Ministère de l'Enseignement Supérieur et de la recherche scientifique

ECOLE NATIONALE POLYTECHNIQUE

Département d'électronique

Rapport Séminaire:

Implémentation d'une blockchain médicale pour la Gestion dans un hôpital intelligent

Réalisé par :

- **Medjdoub Omar**
- **Bouchfer Yousra**

Table des matières :

1. Introduction :

Quoi que nous disions, il nous sera impossible de surestimer l'importance de l'industrie de la santé. Cela dit, il s'agit facilement de l'une des industries à la croissance la plus lente de tout l'espace. Nous nous rendons compte que c'est une chose très controversée à dire, cependant, la preuve est dans le pudding.

Par rapport à il y a deux décennies, les hôpitaux, dans l'ensemble, fonctionnent toujours à peu près de la même manière. La raison, comme le dit Richie Etwaru, est son manque d'innovation. C'est en fait assez surprenant si l'on considère le fait que cet espace, en particulier, compte certaines des personnes les plus intelligentes et les plus éduquées du monde entier.

Cependant, dire qu'aucune innovation n'a été faite dans le domaine médical est vraiment une mauvaise chose à dire. Il suffit de regarder à quel point l'espérance de vie moyenne a augmenté grâce aux médicaments. Il faudra donc creuser un peu plus pour comprendre ce qu'Etwaru voulait dire en disant « manque d'innovation ».

Si vous examinez un peu plus en profondeur, vous remarquerez que cet espace regorge d'innovation verticale, mais il est toujours à la traîne en matière d'innovation horizontale. Alors qu'entend-on par innovation verticale et horizontale ?

L'innovation verticale est une innovation qui se fait spécifiquement dans un domaine particulier, tandis que l'innovation horizontale peut être adoptée par tous.

Prenons un exemple pour rendre cela plus clair :

La pénicilline, le vaccin antipoliomyélitique et les modes opératoires sophistiqués sont tous des exemples d'innovations verticales puisqu'ils ne sont spécifiques qu'à un domaine particulier.

L'électricité, Internet et le Cloud Computing, en revanche, sont des innovations horizontales qui ont été adoptées par de multiples domaines et industries pour rendre leurs fonctionnalités plus efficaces.

Le fait que la plupart des hôpitaux utilisent encore des papiers et des fichiers pour faire leurs dossiers montre qu'ils sont loin derrière en ce qui concerne innovation horizontale.

2. Présentation du Problème :

Les dossiers médicaux électroniques (EMRs) sont des informations privées critiques et très sensibles dans le domaine de la santé, et doivent être fréquemment partagées entre des pairs tels que des prestataires de soins de santé, des compagnies d'assurance, des pharmacies, des chercheurs, des familles de patients, entre autres.
L'utilisation de la technologie blockchain pour les soins primaires aux patients peut aider à résoudre les problèmes suivants des systèmes de santé actuels :

Un patient visite souvent plusieurs hôpitaux déconnectés. Il doit conserver l'historique de toutes ses données et maintenir les mises à jour. Cela conduit à la situation où les informations requises peuvent ne pas être disponibles.

En raison de l'indisponibilité des données, le patient peut avoir à répéter certains tests pour les résultats de laboratoire.
Ceci est courant lorsque les résultats sont stockés dans un autre hôpital et ne sont pas immédiatement accessibles.

Les données de santé sont sensibles et leur gestion est lourde.
Pourtant, il n'existe pas de système de protection de la vie privée dans la pratique clinique qui permette aux patients de maintenir une politique de contrôle d'accès de manière efficace.

Le partage de données entre différents prestataires de soins de santé peut exiger des efforts importants et prendre du temps.
S'appuyer sur une entité centralisée qui stockerait et gérerait les données des patients et les politiques de contrôle d'accès signifie avoir un point de défaillance unique et un goulot d'étranglement de l'ensemble du cadre.

La possibilité d'utiliser la blockchain pour la gestion des données de santé a récemment suscité beaucoup d'attention.
Dans nos travaux, nous nous concentrons sur la mise en œuvre pratique d'un système qui utilise la technologie de la blockchain pour crée un réseau qui gère et met à jour la chaîne d'approvisionnement complète, le stockage des données relatives au médicament, à la pharmacie, au pharmacien, à la prescription médicale, médecin, patient, infirmière et dose de médicament.
Cette technologie nous permet de garantir la sécurité et la confidentialité des données ainsi que la disponibilité au regard de la politique de contrôle d'accès définie par le patient.

3. Blockchain :

3.1. Définition :

La blockchain peut être décrite comme un registre immuable qui enregistre les données entrées de manière décentralisée. Il permet aux entités d'interagir sans la présence d'un tiers de confiance central. La blockchain maintient un ensemble sans cesse croissant d'entrées de données, regroupées en blocs de données. Ces blocs sont lors de l'acceptation de la blockchain liés au blocs précédents et futurs avec des protocoles cryptographiques.

Dans la forme originale de la blockchain, ces enregistrements / blocs de données sont : lisible par tous, inscriptible par tous et inviolable par tous. Cela permet par exemple de décentraliser les transactions et la gestion des données. En raison de ces propriétés, la blockchain a attiré beaucoup d'attention pour diverses applications.

De plus, la blockchain permet des contrats intelligents : contrats d'auto-exécutions qui ne nécessitent aucune autorité centrale.

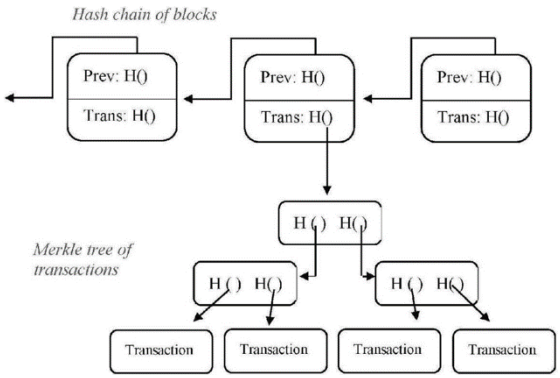
La blockchain Ethereum est à cette date le plus grand facilitateur de smart contracts sur la blockchain.

3.2. Les caractéristiques principales :

Un attribut clé de la blockchain est la décentralisation ; aucune autorité centrale ne contrôle le contenu ajouté à la blockchain. Au lieu de cela, les entrées transmises à la blockchain sont convenues dans un peer-to-peer

Réseau utilisant divers protocoles de consensus. Une autre caractéristique clé de la blockchain est la persistance. C'est pratiquement impossible de supprimer des entrées après avoir été acceptées sur le blockchain grâce au grand livre distribué, stocké sur plusieurs nœuds.

De plus, la possibilité de l'anonymat (ou pseudonymat) est un caractéristique attrayante utilisée dans de nombreuses chaînes de blocs.



-Fig. 1-

Les blockchains rendent l'audit et la traçabilité possibles en associant un nouveau bloquer au précédent en incluant le hachage de ce dernier, et de cette manière formant une chaîne de blocs. Les transactions dans les blocs sont formées dans un Merkle tree où chaque valeur feuille (transaction) peut être vérifiée pour la racine connue. Cela permet à l'arborescence de vérifier l'intégrité de les données en ne stockant que la racine de l'arbre sur la blockchain. Fig. 1 fournit une visualisation de cette structure de base

3.3. Type de blockchains :

Il existe deux types spécifiques de blockchains :

- Blockchains publiques
- Blockchains privées

Comme les deux sont des blockchains, ils fournissent un réseau peer-to-peer qui offre un écosystème décentralisé et immuable synchronisé via des protocoles de consensus.

- **Chaînes publiques :**

Les blockchains publiques sont celles que nous connaissons le mieux. Bitcoin, Ethereum, etc. sont tous des blockchains publiques et la raison pour laquelle ils sont appelés ainsi est assez explicite. Ce sont des écosystèmes complètement ouverts où tout le monde peut participer à l'écosystème. Le réseau dispose également d'un mécanisme d'incitation intégré qui récompense les participants pour leur participation plus approfondie au système.

Cependant, le secteur des soins de santé bénéficiera-t-il d'une blockchain publique ? Enfin… pas tellement.

Premièrement, comme cela a été extrêmement bien documenté, les blocs dans Bitcoin et Ethereum ont un problème de stockage. Bitcoin a un peu plus de 1 Mo d'espace par bloc, ce qui n'est tout simplement pas suffisant pour exécuter le type de transactions et stocker le type de données dont les établissements de santé ont besoin.

Ensuite, nous avons les problèmes de débit qui ont également été assez bien documentés. Bitcoin peut à peine gérer 7 à 8 transactions par seconde. Le temps de confirmation du bloc est de 10 minutes, ce qui ne fait qu'ajouter à la latence. Les grands instituts de santé doivent gérer d'énormes blocs de transactions par jour avec une latence proche de 0. En fait, toute sorte de latence peut potentiellement mettre la vie en danger.

Les blockchains publiques, en particulier celles qui suivent le protocole de preuve de travail comme Bitcoin, nécessitent une énorme quantité de puissance de calcul pour résoudre des énigmes difficiles. En tant que tel, il est vraiment peu pratique pour ces instituts de dépenser autant d'argent dans des mécanismes de consensus.

Enfin, les blockchains publiques sont des chaînes ouvertes, ce qui en soi est un autre inconvénient. Pensez-y, pourquoi les instituts de santé devraient-ils essayer d'interagir les uns avec les autres dans un réseau où tout le monde peut entrer et en faire partie. Les instituts médicaux traitent des données hautement classifiées et sensibles, pourquoi voudront-ils que quelqu'un en dehors de leurs cercles interagisse avec elles ?

Ainsi, les chaînes publiques ne sont pas pratiques à ces fins. Cependant, il existe un autre type de blockchain qui est pratique pour les instituts de santé, et on les appelle des blockchains privées.

- **Chaînes privées :**

Les chaînes privées sont… bien… privées. Contrairement aux blockchains publiques, celles-ci ne sont pas ouvertes à tout le monde.

En conséquence, les personnes qui souhaitent participer à la chaîne privée doivent obtenir la permission de faire partie de ce réseau. C'est la raison pour laquelle les chaînes privées sont également appelées « blockchains autorisées ».

Pour cette raison, il existe des restrictions quant au type de personnes qui peuvent réellement participer au consensus. L'accès pour les nouveaux participants pourrait être accordé par les moyens suivants :

- Les participants existants qui participent à l'écosystème.
- Une autorité réglementée.
- Un consortium.

Une fois qu'une entité a rejoint l'écosystème, elle peut jouer un rôle dans la maintenance du réseau. Hyperledger Fabric de la Linux Foundation est un exemple d'implémentation de framework de blockchain autorisée et de l'un des projets Hyperledger hébergés par la Linux Foundation. Il a été conçu pour répondre à ces exigences d'entreprise.

Ces chaînes privées ont été spécialement conçues pour les besoins des entreprises et offrent de nombreuses fonctionnalités telles que :

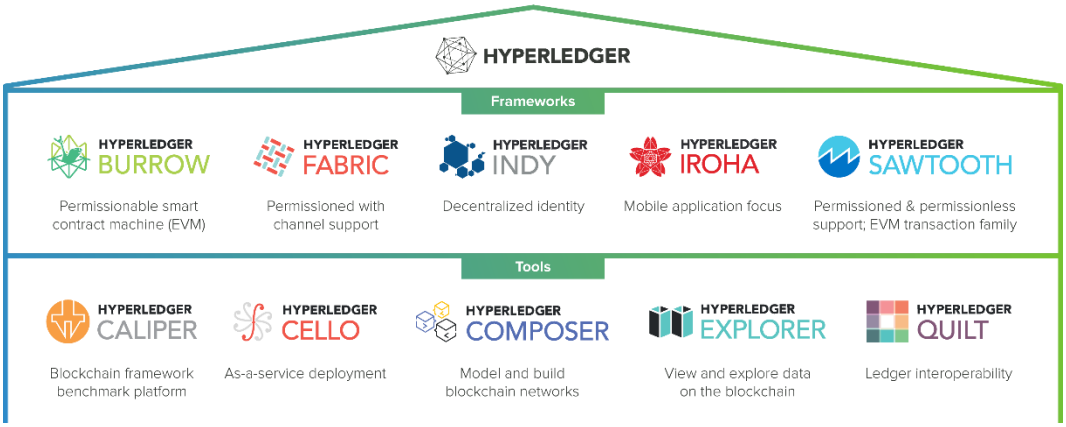
- Transactions rapides
- Immunité.
- Haute sécurité

Ok, donc nous avons un côté de l'équation, qui est la chaîne privée. Cependant, il y a une autre pièce du puzzle que nous devons comprendre avant de comprendre comment l'industrie médicale fonctionnera sur la blockchain.

3.4. Hyperledger :

Hyperledger est une communauté open source axée sur le développement d'une suite de cadres, d'outils et de bibliothèques stables pour les déploiements de blockchain de niveau entreprise.

Il sert de foyer neutre pour divers cadres de registres distribués, notamment Hyperledger Fabric, Sawtooth, Indy, ainsi que des outils tels que Hyperledger Caliper et des bibliothèques comme Hyperledger Ursa.



La Famille Hyperledger

3.4.1. Hyperledger Fabric:

Hyperledger Fabric est une plateforme Open Source Distributed Ledger. La plate-forme a une architecture de module très sophistiquée qui permet au développeur de concevoir le réseau blockchain avec sécurité, évolutivité, confidentialité et haute performance. Hyperledger Fabric prend en charge les contrats intelligents pour écrire la logique de code d'application principale pour un conteneur. Le contrat intelligent ou Chaincode peut être écrit dans les langages de programmation Java, Node.js et Go. Il est plus facile pour les développeurs qui ont une

expérience de ces langages spécifiques d'écrire efficacement un code de chaîne à l'aide du SDK Hyperledger Fabric.

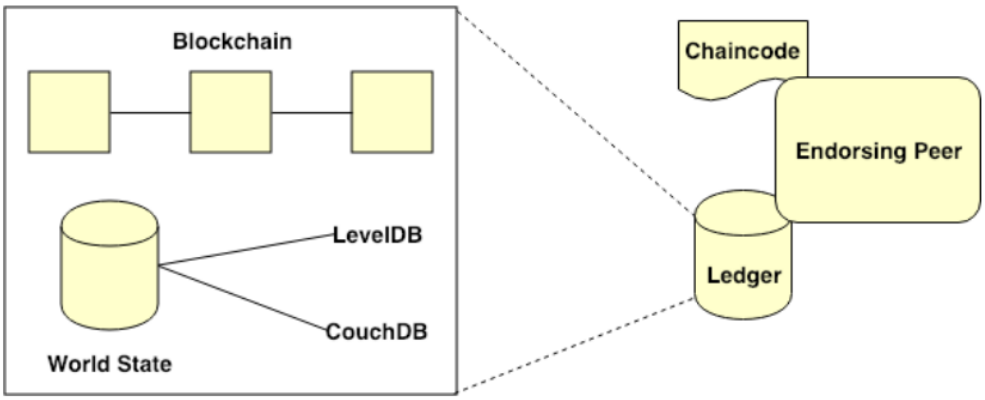
Hyperledger Fabric utilise également Docker comme environnement de conteneur pour stocker toutes les images des composants de fabric, pour ne citer que quelques images comme fabric-ca, fabric-orderer, fabric-CouchDB, fabric-tools et bien d'autres. Le développeur doit définir le nom d'image requis dans un fichier docker-compose.yaml pour télécharger les images dans l'environnement docker local.

3.4.1.1. Composent de Hyperledger Fabric :

Il y a peu de composants majeurs de la structure Hyperledger qui doivent être compris avant de plonger dans le développement de base.

Ledger :

Dans Hyperledger fabric ledger se trouve un journal d'historique des transactions qui contient l'état de chaque transaction validée dans le ledger. Le grand livre se compose de deux parties World State, Blockchain.

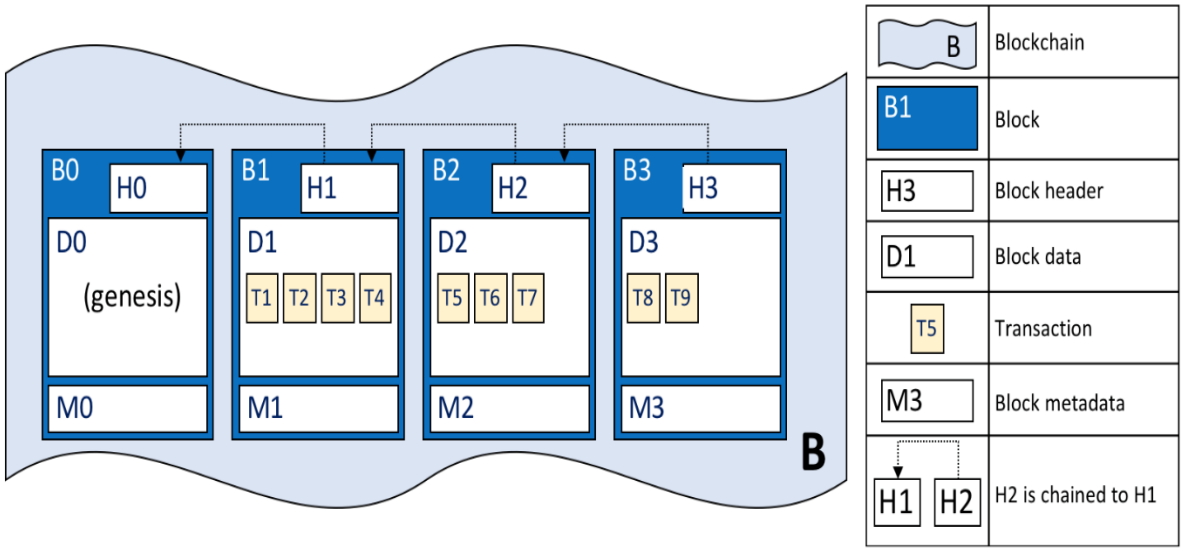


1. World State - C'est une base de données qui contient les enregistrements mis à jour pour une transaction. L'état du monde devient plus confortable pour le programmeur pour vérifier que l'enregistrement est stocké avec précision dans le grand livre ou non. Hyperledger fabric prend en charge LevelDB et CouchDB comme base de données d'état pour stocker le dernier état de l'objet dans une paire clé-valeur. L'état mondial peut être créé, mis à jour et supprimé.

2. Blockchain - est un ensemble de blocs séquentiels contenant la séquence des transactions effectuées dans l'état du monde. Hyperledger Fabric utilise un service de commande pour le séquençage des transactions dans les blocs. Le hachage de la prise en charge des blocs sécurise les données du grand livre, chaque en-tête de bloc comprend un hachage de la transaction du bloc, de sorte que toutes les transactions du grand livre sont séquencées dans l'ordre. Les données du grand livre une fois enregistrées dans la blockchain ne peuvent pas être modifiées.

Blocks :

Dans Blockchain, le premier bloc est appelé le bloc de genèse. Ce bloc initial ne contient aucune transaction utilisateur mais stocke la configuration du canal dans le réseau.



Il y a trois parties dans une structure de bloc :

1. Block Header - Lorsqu'un bloc est créé, son en-tête contient trois champs.

Block number: il s'agit d'un nombre entier commençant par (0) pour le bloc de genèse et par incréments ultérieurs pour chaque nouveau bloc ajouté à la blockchain.

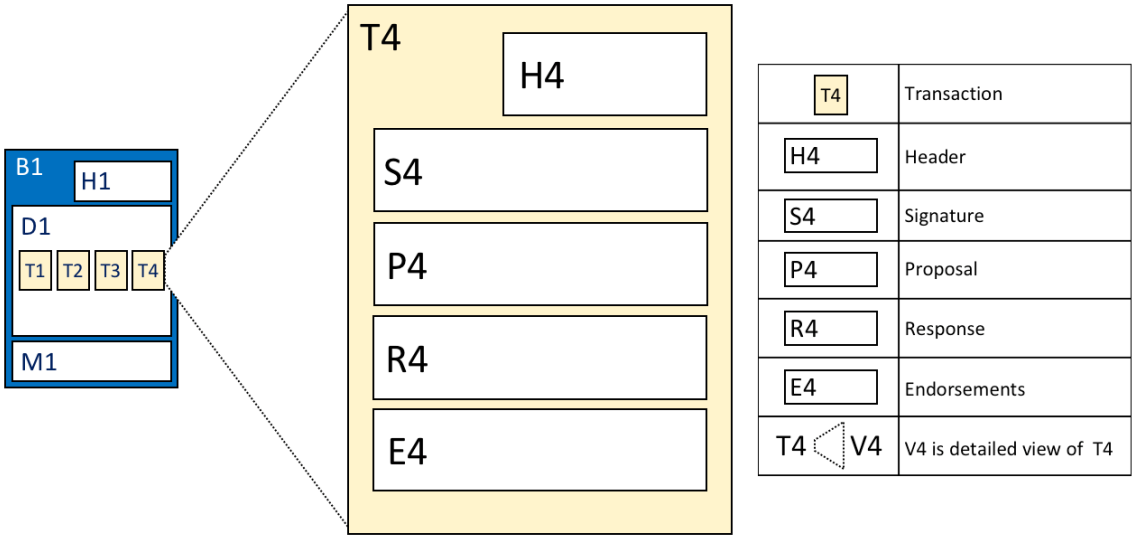
Current Block Hash - Il s'agit de la valeur de hachage de l'enregistrement de bloc actuel. Ainsi, chaque bloc de données fait référence à une valeur de hachage cryptographique unique.

Previous Block Hash - Il s'agit d'une copie du hachage de bloc précédent dans la blockchain. L'avantage majeur d'avoir ce hachage est que chaque bloc sera lié au bloc voisin afin que les données du bloc ne puissent pas être modifiées.

2. Block Data - Il s'agit de la section où chaque transaction est organisée dans l'ordre par le service de commande.

3. Block Metadata - Les métadonnées du bloc contiennent l'heure de création du bloc, les détails du certificat, la clé publique et la signature du rédacteur de bloc.

Transaction :
Le rôle d'une transaction est de stocker l'état d'un objet. Il y a 5 champs pour chaque structure de transaction.



Header - Un Header contient les métadonnées importantes sur la transaction, telles que le nom du code de chaîne et sa version.

Signature - Pour une transaction dans la blockchain, l'application cliente signe la transaction avec une clé privée pour vérifier l'intégrité des enregistrements de la transaction.

Proposal - Pour chaque mise à jour de l'état du grand livre, le Chaincode nécessite quelques ensembles d'une proposition de l'application contenant les paramètres d'entrée pour changer l'état actuel du monde en un nouvel état mondial.

Response - Chaque fois qu'un code de chaîne exécute une proposition de modification de l'état du monde actuel, la réponse de transaction capture l'état du monde avant et après, en tant que jeu de lecture-écriture. Si la sortie du code de chaîne est correcte, la transaction est validée avec succès et le registre met à jour l'état du monde.

Endorsement - Le résultat d'une transaction doit être accepté par chaque organisation du réseau avant de le mettre à jour dans le grand livre. Ainsi, une liste de réponses de transaction signées doit accepter par l'organisation de valider l'état dans le grand livre. Le développeur peut définir des politiques d'approbation pour l'organisation et peut être exécuté à l'aide de Chain

Nodes :

Node ou (nœud) n'est qu'une fonction logique dans le sens où plusieurs nœuds de types différents peuvent s'exécuter sur le même serveur physique. Ce qui compte, c'est la manière dont les nœuds sont regroupés en « domaines de confiance » et associés aux entités logiques qui les contrôlent.

Il existe trois types de nœuds :

1. Client : Un client qui soumet un appel de transaction réel aux endosseurs, et diffuse des propositions de transaction au service de commande. En bref, les clients communiquent avec leurs pairs et le service de commande

2. Peer : un nœud qui valide les transactions et maintient l'état et une copie du grand livre. Un pair reçoit des mises à jour d'état ordonnées sous la forme de blocs du service de commande et maintient l'état et le grand livre. En outre, les pairs peuvent avoir un rôle d'approbation spécial. La fonction spéciale d'un pair endosseur se produit par rapport à un code de chaîne particulier et consiste à endosser une transaction avant qu'elle ne soit validée.

3. Ordering-service-node ou Orderer: un nœud exécutant le service de communication qui met en œuvre une garantie de livraison, telle qu'une diffusion de commande atomique ou totale.

Types de peer :

• **Endoring Peer**: l'approbation des pairs est un type spécial de pairs qui s'engagent qui ont un rôle supplémentaire pour approuver une transaction. Ils approuvent la demande de transaction qui provient du client. Chaque pair endosseur possède la copie du contrat intelligent installé et un registre. La fonction principale d'Endorser est de simuler la transaction. Il est exécuté sur la base du contrat intelligent sur la copie personnelle du grand livre et génère les jeux de lecture / écriture qui sont envoyés au client. Bien que pendant la simulation, la transaction n'est pas validée dans le grand livre.

• **Committing Peer**: Pairs qui commettent le bloc reçu du service de commande, dans leur propre copie de la blockchain. Ce bloc contient la liste des transactions où l'homologue s'engage à valider chaque transaction et à la marquer comme valide ou non valide et à s'engager dans le bloc. Toutes les transactions valides ou non valides sont toutes engagées dans la blockchain à des fins d'audit futur.

• **Anchor Peer**: étant donné que le réseau Fabric peut s'étendre sur plusieurs organisations, nous avons besoin de certains pairs pour communiquer au sein d'une organisation. Tous les pairs ne peuvent pas faire cela, mais ce sont des pairs spéciaux qui sont uniquement autorisés à le faire et qui ne sont rien d'autre que des pairs d'ancrage. Les homologues d'ancrage sont définis dans la configuration du canal.

• **Leader Peer**: les pairs leaders sont ceux qui communiquent ou diffusent les messages du service de commande à d'autres pairs de la même organisation. Ces pairs utilisent le protocole Gossip pour s'assurer que chaque pair reçoit le message. Les pairs de premier plan ne peuvent pas communiquer dans une organisation. Si un pair principal ne répond pas ou est hors du réseau, nous pouvons sélectionner un pair principal parmi les pairs disponibles en fonction du vote ou en choisir un au hasard.

Ordering Service :

Ordering Service ou le service de commande peut être composé d'un ou de plusieurs nœuds de commande. Un ordre est un nœud exécutant le binaire de commande Hyperledger Fabric. Comme son nom l'indique, un service de commande effectue la commande des transactions, mais uniquement pour les demandes d'appel de code de chaîne. Le service de commande prend en charge la multi-location, ce qui signifie qu'un service de commande peut commander des transactions pour plusieurs canaux.

Consensus :

Un consensus est un accord sur l'ordre des transactions entre des nœuds de service de commande individuels. Hyperledger Fabric prend actuellement en charge 3 mécanismes ou implémentations de consensus différents - SOLO, Kafka et Raft. Et pour les réseaux de production, SOLO et Kafka ne sont certainement pas recommandés. SOLO ne fournit ni haute disponibilité (HA), ni décentralisation. Kafka fournit la haute disponibilité mais pas la décentralisation, tandis que Raft fournit les deux.

	High availability	Decentralization
SOLO	**No**	**No**
Kafka	**Yes**	**No**
Raft	**Yes**	**Yes**

Smart Contract (Chaincode) :

Dans les termes d'Hyperledger Fabric, un smart contract est un morceau de code (écrit en Go, node.js ou Java) exécutant le traitement des contrats entre les parties. Chaincode est un smart contract packagé préparé pour l'installation. Essentiellement, un contrat intelligent se trouve dans un package de code de chaîne.

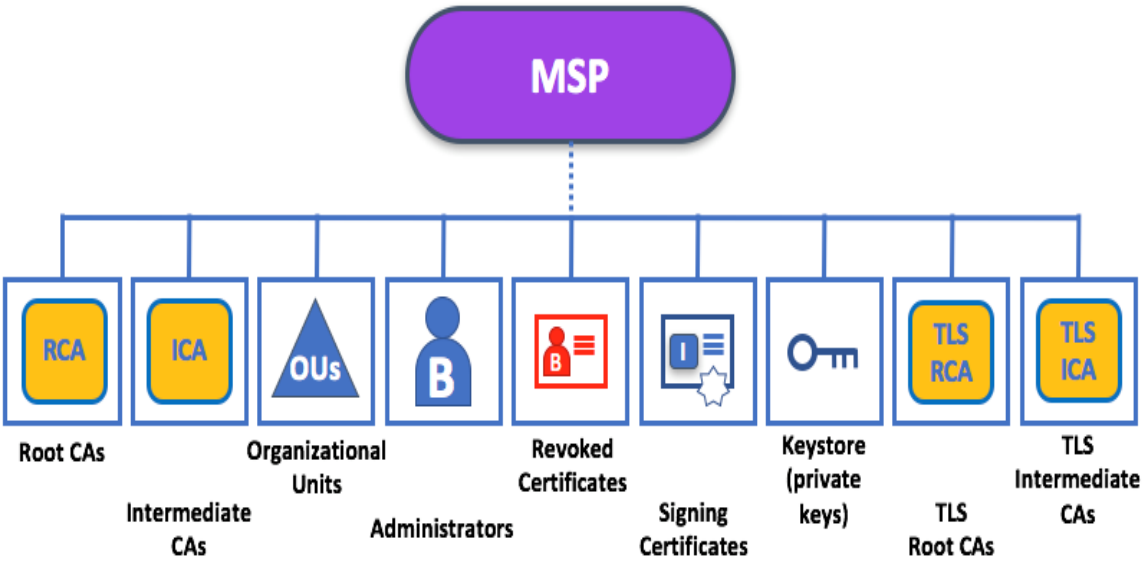
En général, le smart contract est l'endroit où l'automatisation des processus entre les organisations entre en place. Les contrats intelligents fonctionnent avec des données sur la blockchain pour automatiser généralement les activités ainsi que les flux de travail. La plupart de l'automatisation intelligente des contrats est écrite de manière à ce que si quelque chose se produit, cela déclenche une action différente qui sera automatiquement écrite dans le grand livre. Un exemple d'automatisation intelligente des contrats peut être le suivi d'un SLA, et s'il est violé, il engagera automatiquement une remise dans le grand livre pour le service informatique fourni. Un autre exemple peut être de savoir comment, dans les élections, un contrat intelligent engage automatiquement un gagnant sur le grand livre lorsque tous les votes sont comptés. Bien entendu, d'autres actions ou chaînes d'actions peuvent être mises en œuvre.

Membership Service Provider :

Le MSP est un composant majeur pour définir les rôles spécifiques, les identités de chaque participant au réseau. Le MSP fournit une authentification appropriée, les informations d'identification qui souhaitent rejoindre le réseau. MSP gère également l'autorité de certification pour chaque identité. Hyperledger Fabric contrôlait deux types de MSP pour chaque organisation.

Local MSP - Ceci définit le droit d'accès administratif ou de participant pour l'utilisateur, les pairs, l'ordre dans le réseau.

Channel MSP - Ceci définit le droit d'administrateur ou de participant pour les canaux.



Role du MSP

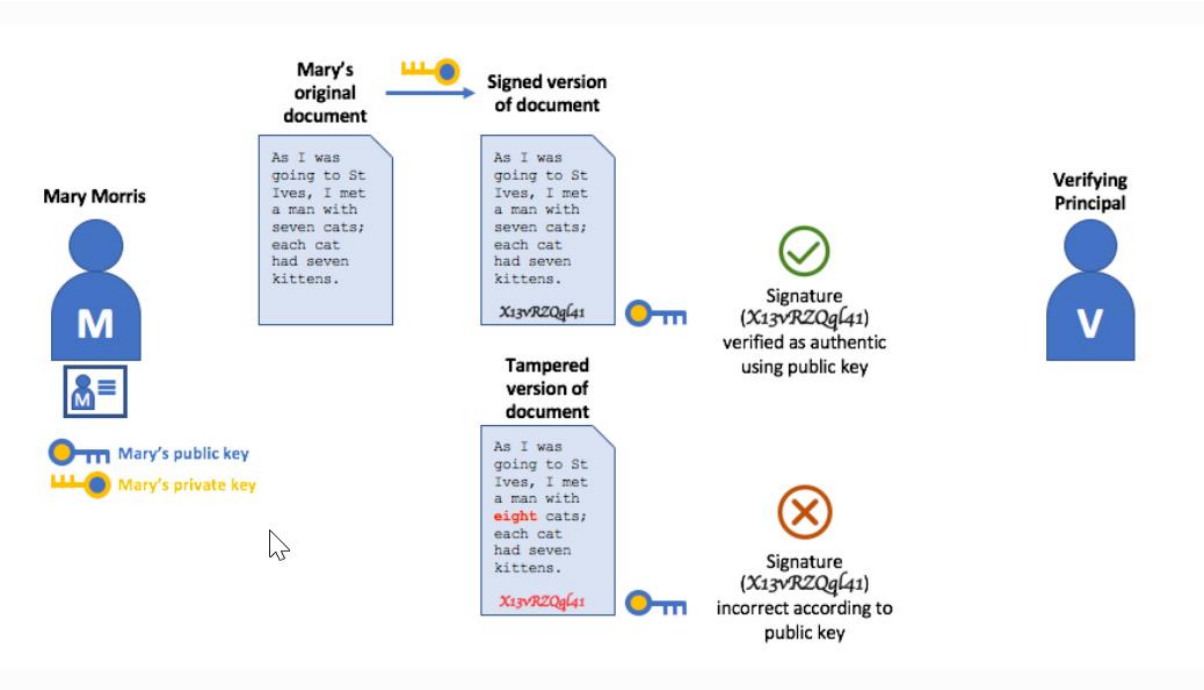
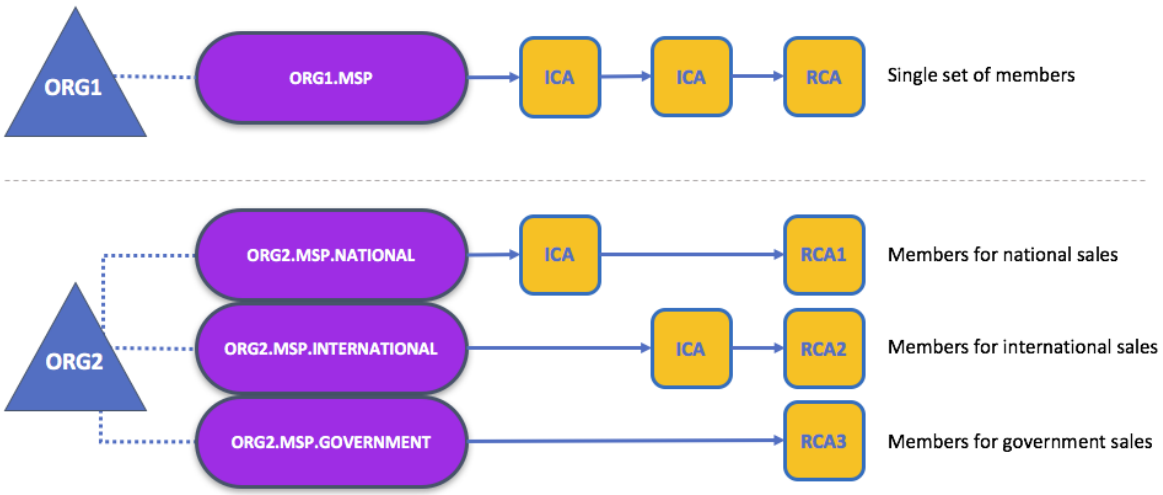


Figure qui indique le processus de génération de certificat par le MSP

Organizations :

Il s'agit d'un ensemble de groupes de membres sous un seul MSP. Une organisation peut être liée à un grand groupe multi-sociétés ou à un petit café. La convention de dénomination pour l'organisation MSP peut être dérivée comme Org1.MSP, ici l'organisation est «Org1». Une organisation peut également avoir de nombreux MSP en fonction de ses besoins.



Channels :

Hyperledger Fabric a une technique qui permet à l'organisation de rejoindre plusieurs réseaux blockchain via un canal. Ainsi, plusieurs organisations peuvent communiquer ou participer à de nombreux réseaux mettant en œuvre divers canaux.

Par exemple :

Org1 & Org2 veulent communiquer puis le canal sera - C1 et Org1 &

Org3 veut communiquer puis le canal sera - C2

Ici Org1 utilise deux canaux C1 et C2 à des fins différentes individuellement.

3.4.1.2. Cycle de vie d'une transaction :

Le client envoie une proposition de transaction aux nœuds homologues d'approbation spécifiés (endosseurs). Les endosseurs, confirmant que le client est valide pour envoyer une proposition de transaction, exécutent le code de chaîne selon les informations spécifiées dans la proposition et dans l'état du monde. Après exécution, ils renvoient la réponse de proposition au client. À l'intérieur de la réponse se trouvent les résultats de simulation effectués par cet endosseur, en termes d'un ensemble de lecture et d'écriture (RWSet), qui montre ce qui est lu à partir de l'état du monde et ce qui est réécrit après l'appel du code de chaîne. En guise d'approbation, les endosseurs signent la réponse. Notez qu'à ce stade, rien n'est encore changé dans le grand livre.

Le client, après avoir obtenu les réponses, les valide pour s'assurer que tout fonctionne bien. Si les réponses collectées atteignent la politique d'approbation requise (par exemple, un pair est requis de

chaque organisation), le client construit une transaction et l'envoie au client. La transaction comprend la proposition de transaction, la réponse à la proposition et les approbations.

Le client reçoit la ou les transactions du (des) client (s) et les valide. Ensuite, le client crée un bloc contenant les transactions validées et diffuse ce bloc nouvellement créé à tous les nœuds homologues. À la réception de ce bloc, le nœud homologue valide le bloc et exécute la transaction à l'intérieur du bloc, ce qui mettra à jour l'état du monde en fonction du contenu de l'ensemble de lecture et d'écriture (RWSet). Ce bloc est maintenant validé dans chaque nœud homologue en tant que bloc valide.

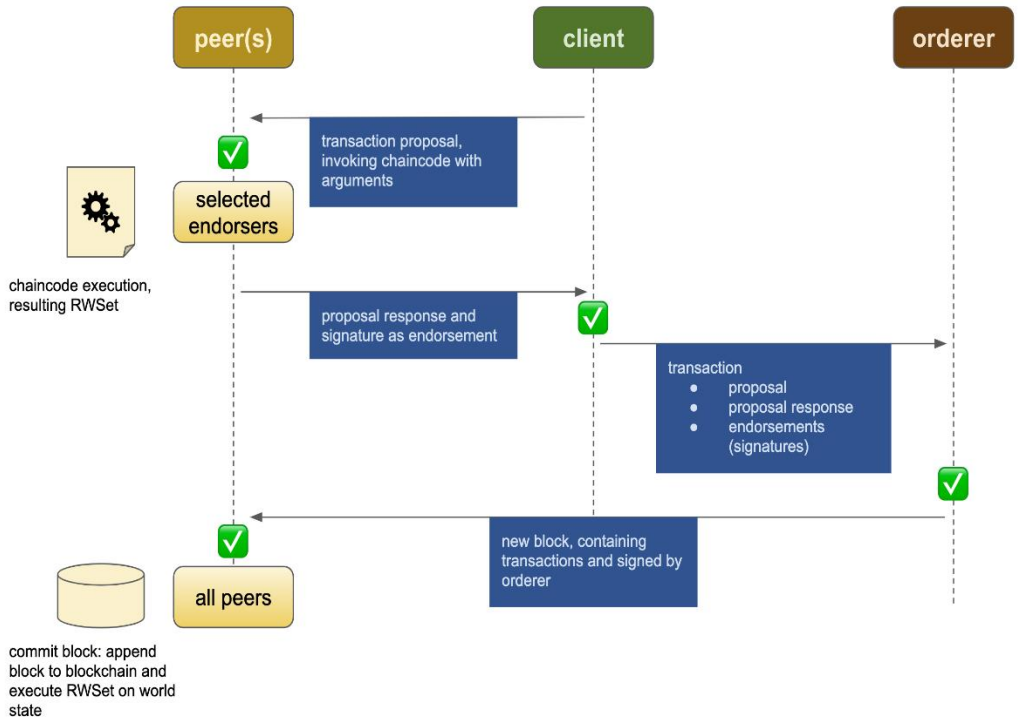
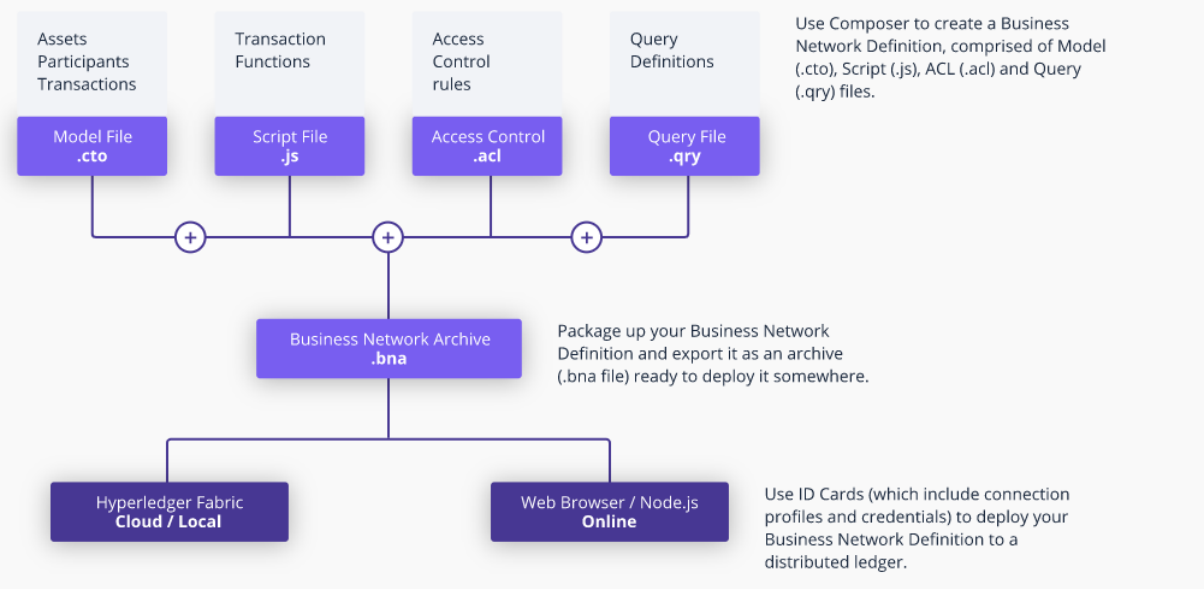


Schéma qui représente le cycle de vie d'une transaction

3.4.2. Hyperledger Composer :

Hyperledger Composer est un ensemble d'outils et un cadre de développement ouverts et complets pour faciliter le développement d'applications blockchain. Notre objectif principal est d'accélérer le délai de rentabilisation et de faciliter l'intégration de vos applications blockchain aux systèmes d'entreprise existants. Vous pouvez utiliser Composer pour développer rapidement des cas d'utilisation et déployer une solution blockchain en quelques semaines plutôt qu'en mois. Composer vous permet de modéliser votre réseau d'entreprise et d'intégrer des systèmes et des données existants à vos applications blockchain.

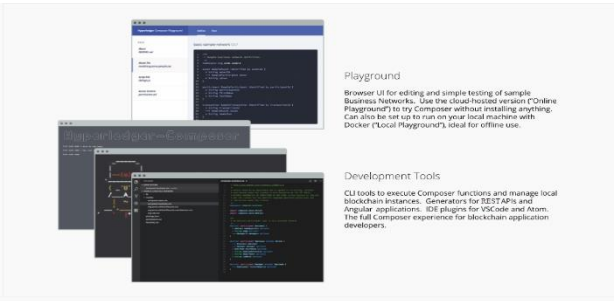
Hyperledger Composer prend en charge l'infrastructure et le runtime de blockchain Hyperledger Fabric existants, qui prend en charge les protocoles de consensus de blockchain enfichables pour garantir que les transactions sont validées conformément à la politique par les participants du réseau d'entreprise désignés.



Architecture de Hyperledger Composer

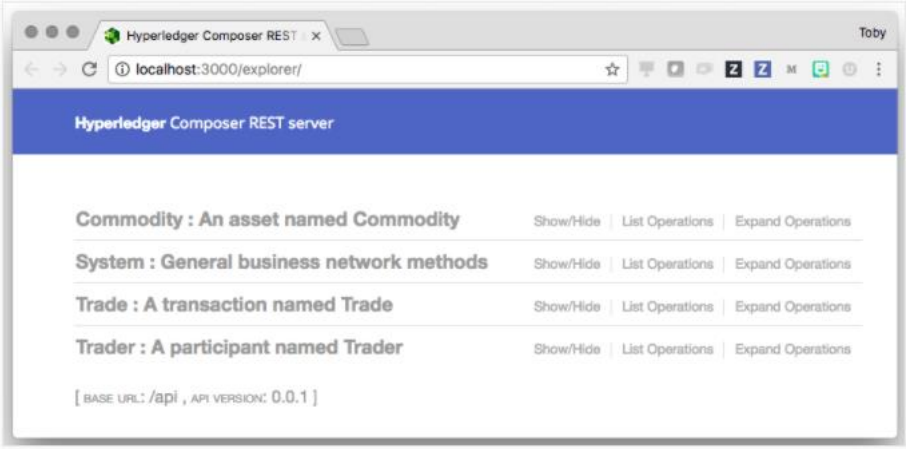
3.4.2.1. Composant de Hyperledger Composer:

Hyperledger Composer avec son interface utilisateur basée sur un navigateur appelée Hyperledger Composer Playground. Playground est disponible en version hébergée (aucune installation nécessaire) ou en installation locale (idéale pour éditer et tester des exemples de réseaux d'entreprise hors ligne).



REST server API :

Le serveur REST Hyperledger Composer, composer-rest-server, peut être utilisé pour générer une API REST à partir d'un réseau d'entreprise blockchain déployé qui peut être facilement consommée par les clients HTTP ou REST.



3.4.2.2. Concept clé sur Hyperledger Composer:

Blockchain State Storage

Toutes les transactions soumises via un réseau d'entreprise sont stockées dans le grand livre de la blockchain, et l'état actuel des actifs et des participants est stocké dans la base de données d'état de la blockchain. La blockchain distribue le registre et la base de données d'état sur un ensemble de pairs et garantit que les mises à jour du registre et de la base de données d'état sont cohérentes entre tous les pairs à l'aide d'un algorithme de consensus.

Connection Profiles

Hyperledger Composer utilise des profils de connexion pour définir le système auquel se connecter. Un profil de connexion est un document JSON qui fait partie d'une carte réseau professionnelle. Ces profils sont généralement fournis par le créateur du système auquel ils se réfèrent et doivent être utilisés pour créer des cartes réseau professionnelles afin de pouvoir se connecter à ce système.

Assets

Les actifs sont des biens corporels ou incorporels, des services ou des biens, et sont stockés dans des registres. Les actifs peuvent représenter presque tout dans un réseau d'entreprise, par exemple, une maison à vendre, la liste de vente, le certificat de cadastre de cette maison et les documents d'assurance pour cette maison peuvent tous être des actifs dans un ou plusieurs réseaux commerciaux.

Les assets doivent avoir un identifiant unique, mais à part cela, ils peuvent contenir les propriétés que vous définissez. Les actifs peuvent être liés à d'autres actifs ou participants.

Participants

Les participants sont membres d'un réseau d'entreprises. Ils peuvent posséder des actifs et soumettre des transactions. Les types de participants sont modélisés et, comme les actifs, doivent avoir un identifiant et peuvent avoir toutes les autres propriétés requises. Un participant peut être associé à une ou plusieurs identités.

Identities

Une identité est un certificat numérique et une clé privée. Les identités sont utilisées pour effectuer des transactions sur un réseau d'entreprise et doivent être mappées à un participant du réseau d'entreprise. Une identité unique est stockée dans une carte de réseau d'entreprise et si cette identité a été mappée à un participant, elle permet à l'utilisateur de cette carte de réseau d'entreprise d'effectuer des transactions sur un réseau d'entreprise en tant que ce participant.

Business Network cards

Les cartes réseau professionnelles sont une combinaison d'une identité, d'un profil de connexion et de métadonnées, les métadonnées contenant éventuellement le nom du réseau professionnel auquel se connecter. Les cartes réseau professionnelles simplifient le processus de connexion à un réseau d'entreprise et étendent le concept d'identité en dehors du réseau d'entreprise à un «portefeuille» d'identités, chacune associée à un réseau d'entreprise et un profil de connexion spécifiques.

Transactions

Les transactions sont le mécanisme par lequel les participants interagissent avec les actifs. Cela peut être aussi simple qu'un participant plaçant une offre sur un actif dans une enchère, ou un commissaire-priseur marquant une enchère close, transférant automatiquement la propriété de l'actif au plus offrant.

Queries

Les requêtes sont utilisées pour renvoyer des données sur l'état du monde de la blockchain. Les requêtes sont définies au sein d'un réseau d'entreprise et peuvent inclure des paramètres variables pour une personnalisation simple. En utilisant des requêtes, les données peuvent être facilement extraites de votre réseau blockchain. Les requêtes sont envoyées à l'aide de l'API Hyperledger Composer.

Events

Les événements sont définis dans la définition du réseau d'entreprise de la même manière que les actifs ou les participants. Une fois les événements définis, ils peuvent être émis par les fonctions du processeur de transaction pour indiquer aux systèmes externes que quelque chose d'important est arrivé au grand livre. Les applications peuvent s'abonner aux événements émis via l'API composer-client.

Access Control

Les Bussines network peuvent contenir un ensemble de règles de contrôle d'accès. Les règles de contrôle d'accès permettent un contrôle précis de ce que les participants ont accès à quels actifs du réseau d'entreprise et à quelles conditions. Le langage de contrôle d'accès est suffisamment riche

pour capturer des conditions sophistiquées de manière déclarative, telles que «seul le propriétaire d'un véhicule peut transférer la propriété du véhicule». L'externalisation du contrôle d'accès à partir de la logique fonctionnelle du processeur de transaction facilite l'inspection, le débogage, le développement et la maintenance.

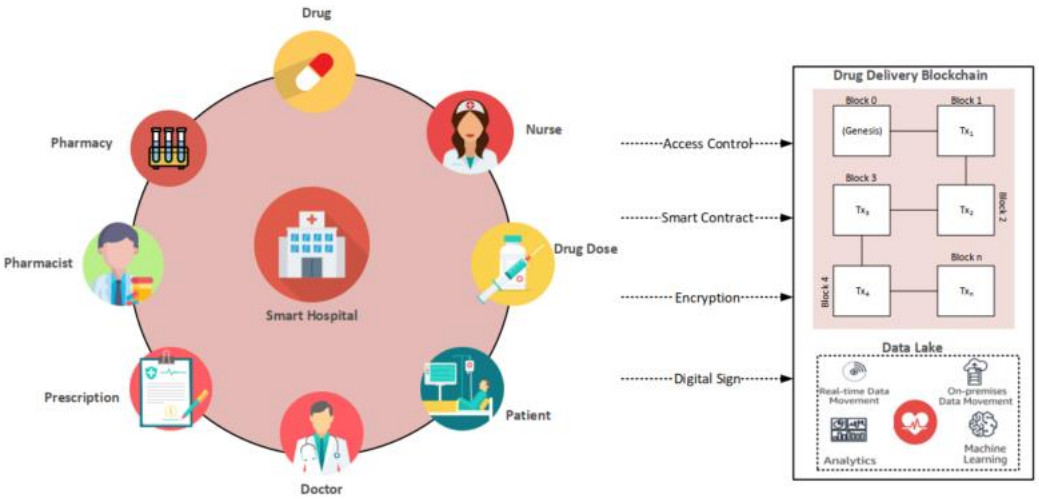
Historian registry

L'historien est un registre spécialisé qui enregistre les transactions réussies, y compris les participants et les identités qui les ont soumises. L'historien stocke les transactions en tant qu'actifs Historian Record, qui sont définis dans l'espace de noms système Hyperledger Composer.

4. Solution :

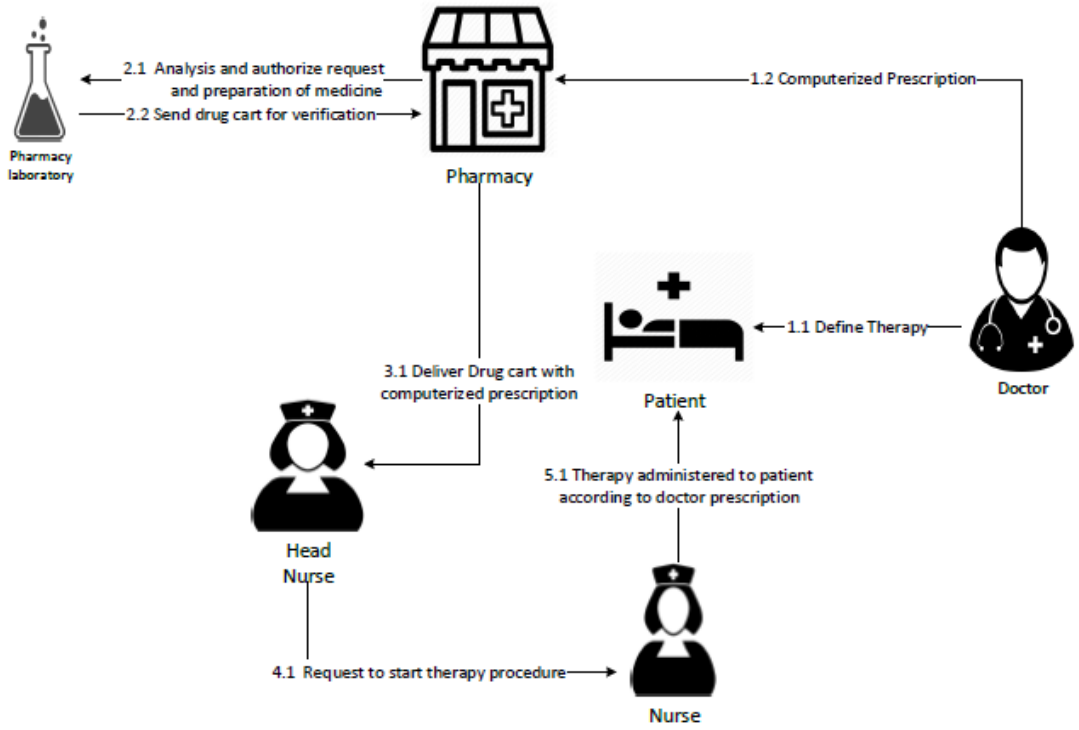
La blockchain offre une opportunité unique de fournir des avantages dans le secteur de la santé. Figure 1 présente une description graphique de l'architecture conceptuelle, où la blockchain médicale conserve un historique complet à jour de toutes les données médicales, couvrant le DME, les visites, les ordonnances, facturation et données IoT, qui suivraient un utilisateur individuel à vie.

Le Data Lake représente un outil qui sera développer dans les versions futures du projet, il est utilisé pour des tâches telles que l'analyse, visualisation et rapport de données médicales.



4.1. Scénario :

- Initialement le patient donne l'accès a son dossier médical au médecin.
- Puis, le médecin examine le patient et définit une thérapie, une dose de médicament et d'autres conseils sous la forme d'une prescription informatisée.
- Ensuite, l'ordonnance informatisée est envoyée au personnel de la pharmacie qui analyse l'authenticité de l'ordonnance et demande au pharmacien de préparer le chariot de médicaments.
- Le pharmacien envoie ensuite le chariot de médicaments au personnel de la pharmacie pour vérification croisée avec l'ordonnance informatisée.
- Le chariot de médicaments préparé avec la prescription informatisée est envoyé à l'infirmière en chef, qui vérifie et met à jour l'inventaire des médicaments de la salle et demande à l'infirmière de commencer la procédure de thérapie du patient.
- Enfin, l'infirmière administre la thérapie au patient conformément à la prescription du médecin.

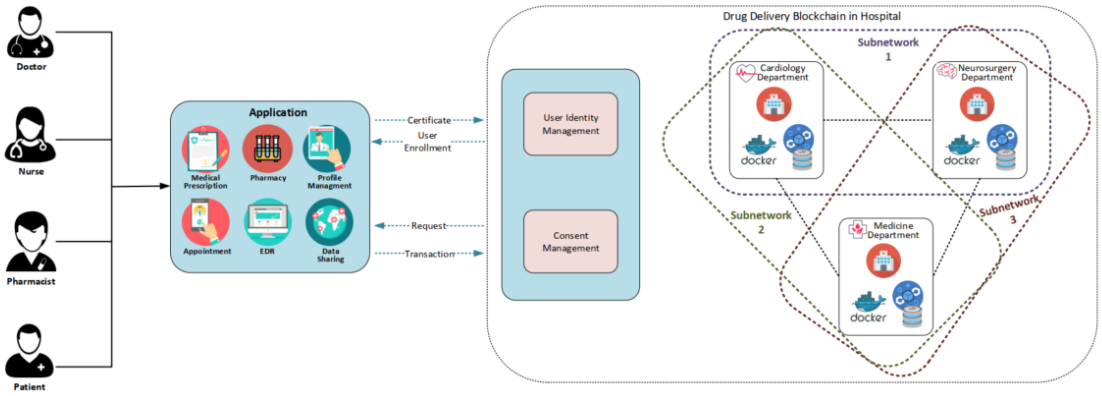


Scenario

4.2. Architecture de la supply chain :

- Dans le système proposé, la proposition de transaction est envoyée par l'utilisateur final (c'est-à-dire le médecin, l'infirmière, le pharmacien et le patient) via l'application pour appeler des services backend tels que prescription médicale, gestion de profil, rendez-vous, EDR (dossier électronique de médicament), données partage, gestion de pharmacie, etc., assuré par le réseau de blockchain proposé.
- Le système d'administration de médicaments proposé permet à chaque département de créer ses propres sous-réseaux dans le but de partager les données en toute sécurité.
- Seuls les participants autorisés peuvent participer et s'inscrire au réseau, pour cela ils passent par un gestionnaire d'identité « user identity manager »
- Le « user identity manager » fournit des certificats pour l'inscription et l'authentification des utilisateurs. De plus, ces services sont liés aux validations d'identité des utilisateurs, à la vérification et à la génération de signatures pour les utilisateurs individuels qui participent au réseau de la chaîne de blocs.

Gestionnaire de consentement représente le consensus qui est responsable de la connexion d'un département avec un sous-réseau via une interface fournie et également le contrôle de l'ordre de transaction. Chaque département d'une blockchain d'administration de médicaments contient des nœuds et un registre distribué



Architecture du system proposé

La responsabilité d'un nœud dans un réseau est de soutenir la réplique d'une blockchain, il est également responsable du traitement de la transaction. La figure 5 donne la vue intérieure d'un nœud blockchain. Le nœud dans la blockchain comprend un contrat intelligent, des blocs, une base de données d'état et des politiques. L'état La base de données est utilisée pour représenter et stocker l'état du grand livre à un moment et à un moment donné. L'échantillon La représentation en boîte blanche d'un nœud indique la relation entre la base de données d'état et le bloc.

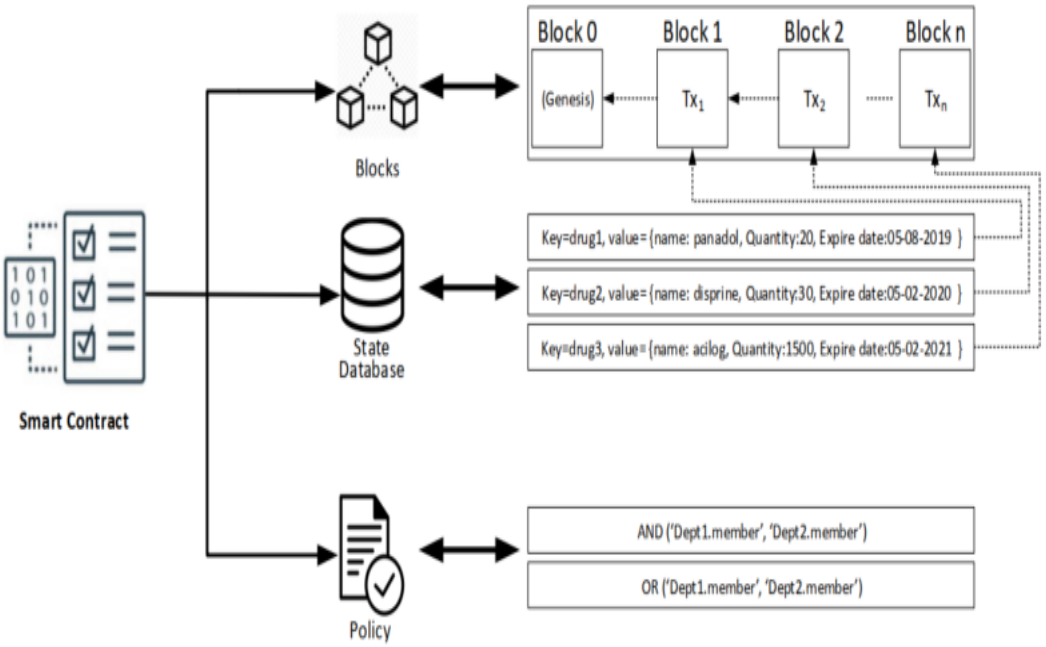


Figure 5

La figure 6 représente la liste des docker container qui hébergé diffèrent nœud dans la blockchain

```
omar@omar-VirtualBox:~/main/test$ docker ps
CONTAINER ID        IMAGE
    COMMAND                  CREATED             STATUS              PORTS                                                NAM
ES
631071de23f9        dev-peer0.org1.example.com-test-0.0.5-fc5aea9b0d4b41d40fe0eed0c61f894a527d42319fb73c2ccff445531e7c294
7   "/bin/sh -c 'cd /usr…"   2 days ago          Up 2 days                                                            dev
-peer0.org1.example.com-test-0.0.5
c59bc7f8b45f        hyperledger/fabric-peer:x86_64-1.1.0
    "peer node start"        2 days ago          Up 2 days           0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp   pee
r0.org1.example.com
df8cf6fae2f2        hyperledger/fabric-orderer:x86_64-1.1.0
    "orderer"                2 days ago          Up 2 days           0.0.0.0:7050->7050/tcp                           ord
erer.example.com
374c755b86cd        hyperledger/fabric-ca:x86_64-1.1.0
    "sh -c 'fabric-ca-se…"   2 days ago          Up 2 days           0.0.0.0:7054->7054/tcp                           ca.
org1.example.com
401e7c626875        hyperledger/fabric-couchdb:x86_64-0.4.6
    "tini -- /docker-ent…"   2 days ago          Up 2 days           4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp       cou
chdb
omar@omar-VirtualBox:~/main/test$
```

Figure 6

Pour configure le réseaux blockchain hyperléger nous met en disposition des fichiers de configuration en note :

Crypto-config.yaml : Ce fichier de configuration générera quelques certificats et clés pour l'organisation. Le binaire Cryptogen prendra le fichier de configuration comme entrée et créera un dossier crypto-config dans le répertoire de configuration, qui contiendra tous les certificats et clés générés.

```
# ---------------------------------------------------------------------------
# "OrdererOrgs" - Definition of organizations managing orderer nodes
# ---------------------------------------------------------------------------
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com
    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer
# ---------------------------------------------------------------------------
# "PeerOrgs" - Definition of organizations managing peer nodes
# ---------------------------------------------------------------------------
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Org1
  # ---------------------------------------------------------------------------
  - Name: Org1
    Domain: org1.example.com
    # ---------------------------------------------------------------------------
    # "Specs"
    # ---------------------------------------------------------------------------
    # Uncomment this section to enable the explicit definition of hosts in your
    # configuration.  Most users will want to use Template, below
    #
    # Specs is an array of Spec entries.  Each Spec entry consists of two fields:
    #   - Hostname:   (Required) The desired hostname, sans the domain.
    #   - CommonName: (Optional) Specifies the template or explicit override for
    #                 the CN.  By default, this is the template:
    #
    #                              "{{.Hostname}}.{{.Domain}}"
    #
    #               which obtains its values from the Spec.Hostname and
    #               Org.Domain, respectively.
    # ---------------------------------------------------------------------------
```

Crypto-config.yaml

Configtx.yaml : Ce fichier de configuration contiendra les détails complets d'un canal lié à une organisation.

```
################################################################################
#
#   SECTION: Orderer
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters
#
################################################################################
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: kafka

    Addresses:
        - orderer.example.com:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 10

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 98 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB
```

Configtx.yaml

Remarque : Kafka est définie comme le consensus de notre projet pour 2 raisons :

- Le projet est développé sur la version 1.2 de Hyperledger fabric cela nous donne la possibilité d'implémenter soit solo ou Kafka, raft n'est malheureusement que on version beta.
- Solo Ordrer est appelé ainsi car il exécute une seule instance du Orderer et dans ce cas il est trivial d'établir un ordre total sur les messages. Dans la pratique, dans un système de production, nous ne voulons pas d'un point de défaillance unique et souhaitons donc avoir plus d'un nœud de commande.

docker-compose.yaml : Ce fichier défini tous les docker container qui vont hébergé les noeud du réseau blockchain

```
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  image: hyperledger/fabric-peer:$ARCH-1.1.0
  environment:
    - CORE_LOGGING_LEVEL=debug
    - CORE_CHAINCODE_LOGGING_LEVEL=DEBUG
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=composer_default
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_CHAINCODE_STARTUPTIMEOUT=1200s
    - CORE_CHAINCODE_EXECUTETIMEOUT=800s
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/peer/msp
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
  command: peer node start
  ports:
    - 7051:7051
    - 7053:7053
  volumes:
      - /var/run/:/host/var/run/
      - ./:/etc/hyperledger/configtx
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/peer/msp
      - ./crypto-config/peerOrganizations/org1.example.com/users:/etc/hyperledger/msp/users
  depends_on:
    - orderer.example.com
```

Docker-composer.yaml

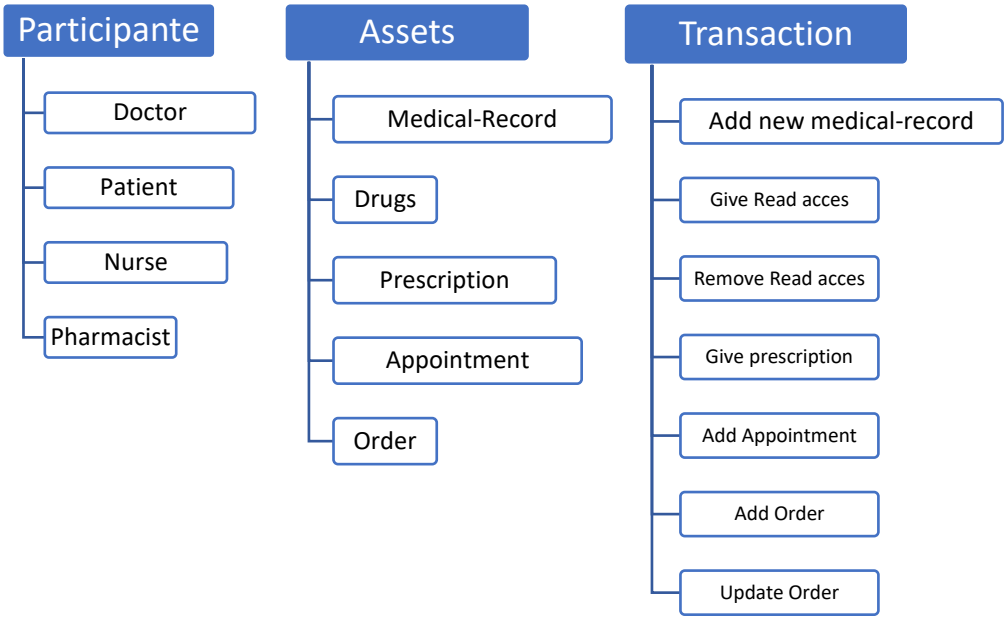
4.3. Implémentation de la solution :

4.3.1. Environnement de développement :

L'environnement de développement de l'étude du cas proposée est divisé en deux parties.
Le front-end et le back-end ont été développés dans des environnements séparés. La mise en œuvre back-end pour la gestion de la chaîne d'approvisionnement dans un environnement docker est décrit dans le tableau 1. Tous les travaux de mise en œuvre et expérimentaux de cette étude ont été réalisés sur Ubuntu Linux 18.04 LTS .
De plus, pour l'environnement d'exécution de docker, nous avons utilisé un moteur Docker (version 18.06.1-ce), et pour la configuration du conteneur et de l'image docker et dans la machine virtuelle, nous utilisé docker-compose (version 1.13.0). Nous avons utilisé un framework open-source Hyperledger Fabric (v1.2) projet hébergé par Linux Foundation. Python (v2.7.15) et Node (v8.11.4) sont des prérequis d'un réseau Fabric pour développer le SDK client. Nous avons utilisé le Composer web-playground pour concevoir et développer le définition du Business network et pour le déploiement, nous avons utilisé un outil Composer CLI pour la plateforme blockchain. Nous avons utilisé un serveur REST Composer pour créer l'API REST pour communique avec le service front end et visualiser la logique back-end sur une interface utilisateur graphique (GUI).
Le front-end a été développée en utilisant le framwork java script Vue js qui utilise HTML5, CSS3 pour le front end et java script pour la programmation dynamique. Afin de rendre l'application Web plus efficaces et conviviaux, nous avons utilisé un toolkits tiers comme Bootstrap. Le back-end et front-end interagissent les uns avec les autres à l'aide d'un serveur d'API REST. Le client effectue une action sur l'application Web qui déclenchera la méthode HTTP comme POST, GET, PUT et DELETE, qui dans réponse exécutée selon la requête HTTP du client.

4.3.2. Composants du network :

Dans Hyperledger Fabric, il existe trois composants principaux pour la création d'un réseau d'entreprise, c'est-à-dire les participants, les assets et les transactions. Le réseau d'entreprise proposé est visualisé sur la figure suivant



4.3.3. Smart Contract :

Le contrat intelligent dans Hyperledger comprend quatre composants, à savoir le modèle, la définition de requête, le script et les règles de contrôle d'accès.

Les transactions sont également définies lors de la modélisation des contrats intelligents, qui vise à interagir avec les actifs.

Le fichier de script contient une fonction de processus de transaction qui est déclenchée. Il contient des fonctions comme CREATE, DELETE, UPDATE, etc. pour modifier les valeurs des actifs et des participants au réseau de la blockchain. Les contrats intelligents sont écrits en JavaScript et confinés dans un fichier isolé en tant que partie d'une définition de contrat intelligent.

```
'use strict';
/**
 * Write your transction processor functions here
 */

/**
 * Add new Product
 * @param {org.lms.ehr.CreatePrescription} addprescription - new prescription addition
 * @transaction
 */
function addprescription(newprescription) {
  var  NS =  'org.lms.ehr';
  var prescription = getFactory().newResource(NS, 'Prescription', newprescription.recordId);
  prescription.comments = newprescription.comments;
  prescription.time = newprescription.time
  prescription.drug = newprescription.drug;
   prescription.pharma = newprescription.pharma;
    prescription.doctor = newprescription.doctor;
     prescription.patient = newprescription.patient;
     prescription.meds = newprescription.meds;

  if(!prescription.meds.prescription) {
    prescription.meds.prescription = [];
  }
  prescription.meds.prescription.push(prescription);
  return getAssetRegistry(NS + '.Prescription').then(function(registry) {
    return registry.add(prescription);
  }).then(function() {
    return getAssetRegistry(NS + '.MedicalRecords');
  }).then(function(medsRegistry) {
    return medsRegistry.update(newprescription.meds);
  });
}
```

Fichier logic.js (définir la logique des transaction)

```
namespace org.lms.ehr

participant Patient identified by patientId {
  o String patientId
  o String firstName
  o String lastName
  o String job optional
  o String Adresse optional

}
participant agent identified by agentId {
  o String agentId
  o String firstName
  o String lastName
}



participant Doctor identified by doctorrId {
  o String doctorrId
  o String firstName
  o String lastName
  o String registrationNumber optional
  o String Specialisation optional
  o String Adresse optional
}
```

Fichier .CTO pour décrire le model

4.3.4. Access Control Language :

(ACL) fournit un contrôle d'accès déclaratif sur les éléments du modèle de domaine. En définissant des règles ACL, nous pouvons déterminer quels utilisateurs / rôles sont autorisés à créer, lire, mettre à jour ou supprimer des éléments dans le modèle de domaine d'un réseau d'entreprise**.**

```
rule NetworkAdminUser {
    description: "Grant business network administrators full access to user resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "**"
    action: ALLOW
}


rule NetworkAdminSystem {
    description: "Grant business network administrators full access to system resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}
//patient
rule DoctorSeeThemselves {
  description: "Doctor can see their own participant"
  participant(t): "org.lms.ehr.Doctor"
  operation: ALL
  resource(v): "org.lms.ehr.Doctor"
  condition: (v.getIdentifier() == t.getIdentifier())
  action: ALLOW
}
rule DoctorSeePAtient {
  description: "Doctor see patient"
  participant(t): "org.lms.ehr.Doctor"
  operation: ALL
  resource: "org.lms.ehr.Patient"
  condition: (testOwnership(org.lms.ehr.MedicalRecords, t))
  action: ALLOW
}
```

Fichier .ACL (définir les permission)

4.4. Démonstration

4.4.1. Back-end

Cette section illustre l'exécution du système proposé et fournit quelques snapshots illustrant le processus d'exécution, la figure 13 montre la demande d'API détaillée et la réponse de différents modules du système proposé avec le serveur REST Hyperledger Composer.

Premièrement, le réseau autorise l'utilisateur en validant l'ID utilisateur, puis la requête sera initiée par le client au REST serveur afin de soumettre la transaction à la plate-forme blockchain proposée. Pour effectuer les transactions, les fonctions de contrat intelligent sont déclenchées par la plateforme blockchain, qui retourne une réponse au client après l'exécution réussie d'une transaction.

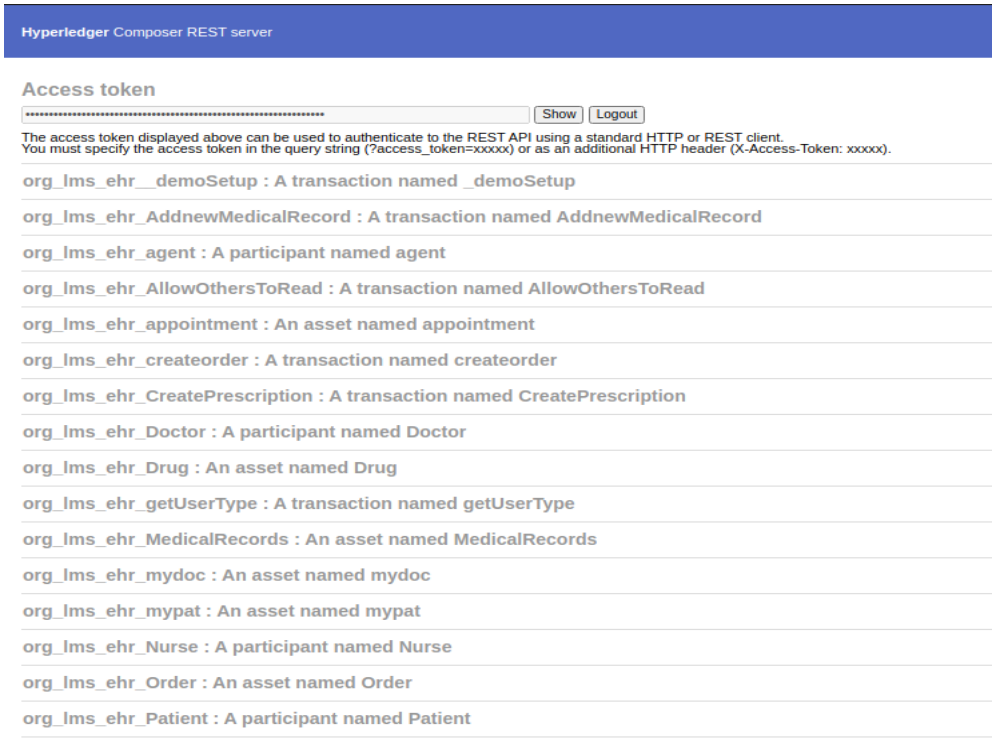


Figure 13

La figure suivante montre une réponse d'une requête GET initialise par l'utilisateur

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:3000/api/org.lms.ehr.MedicalRecords'
```

Request URL

```
http://localhost:3000/api/org.lms.ehr.MedicalRecords
```

Response Body


```
[
  {
    "$class": "org.lms.ehr.MedicalRecords",
    "record_id": "21bd300ef64ae6539005f7e2f9dc9d9838b7a6cd1caad4fb2ee6df1a9d0ca8c4",
    "patient": "medj",
    "doctor": "string",
    "authorized": [],
    "description": "string",
    "prescription": [],
    "app": [],
    "encounter_time": "2020-09-07T17:29:43.134Z"
  }
]
```


Response Code

```
200
```

Response Headers


```
{
  "access-control-allow-credentials": "true",
  "content-length": "265",
  "content-type": "application/json; charset=utf-8",
  "date": "Mon, 07 Sep 2020 17:34:27 GMT",
  "etag": "W/\"109-oDvyZdAHR2JvKaOfYE9KzlUlHkg\"",
  "vary": "Origin",
  "x-content-type-options": "nosniff",
  "x-download-options": "noopen",
  "x-frame-options": "DENY",
  "x-xss-protection": "1; mode=block"
}
```


Grace a l'outil Composer-Playground on peut visualiser les transactions effectué dans le réseau blockchain :

Date, Time	Entry Type	Participant	
2020-08-20, 06:40:56	ActivateCurrentIdentity	none	view record
2020-08-20, 06:40:44	AllowOthersToRead	pat1 (Patient)	view record
2020-08-20, 06:35:12	AddAsset	admin (NetworkAdmin)	view record
2020-08-20, 06:34:41	ActivateCurrentIdentity	none	view record
2020-08-20, 06:34:38	IssueIdentity	admin (NetworkAdmin)	view record
2020-08-20, 06:34:28	IssueIdentity	admin (NetworkAdmin)	view record
2020-08-20, 06:12:49	AddnewMedicalRecord	admin (NetworkAdmin)	view record
2020-08-04, 07:38:51	AddnewMedicalRecord	admin (NetworkAdmin)	view record

Liste de transaction

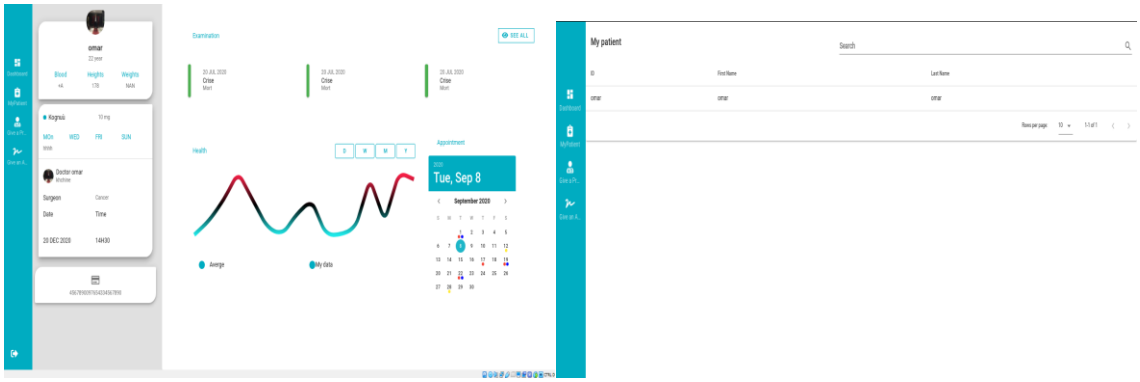
4.4.2. Front-end

Le front-end a été développe grâce au Framework Vue js chaque participant aura accès au Dashboard

Les différèrent participant auront la possibilité de visualise différent information, par exemple le médecin peut initier une prescription pour un malade et voir son dossier médical s'il a accès.

Aussi le patient peut visualiser ses informations personnelles et les rende vous qui là à l'hôpital, il peut initier des transactions pour donner l'accès a son dossier médical au médecine.

La figure suivante montre l'interface graphique :



5. Conclusion :

La blockchain a montré sa capacité à transformer l'industrie traditionnelle de la chaîne d'approvisionnement en une chaîne d'approvisionnement sécurisée, automatisée, anonyme, persistante, audible et décentralisée. Dans ce projet, nous avons décrit la conception, la mise en œuvre de l'intégrité de la chaîne d'approvisionnement dans un hôpital intelligent basé sur Hyperledger Fabric. Le système proposé est une application de proof-of-concept qui assure le suivi des dossiers médicaux individuels à l'aide de la technologie blockchain de manière décentralisée. Il permet aux médecins, aux infirmières, aux patients et aux pharmaciens de gérer, d'accéder, et partager des dossiers médicaux personnels ainsi qu'un cycle de vie complet des médicaments.

Afin d'atteindre la transparence, la sécurité, et confidentialité du système proposé, nous avons utilisé un contrat intelligent. De plus, nous avons également conçu une application web qui interagit avec la plateforme blockchain pour exposer les services au front-end.

L'orientation future potentielle pourrait être d'augmenter la taille du réseau, puis un déploiement du système dans un environnement réel pour vérifier les performances et la faisabilité.