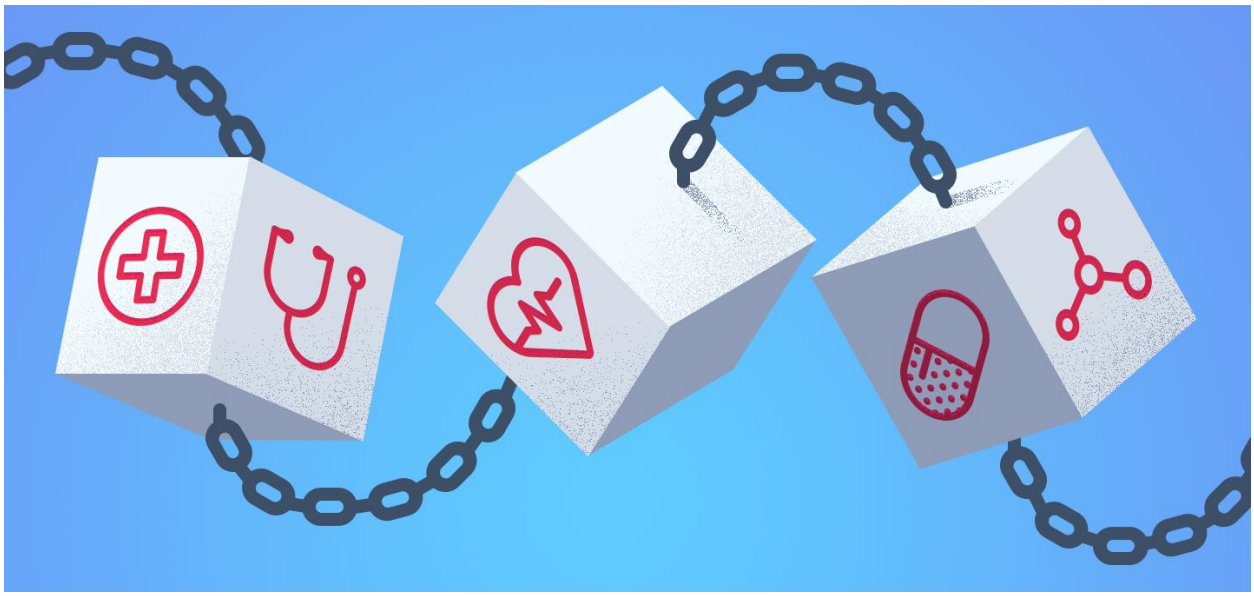
Rapport

Rapport de Projet : Dossier Médical avec Blockchain



Fait par : **EL MEZIANE Chaïma**

OHMAD Abdessamade

Encadré par Pr : **Ikram Ben abdel ouahab**

Contexte

La gestion sécurisée des données médicales constitue un défi majeur dans le secteur de la santé. Les systèmes traditionnels de gestion des dossiers médicaux sont souvent vulnérables aux cyberattaques, aux erreurs humaines et aux accès non autorisés. Cela peut entraîner des fuites de données sensibles, compromettant la confidentialité des patients et la qualité des soins. Pour surmonter ces défis, il devient crucial d'adopter des technologies innovantes, permettant de garantir la sécurité, la traçabilité et la confidentialité des informations médicales.

Introduction

Le projet "Dossier Médical avec Blockchain" propose une solution décentralisée et sécurisée pour gérer les données médicales des patients. En utilisant la technologie blockchain, ce projet vise à offrir une alternative aux systèmes centralisés traditionnels. La blockchain, grâce à sa nature immuable et transparente, permet de stocker les informations médicales de manière sécurisée, garantissant ainsi leur confidentialité et leur intégrité.

Objectif

L'objectif principal du projet "Dossier Médical avec Blockchain" est de développer une application décentralisée (DApp) utilisant la technologie blockchain pour gérer et sécuriser les dossiers médicaux des patients. Ce projet vise à résoudre les problèmes de sécurité, de confidentialité et de traçabilité des données médicales, souvent rencontrés dans les systèmes traditionnels de gestion des informations de santé.

1. **Sécurisation des données médicales :** Assurer l'immuabilité et la confidentialité des dossiers médicaux en utilisant la blockchain pour garantir qu'aucune modification non autorisée ne soit effectuée sur les informations sensibles des patients.
2. **Confidentialité des informations :** Mettre en place des mécanismes de cryptage avancés pour protéger les données médicales, assurant que seules les personnes autorisées puissent accéder et consulter ces informations.
3. **Transparence et traçabilité :** Permettre aux utilisateurs de tracer toutes les actions effectuées sur les dossiers médicaux, telles que les mises à jour ou les accès, en enregistrant ces actions sur la blockchain pour garantir une traçabilité complète et une auditabilité des opérations.
4. **Décentralisation :** Réduire la dépendance vis-à-vis d'une autorité centrale en utilisant la blockchain pour stocker et gérer les données, ce qui permet une gestion des dossiers médicaux plus résiliente, transparente et moins vulnérable aux attaques ou aux défaillances d'un serveur centralisé.
5. **Interopérabilité :** Faciliter le partage sécurisé des informations entre les différents acteurs du secteur médical (médecins, hôpitaux, laboratoires, etc.) en garantissant l'intégration fluide et sécurisée des systèmes via des mécanismes de gestion des accès basés sur des smart contracts.

Technologies et Outils Utilisés

Python

Python est le langage de programmation principal utilisé dans ce projet pour le développement du backend et l'intégration avec la blockchain. Il offre une grande flexibilité, une riche bibliothèque et une syntaxe simple, ce qui permet une gestion efficace des transactions et interactions avec Ethereum via la bibliothèque Web3.py.

Python est utilisé pour la logique métier, la gestion des utilisateurs, et l'interaction avec les smart contracts déployés.

CustomTkinter

CustomTkinter est un framework Python qui permet de créer des interfaces graphiques modernes et performantes avec Tkinter, tout en offrant une personnalisation avancée. Dans ce projet, CustomTkinter est utilisé pour créer l'interface graphique (frontend) de l'application décentralisée (DApp). L'interface permet aux utilisateurs d'interagir de manière fluide et intuitive avec les dossiers médicaux stockés sur la blockchain. Grâce à CustomTkinter, l'application bénéficie d'une interface simple, responsive et esthétique.

Flask

Flask est un micro-framework Python utilisé pour le développement du backend de l'application. Il permet de créer une API RESTful pour gérer la communication entre le frontend et le backend. Flask est utilisé pour traiter les requêtes des utilisateurs, comme l'enregistrement des dossiers médicaux, la gestion des permissions d'accès, et l'interaction avec les smart contracts via Web3.py. Flask permet une architecture légère et modulaire pour ce projet, avec une gestion simple des routes et des requêtes.

Postman

Postman est un outil de test d'API qui facilite la création, l'envoi et l'analyse des requêtes HTTP pour tester le backend de l'application. Il est utilisé dans ce projet pour tester les routes définies avec Flask, simuler des actions des utilisateurs, et valider la communication entre le frontend et le backend avant l'intégration avec la blockchain. Postman permet de s'assurer que toutes les fonctionnalités de l'API sont correctes et de résoudre les problèmes potentiels avant de passer au déploiement sur la blockchain.

Ganache

Ganache est un simulateur de blockchain Ethereum local utilisé pour créer un réseau privé où les smart contracts peuvent être déployés et testés. Dans ce projet, Ganache fournit des adresses Ethereum pour tester les transactions et les interactions avec les smart contracts sans avoir besoin de dépenser de l'Ether réel. Il permet également de tester les fonctionnalités du projet dans un environnement sécurisé avant le déploiement final sur le réseau principal d'Ethereum.

MetaMask

MetaMask est un portefeuille Ethereum qui permet aux utilisateurs de gérer leurs clés privées et d'interagir avec les dApps via leur navigateur. MetaMask est utilisé dans ce projet pour permettre l'authentification décentralisée des utilisateurs. Les patients et les professionnels de santé peuvent utiliser MetaMask pour se connecter à l'application, signer des transactions, et accéder

aux dossiers médicaux stockés sur la blockchain. Cela assure un contrôle total des utilisateurs sur leurs propres données et interactions avec le système.

Remix IDE

Remix est un environnement de développement intégré (IDE) pour le développement de smart contracts en Solidity. Il est utilisé dans ce projet pour écrire, compiler, et déployer les smart contracts sur la blockchain Ethereum. Remix facilite le processus de développement en permettant des tests interactifs des smart contracts, le débogage, et le déploiement direct sur un réseau de test ou sur le réseau principal. Les smart contracts créés dans Remix gèrent l'enregistrement des patients, la gestion des permissions d'accès, et la traçabilité des actions sur les dossiers médicaux.

IPFS (InterPlanetary File System)

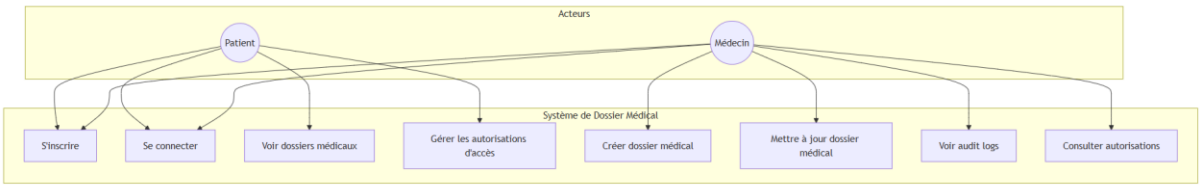
IPFS est un système de stockage décentralisé utilisé pour stocker de grands fichiers, comme les dossiers médicaux. Dans ce projet, IPFS est utilisé pour héberger les fichiers des dossiers médicaux, en garantissant leur sécurité et leur disponibilité. Seuls les hachages de ces fichiers sont stockés sur la blockchain, assurant ainsi leur immuabilité et permettant de vérifier leur intégrité sans exposer directement les données sensibles. IPFS contribue à la décentralisation du système de stockage des données médicales.

Web3.py

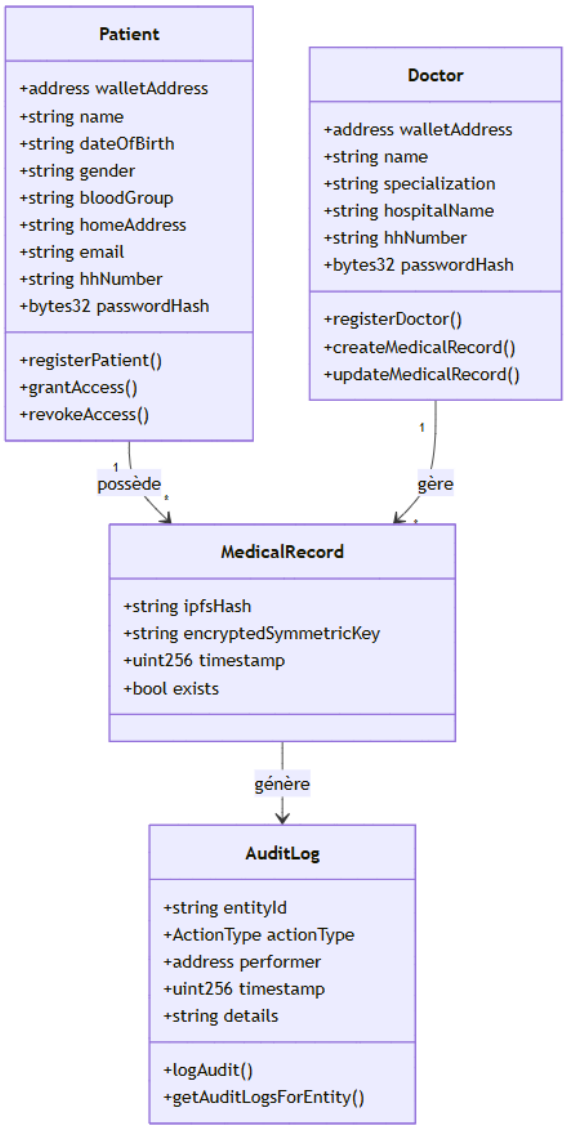
Web3.py est une bibliothèque Python permettant d'interagir avec la blockchain Ethereum via des smart contracts. Dans ce projet, Web3.py est utilisé pour effectuer des transactions, lire et écrire des données sur la blockchain, et interagir avec les smart contracts déployés sur Ethereum. Il permet au backend en Python de communiquer efficacement avec le réseau Ethereum, garantissant ainsi la sécurité et l'intégrité des actions effectuées sur les dossiers médicaux des patients.

Conception

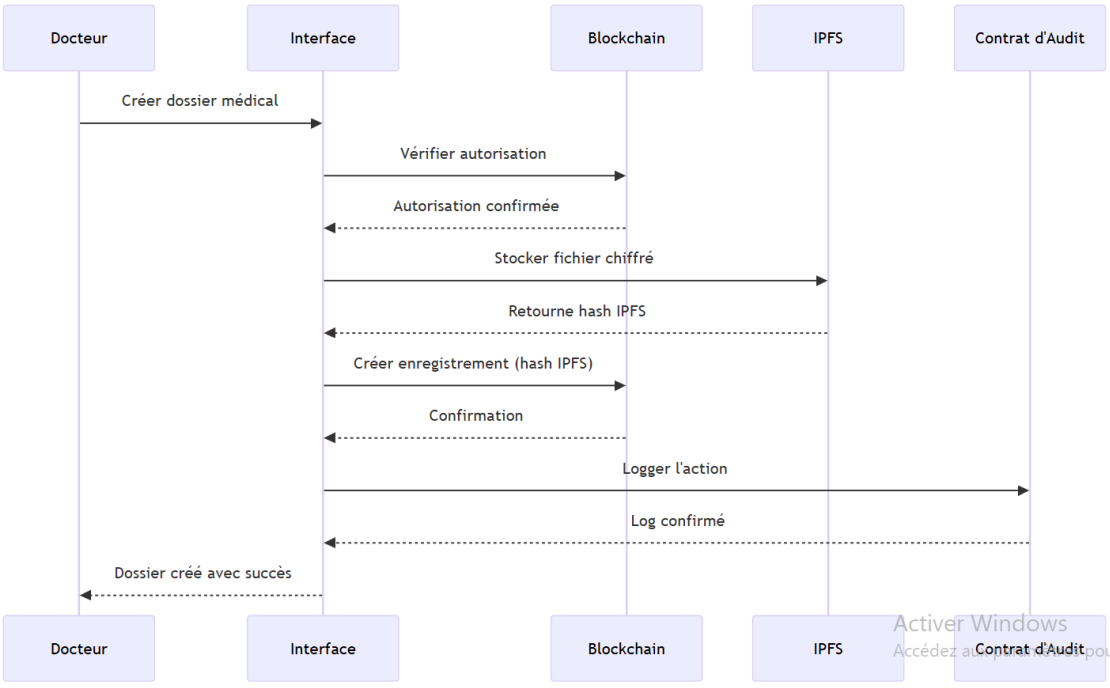
Le diagramme de cas d'utilisation montre les principales interactions des acteurs (patients et médecins) avec le système.



Le diagramme de classes représente la structure des données et les relations entre les différentes entités du système.



Le diagramme de flux détaille le processus de création d'un dossier médical, montrant les interactions entre les différentes composantes du système.



Architecture du Système

Interaction Frontend-Backend-Blockchain

1. Frontend : Permet aux utilisateurs (patients et médecins) d'interagir avec les dossiers médicaux via une interface graphique intuitive.
2. Backend :
   - Gère les requêtes des utilisateurs et les transmet aux smart contracts.
   - Stocke temporairement les données avant de les envoyer à IPFS ou à la blockchain.
3. Blockchain :
   - Assure la traçabilité et l'immuabilité des données.
   - Utilise les smart contracts pour contrôler les accès et les permissions.
4. IPFS : Conserve les fichiers volumineux (images, documents médicaux). Seuls les hachages de ces fichiers sont enregistrés sur la blockchain.

Smart Contracts

Les smart contracts constituent le cœur de l'application. Ils gèrent toutes les opérations relatives aux dossiers médicaux, aux permissions et à l'audit des actions. Voici une description détaillée des principaux smart contracts :

ContractPatient

Ce contrat gère l'enregistrement et l'accès des patients.

1. Structures et Mappings:
   - Patient: Représente un patient avec des informations telles que son nom, sa date de naissance, son groupe sanguin, etc.
   - patients: Un mapping pour stocker les patients par leur numéro de ménage (hhNumber).
   - isPatientRegistered: Permet de vérifier si un patient est déjà enregistré.
   - addressToHhNumber: Relie une adresse de portefeuille à un numéro de ménage.
2. Événements:
   - PatientRegistered: Déclenché lors de l'enregistrement d'un patient.
   - PatientInfoUpdated: Déclenché lors de la mise à jour des informations d'un patient.
   - AccessGranted: Déclenché lorsqu'un patient accorde l'accès à un médecin.
   - AccessRevoked: Déclenché lorsqu'un patient révoque l'accès d'un médecin.
3. Fonctions:
   - registerPatient: Permet à un patient de s'enregistrer, vérifiant que l'adresse est une EOA et que le patient n'est pas déjà enregistré.
   - grantAccess: Permet à un patient de donner accès à son dossier médical à un médecin.
   - revokeAccess: Permet à un patient de révoquer l'accès à son dossier médical pour un médecin.
   - checkAccess: Vérifie si un médecin a accès aux informations médicales d'un patient.

ContractDoctor

Ce contrat gère l'enregistrement des médecins et la création/mise à jour des dossiers médicaux.

1. Structures et Mappings:
   - Doctor: Représente un médecin avec des informations telles que son nom, sa spécialisation, son hôpital, etc.

- MedicalRecord: Représente un dossier médical avec un hash IPFS, une clé symétrique cryptée et un timestamp.
- doctors: Un mapping pour stocker les informations des médecins par leur numéro de ménage (hhNumber).
- isDoctorRegistered: Vérifie si un médecin est enregistré.
- patientRecords: Stocke les dossiers médicaux des patients.
- recordCount: Suit le nombre de dossiers médicaux par patient.

2. Événements:
   - DoctorRegistered: Déclenché lorsqu'un médecin est enregistré.
   - MedicalRecordCreated: Déclenché lorsqu'un dossier médical est créé.
   - MedicalRecordUpdated: Déclenché lorsqu'un dossier médical est mis à jour.
3. Fonctions:
   - registerDoctor: Permet à un médecin de s'enregistrer, vérifiant que le médecin n'est pas déjà enregistré.
   - createMedicalRecord: Permet à un médecin de créer un dossier médical pour un patient, si l'accès est accordé.
   - updateMedicalRecord: Permet à un médecin de mettre à jour un dossier médical pour un patient, si l'accès est accordé et si le dossier existe.

ContractAudit

Ce contrat gère les journaux d'audit des actions liées aux dossiers médicaux.

1. Structures et Mappings:
   - AuditLog: Représente un log d'audit avec l'ID de l'entité, le type d'action, l'adresse de l'exécutant, un timestamp et les détails de l'action.
   - auditLogs: Un tableau pour stocker les logs d'audit.
2. Événements:
   - AuditLogCreated: Déclenché lorsqu'un log d'audit est créé.
3. Fonctions:
   - logAudit: Permet d'ajouter un log d'audit pour une entité donnée avec le type d'action et les détails.
   - getAuditLogsForEntity: Permet de récupérer les logs d'audit associés à une entité spécifiée.

Fonctionnalités Implémentées

Interface Principale :

1. Page d'Accueil (Landing Page) :

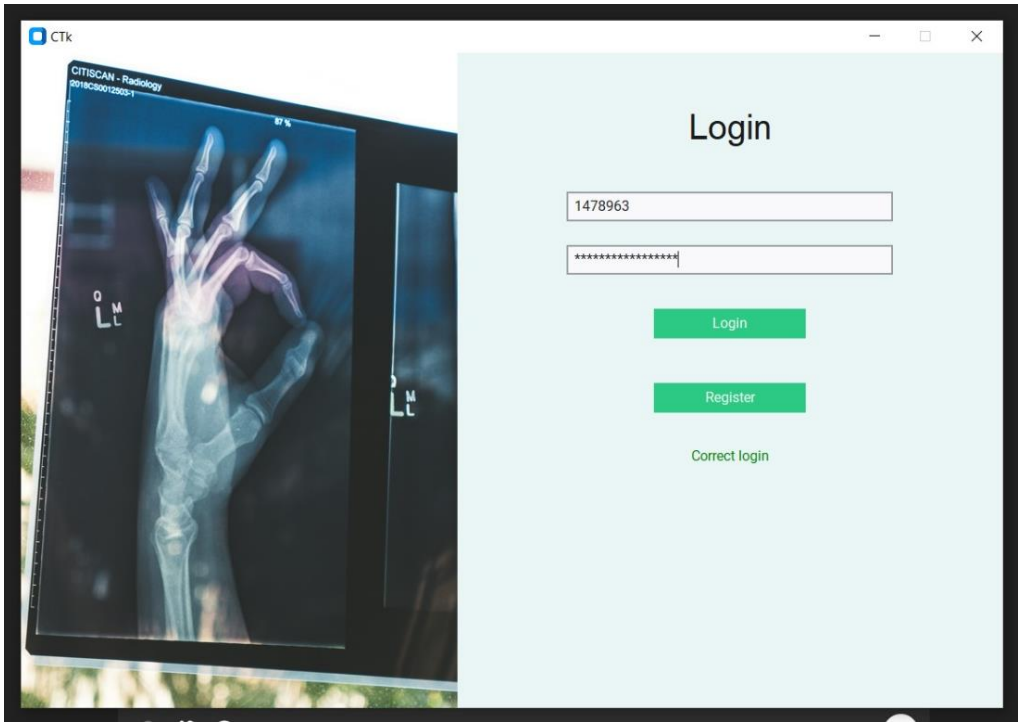
- Interface d'accueil avec une navigation intuitive.
- Choix pour les patients et les professionnels de santé, permettant l'accès à des sections spécifiques.

Gestion des Patients

1. Connexion des Patients :

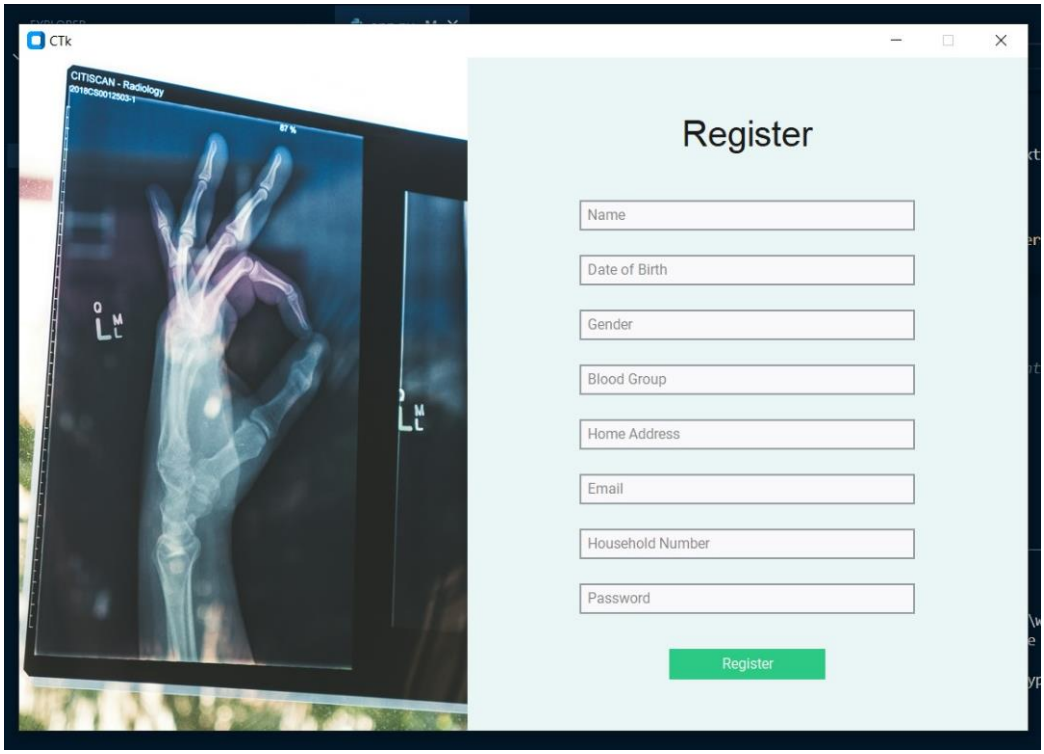
- Champs pour le numéro de santé et le mot de passe.

- Validation des identifiants pour l'accès sécurisé.

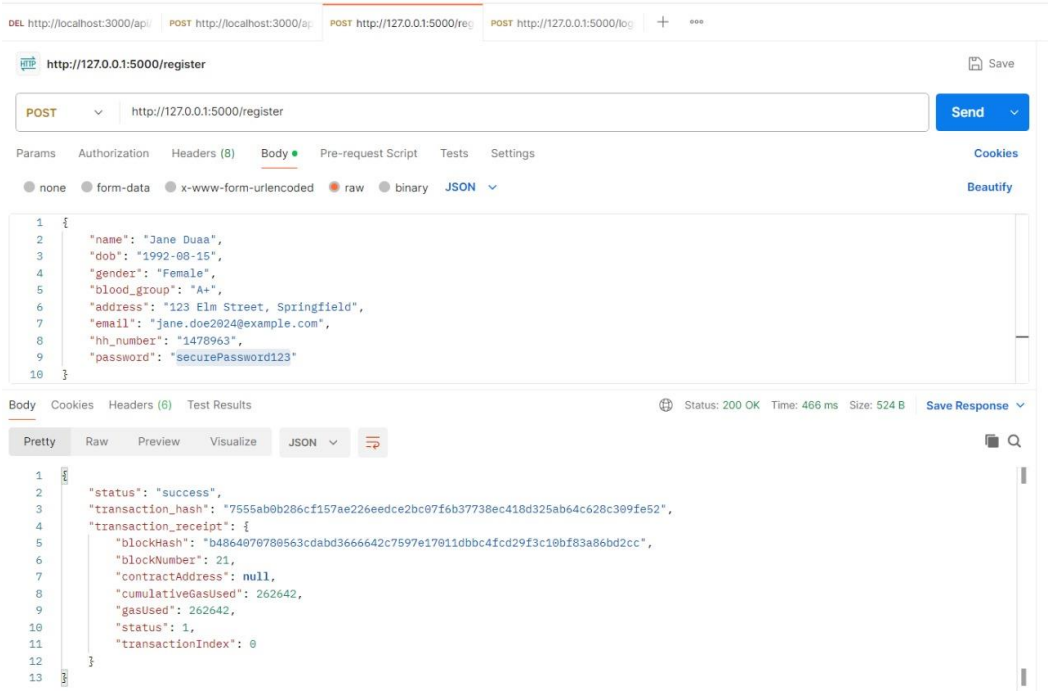


2. Inscription des Patients :

- Formulaire complet pour enregistrer les informations personnelles (nom, date de naissance, genre, groupe sanguin, etc.).
- Vérification de la correspondance entre les champs « mot de passe » et « confirmation du mot de passe ».
- Gestion des entrées dans un cadre déroulant pour optimiser l'expérience utilisateur sur des écrans réduits.



Les fonctionnalités d'authentification ont été testées à l'aide de Postman pour garantir leur bon fonctionnement et la sécurité des échanges.



Gestion des Professionnels de Santé

1. Connexion des Professionnels de Santé :

- Champs pour l'identifiant professionnel et le mot de passe.
- Validation des informations pour un accès sécurisé.

2. Inscription des Professionnels de Santé :

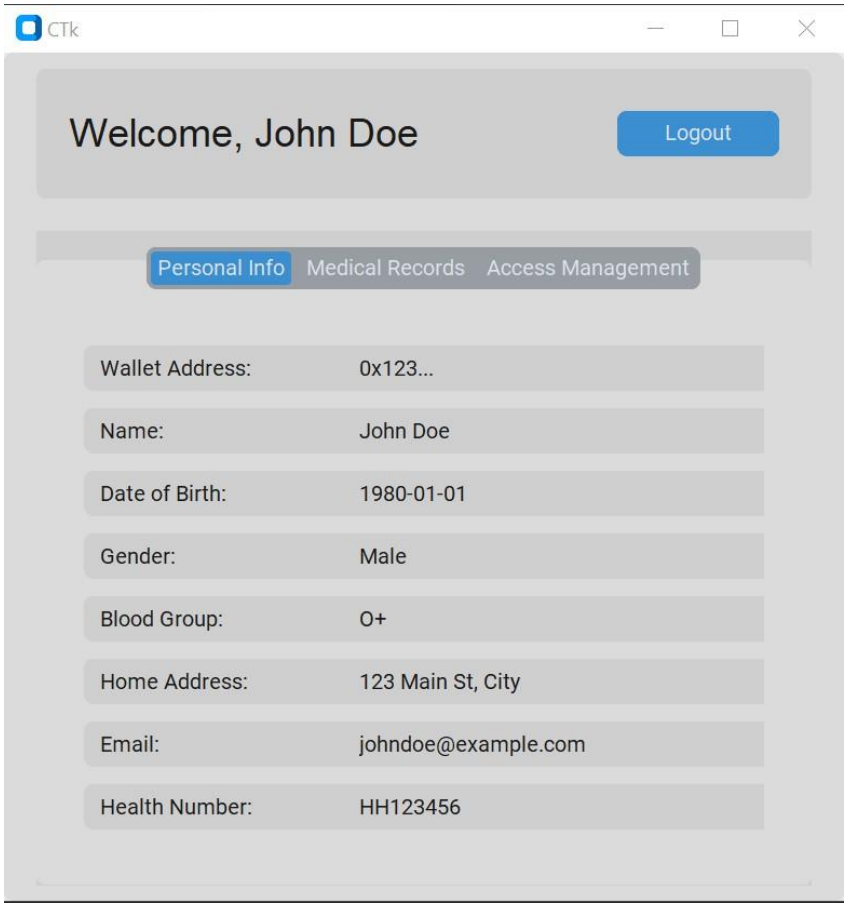
- Formulaire de création de compte avec des champs spécifiques comme la spécialisation et l'hôpital.
- Vérification de la correspondance des mots de passe.

Dashboard du Patient :

Cette interface de tableau de bord pour les patients permet de gérer les informations personnelles, les dossiers médicaux et les autorisations d'accès de manière efficace et sécurisée. Voici les principales fonctionnalités intégrées :

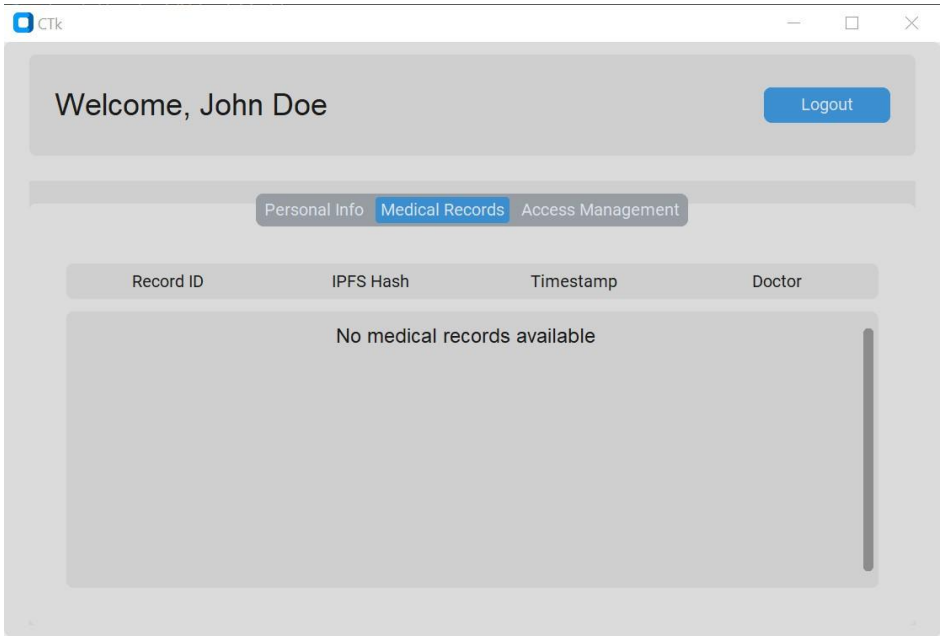
1. Affichage des informations personnelles

- Présentation des informations personnelles des patients, comme le nom, l'adresse e-mail, la date de naissance, et le groupe sanguin.
- Extraction et affichage des données directement depuis les informations fournies par les contrats intelligents sur la blockchain.



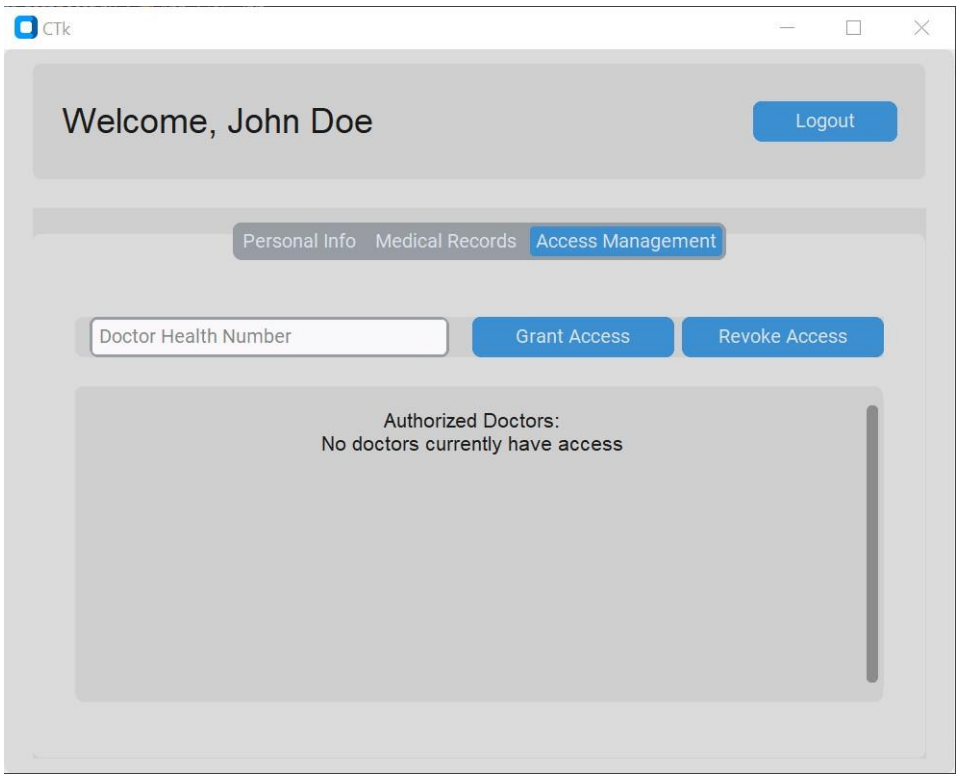
2. Gestion des dossiers médicaux

- Consultation des dossiers médicaux, avec des métadonnées importantes comme :
- Identifiant du dossier.
- Hash IPFS pour le stockage décentralisé.
- Horodatage des entrées.
- Nom du médecin associé.
- Intégration avec IPFS pour garantir un stockage sécurisé et distribué des fichiers.



3. Gestion des autorisations d'accès

- Ajouter ou révoquer des autorisations d'accès pour les médecins à l'aide de leur numéro de santé.
- Affichage des médecins autorisés, avec une interface conviviale pour mettre à jour les autorisations.
- Interaction avec des fonctions de contrat intelligent telles que grantAccess et revokeAccess.



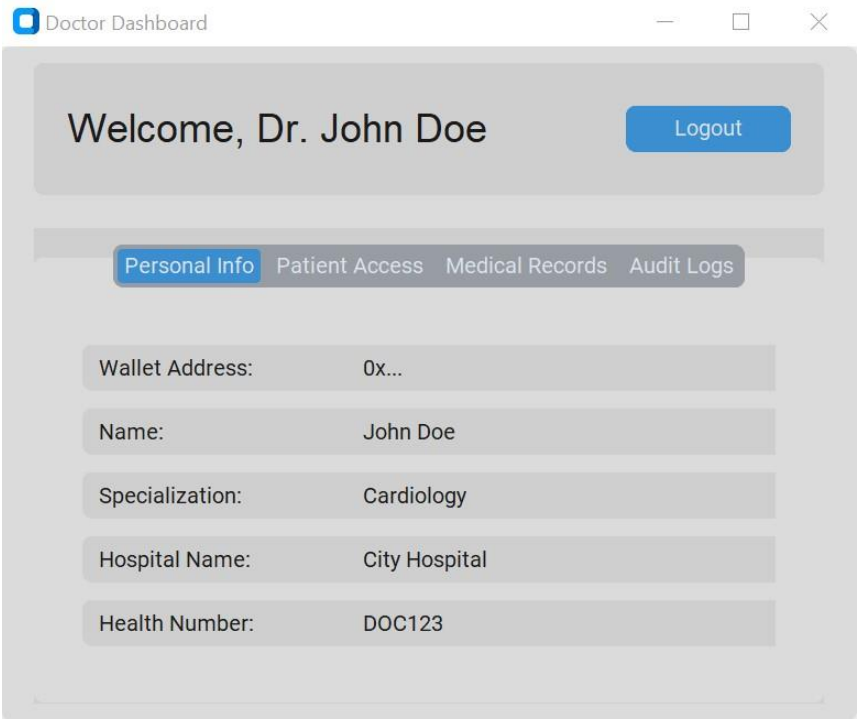
4. Déconnexion sécurisée

- Fonctionnalité de déconnexion pour garantir la sécurité et la confidentialité de la session patient.

Dashboard du Docteur:

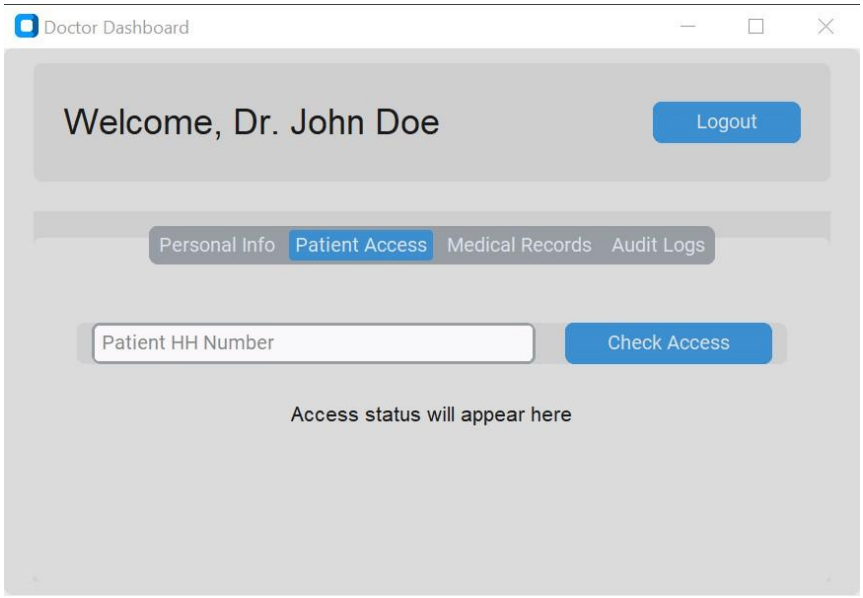
1. Gestion des Informations Personnelles

- Affiche les informations personnelles du médecin (nom, spécialisation, adresse de portefeuille, etc.).
- Permet de structurer les données conformément aux champs d'un contrat intelligent (smart contract).



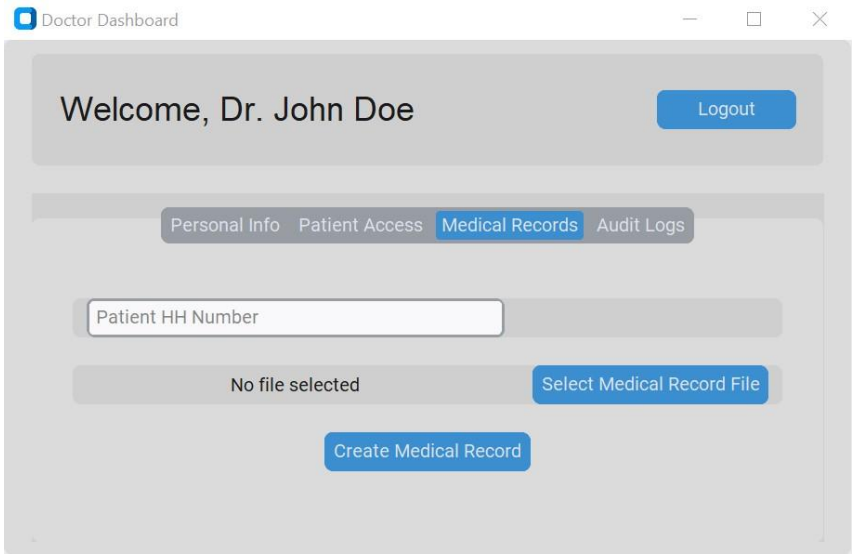
2. Accès Patient

- Vérifie si un médecin peut accéder aux informations médicales d'un patient via la vérification des permissions.
- Fournit un retour d'état (accès accordé/refusé) avec une interface utilisateur intuitive.



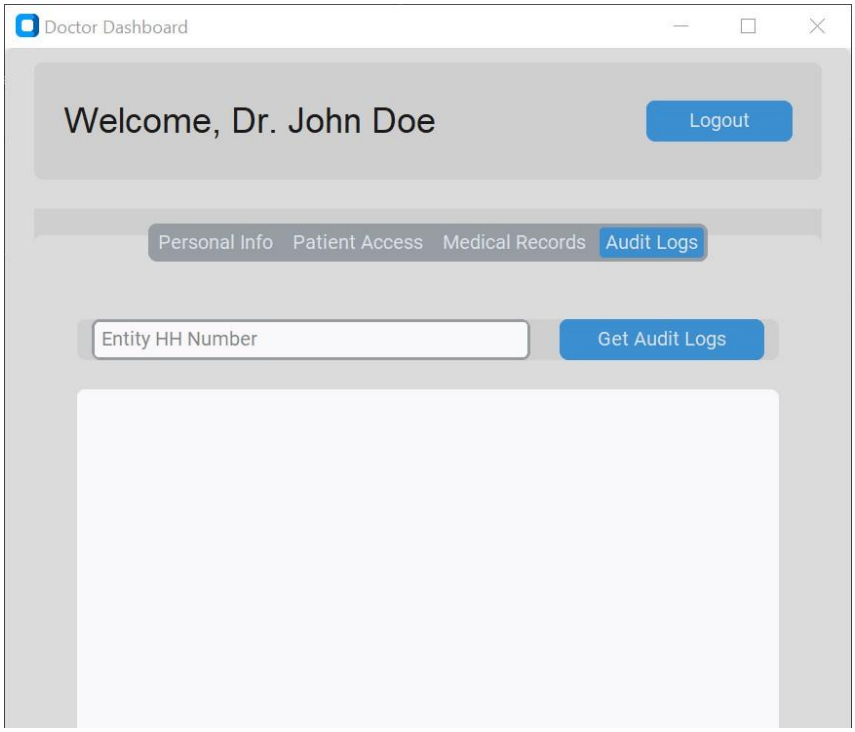
3. Création de Dossiers Médicaux

- Sélectionne des fichiers médicaux via une interface utilisateur conviviale.
- Crypte les fichiers avec une clé symétrique pour sécuriser les données.
- Télécharge les fichiers chiffrés sur IPFS via Pinata et enregistre le hash IPFS.
- Enregistre les métadonnées (hash IPFS et clé symétrique chiffrée) sur la blockchain.
- Log des activités liées à la création des dossiers médicaux pour audit.



4. Journaux d'Audit

- Recherche et affiche les journaux d'audit associés à des entités spécifiques (patients ou médecins).
- Log des opérations comme la création et l'accès aux dossiers médicaux pour traçabilité.



5. Mise à Jour de Dossiers Médicaux

- Fournit une fonctionnalité pour mettre à jour les informations d'un dossier médical existant.

6. Sécurisation et Cryptage

- Crypte les fichiers à l'aide de la bibliothèque cryptography pour protéger les informations sensibles.
- Implémente la gestion des clés symétriques pour le cryptage/décryptage.

7. Intégration avec Blockchain et IPFS

- Interaction avec des contrats intelligents via Web3 pour enregistrer et gérer les dossiers médicaux.
- Téléversement de fichiers sécurisés sur IPFS pour un stockage décentralisé.

Conclusion

Ce projet a permis de développer une solution innovante pour la gestion des dossiers médicaux, en intégrant la blockchain pour assurer l'intégrité, la sécurité et la traçabilité des données. Grâce à l'utilisation d'IPFS, nous avons optimisé le stockage des fichiers volumineux tout en garantissant leur accessibilité via des hachages sécurisés sur la blockchain.

Le système assure une gestion fluide des accès grâce aux smart contracts, offrant ainsi une solution décentralisée et fiable pour les patients et médecins. Ce projet marque une avancée dans la sécurisation des données de santé et ouvre la voie à une gestion de plus en plus décentralisée et sécurisée des informations sensibles.

En somme, cette solution répond aux besoins de transparence et de sécurité tout en offrant une architecture évolutive, adaptée aux enjeux futurs de la gestion des données médicales.