participation des utilisateurs au Mobile Crowdsensing. En effet, les utilisateurs peuvent hésiter à partager leurs données, craignant une utilisation abusive ou non autorisée de leurs informations personnelles. De plus, le manque de certitude quant à une rétribution équitable après avoir accompli une tâche constitue également une source d'appréhension pour les participants potentiels.

2.4 Conclusion

Dans ce chapitre, nous avons présenté les fondements des deux technologies. Cependant, l'exploration n'est pas terminée, car la prochaine étape de notre projet consistera à examiner de près la manière dont ces deux technologies peuvent converger pour créer une solution novatrice permettant de surmonter les défis de MCS en termes de sécurité et vie privée.

Chapitre 3

Analyse et conception

3.1 Introduction

Ce chapitre se focalise sur l'analyse et la conception de notre projet. Nous explorerons l'intégration de la technologie Blockchain dans le contexte du Mobile Crowdsensing (MCS). De plus, nous détaillerons le modèle et l'architecture de la solution proposée où nous examinerons aussi les aspects liés aux Smart Contracts.

3.2 Intégration de la Blockchain au MCS

3.2.1 Qu'apporte la Blockchain au MCS ?

L'adoption de Mobile Crowdsensing soulève des défis cruciaux, comme déjà expliqué précédemment, surtout en matière de sécurité et confidentialité. En effet, sa nature distribuée et participative expose les participants à des risques potentiels liés à la vie privée, l'authentification décentralisée de ces participants, révélation d'informations personnelles sensibles, etc.

Par ailleurs, la blockchain émerge comme une solution efficace pour atténuer ces défis de sécurité et de confidentialité dans le MCS. Elle offre une plateforme idéale pour instaurer un environnement de confiance, en introduisant des mécanismes qui renforcent la protection de la vie privée, et ce à travers [6] :

- Une authentification décentralisée : les participants au MCS s'inscrivent sur la blockchain, puis des identifiants cryptographiques uniques sont générés pour identifier ces participants, et seront vérifiés de manière décentralisée, éliminant donc le besoin d'une tierce partie de confiance.
- Anonymat : les participants sont identifiés par leurs ID généré, donc leurs identités réelles ne sont pas connues ni divulguées.
- Confidentialité : les données collectées sont cryptées à l'aide de clés privées, alors seuls ceux avec les clés appropriées peuvent y accéder.
- Transparence et traçabilité : chaque transaction et chaque étape du cycle de vie MCS sont enregistrées de manière immuable sur la blockchain, ce qui assure aux participants un historique transparent et une traçabilité complète.
- Les contrats intelligents : permettent l'automatisation des transactions sur la blockchain en fonction des conditions prédéfinies. Cela inclut toutes les phases de cycle de vie de MCS, garantissant des transactions transparentes et sécurisées.

3.2.2 Modèle de MCS basé sur la Blockchain

Dans cette section, nous présentons le modèle, qui est une architecture décentralisée de crowdsensing basée sur la blockchain privée, comme le montre la figure 3.1. Tous les utilisateurs interagissent les uns avec les autres sous forme de transactions, et tous les enregistrements de transactions sont stockés sur la blockchain. Pour mettre en œuvre ces transactions, chaque entité impliquée doit disposer d'un compte.

Il existe trois rôles impliqués pour la gestion du réseau blockchain :

- Mineur (Miner) : responsable de la vérification des enregistrements de transaction nouvellement générés et de leur ajout au dernier bloc. De plus, un mineur peut changer de rôle, par exemple en devenant un demandeur qui publie une tâche ou un travailleur qui accepte une tâche.
- Demandeur (Requester) : définit le contenu de la tâche et les récompenses correspondantes, puis publie la tâche dans une annonce sur la blockchain pour la collecte et le traitement des données.

- **Travailleur (Worker)** : un participant qui termine la tâche publiée par le demandeur sur la blockchain. Le travailleur peut choisir la tâche qu'il souhaite accomplir, et une fois accepté, il peut télécharger les données de détection sur la blockchain. Après la vérification des données, le travailleur peut recevoir la récompense envoyée par le demandeur. En raison de l'anonymat de la blockchain, tous les travailleurs peuvent effectuer des tâches sans révéler leur véritable identité.

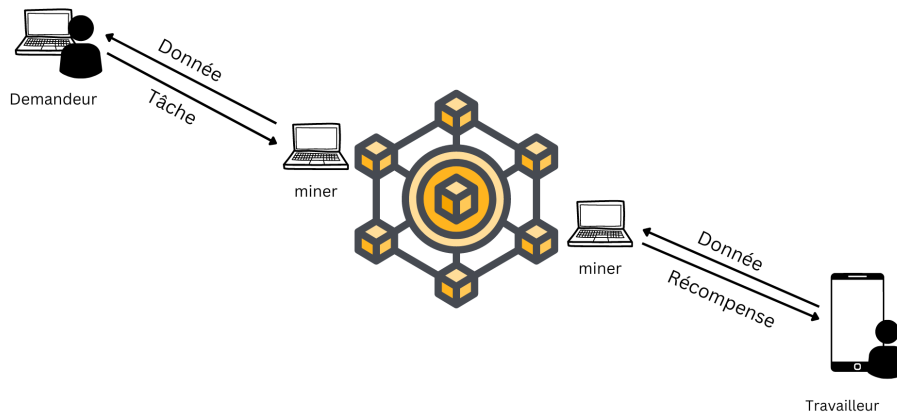


FIGURE 3.1 – Le modèle de crowdsensing basé sur la Blockchain

3.2.3 L'architecture de solution proposée

La figure 3.2 illustre l'architecture proposée et les différentes interactions des participants (travailleur et demandeur) avec une Blockchain privée :

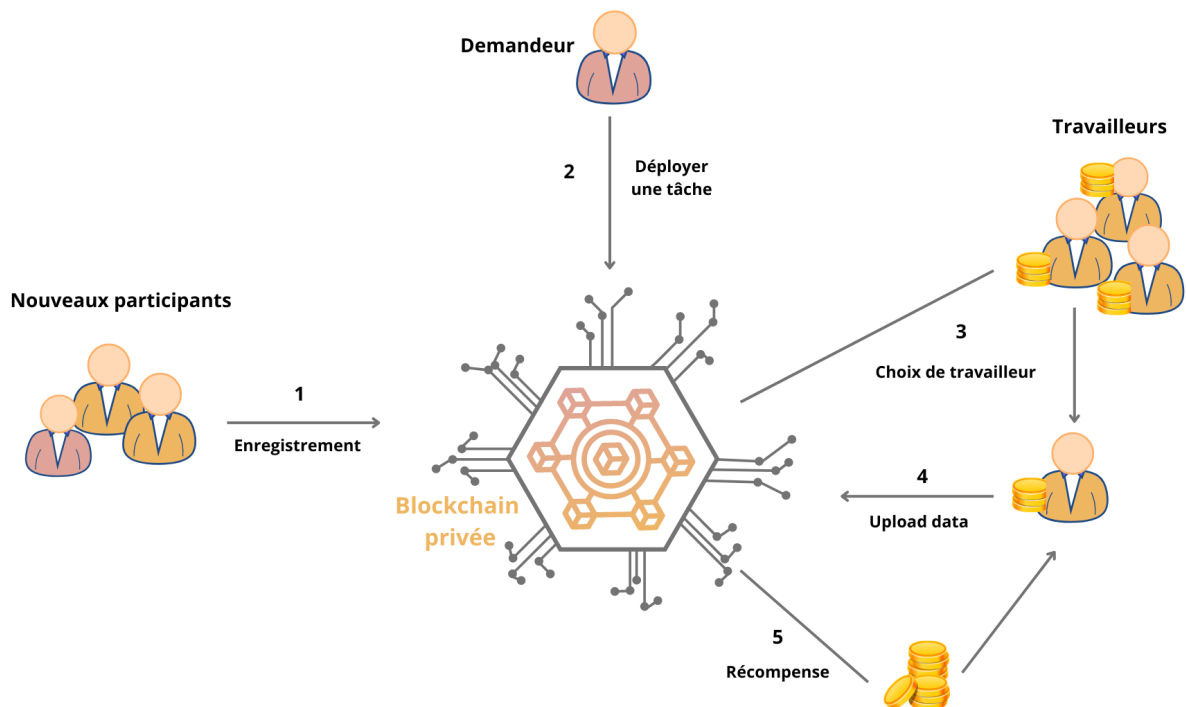


FIGURE 3.2 – L'architecture du système

Maintenant, nous allons détailler chaque transaction et ses spécificités :

1. L'enregistrement de participants : Tous les utilisateurs mobiles qui souhaitent participer au système doivent s'inscrire, que ce soit travailleur ou demandeur. Une paire de clés est ensuite attribuée à chaque utilisateur enregistré, et l'identifiant qui lui est associé est stocké dans la réserve d'utilisateurs. Les autres informations d'enregistrement de l'utilisateur sont enregistrées sous forme de transactions dans le grand livre de la blockchain. Dans un réseau blockchain, l'identité de l'utilisateur est représentée par la clé publique générée, sans révéler son identité réelle, et c'est elle qui est utilisée pour les transactions.

En résumé, un participant :

- Envoie une demande d'enregistrement à la Blockchain.
 - La blockchain reçoit la demande et s'assure que l'utilisateur n'est pas déjà enregistré. Pour procéder à l'enregistrement, la blockchain demande les données de l'utilisateur.
 - L'utilisateur envoie sa clé publique comme paramètre d'authentification.
 - La blockchain enregistre ces données.
2. Déploiement des tâches : Le demandeur publie des tâches sur la blockchain et fixe une récompense pour chaque tâche, en plus d'autres informations reliées à la tâche comme son temps limite, le type de données désirées.
 3. Choix des travailleurs : Les travailleurs déjà enregistrés sur la blockchain reçoivent les tâches publiées par les demandeurs. Le système par la suite déclenche le contrat intelligent pour choisir les travailleurs les plus appropriés pour réaliser la tâche selon l'algorithme et les règles définies dans ce contrat qui prennent en considération le contexte de la tâche et les informations de travailleur.

Dans ce projet, nous avons opté vers un cas d'étude simple où le choix du travailleur est basé sur le nombre de points qu'il possède. c'est-à-dire le travailleur qui a réalisé plus de tâches avec succès donc qu'il a cumulé le grand nombre de points.

4. Upload de données collectées : Une fois que le travailleur termine sa tâche, il upload les données collectées sur la Blockchain, et un contrat intelligent est déclenché, si la tâche prend plus de temps que prévu, le contrat est résilié et la tâche échoue. Sinon, ces données seront stockées dans la Blockchain et le demandeur peut les obtenir en les décryptant, puis il vérifie leur qualité.

Par ailleurs, la manière dont la qualité des données est évaluée dépasse le cadre du présent document. Dans notre cas, deux niveaux de qualité seront inclus : valide et non valide. Si les données répondent aux exigences définies par le demandeur (valide), le travailleur peut recevoir sa récompense, sinon (non valide) le travailleur ne reçoit aucune récompense et les données seront détruites.

5. Attribution des récompenses : Il s'agit de la dernière étape du processus, au cours de laquelle une récompense est attribuée aux travailleurs inscrits lorsqu'ils ont réussi la tâche assignée. Un contrat intelligent sera automatiquement déclenché pour payer le travailleur par le biais de sa clé publique correspondante. La réputation des travailleurs qui ont fourni des efforts importants et de bonnes performances pour l'accomplissement de la tâche sera améliorée.

3.2.4 Les contrats intelligents

Pour faciliter l'interaction entre le demandeur et le travailleur, nous utiliserons un contrat intelligent comme tiers. Le contrat intelligent agit comme un intermédiaire, garantissant l'exécution sécurisée et automatisée de l'accord entre les deux parties.

Nous allons utiliser trois contrats intelligents :

- Contrat d'inscription des utilisateurs (Travailleur ou Demandeur) : ce contrat intelligent prend en entrée les informations d'un nouvel utilisateur à travers la fonction `addUser()`. Lorsqu'un utilisateur souhaite s'inscrire, cette fonction est utilisée pour enregistrer ses données et l'ajouter à la Blockchain.
- Contrat de création des tâches : ce contrat prend en entrée les détails fournis par un demandeur via la fonction `addTask()`. Ces détails incluent la description de la tâche, le nombre de travailleurs requis, la récompense, le dépôt et le type de données. En retour, cette fonction publie la tâche dans la liste des tâches disponibles.
- Contrat de gestion de cycle MCS : Le contrat MCS gère efficacement le processus de collecte de données en prenant en entrée le demandeur, une liste de travailleurs, et la tâche choisie. Il effectue une sélection de travailleur via `workerSelection()`, importe les données collectées auprès de travailleur via `uploadData()`, valide ces données à l'aide de `dataValidation()`, et enfin distribue les récompenses grâce à `paymentDistribution()`.

Dans le tableau 3.1, nous présentons un ensemble de notations utilisées tout au long du document pour représenter les entités et concepts clés dans le contexte d'un système orienté tâches

Notation	Description
R	Demandeur
W	Travailleur
P_i	Les points cumulés d'un travailleur i
$Pool_W$	Ensemble des travailleurs
$Pool_P$	Ensemble des participants
$Pool_R$	Ensemble des demandeurs
ID_T	Identifiant d'une tâche
ID_W	Identifiant d'un travailleur
ID_R	Identifiant d'un demandeur

TABLE 3.1 – Tableau de notation

Algorithm 1 L'ajout d'un nouveau participant

Entrées : $Type$ - Type de nouveau participant (R ou W)**Sortie :** Résultat de l'enregistrement Res Initier la valeur de Res en $Faux$ $\{P_k, S_k\} \leftarrow$ générer la paire de clés (privée et publique) $ID_{user} \leftarrow P_k$ **Si** $ID_{user} \in Pool_P$ **Alors** $Res \leftarrow Vrai$ **retourner** Res **Sinon****Si** $Type == D$ **Alors** $Pool_R \leftarrow Pool_R \cup \{ID_{user}\}$ **Sinon** $Pool_W \leftarrow Pool_W \cup \{ID_{user}\}$ **Fin Si** $Res \leftarrow Vrai$ **retourner** Res **Fin Si**

L'algorithme 2 a pour but de créer une nouvelle tâche dans le système de crowdsensing. Il prend en entrée les informations du demandeur telles que l'objectif de la tâche, la récompense, le dépôt, le type de données et la date limite. En retour, il génère un identifiant unique pour la tâche créée, crée un nouvel objet tâche avec les détails fournis, l'ajoute au pool de tâches et incrémente le compteur de tâches. L'algorithme renvoie l'identifiant de la tâche nouvellement créée.

Algorithm 2 Création de Tâche

Entrées : ID_R - Demandeur, obj - Objectif de la tâche, $Reward$, $deposit$, $TypeData$, $Deadline$ **Sortie :** ID_T $taskID \leftarrow generateUniqueTaskID()$ \triangleright Générer un identifiant unique pour la tâche

Créer un nouvel objet tâche avec les propriétés suivantes :

— Adresse du demandeur ID_R — Objectif obj — Récompense $Reward$ — Dépôt $Deposit$ — Type de données $TypeData$ — Date limite $Deadline$

— Statut (initialisé à "Disponible" ou statut par défaut)

retourner ID_T

Cet algorithme gère le processus de gestion des tâches entre le demandeur et les travailleurs. On commence par initialiser le statut de la tâche à "Disponible". Ensuite, on choisit le travailleur qui a le plus de points grâce à la fonction **workerSelection()**. Tant que le travailleur n'a pas annulé la tâche, il peut continuer, sinon le statut reste "Disponible". Si les points du travailleur sélectionné sont inférieurs au dépôt requis, le statut reste "Disponible". Si le temps dépasse la date limite, le statut devient "Échouée". Sinon, le travailleur termine la tâche et soumet les données via **UploadData()**. Cette fonction récupère les données, les enregistre et renvoie un identifiant unique. Ensuite, les données sont vérifiées par **DataValidation()**. Si le type ne correspond pas ou si le demandeur l'a marqué comme "Non", les données ne sont pas

valides. Sinon, elles sont validées dans ce cas **PaymentDistribution()** met à jour les points du travailleur en ajoutant la récompense et du dépôt.

Algorithm 3 Gestionnaire MCS

Entrées : ID_R - Demandeur , Liste des Travailleurs L , ID_T

Sortie : Statut de la tâche

Initialiser la valeur de Statut \leftarrow "Disponible"

$ID_W \leftarrow workerSelection(L)$

Si $Point_W < deposit$ **Alors**

retourner Statut \leftarrow "Disponible"

Sinon

Si ID_W annule **Alors**

retourner Statut \leftarrow "Disponible"

Sinon

Si $Temps_T > deadline$ **Alors**

retourner Statut \leftarrow "Échouée"

Sinon

$ID_D \leftarrow UploadData(ID_W, ID_R, ID_T)$

 Feedback $\leftarrow DataValidation(ID_D, ID_T, ID_R)$

Si Feedback == "valide" **Alors**

 paymentDistribution(ID_T, ID_W, ID_R)

retourner Statut \leftarrow "Complète"

Sinon

retourner Statut \leftarrow "Disponible"

Fin Si

Fin Si

Fin Si

Fin Si

Algorithm 4 Sélection du travailleur

Entrée : Liste des Travailleurs L de taille n **Sortie** : ID_W $Max \leftarrow L[0][1]$ $J \leftarrow 0$ **for** i de 1 à n **do** **Si** $Max < L[i][1]$ **Alors** $Max \leftarrow L[i][1]$ $J \leftarrow i$ **Fin Si****Fin for****retourner** ID_W de Travailleur J

3.3 Intégration de la solution proposée à une application mobile de MCS

3.3.1 Description de l'application

Nous envisageons de développer une application mobile basée sur l'architecture définie dans la section précédente. Cette application reposera sur la blockchain dans un souci de confidentialité, garantissant ainsi l'anonymat des utilisateurs. En ce qui concerne la collecte de données, le mobile crowdsensing implique que les utilisateurs de l'application nous fourniront les données collectées par leur application mobile. Ces données peuvent prendre la forme de photos, de vidéos, d'audios ou simplement de texte, en réponse aux tâches proposées par le propriétaire de l'application (demandeur).

L'utilisateur doit d'abord créer un compte pour s'authentifier auprès de l'application. Ainsi, il doit commencer par créer un compte, puis se connecter. Une fois connecté, l'utilisateur accède à un tableau de bord où il peut retrouver les tâches réalisées dans le passé, les points accumulés en accomplissant ces tâches, ainsi que la liste des tâches disponibles à accomplir.

Après avoir choisi une tâche, l'utilisateur attend d'être accepté par le propriétaire de l'application. Une fois accepté, il peut effectuer la tâche en transférer les données nécessaires vers l'application. Ensuite, il voit la valeur des points augmenter s'il a bien réalisé la tâche.

3.3.2 Diagramme de cas d'utilisation

La figure ?? représente le diagramme de cas d'utilisation qui illustre les différentes interactions des utilisateurs avec l'application mobile :

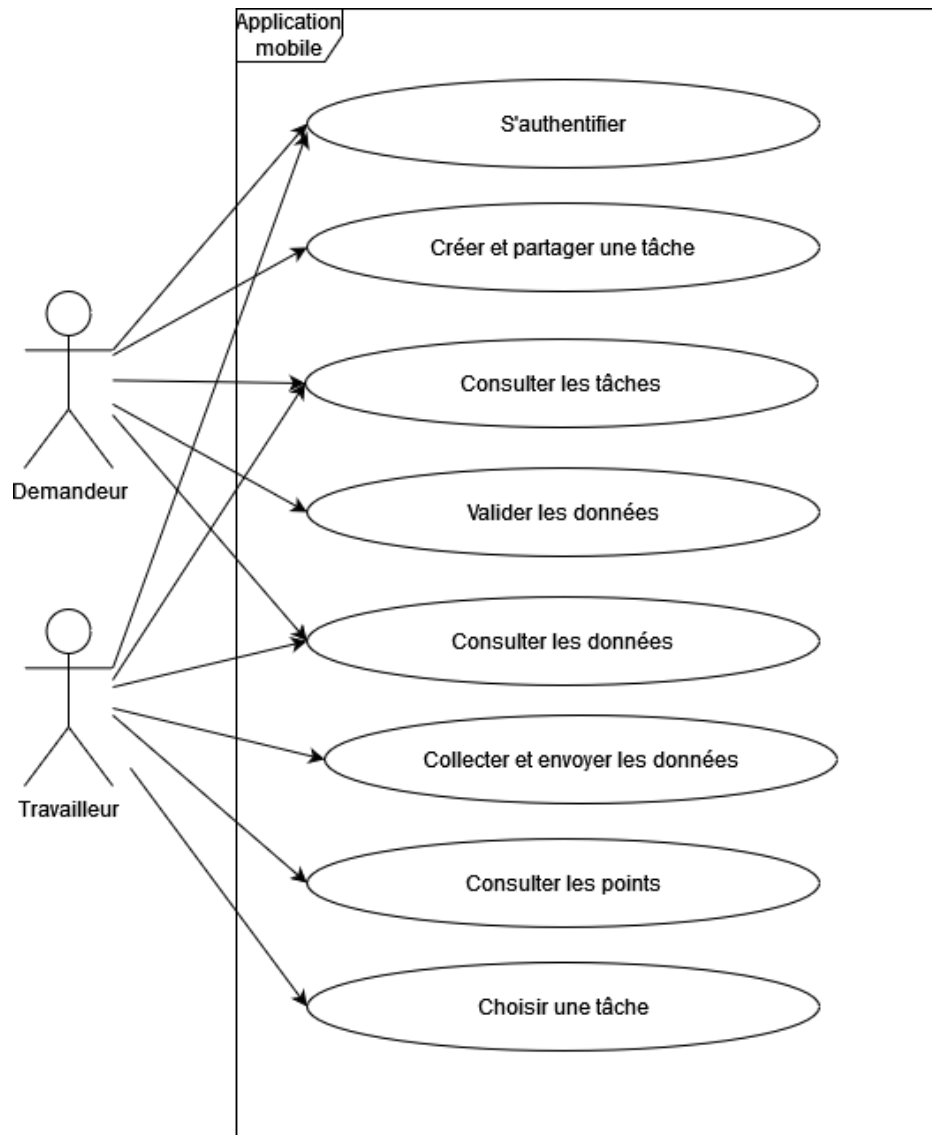


FIGURE 3.3 – Diagramme de cas d'utilisation

3.3.3 Les fonctionnalités de l'application

- Authentification de l'utilisateur : L'authentification de l'utilisateur est le processus initial où l'utilisateur doit créer un compte, en tant que demandeur ou travailleur, en fournissant des informations d'identification.
- Après la connexion, l'utilisateur est dirigé vers une interface d'accueil.
- **Si l'utilisateur est un travailleur :**
 - A travers l'interface d'accueil, il peut consulter l'historique de ses tâches précédemment accomplies, la somme de ses points accumulés, ainsi qu'une liste des tâches actuellement disponibles pour être réalisées.
 - Le travailleur a la possibilité de parcourir la liste des tâches disponibles et de sélectionner celle qui veut réaliser. Cette sélection dépend de la nature de la tâche, des compétences de l'utilisateur et de ses préférences.
 - Après avoir choisi une tâche, une demande est envoyée. Le travailleur doit attendre le résultat du contrat intelligent qui décide si la tâche lui sera assignée ou non.
 - Une fois accepté, l'application Bloque une somme de points pour la tâche choisie, et le travailleur peut uploader les données collectées à travers les capteurs de son appareil mobile.

- Après avoir envoyé ces données, le travailleur doit attendre l'évaluation de données envoyées qui détermine la qualité de sa contribution. Si ses données sont "valides" alors il reçoit sa récompense sous forme de points sur son compte, sinon, il va perdre les points qui a bloqué avant.
- **Si l'utilisateur est un demandeur :**
 - A travers l'interface d'accueil, il peut consulter l'historique de ses tâches précédemment tâches partagées. Il contient aussi une section où le demandeur peut publier une nouvelle tâche.
 - En appuyant sur "Créer une tâche", le demandeur est redirigé vers un formulaire où il peut définir les détails de la tâche. Le formulaire inclut des champs tels que l'objectif de la tâche, le type de données demandées, la date limite de la tâche et la récompense proposée.
 - Il remplit le formulaire en spécifiant les informations requises pour la nouvelle tâche.
 - Une fois le formulaire rempli, le demandeur publie la tâche sur la plateforme de collecte de données, la rendant accessible aux travailleurs.
 - Le demandeur attend la réception des données provenant des travailleurs qui effectuent la tâche. Pendant cette période, il peut suivre le statut de la tâche et recevoir des notifications lorsqu'il y a des soumissions de données.

3.3.4 Diagramme de séquence

Le diagramme de séquence 3.4 illustre le flux d'interaction entre les composants clés de notre application lorsqu'un utilisateur, soit le demandeur ou le travailleur, contribue en interagissant avec l'application.

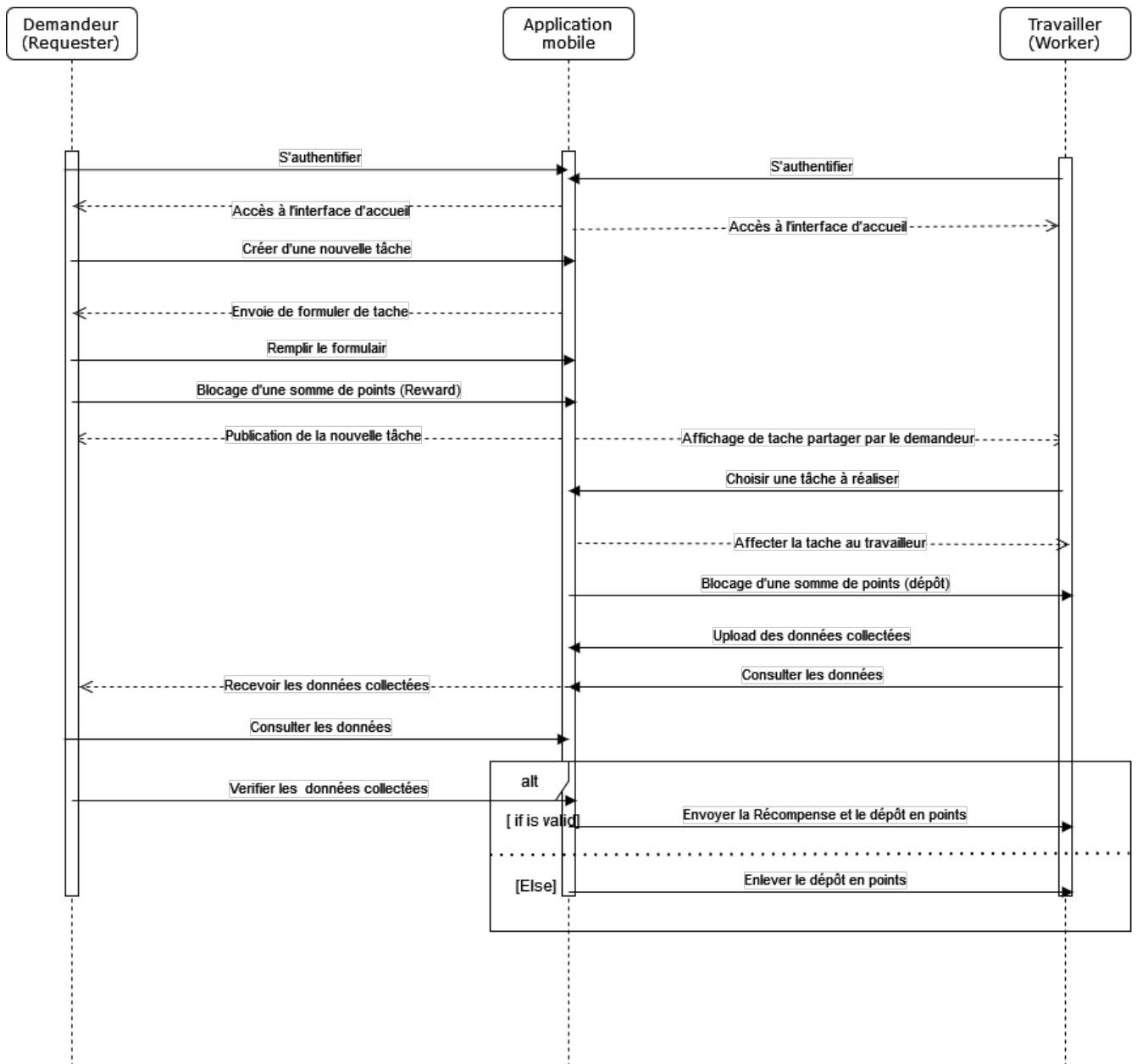


FIGURE 3.4 – Diagramme de séquence

3.3.5 Les outils

Dans le cadre de ce projet, nous avons sélectionné un ensemble d'outils essentiels pour optimiser le développement et le déploiement de notre application sur la blockchain Ethereum.

Le langage de programmation pour les contrats intelligents qui sera utilisé est **Solidity 0.8.17**, en raison de sa conception spécifique pour Ethereum, offrant une syntaxe puissante et une intégration directe avec la plateforme, ce qui garantit une exécution efficace de ces contrats au sein de l'environnement Ethereum.

En ce qui concerne les tests des contrats intelligents, **Truffle v7.9.1**, un framework renommé dans le développement Ethereum, a été choisi. Truffle Develop, un composant intégré de Truffle, fournit un environnement de test créant une blockchain locale, souvent associé à Truffle Suite pour une gestion complète du cycle de vie des contrats intelligents.

Parallèlement, **Ganache v7.9.1** est un environnement de développement Ethereum local. Sa principale utilité réside dans la simulation d'une blockchain privée, facilitant ainsi le développement et les tests sans recourir au réseau principal. En effet, il offre une blockchain locale

dédiée aux tests, avec des comptes virtuels initialement dotés de 100 Ether chacun, facilitant ainsi les simulations de transactions pour assurer la robustesse de notre application.

En combinant Truffle et Ganache, nous nous appuyons sur des outils complémentaires simplifiant le processus de développement, de test et de déploiement des contrats intelligents sur la blockchain Ethereum.

On a choisi comme environnement de développement intégré **Visual Studio Code** qui est éditeur de code léger mais puissant prend en charge le langage Solidity, il permet de réaliser la conception, le test et le débogage efficaces des contrats intelligents, avec des fonctionnalités intégrées pour le déploiement sur la blockchain, assurant une gestion transparente tout au long du cycle de développement.

Pour le développement de l'application mobile, **Android Studio** offre des fonctionnalités avancées, offrant une expérience de développement complète et bien intégrée avec le système d'exploitation Android, assurant ainsi une conception robuste et des performances optimales.

En explorant les moyens d'interagir avec la blockchain via Android Studio, différentes options ont été considérées. **Web3.js v1.10.0**, une bibliothèque JavaScript populaire, offre une interface prisée pour les applications basées sur Ethereum. Cependant, pour une approche plus intégrée avec Java, Ethers.js a été envisagé comme une alternative légère et moderne. Le SDK officiel d'Ethereum pour Android propose une solution native complète, tandis que Web3j, une bibliothèque Java, constitue également un choix solide. Infura, en fournissant des points d'accès Ethereum, offre une solution contournant la nécessité de gérer un nœud complet. Enfin, pour ceux préférant les appels REST, certaines solutions Ethereum exposent des fonctionnalités via des API REST, offrant une approche alternative pour interagir avec la blockchain depuis notre application Android. Cette exploration vise à évaluer la meilleure solution en fonction de nos besoins spécifiques en termes de simplicité d'intégration, de performances et de compatibilité avec notre architecture existante.

3.4 Conclusion

Après avoir exposé en détails la conception de notre projet, le chapitre suivant se consacrera à sa mise en œuvre concrète pour l'implémentation.

Chapitre 4

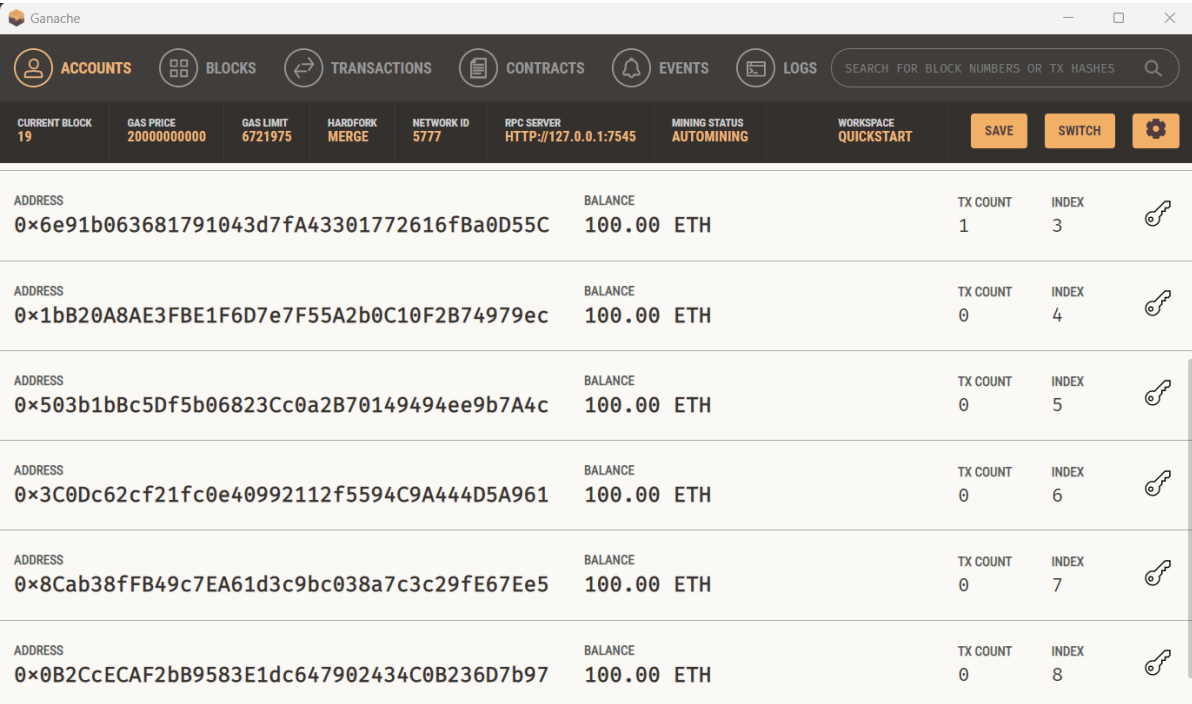
Réalisation

4.1 Introduction

Ce chapitre traitera la mise en œuvre pratique du projet, couvrant le déploiement et les tests des contrats intelligents, ainsi que la présentation de l'application mobile MCS basée sur la Blockchain, avec une navigation de ses interfaces.

4.2 Déploiement et test des contrats intelligents

Le développement et le test des contrats intelligents nécessite un environnement Ethereum qui permet à deployer ces contrats, exécuter des transations, simuler des scénarios de test, etc. Nous avons utilisé un environnement en local à travers Ganache, qui offre des comptes identifiés par des adresses Ethereum uniques, qui possèdent des clés privées associées pour signer les transactions effectuées, et en ayant aussi un solde initial de 100 Ether, ce qui élimine les contraintes liées à la disponibilité réelle de cryptomonnaie.



ADDRESS	BALANCE	TX COUNT	INDEX
0x6e91b063681791043d7fA43301772616fBa0D55C	100.00 ETH	1	3
0x1bB20A8AE3FBE1F6D7e7F55A2b0C10F2B74979ec	100.00 ETH	0	4
0x503b1bBc5Df5b06823Cc0a2B70149494ee9b7A4c	100.00 ETH	0	5
0x3C0Dc62cf21fc0e40992112f5594C9A444D5A961	100.00 ETH	0	6
0x8Cab38fFB49c7EA61d3c9bc038a7c3c29fE67Ee5	100.00 ETH	0	7
0x0B2CcECAF2bB9583E1dc647902434C0B236D7b97	100.00 ETH	0	8

FIGURE 4.1 – Les comptes Ganache

La première étape à faire pour utiliser ce réseau Blockchain local consiste à configurer truffle, et plus précisément le fichier truffle-config.js de notre projet, en ajoutant ganache comme réseau de développement.

Ensuite, on compile nos contrats intelligents et on crée des fichiers de migration dans le répertoire migrations de truffle afin de les déployer.

La figure ci-dessous montre les contrats déployés sur notre environnement Ganache, où on peut remarquer que chaque contrat possède une adresse Ethereum unique :

ProjetMCS C:\Users\HP\OneDrive\Escritorio\workspace\projetMCS

NAME	ADDRESS	TX COUNT	
HelloWorld	0x23664b08d99304C3dfC2BBf0a866fC136ead78B1	0	DEPLOYED
MCSContract	0xFE2e233B76178Ef893bf3438F440F26c7B398838	0	DEPLOYED
TaskCreationContract	0xacac5B1Cc68e446b4b904c467647c109fFa0aDCc	0	DEPLOYED
UserRegistrationContract	0x6bce73d95C3961E58704194C2Eb4669F61185021	5	DEPLOYED

FIGURE 4.2 – Les contrats deployes sous Ganache

Passons maintenant au test de ces contrats. On propose le scénario suivant :

- Deux demandeurs ayant respectivement 100 et 99 dans leurs soldes de points.
- Trois travailleurs avec des soldes de points que nous avons initialisé par 70, 10, 5 respectivement.
- Deux tâches à réaliser, chacune d’elles est créée par un demandeur différent.

Ce scénario va ne permettre de tester toutes les fonctions contenues dans nos contrats intelligents.

Contrat intelligent 1 : UserRegistrationContract

A travers des scripts que nous avons créé : `registerWorker.js` et `registerRequester.js`, on a pu faire appel au contrat `UserRegistrationContract` qui, grace à la fonction `addUser()`, permet d’ajouter de nouveaux participants à notre Blockchain.

La figure 4.3 montre que trois travailleurs ont été ajoutés, et la figure 4.4 représente l’ajout des deux demandeurs. Tous ces nouveaux participants sont identifiées par des adresses Ethereum uniques.

```

PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/registerWorker.js --network development
Using network 'development'.

User registered successfully as a worker!
Requesters List: []
Workers List: [ '0x694a095d67a048AbCEA0545f8C5570Afa8C73141' ]
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/registerWorker.js --network development
Using network 'development'.

User registered successfully as a worker!
Requesters List: []
Workers List: [
  '0x694a095d67a048AbCEA0545f8C5570Afa8C73141',
  '0x4C622Fd04B4bC4144159644C048F5E038Cd5670f'
]
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/registerWorker.js --network development
Using network 'development'.

User registered successfully as a worker!
Requesters List: []
Workers List: [
  '0x694a095d67a048AbCEA0545f8C5570Afa8C73141',
  '0x4C622Fd04B4bC4144159644C048F5E038Cd5670f',
  '0x5D0b8340394A789099a69B4829036323755f8501'
]

```

FIGURE 4.3 – Ajout des travailleurs

```

PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/registerRequester.js --network development
Using network 'development'.

User registered successfully as a requester!
Requesters List: [ '0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A' ]
Workers List: [
  '0x694a095d67a048AbCEA0545f8C5570Afa8C73141',
  '0x4C622Fd04B4bC4144159644C048F5E038Cd5670f',
  '0x5D0b8340394A789099a69B4829036323755f8501'
]
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/registerRequester.js --network development
Using network 'development'.

User registered successfully as a requester!
Requesters List: [
  '0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A',
  '0xcF835784D1c6705743887B6D78cb6F357F97799A'
]
Workers List: [
  '0x694a095d67a048AbCEA0545f8C5570Afa8C73141',
  '0x4C622Fd04B4bC4144159644C048F5E038Cd5670f',
  '0x5D0b8340394A789099a69B4829036323755f8501'
]
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> 

```

FIGURE 4.4 – Ajout des demandeurs

En somme, on a les participants suivants :

Type	Adresse Ethereum	Solde de points
Demandeur	0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A	100
Demandeur	0xcF835784D1c6705743887B6D78cb6F357F97799A	99
Travailleur	0x694a095d67a048AbCEA0545f8C5570Afa8C73141	70
Travailleur	0x4C622Fd04B4BC4144159644C048F5E038cd5670f	10
Travailleur	0x5D0b8340394A789099a69B4829036323755f8501	5

TABLE 4.1 – Les participants (Demandeur et Travailleur)

On peut aussi voir le stockage de ces participants dans la Blockchain à travers Ganache. La

figures suivants montre que les deux demandeurs ont été ajouté au requesterPool et les trois travailleurs au workerPool :

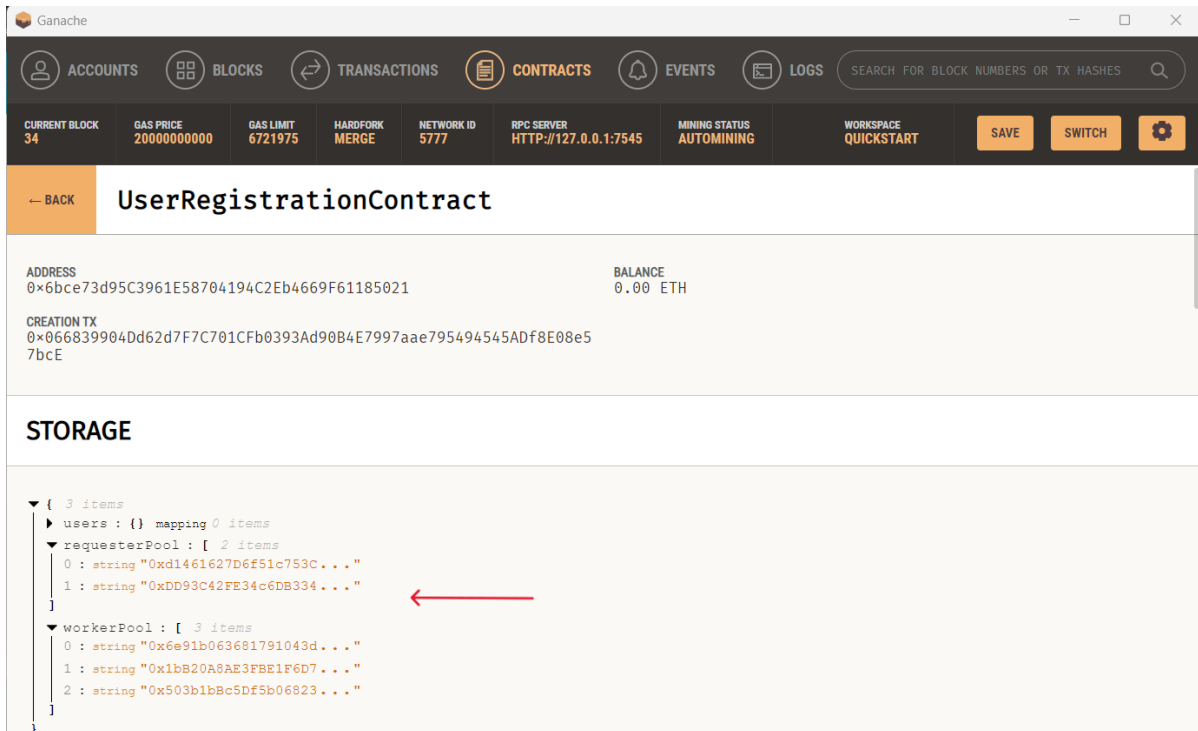


FIGURE 4.5 – Les participants ajoutés sous Ganache

Contrat intelligent 2 : TaskCreationContract

Maintenant, on va faire appel au deuxième contrat intelligent TaskCreationContract, pour tester sa fonction addTask(), en lançant le script taskCreation.js qu'on a créé :

```
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/taskCreation.js --network development
Using network 'development'.

Task created successfully with ID: 1
Task Details:
Requester Address: 0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A
Objective: Sample Objective
Reward: 50
Deposit: 5
Type Data: Image
Deadline: 30/01/2024 17:52:49
Status: Available
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/taskCreation.js --network development
Using network 'development'.

Task created successfully with ID: 2
Task Details:
Requester Address: 0xCf835784D1c6705743887B6D78cb6F357F97799A
Objective: Sample Objective
Reward: 100
Deposit: 10
Type Data: Image
Deadline: 30/01/2024 17:53:52
Status: Available
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest>
```

FIGURE 4.6 – Création des tâches

Deux tâches sont créées, ses informations sont contenues dans le tableau suivant :

ID de la tâche	1	2
Demandeur	0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A	0xcf835784D1c6705743887B6D78cb6F357F97799A
Objectif	Sample Objective	Sample Objective
Récompense	50	100
Deposit	5	10
Type de données	Image	Image
Deadline	30/01/2024 17 :52 :49	30/01/2024 17 :53 :52
Statut	Disponible	Disponible

TABLE 4.2 – Les tâches créées

NB : Les valeurs des paramètres de chaque tâche sont pris juste pour faire des tests. Par ailleurs, on peut visualiser le stockage de ces nouvelles tâches au niveau de Ganache :

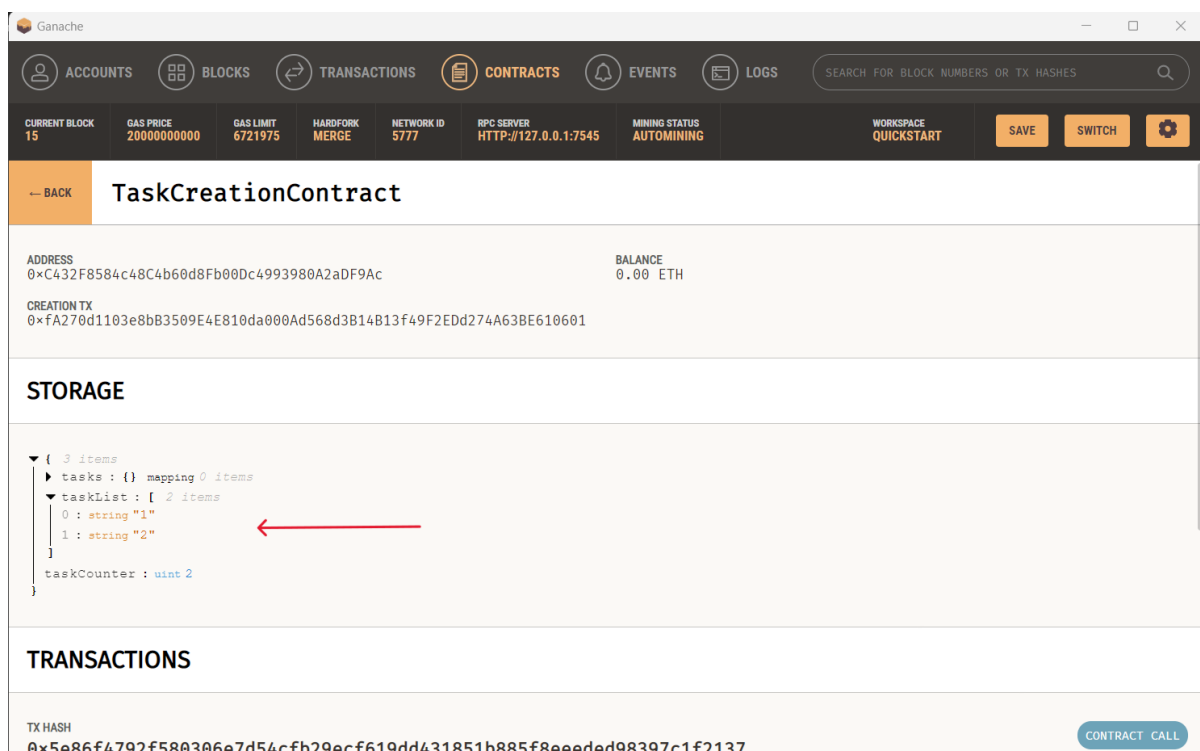


FIGURE 4.7 – Les tâches créées sous Ganache

Contrat intelligent 3 : MCSCContract

Comme déjà mentionné dans la partie 3.2.4, ce contrat contient quatre fonctions principales : `workerSelection()`, `uploadData()`, `dataValidation()`, `paymentDistribution()`.

Après avoir ajouté les participants au processus de MCS et les tâches à réaliser, les travailleurs postulent pour réaliser une tâche spécifique, à ce moment-là le contrat MCSCContract est déclenché, en utilisant sa première fonction qui permet la sélection de travailleur approprié selon l'algorithme 3.2.4.4, on lançant le script `testWorkerSel.js`, on obtient le travailleur qui a postulé pour la tâche 2. Il s'agit du travailleur qui a le maximum de points cumulés : 70.

```
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/testWorkerSel.js --network development
Using network 'development'.

Selected Worker: 0x694a095d67a048AbCEA0545f8C5570Afa8C73141
```

FIGURE 4.8 – Le travailleur sélectionné pour la réalisation du tâche

Ensuite, ce travailleur upload les données collectées à l'aide de la fonction `dataUpload()`, qui affecte aussi un identifiant de ses données sur la Blockchain :

```
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/testUploadData.js --network development
Using network 'development'.

Data uploaded successfully with ID: 1
Data Details:
Worker Address: 0x694a095d67a048AbCEA0545f8C5570Afa8C73141
Requester Address: 0xCf835784D1c6705743887B6D78cb6F357F97799A
Task ID: 2
Data Type: Data Type
Deposit Date: 30/01/2024 17:17:07
Data Content: Sample Data
```

FIGURE 4.9 – Upload des données

Maintenant, le demandeur de la tâche entre en jeu pour juger la validité des données collectées en utilisant la fonction `dataValidation()`. Dans ce cas, on a supposé que le travailleur n'a pas respectés les exigence de la tâche, donc les données sont jugées "Non valide" :

```
PS C:\Users\HP\OneDrive\Escritorio\workspace\projetTest> truffle exec script/dataValidation.js --network development
Using network 'development'.

Validation Status: Non valide
```

FIGURE 4.10 – Validation des données par le demandeur

Finalement, c'est la phase de distribution des récompenses. Dans ce cas la tâche est échouée "Failed", le travailleur ne reçoit pas la récompense fixée dans la tâche à réaliser et il perd son deposit (qui été fixé à 10 pour cette tâche, selon le tableau 4.2) :

```
====> Informations de sélection de l'utilisateur avant le contrat MCS <====
Travailleur sélectionné : 0x694a095d67a048AbCEA0545f8C5570Afa8C73141
Points du travailleur sélectionné (Avant MCS) : 70 <-----
====> Informations de sélection de l'utilisateur après le contrat MCS <====
Points du travailleur sélectionné (Après MCS) : 60 <-----
Détails mis à jour de la tâche :
Requester Address: 0xCf835784D1c6705743887B6D78cb6F357F97799A
Objective: Sample Objective
Reward: 100
Deposit: 10
Type Data: Image
Deadline: 30/01/2024 17:53:52
Statut : Failed <-----
PS C:\Users\USER\OneDrive\Documents\ensias\projet_federateur\Project> █
```

FIGURE 4.11 – Distribution de récompense - cas de données non valides

Par ailleurs, l'autre travailleur a été choisi pour effectuer la tâche 1. Après qu'il a fait l'upload de données collectées, ces données sont jugées valide de la part de demandeur de cette tâche, donc le travailleur a reçu bien sa récompense en gardant sa deposit, ainsi que le statut de deuxième tâche devient "Completed" :

```

====> Informations de sélection de l'utilisateur avant le contrat MCS <====
Travailleur sélectionné : 0x4C622Fd04B4BC4144159644C048F5E038cd5670f
Points du travailleur sélectionné (Avant MCS) : 10
====> Informations de sélection de l'utilisateur après le contrat MCS <====
Points du travailleur sélectionné (Après MCS) : 60
Détails mis à jour de la tâche :
Requester Address: 0xd03ae5AE8b2fF8A3D855A969869c666c48b8261A
Objective: Sample Objective
Reward: 50
Deposit: 5
Type Data: Image
Deadline: 30/01/2024 17:52:49
Statut : Completed
PS C:\Users\USER\OneDrive\Documents\ensias\projet_federateur\Project>

```

FIGURE 4.12 – Distribution de récompense - cas de données valides

Finalement, on peut visualiser toutes ces transactions effectuées, dans l'onglet "Transactions" et les nouveaux blocs ajoutés à notre Blockchain dans l'onglet "Blocks", avec transparence sur Ganache :

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
15

GAS PRICE
2000000000

GAS LIMIT
6721975

HARDFORK
MERGE

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

WORKSPACE
QUICKSTART

SAVE

SWITCH

TX HASH

CONTRACT CALL

0x146fff970b2c71af7b05a192a7a2dd2c5a34d4f00703aa448475bde41219d752

FROM ADDRESS

TO CONTRACT ADDRESS

GAS USED

VALUE

0x8FCC6E47BD9714F2aF9B42fE9638A1edeB37F181

MCSContract

61707

0

TX HASH

CONTRACT CALL

0x146fff970b2c71af7b05a192a7a2dd2c5a34d4f00703aa448475bde41219d752

FROM ADDRESS

TO CONTRACT ADDRESS

GAS USED

VALUE

0x8FCC6E47BD9714F2aF9B42fE9638A1edeB37F181

MCSContract

61707

0

TX HASH

CONTRACT CALL

0xed4c8746c59afb762987e3a6a41561e576edfd857cb804ce0d3d17ffe65fed7

FROM ADDRESS

TO CONTRACT ADDRESS

GAS USED

VALUE

0x8FCC6E47BD9714F2aF9B42fE9638A1edeB37F181

MCSContract

211098

0

TX HASH

CONTRACT CALL

0xed4c8746c59afb762987e3a6a41561e576edfd857cb804ce0d3d17ffe65fed7

FROM ADDRESS

TO CONTRACT ADDRESS

GAS USED

VALUE

0x8FCC6E47BD9714F2aF9B42fE9638A1edeB37F181

MCSContract

211098

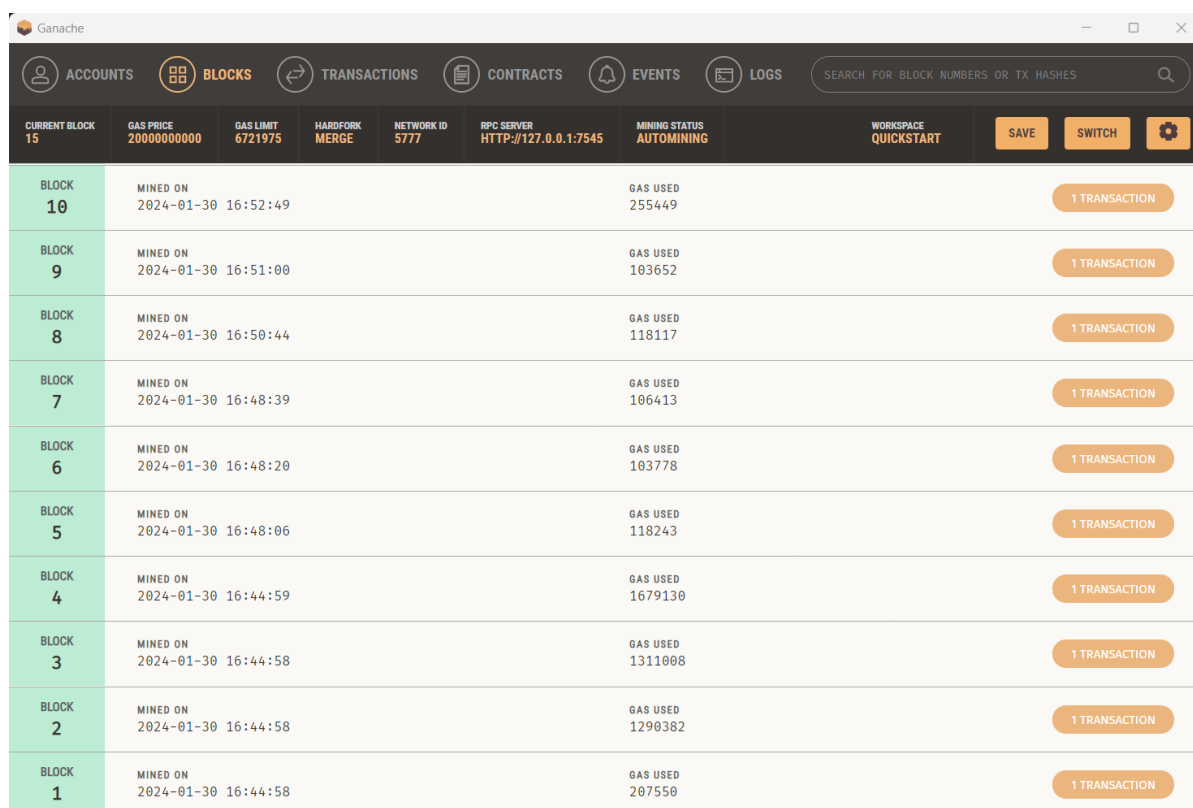
0

TX HASH

CONTRACT CALL

0xbd52e6d2dd072aa10ca16899f7f5db0db3e7ddf4ebd4b5e02ae4574e72e802e1

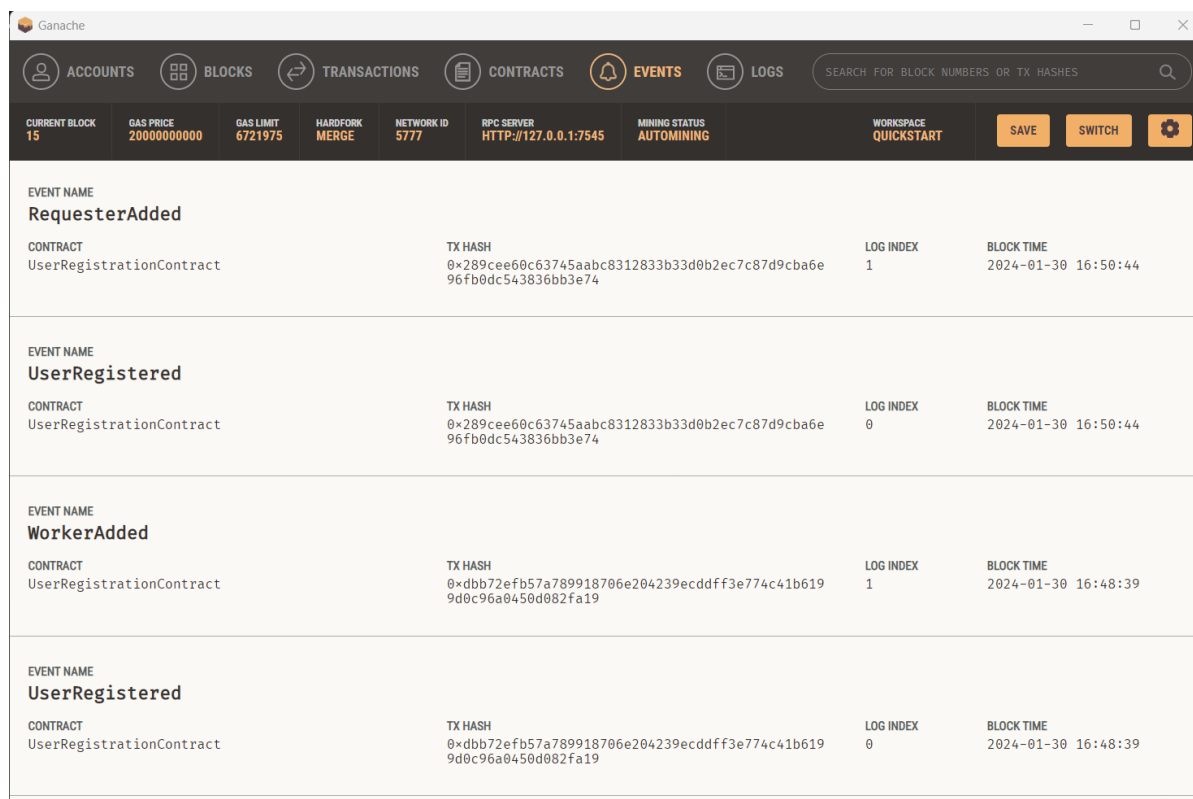
FIGURE 4.13 – Les Transactions effectuées



BLOCK	MINED ON	GAS USED	TRANSACTIONS
BLOCK 10	2024-01-30 16:52:49	255449	1 TRANSACTION
BLOCK 9	2024-01-30 16:51:00	103652	1 TRANSACTION
BLOCK 8	2024-01-30 16:50:44	118117	1 TRANSACTION
BLOCK 7	2024-01-30 16:48:39	106413	1 TRANSACTION
BLOCK 6	2024-01-30 16:48:20	103778	1 TRANSACTION
BLOCK 5	2024-01-30 16:48:06	118243	1 TRANSACTION
BLOCK 4	2024-01-30 16:44:59	1679130	1 TRANSACTION
BLOCK 3	2024-01-30 16:44:58	1311008	1 TRANSACTION
BLOCK 2	2024-01-30 16:44:58	1290382	1 TRANSACTION
BLOCK 1	2024-01-30 16:44:58	207550	1 TRANSACTION

FIGURE 4.14 – Les blocs ajouté à la Blockchain

En ajoutant les événements "Events" qui sont déclenchés à chaque exécution d'une fonction, ce qui ajoute plus de traçabilité à notre processus de MCS :



EVENT NAME	CONTRACT	TX HASH	LOG INDEX	BLOCK TIME
RequesterAdded	UserRegistrationContract	0x289cee60c63745aabc8312833b33d0b2ec7c87d9c6a6e96fb0dc543836bb3e74	1	2024-01-30 16:50:44
UserRegistered	UserRegistrationContract	0x289cee60c63745aabc8312833b33d0b2ec7c87d9c6a6e96fb0dc543836bb3e74	0	2024-01-30 16:50:44
WorkerAdded	UserRegistrationContract	0xdbb72efb57a789918706e204239ecddff3e774c41b6199d0c96a0450d082fa19	1	2024-01-30 16:48:39
UserRegistered	UserRegistrationContract	0xdbb72efb57a789918706e204239ecddff3e774c41b6199d0c96a0450d082fa19	0	2024-01-30 16:48:39

FIGURE 4.15 – Les événements déclenchés

4.3 Application mobile de MCS basée sur la Blockchain

4.3.1 Présentation des défis confrontés

Après avoir créé les contrats intelligents avec Solidity dans notre IDE et les avoir testés sous Ganache, nous avons passé à rendre ces contrats opérationnels en les intégrant dans notre application mobile.

Pour cela, nous avons travaillé sous Android Studio pour réaliser notre application mobile, et pour la partie communication avec les contrats intelligents, nous avons utilisé la bibliothèque web3js spécialement conçue à cet effet.

Ainsi, pour tester la communication, nous avons commencé par la parties d'enregistrement des participants en utilisant le contrat UserRegistrationContract.

Cependant, nous avons rencontré plusieurs problèmes techniques : nous n'avons pas pu obtenir l'adresse de l'application locale de test Ethereum (Ganache), ce qui crée un problème de communication entre notre application et Ganache. Ensuite, nous avons tenté une autre approche en intégrant les sockets pour la communication entre les deux, mais nous n'avons pas pu réussir aussi cette approche.

Malgré cela, nous avons décidé de présenter l'idée de notre application sous forme de maquettes que nous avons réalisé. Où nous avons essayé d'expliquer la logique de l'application avec ses différentes interfaces, ainsi que les différentes interactions entre les participants (que ce soient travailleurs ou demandeurs) et l'application mobile.

4.3.2 Les interfaces de l'application

Page de connexion

C'est la première page qui s'affiche lorsque les utilisateurs accèdent à l'application.

S'il s'agit d'un nouvel utilisateur, il a le choix entre participer en tant que demandeur, en cliquant sur le bouton "Requester", où en tant que travailleur, en cliquant su "Worker".

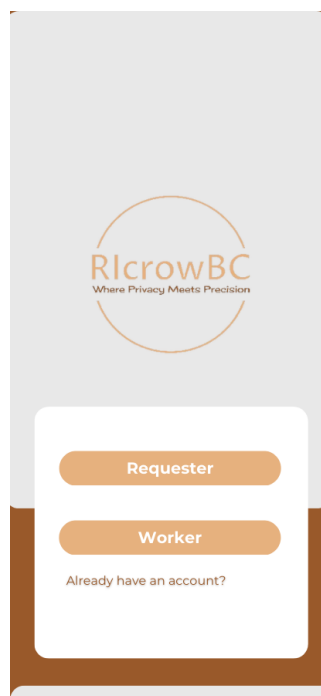


FIGURE 4.16 – Page d'inscription

Un fois il clique sur l'un des boutons, une paire de clés sera générée sur son appareil, dont la clé publique sera son identifiant dans la Blockchain.

Sinon, s'il s'agit d'un utilisateur qui a déjà inscrit, il peut cliquer sur "Already have an account ?", et il peut saisir sa clé publique pour se connecter à son compte.

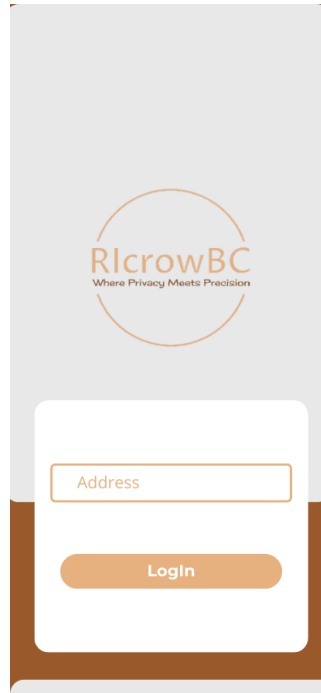


FIGURE 4.17 – Page de connexion

En ajoutant que selon le role de participant (son profil), un espace approprié de l'application lui est attribué pour continuer ses interactions.

Profil Demandeur (Requester)

Après l'authentification, c'est la première interface qui s'affiche pour un demandeur :

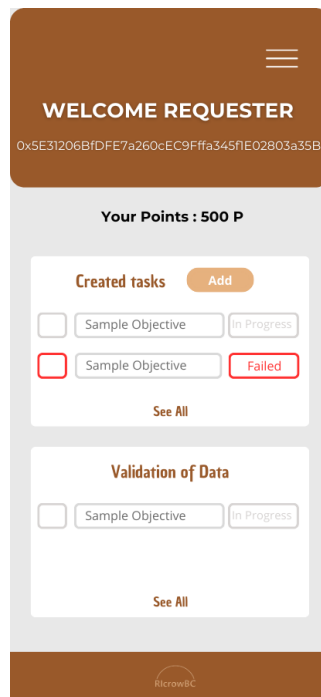


FIGURE 4.18 – Interface principale de demandeur

Elle contient son adresse sur Ethereum, son solde de points, et deux rubriques en dessous :

- La première "Created tasks" permet de lister les tâches qui a déjà créé et publié sur la Blockchain avec une étiquette à droite montrant le statut de la tâche. Avec un bouton "Add" qui, en cliquant ci-dessus, lui permet d'ajouter une nouvelle tâche, en remplissant le formulaire qui s'affiche :

The screenshot shows a mobile application interface for creating a new task. It features a brown header bar with a back arrow and a hamburger menu icon, and the title "CREATE TASK". Below the header, there is a form with five input fields: "Objective", "Reward", "Deposit", "T ypeData" (with a dropdown arrow), and "Deadline". At the bottom of the form is a brown button labeled "Create Task".

FIGURE 4.19 – Formulaire pour créer une nouvelle tâche

- La deuxième "Validation of Data" concerne les tâches déjà réalisées par les travailleurs, et que le demandeur doit les valider. En effet, il peut cliquer sur la tâche pour consulter

les données collectées par le demandeur. S'il s'agissent des données valides il clique sur "oui", sinon il clique sur "non".

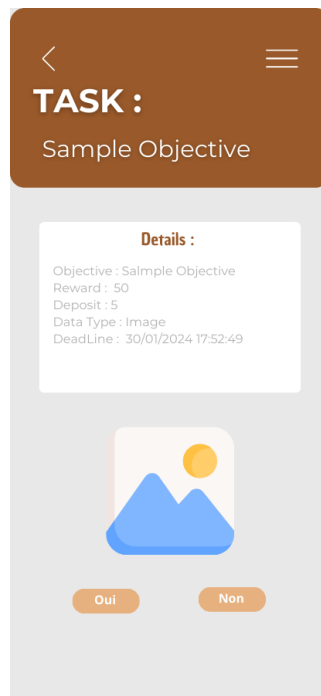


FIGURE 4.20 – Validation de la tâche

Après cette validation, le statut des tâches se met à jour :

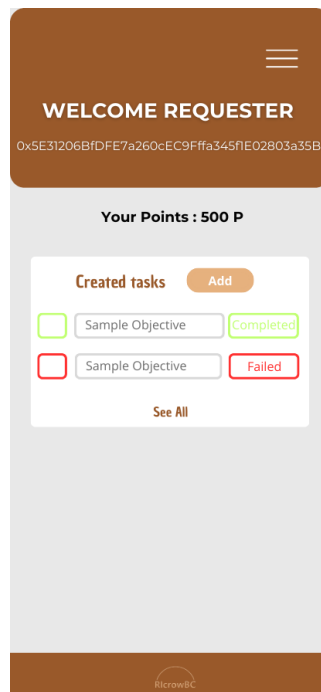


FIGURE 4.21 – Interface de demandeur après la validation des tâches

Et il peut toujours consulter l'historique des tâches, par exemple q'il clique sur la tâche échoué il peut avoir :

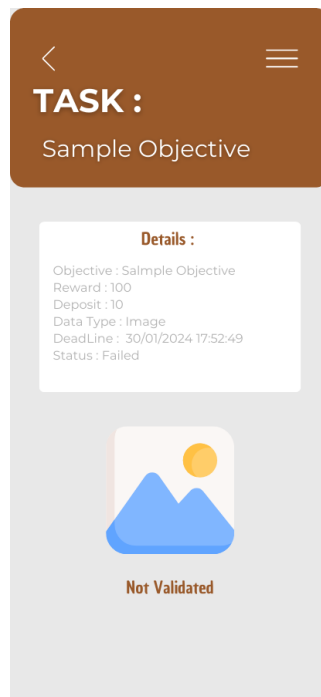


FIGURE 4.22 – Les details de tâche

Profil Travailleur (Worker)

Après l'authentification en tant que travailleur, ce dernier peut consulter ses points, son adresse Ethereum, et la liste des tâches disponibles dans "Available Tasks".

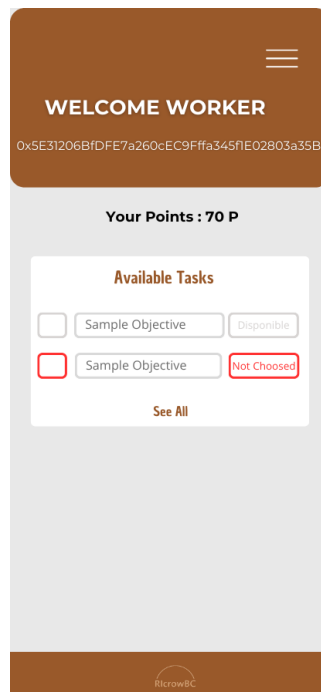


FIGURE 4.23 – Interface principale de travailleur

Pour choisir une tâche, il peut cliquer sur le petit carré à gauche pour postuler à la réaliser.

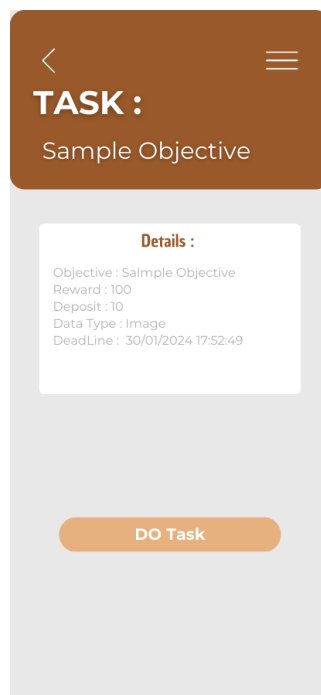


FIGURE 4.24 – Les détails d'une tâche

S'il est choisi pour la tâche, il pourra uploader les données collectées en respectant les exigences représentées dans la description de la tâche :

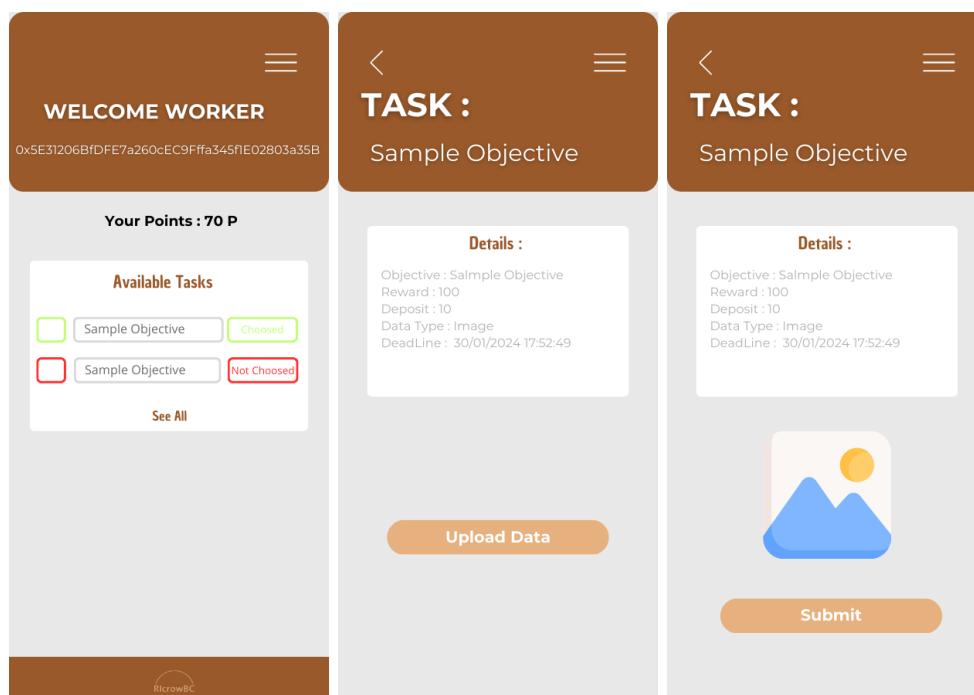


FIGURE 4.25 – Etapes de réalisation d'une tâche

Après l'envoi des données collectées, il attend la validation de demandeur.

Les tâches validés (que ce soit par oui ou par non) s'affiche dans une rubrique des tâches complétées "Completed Tasks". Si le demandeur a jugé les données comme valide, une étiquette s'affiche près de la tâche indique de résultat "completed successfully", et le solde de points du travailler s'est mis à jour selon la récompense fixée par le demandeur de la tâche :

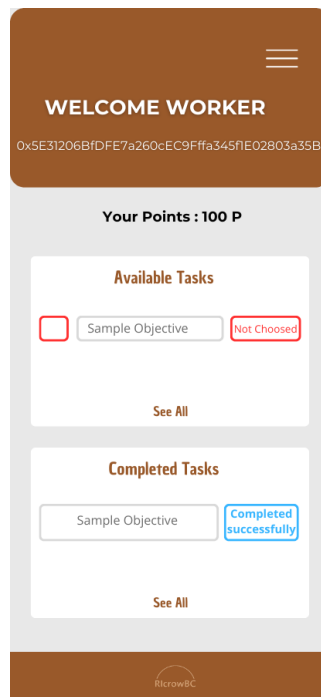


FIGURE 4.26 – Interface de travailleur

4.4 Conclusion

En conclusion de ce chapitre de réalisation, nous avons passé à la pratique pour donner vie à notre vision de l'intégration de la blockchain au Mobile Crowdsensing.

Conclusion Générale et Perspectives

En conclusion, ce projet illustre le potentiel de la technologie Blockchain pour renforcer la confidentialité dans le contexte du Mobile Crowdsensing. Une conception approfondie du solution proposée a été présentée en mettant la lumière sur la manière dont la décentralisation et l'immuabilité de la Blockchain peuvent contribuer à surmonter les défis de sécurité et vie privée au MCS.

La mise en œuvre de ce projet a été un processus complexe et instructif. En fait, nous avons réussi le déploiement et les tests des contrats intelligents, mais pour la suite nous avons confronté des obstacles techniques ont entravé la communication entre l'application mobile et la Blockchain. Ce qui nous a poussé à travailler sur le front-end de l'application en réalisant des maquettes pour continuer l'explication de notre idée initiale et des résultats que nous avons voulu d'obtenir à la fine de projet. Malgré ces difficultés, cette expérience nous a permis d'acquérir des connaissances précieuses sur les aspects pratiques de l'intégration de la Blockchain dans le domaine du MCS.

Dans une perspective future, il est impératif d'explorer des solutions alternatives pour résoudre les problèmes techniques de communication entre l'application mobile et la Blockchain. En approfondissant plus nos connaissances sur la librairie Web3j.

D'autre part, ce projet peut être plus développé pour englober aussi la partie validation de données, car cela n'a pas fait partie de notre cadre d'étude, en adoptant des techniques et algorithmes de l'intelligence artificielle qui peuvent permettre un traitement initial au niveau de l'appareil de l'utilisateur pour éliminer s'il y a des données très sensibles dans les images où les vidéos importées afin (en mettant par exemple en flou les visages humains,...) pour ajouter une augmentation de plus de Privacy.

Références

- [1] La vie privée <https://www.wavestone.com/app/uploads/2017/01/Vie-privee-numerique-confiance.pdf>
- [2] The Timeline History of Blockchain <https://intellipaat.com/blog/tutorial/blockchain-tutorial/history-of-blockchain/>
- [3] Blockchain Technology Overview <https://doi.org/10.6028/NIST.IR.8202>
- [4] Les types de noeuds de Blockchain <https://utimaco.com/service/knowledge-base/blockchain/what-are-the-types-of-nodes-in-blockchain>
- [5] Type de réseaux de blockchain <https://www.ibm.com/topics/blockchain>
- [6] Blockchain Based Mobile Crowd Sensing in Industrial Systems https://www.researchgate.net/publication/338379043_Blockchain_Based_Mobile_Crowd_Sensing_in
- [7] Mobile crowdsensing : current state and future challenges <https://ieeexplore.ieee.org/document/6069707>
- [8] A Survey of Sparse Mobile Crowdsensing : Developments and Opportunities <https://ieeexplore.ieee.org/document/9780171>
- [9] Facilitating Mobile Crowdsensing from both Organizers' and Participants' Perspectives <https://theses.hal.science/tel-01388141>
- [10] APISENSE : une plate-forme répartie pour la conception, le déploiement et l'exécution de campagnes de collecte de données sur des terminaux intelligents <https://inria.hal.science/tel-01087240/document>