الجـــــــــــــمهـوريـــــة الجـــــــــــــزائـريـة الديمـــــــقـراطيـة الشـــــــــعبيـة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــــــعـليـم العـــــــالي والبـــــــــــحث العـــــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــــــامعة أبي بكـــر بـلـقـايد – تـلمســـــــــــــــان –

Université Aboubakr Belkaïd – Tlemcen –

Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du **diplôme** de **MASTER**

En : Télécommunications

Spécialité : Réseaux et Télécommunications

Par : LARABI Sofiane

KHIAT Nesrine

Thème

Conception et implémentation d'un système basé sur la blockchain pour le secteur médical

Soutenu le 29/08/2020 devant le jury composé de :

Mr. MERZOUGUI Rachid	PR	Univ. Tlemcen	Président
Mr. ZERROUKI Hadj	MCB	Univ. Tlemcen	Examinateur
Mr. HADJILA Mourad	MCA	Univ. Tlemcen	Directeur de mémoire
Mr. GHAFFOUR Ayoub	Ing. Rech.	Algérie Télécom.	Co-Directeur de mémoire

Année Universitaire 2019-2020

Remerciements

Tout d'abord, nous tenons à remercier Allah, le clément et le miséricordieux de nous avoir donné la force et le courage de mener à bien ce travail.

Nous voudrions exprimer nos vifs remerciements à notre encadreur ***Dr.Hadjila Mourad*** *et* ***Mr.Ghafour Ayoub****, pour ses orientations, ses encouragements, sa disponibilité et ses précieux conseils qui nous ont permis de mener à bien ce travail.*

Nous tenons à remercier les membres du jury qui ont bien voulu accepter de juger notre modeste travail.

Nos remerciements s'adressent également à tous les enseignants du département de Télécommunication ayant participé d'une manière ou d'une autre à notre formation master et licence.

A la fin nos remerciements les plus sincères à toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année Universitaire.

Dédicaces

Ce modeste travail est dédié à :

A celui qui a été toujours la source d'inscription, de courage a tout le long de mes études...mon Père qu'Allah paix à son âme.

A celle qui a été toujours la source de grande affection...ma Mère qu'Allah la garde et prolonge sa vie toujours.

A toutes ma petite famille, mes proches et mes amis sons exception.

A toute la promotion Télécommunication 2019/2020 que je leurs souhaite un bon avenir.

A tous ceux qui m'ont aidé de près ou de loin à la réalisation de ce modeste travail.

LARABI SOUFIANE

Dédicaces

Je dédie ce modeste travail

À mes très chers parents, mes estimes pour eux sont immenses, Pour leurs soutiens tout au long mes études. Je vous remercie pour tout ce que vous avez fait pour moi, et sur tout à ma mère qui représente la source de tendresse et l'exemple de dévouement qui n'a pas cessé de m'encourager. Qu'Allah vous bénisse avec une longue vie heureuse.

À ma chère sœur, source de joie et de bonheur

À une personne qui est chère pour moi je le remercie pour leurs sacrifices, leur soutien moral qui m'a permis de réussir mes études. Ce travail soit témoignage de ma reconnaissance et de mon amour sincère et fidèle.

À mes chères amies qui m'ont soutenues, et avec qui j'ai passé des meilleurs moments de ma vie et gardé de très bon souvenirs.

A mon binôme et toute sa famille.

A toute la promotion Télécommunication 2019/2020 que je leurs souhaite un bon avenir

Et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci.

KHIAT NESRINE

Résumé

La technologie Blockchain a le potentiel de transformer le secteur de la santé, en plaçant le patient au centre de l'écosystème de la santé et en augmentant la sécurité, la confidentialité et l'interopérabilité des données. Cette technologie pourrait fournir un nouveau modèle pour les échanges d'informations sur la santé en rendant les dossiers médicaux électroniques plus efficaces et sécurisés. Bien qu'il ne s'agisse pas d'une panacée, ce nouveau domaine en évolution rapide offre un terrain fertile pour l'expérimentation, l'investissement et les tests. Étant donné que les antécédents médicaux d'un patient sont la première pierre angulaire d'une bonne médecine, la recherche dans ce domaine est relativement nouvelle mais en croissance rapide. Ainsi, les chercheurs et praticiens en informatique de la santé ont toujours du mal à suivre le rythme des progrès de la recherche dans ce domaine. Dans ce modeste travail nous appliquerons le concept de la technologie blockchain pour implémenter un système de stockage et partage de DES du patient. L'objectif de notre projet de fin d'études est de construire un réseau blockchain décentralisée, synchronisée et fonctionne sans organe central de contrôle grâce aux mécanismes de consensus bien défini et créons une API qui nous permettra d'interagir avec notre blockchain. Ce travail a été développé au sein de l'entreprise Algérie Télecom Division commercial, Communication et Marketing, Unité de Recherche et Développement.

Mots clés : Blockchain, réseau, API, sécurité, domaine médicale, javascript.

Abstract

Blockchain technology has the potential to transform the healthcare industry, placing the patient at the center of the healthcare ecosystem and increasing data security, privacy and interoperability. This technology could provide a new model for the exchange of health information by making electronic medical records more efficient and secure. While not a panacea, this new and rapidly evolving field provides fertile ground for experimentation, investment and testing. Since a patient's medical history is the first cornerstone of good medicine, research in this area is relatively new but growing rapidly. Thus, health informatics researchers and practitioners continue to struggle to keep pace with advances in research in this domain. In this modest work we will apply the concept of blockchain technology to implement a patient DES storage and sharing system. The goal of our graduation project is to build a decentralized, synchronized and functioning blockchain network without a central control body thanks to well-defined consensus mechanisms and create an API that will allow us to interact with our blockchain. This work was developed within the company Algérie Télecom

Commercial, Communication and Marketing Division Research and Development Unit.
Keywords: Blockchain, network, API, security, medical field, javascript.

ملخص

تقنية البلوكتشين لديها القدرة على تحويل قطاع الرعاية الصحية، ووضع المريض في مركز النظام البيئي للرعاية الصحية وزيادة أمن البيانات والخصوصية والتشغيل البيني. يمكن أن توفر هذه التكنولوجيا نموذجًا جديدًا لتبادل المعلومات الصحية من خلال جعل السجلات الطبية الإلكترونية أكثر كفاءة وأمانًا. على الرغم من أنه ليس حلا سحريا، إلا أن هذا المجال الجديد سريع التطور يوفر أرضًا خصبة للتجربة والاستثمار والاختبار. نظرًا لأن التاريخ الطبي للمريض هو حجر الزاوية الأول للطب الجيد، فإن البحث في هذا المجال جديد نسبيًا ولكنه ينمو بسرعة. وبالتالي، يواصل الباحثون والممارسون في مجال المعلوماتية الصحية النضال من أجل مواكبة التقدم في البحث في هذا المجال.

في هذا العمل المتواضع، سنطبق مفهوم تقنية البلوكتشين لإنشاء نظام تخزين ومشاركة السجلات الخاصة بالمريض. الهدف من مشروع التخرج الخاص بنا هو بناء شبكة بلوكتشين لامركزية ومتزامنة وعاملة بدون هيئة تحكم مركزية بفضل آليات الإجماع المحددة جيدًا وإنشاء واجهة برمجة تطبيقات تتيح لنا التفاعل مع البلوكتشين الخاصة بنا. تم تطوير هذا العمل في شركة اتصالات الجزائر، قسم التجارة والاتصالات والتسويق، وحدة البحث والتطوير.

Table des matières

Introduction générale

Chapitre 1 : Généralités sur la technologie blockchain

Chapitre 2 : L'intérêt de la blockchain dans le monde médical

Chapitre 3 : Conception et réalisation

Chapitre 1

Chapitre 2

Chapitre 3

Liste des tableaux

Introduction générale

Introduction générale

Le domaine médical et un domaine clinique à forte intensité de données dans lequel une énorme quantité de données est générée, consultée et diffusée régulièrement. Le stockage et la diffusion de cette quantité de données sont cruciaux, mais aussi très difficiles, en raison de leur nature sensible et de facteurs limitatifs, tels que la sécurité et la confidentialité.

La pratique de partage de données est tout à fait essentielle pour permettre aux médecins et aux prestataires de soin de santé de transférer et partager les dossiers médicaux de leurs patients afin de garantir les informations complètes et à jour sur l'état de santé de chaque patient, par conséquent un diagnostic et suivi rapide.

De nos jours, plusieurs établissements sanitaires stockent ses informations de manière traditionnelle c'est-à-dire en archive physique (en papiers), et d'autre dans des serveurs. Toutes les deux méthodes ont un élément en commun : il s'agit de méthodes de stockage centralisées. En effet, toutes les données sont conservées au même endroit et sont contrôlées par un nombre restreint d'individus. Ce caractère centralisé du support de stockage permet de les cibler plus facilement, ce qui compromet les données qu'ils contiennent.

La technologie blockchain a un intérêt remarquable pour le domaine médical pour sécuriser le partage et le stockage des données critiques, ainsi d'améliorer et de réaliser les interconnections entre différents prestataires de santé.

D'abord, cette technologie suit un réseau peer-to-peer décentralisée qui néglige tout intermédiaire ou un tiers de confiance. De plus, elle est une base de données distribuée entre tous les nœuds du réseau afin que chaque nœud dispose une copie de toute la blockchain. Et elle permet également de tracer l'origine et le devenir de chacune des données enregistrées.

Ce mémoire est organisé en trois chapitres :

Le premier chapitre définit la technologie blockchain de manière générale, cite ses caractéristiques et son principe de fonctionnement. Cette première partie a pour but d'acquérir une culture générale autour de la blockchain pour mieux la comprendre.

Dans le deuxième chapitre, nous serons exposés des cas pratiques, où la blockchain peut faire une différence en apportant de la valeur au système de santé et ses usagers. Pour cela, nous examinerons des cas pratiques, pour lesquels nous rappellerons les enjeux, et enfin nous verrons l'apport mais aussi les limites que peut avoir la blockchain.

Dans le troisième chapitre, on parle d'outils de développement logiciel, on parle aussi des fonctions et des méthodes utilisées afin de construire notre propre application blockchain décentralisée, synchronisée et créerons une API qui nous permettra d'interagir avec

Chapitre 1

Généralités sur la technologie blockchain

1.1.Introduction

Le 21ème siècle est une question de technologie, avec le besoin croissant de modernisation dans notre vie de tous les jours, les gens sont prêts à accepter des nouvelles technologies et surtout des technologies révolutionnaires, au même niveau que ce que représentait à l'époque l'apparition d'internet pour mieux vivre lorsqu'elle s'intègre dans une approche sociale large et durable.

La blockchain est la technologie révolutionnaire impactant miraculeusement différentes industries. Elle a été introduite sur les marchés avec sa toute première application moderne Bitcoin. On parle parfois aussi de Ethereum et d'autres devises virtuelles ou jetons numériques, et parfois de contrats intelligents. Mais, la plupart du temps, il s'agit de registres distribués c'est-à-dire d'une liste de transactions qui est répliquée sur plusieurs ordinateurs, plutôt que d'être stockée sur un serveur central**.**

1.2.Historique

En 1991, les deux chercheurs scientifiques « Stuart Haber » et « W. Scott Stornetta » avaient une idée de mettre en œuvre un système où les documents numériques sont horodatés pour ne pas être falsifiés ou altérés. Leur système utilisait une blockchain sécurisée cryptographique pour stocker des documents horodatés [1].

Par la suite, en 1992, un protocole dit « arbre de Merkle » fut introduit au fonctionnement du système, ce qui le rend plus efficace en permettant à plusieurs documents d'être rassemblés en un seul bloc [1].

Cette technologie tomba des années dans l'oubli jusqu'à l'arrivée de l'informaticien et activiste cryptographique « Harold Thomas Finney » en 2004 (quatre ans avant la création du Bitcoin) qui a lancé le système RPoW « Reusable Proof Of Work » pour « Preuve de travail réutilisable »). Ce dernier fonctionnait en recevant un jeton preuve du travail non échangeable et non fongible basé sur le système Hashcash, celui-ci créait en retour un jeton possédant une signature RSA qui pouvait ensuite être transféré de personne en personne [1].

Fin 2008, une personne connue sous le pseudonyme de « Satoshi Nakamoto » avait publié un livre blanc (white paper) distribué par le biais d'une liste de diffusion e-mail en rapport avec la

cryptographie. Ce livre introduit un système de paiement électronique décentralisé de pair à pair (peer-to-peer), appelé Bitcoin [1].

Le 3 Janvier 2009, le Bitcoin est né quand le premier bloc de Bitcoin est miné par Satoshi Nakamoto comme la première unité de la crypto-monnaie. Le bloc offrait une récompense de 50 Bitcoins. Le premier destinataire de Bitcoin fut Hal Finney, qui reçut 10 Bitcoins de la part de Satoshi Nakamoto dans la première transaction mondiale de Bitcoins, le 12 Janvier 2009 [1].

En 2013, Vitalik Buterin, un programmeur et co-fondateur du Bitcoin Magazine lança le développement d'une nouvelle plate-forme informatique distribuée et basée sur la blockchain : l'Ethereum, dotée d'une fonctionnalité de script appelée « smart contracts » (des contrats intelligents en français) [1].

1.3.Qu'est-ce qu'une blockchain ?

1.3.1. Définition générale

« L'idée d'un grand cahier informatique, partagé, infalsifiable et indestructible du fait même de sa conception est au cœur d'une nouvelle révolution, celle de la blockchain. »

- Jean-Paul Delahaye, chercheur au centre de recherche en informatique, signal et automatique de Lille –

La blockchain, mot anglais, se définit comme une chaîne de blocs de données bien ordonnées, c'est une nouvelle technologie de stockage et de transmission, techniquement une nouvelle génération de base de données distribuée, sécurisée, transparente et fonctionne sans organe centrale, s'appuyant et tirant pleinement profit d'Internet, du protocole libre, de la puissance de calcul et de la cryptographie [2].

C'est avant tout une technologie basée sur le système de transfert pair à pair décentralisé, autrement dit sans intermédiaire, sans tiers de confiance ou organe central de contrôle (établissement bancaire, organisation gouvernementale, etc.) [3].

Cette nouvelle technologie semblable à un grand registre numérique public partagé qui contient l'enregistrement des données, des informations et des transactions effectuées par les

utilisateurs, regroupées dans des blocs reliés entre eux de manière irréversible à l'aide d'un processus cryptographique. Chaque bloc est validé par tous les membres de réseau (les nœuds), une fois validé, il sera horodaté et intégré à la chaîne de blocs. Chaque membre du réseau dispose donc d'une copie de la blockchain enregistrée en local et la met à jour à chaque fois qu'un bloc est ajouté [3].

Cette chaine est visible et accessible par tous les nœuds du réseau, ce qui donne un haut niveau de fiabilité. Donc, il est impossible de modifier ni supprimer un bloc existant ou changer leur ordre d'ajout [3].

1.3.2. Définition et structure d'un bloc

Un bloc est un enregistrement (petite base de données) dans la blockchain qui contient des données et/ou l'historique des transactions effectuées. La taille du bloc peut varier (pour le Bitcoin, la taille d'un bloc est de 1Mo, alors que, pour le Bitcoin Cash elle est de 8Mo).

Chaque bloc contient :

- **Indice de bloc :** numéro du bloc par rapport à leur emplacement dans la blockchain.
- **L'horodatage :** ou timestamping en anglais, est une valeur représentant la date et l'heure de création d'un bloc.
- **Tx** : la partie centrale qui contient une liste de transactions (les données).
- **Block hash** : Une empreinte digitale unique pour toutes les données du bloc. Lorsque les données d'un bloc changent, le hachage change également.
- **Previous hash** : c'est un champ qui contient le Hash de bloc précédant, cela permet de relier chaque bloc avec celui qui le précède et forme une chaine de blocs, sans qu'il soit possible de changer leur ordre d'ajout**.** Le premier bloc d'une blockchain est appelé "Genesis Block" et est le seul qui ne contient pas les données des blocs précédents.
- **Nonce** : Un nonce c'est un nombre aléatoire ou pseudo-aléatoire utilisé une seule fois qui est ajouté à un bloc par les mineurs pour garantir que les anciennes données ne peuvent pas être réutilisées dans des attaques par rejeu. Ce qui fait même si on a toutes les données enregistrées dans le bloc, on n'obtient pas le même Hash [4].

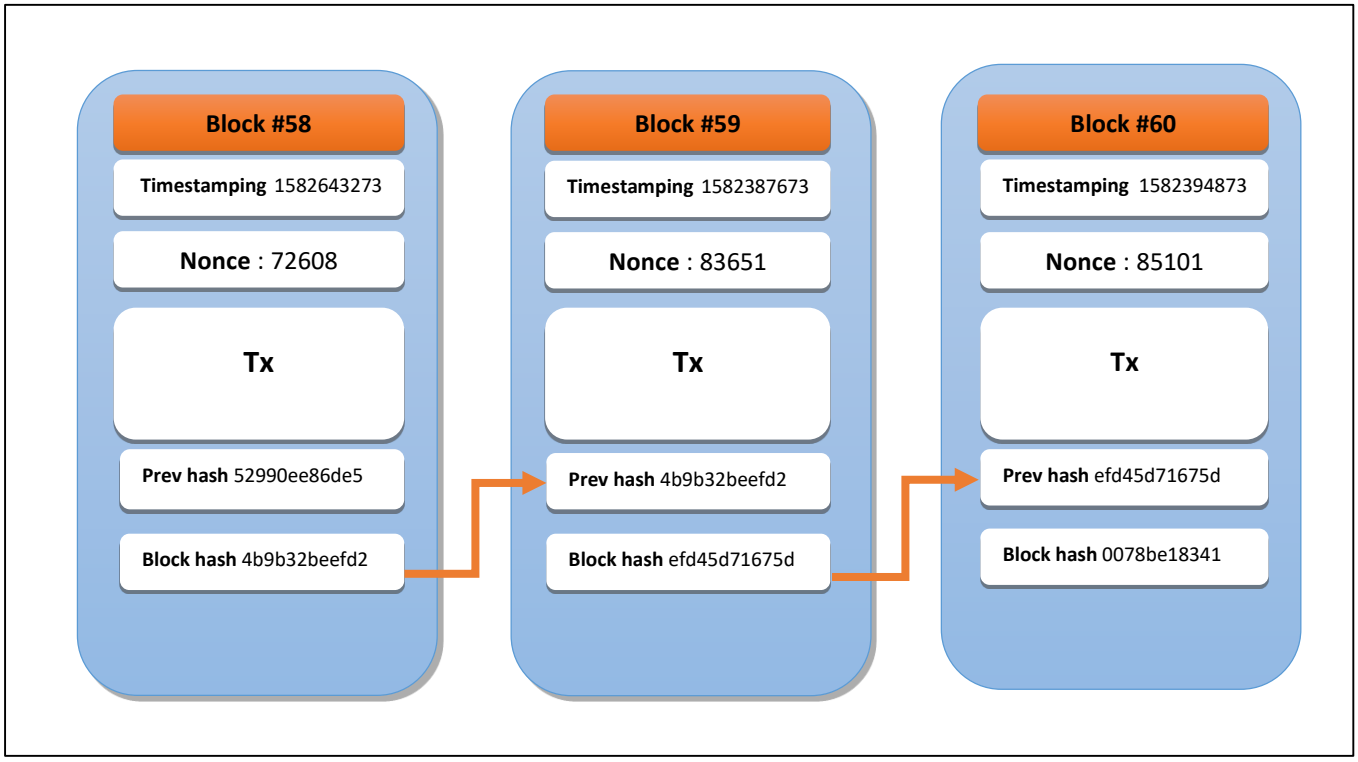

Figure 1.1 : Exemple d'une blockchain contenant trois blocs.

1.3.3. Types de la Blochchain

On peut distinguer trois types de blockchain fondamentaux.

1.3.3.1. La blockchain publique

La blockchain publique est un réseau décentralisé qui consiste à faire des échanges entre les nœuds du réseau sans intermédiaire (pair-à-pair) grâce à une relation de confiance. De ce fait, une blockchain publique est accessible par n'importe qui dans le monde grâce à leur code source public. Chacun peut effectuer des transactions et/ou en vérifier et chacun peut lire le registre et en obtenir une copie, enfin chacun peut participer librement au processus d'approbation (celui qui permet de décider quel bloc sera ajouté à la chaîne et qui définit l'état du système) [5].

Dans ce type de blockchain, les acteurs sont ceux qui vérifient les transactions et valident les blocs. Les mineurs garantissent la sécurité, la fiabilité et la mise à jour continuelle des données.

Les transactions dans la blockchain publique entre les acteurs sont pseudonymes, Il est donc possible de retrouver l'identité d'une personne via son adresse publique (connue par tout le réseau et indispensable pour effectuer une transaction) [6].

Deux innovations reposent sur ce type de blockchain, le Bitcoin et l'Ethereum et d'autres existent également, de moindre ampleur comme : Litecoin, Dogecoin, etc.

1.3.3.2. La blockchain privée

Les blockchain privées sont totalement centralisées, parfois appelées « permissioned ». En effet, elles fonctionnent comme un réseau privé (un peu comme un intranet) dirigé par un organe central appelé le gérant. Ce dernier peut modifier, valider en particulier les droits d'accès et décider les règles de fonctionnement de la blockchain comme il le souhaite et a pour mission d'ajouter les blocs à la chaîne. Le protocole peut donc être modifié selon le bon vouloir de l'administrateur du système (gérant) [6].

Dans ce système, il n'existe pas des liens entre les différents participants. Bien que tout le monde peut y participer, l'accès à cette blockchain est restreint et nécessite l'approbation du gérant et donc d'être « autorisé ». Les autres participants peuvent également refuser ou non un accès suivant les mécanismes de contrôle utilisés [6].

Contrairement au blockchain publique, l'existence d'une crypto-monnaie n'est pas nécessaire pour les blockchains privées. Ce type est principalement utilisé par les entreprises comme les banques qui souhaitent utiliser la technologie blockchain mais restent frileuses quant à l'utilisation d'une blockchain publique. Donc, elles ont besoin d'un tiers de confiance pour garder leurs transactions privées, en garantissant une sécurité intérieure et une confidentialité élevée [6].

1.3.3.3. La blockchain de consortium

La blockchain de consortium est un peu différente que les deux types précédents. En fait, elle se situe sur la limite entre chaîne publique et chaîne privée.

Dans ce type, il existe une double modification au système originel. La première c'est que les participants au processus d'approbation (validateurs) sont limités contrairement au cas d'une blockchain publique [5].

En outre, ce n'est plus la règle de la majorité qui s'impose pour lire la blockchain, c'est-à-dire l'accès au registre s'effectue soit par tous les utilisateurs soit réservé à certain participants du réseau [4].

Les transactions dans la blockchain de consortium entre les acteurs ne sont pas pseudonymes contrairement au blockchain publique. Les acteurs principaux connaissent l'identité des utilisateurs [6].

1.3.3.4. La comparaison entre les trois systèmes

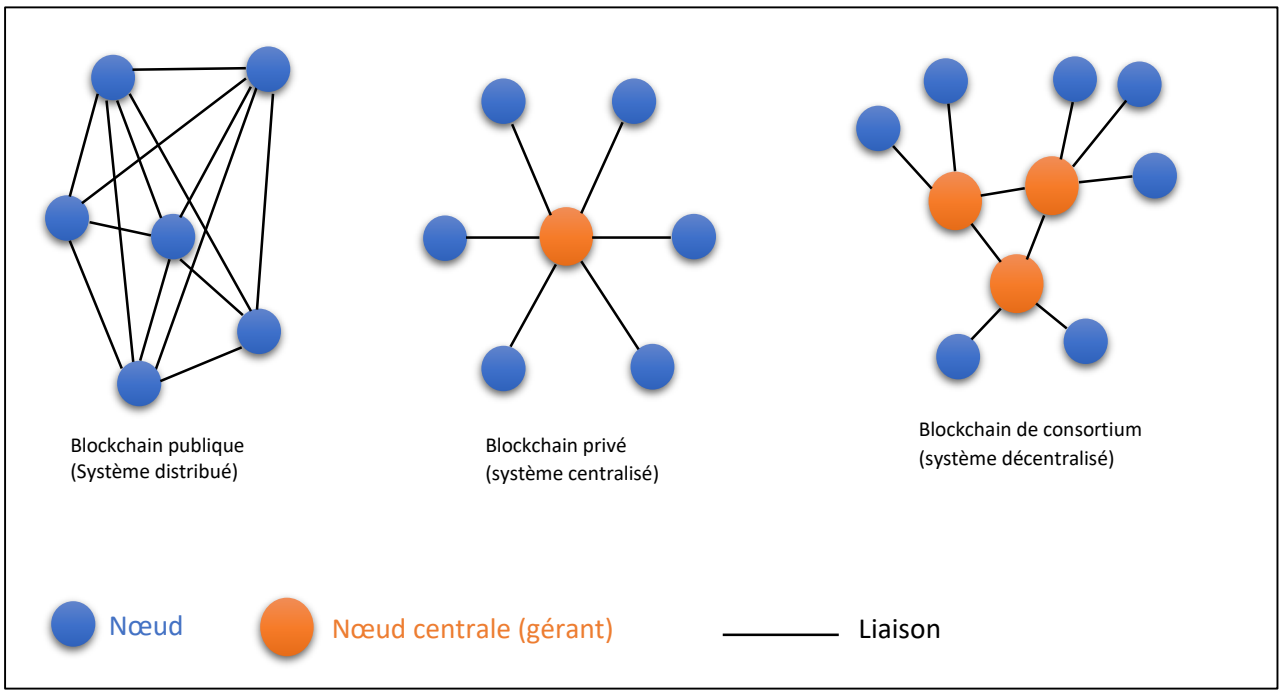

Figure 1.2 : Distribution des nœuds dans les trois types de la blockchain.

Les trois types de la blockchain publique, privée et à consortium se sont des technologies différentes et chacun se spécialisant sur les applications les plus adéquates. La blockchain publique est idéale pour les transactions pair à pair, tandis que les blockchains privée et à consortium sont davantage adaptées par exemple à l'ensemble des activités commerciales nouées entre deux entreprises**.**

- Une blockchain publique est bien résistante à la censure et elle représente une solution de confiance décentralisée pour les transactions et les données.
- Une blockchain privée est caractérisée par la rapidité du système, et une garantie de confidentialité et de confiance, puisqu'elle est idéalement déployée dans des situations

où une personne ou une organisation doit rester au contrôle et où les informations doivent rester privées.

- Une blockchain à consortium réduit certains risques par rapport à la blockchain privée, (en cas ou le contrôleur centralisé est supprimé), et leur nombre de nœuds limité donne des performances généralement beaucoup plus efficaces que celles d'une blockchain publique. Les consortiums sont susceptibles de séduire les organisations qui veulent rationaliser la communication entre elles [7].

	Les types de la Blockchain		
	Publique	Privé	à consortium
Sans permission ?	Oui	Non	Non
Qui peut la consulter ?	Tout le monde	Seulement les utilisateurs invités	Cela vrais
Qui peut y écrire ?	Tout le monde	Des participants approuvés	Des participants approuvés
Propriété	Personne	Entité unique	Plusieurs entités
Participants connus ?	Non	Oui	Oui
Vitesse de transaction	Lente	Rapide	Rapide

Tableau 1.1 : Différences majeures entre les trois types de la blockchain.

1.3.4. Caractéristiques principales

Plusieurs caractéristiques sont associées à la blockchain : désintermédiation, traçabilité, transparence, consensus distribué, ineffaçable, structure distribuée, résilience, sécurité et confiance. Toutes ces caractéristiques constituent le potentiel innovateur de la blockchain.

1.3.4.1. La désintermédiation

Le réseau blockchain fonctionne en pair à pair, c'est-à-dire échanger des informations sans un tiers de confiance de sorte que personne n'est au-dessus des lois. De plus, ce système qui permet de supprimer les intermédiaires va également réduire les coûts de transfert [8].

La validation et l'ajout d'un bloc résultent d'un consensus entre les utilisateurs (validateurs), ce qui fait que personne ne peut effacer ou falsifier les informations dans la blockchain grâce à la duplication. Chaque participant possède une copie de la blockchain.

La technique de distribution permet au réseau d'être difficilement interruptible, c'est-à-dire lorsqu'un nœud est devenu défaillant, le réseau continuera à fonctionner normalement sans aucun problème parce que tous les autres nœuds resteront liés entre eux [8].

1.3.4.2. La transparence

La blockchain est transparente, car tout le monde vérifie son état et son honnêteté à tout moment. Tous les utilisateurs ont un exemplaire local de toute la blockchain et peuvent la mettre à jour à chaque fois un nouveau bloc est inséré ou modifié, c'est-à-dire ils peuvent voir les transactions présentes et passées ce qui permet de régler d'éventuels conflits. Par exemple, si on prétend avoir envoyé de l'argent sans le faire en réalité, tout le monde peut voir que le destinataire n'a rien reçu [9].

Le caractère de la transparence a pour objectif d'assurer les transactions, mais l'anonymisation des utilisateurs peut casser ce caractère. En effet, l'anonymat sur la blockchain peut être utilisé pour des activités frauduleuses, difficiles voire impossibles à détecter et à réguler [9].

1.3.4.3. La sécurité

La sécurité est l'une des atouts prometteurs de la blockchain qui lui permet d'être plus résistante contre les cyber-attaques ou au contrôle de l'État. Elle peut être considérée comme triplement sécurisée.

Premièrement, cette technologie est basée sur un système cryptographique moderne dit « asymétrique », employant deux clefs, la première dite « publique » utilisée pour le déchiffrement (transmissible sans restriction) et la seconde dite « privée » utilisée pour le chiffrement (jamais transmise à personne). On peut comparer ce couple de clefs à celui de RIB/PIN dans le monde bancaire. La clé publique est l'équivalent du RIB (l'adresse publique du compte d'un utilisateur donné). Cette clé n'a pas d'autre fonction que la réception des paiements. Le code PIN bancaire est représenté dans ce cas-là comme une clé privée qui permet de signer les transactions numériquement. Ainsi, personne ne peut signer les transactions au nom d'un autre individu. Cela signifie que chacun a son propre clé privée « PIN » et que cette dernière ne se perde pas et ne soit pas révélée [5].

Deuxièmement, le calcul du hash du bloc qui a pour but de convertir toutes les données d'un bloc plus le hash du bloc précédant en une suite pseudo-aléatoire de chiffres fixes quel que soit la longueur des données en entrée basée sur la fonction de hachage. Cette fonction est dite aussi à sens unique car il est impossible de retrouver l'entrée de la fonction (l'information contenue dans le bloc) à partir de son hash, et il est impossible de produire la même valeur de hachage pour des entrées différentes, ce qui fait que la moindre modification d'une entrée modifie complètement le hachage. Ce point est essentiel car il assure l'intégrité des données [10]. Maintenant, si quelqu'un veut modifier ou falsifier un bloc, il ne doit pas seulement modifier ce bloc et trouver leur hach, mais, il doit modifier tous les blocs qui ont été ajoutés après lui, car chaque bloc contient le hach du bloc précédant ce qui rend cette opération quasiment impossible.

Troisièmement, L'hébergement décentralisé est le second mécanisme permettant à la blockchain d'être une technologie sûre. Ce mécanisme demande la participation d'un grand nombre d'acteurs à travers le monde, chacun entre eux a une copie de toutes la blockchain c'est à dire toutes les données et les transactions s'inscrivent dans elle. Cela offre un haut niveau de sécurité. En effet, s'il est possible de s'attaquer à un ou plusieurs ordinateurs, il est plus compliqué de s'attaquer aux blocs d'informations copiés dans l'ensemble des ordinateurs connectés au réseau [9].

Donc, la blockchain est considérée comme un système inattaquable et inviolable. En cas de tentative de fraude, la majorité des acteurs détectent rapidement l'incohérence par rapport à l'historique du systeme.

1.3.4.4. L'autonomie

On dit que la blockchain est autonome parce qu'elle est indépendante des infrastructures centrales. Cela signifie qu'aucun acteur ni organisation ne puissent prendre la main sur la communauté des acteurs où la puissance de calcul et l'espace d'hébergement (stockage) sont fournis par les nœuds du réseau, c'est-à-dire les acteurs eux-mêmes [9]. Donc, la blockchain n'est pas seulement indépendante mais aussi autoportante**.**

1.4. Fonctionnement de la blockchain

Il est maintenant le temps de détailler le fonctionnement de la blockchain, et avant de commencer, nous avons besoin d'un réseau d'ordinateurs ou serveurs afin de bénéficier cette technologie.

Pour mieux comprendre le mécanisme, on prend l'exemple d'une blockchain avec des jetons (actif numérique émis et échangeable) effectués entre deux participants X et Y.

Les différentes étapes de ce scénario sont illustrées dans le schéma suivant (voir figure 1.3).

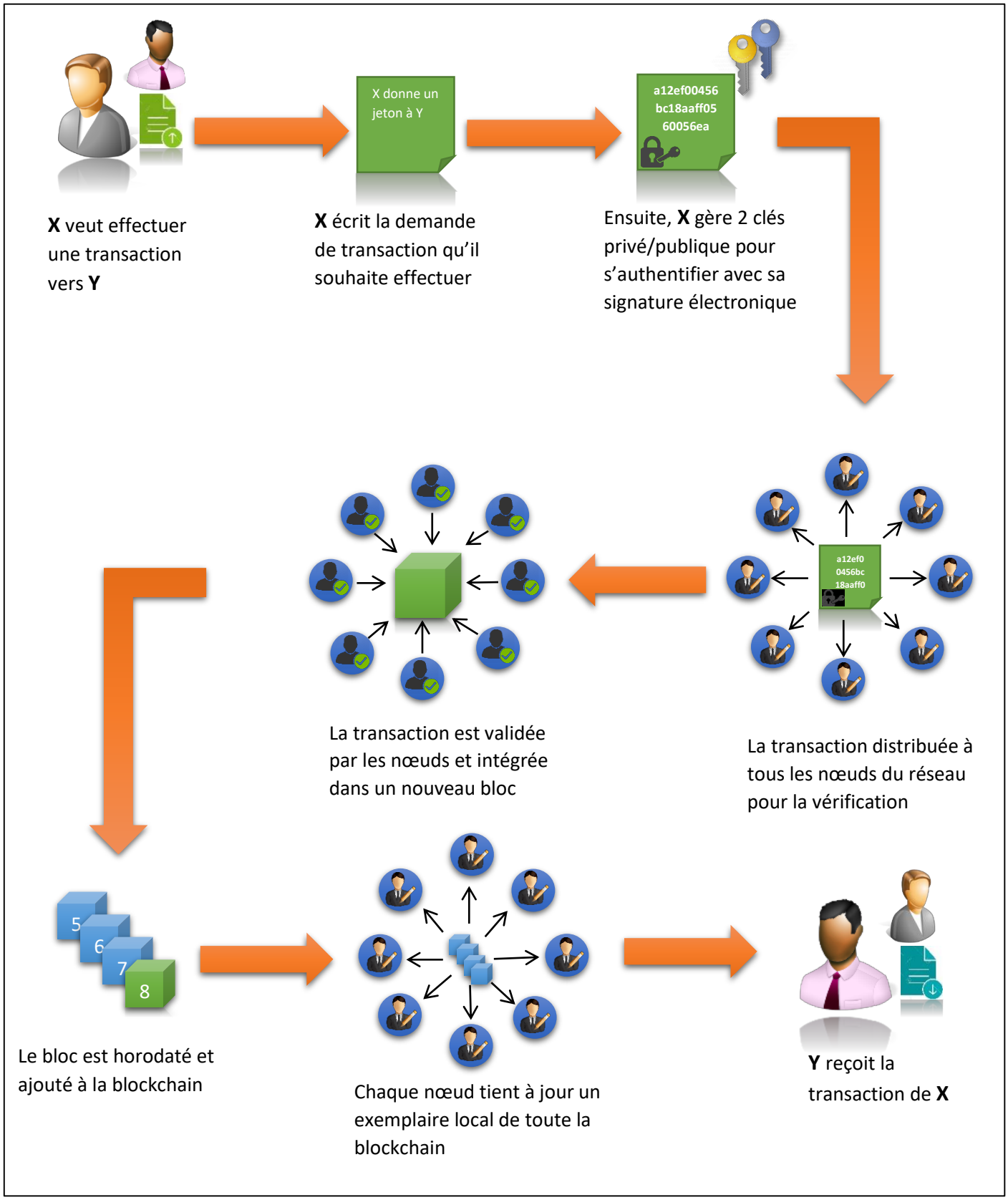

Figure 1.3 : Processus de fonctionnement de la blockchain

X veut effectuer une transaction d'un jeton vers **Y** qui peut être un actif monétaire ou non monétaire (titre, action, vote, …). Mais avant que **Y** reçoit leur jeton, plusieurs opérations vont être mises en jeux :

- **La première opération**

Une demande de transaction est formée sous forme d'un message, l'émetteur **X** annonce à tous les nœuds du réseau qu'il veut donner un jeton à **Y**.

- **La deuxième opération**

X doit ajouter une signature électronique à son message pour garantir l'intégrité des données transmises. Pour cela, il doit gérer une clé privée et une clé publique reliées mathématiquement entre eux.

Le message de transaction est alors signé (chiffré) avec la clef privée qu'elle reste secrète que pour l'émetteur (**X**), et puis, il intègre sa clef publique dans ce message sur laquelle tout le monde dans le réseau est d'accord [11].

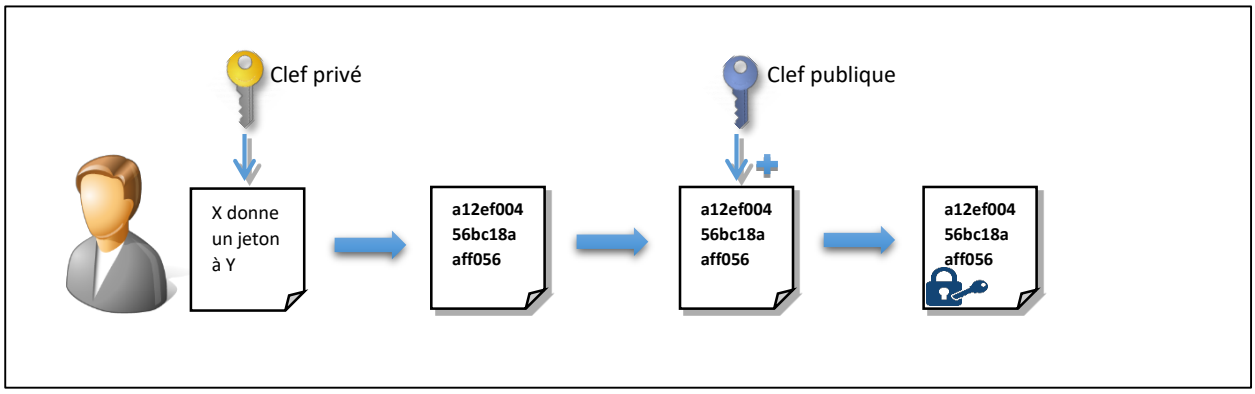

Figure 1.4 : Les étapes de la signature numérique asymétrique.

- **La troisième opération**

Le message signé se distribue à tous les nœuds du réseau qu'ils vont le vérifier pour s'assurer que **X** possède bien ce jeton (voir l'historique des transactions enregistrées dans la blockchain localement), et ils vont assurer l'authenticité de la personne à l'origine de cette transaction et vérifier la signature (déchiffrement) grâce à sa clef publique que tout le monde connaît.

- **La quatrième opération**

Si tout est correcte, cette transaction doit être intégrée à un bloc.

Des nœuds caractérisés par une grande puissance de calcul et d'énergie appelés « mineurs », travaillent simultanément pour forger un nouveau bloc en résolvant un problème mathématique très complexe et ils complètent le bloc en lui ajoutant un en-tête, qui contient le hach du block et du bloc précédent ainsi la date et l'heure exacte où le bloc a été créé.

Le premier qui réussit à créer le bloc transmet ce bloc aux autres membres du réseau pour la vérification [10].

Mais, pour que le bloc soit validé, il faut qu'il ait un mécanisme de consensus entre les membres du réseau qui assure que les règles du protocole soient respectées et pour mettre d'accord sur la validité et sur l'ordre d'ajout d'un nouveau bloc.

- **La cinquième opération**

Une fois le bloc est validé par tous les nœuds du réseau, il est horodaté et ajouté à la chaîne de blocs, et chaque nœud tient à jour un exemplaire local de toute la blockchain. La transaction est alors visible pour le récepteur ainsi que l'ensemble du réseau. (Y) possède maintenant leur jeton [5].

1.5.Mécanismes de consensus

Il existe de nombreux mécanismes de consensus dépendant du type de la blockchain, mais les deux plus couramment utilisés sont les algorithmes de « preuve de travail » et de « preuve d'enjeu ». Les autres sont généralement des dérivés ou des hybrides de ces deux.

1.5.1. Preuve de travail

En anglais proof-of-work ou (PoW) également appelé « minage », cet algorithme est connu comme le principal processus de consensus des blockchains et l'un des piliers de la crypto-monnaie Bitcoin.

Ce procédé doit établir des règles permettant de choisir le mineur qui aura le droit d'inséré le prochain bloc dans la blockchain, donc il demande aux mineurs de résoudre un problème mathématique complexe dit « puzzle » reposant sur la fonction de hachage SHA-256 (dont le résultat est un hash) qui nécessite une puissance de calcul et d'énergie importante mais son résultat est facilement véritable par l'ensemble du réseau afin de trouver un hash du bloc mais

avec une certaine difficulté cible (un hash qui commence par un nombre de zéros défini). Le fait de parvenir à la solution permet de prouver aux autres nœuds du réseau qu'on a bien dû travailler pour parvenir à la solution, et celui qui réussit de trouver la solution et créer le nouveau bloc, il va gagner des frais [12].

Cette difficulté de hachage est liée à un nombre aléatoire différent appelé le nonce qu'on doit l'ajouter à la fonction SHA-256 avec les données fournies dans le bloc ainsi que le hach du bloc précédent et tester différentes valeurs uniques jusqu'à ce que nous obtinrent le hachage cible du bloc correspondant (qui commence par quatre zéros par exemple). Ce nonce est une valeur entière d'une taille de 32 bits ayant environ 4 milliards de possibilités [13].

L'exemple suivant, montre l'algorithme proposé par les mineurs pour trouver un hash à un nouveau bloc avec une complexité (nombre de zéros en début de hash) est de 7 zéros :

- **Premièrement,** inséré le chiffre « Nonce » qui vérifier la propriété suivante :

  HASH (Information du bloc, Hach bloc précédent, Nonce) < **C**

  Avec C est la valeur dépendante de la complexité, c'est-à-dire il faudra trouver un C inférieur à : 0000000fffffffffffffffffffffffffffffffffffffffffffffffffffffffff.

  (Le calcul s'effectue en hexadécimal (base 16), c'est-à-dire en une base allant de 0 a f)
- **Deuxièmement,** recommencer le calcul de l'inégalité précédente en modifiant uniquement le nombre aléatoire, tant que la condition n'est pas réalisée.
- **Troisièmement,** Une fois que le résultat donne le hash avec la difficulté requise, le bloc ainsi formé est envoyé à tous les nœuds du réseau.

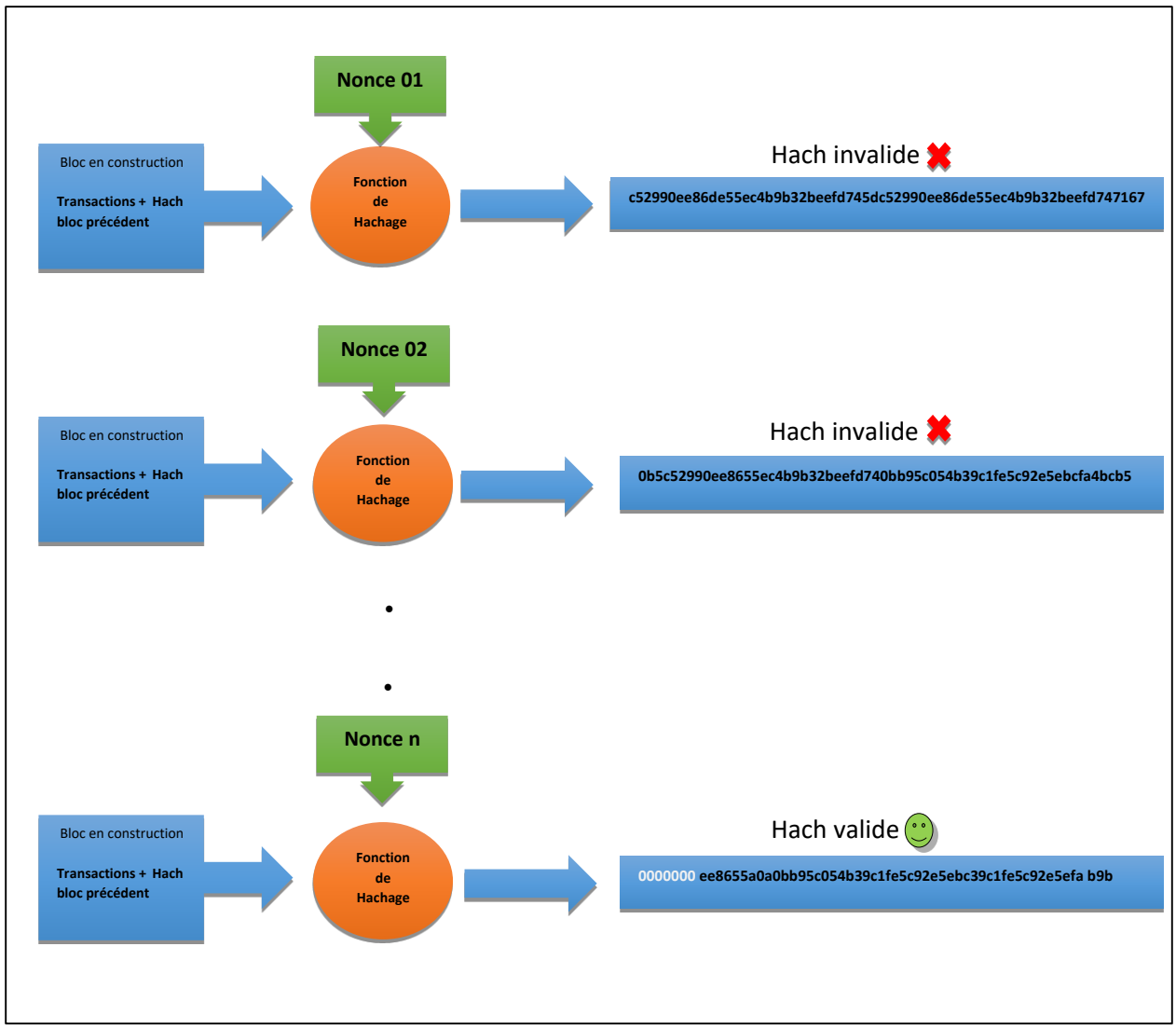

Figure 1.5 : Réalisation de la preuve de travail trouvée en N essais.

Il est important de noter que le niveau de difficulté de hachage dépend du nombre d'utilisateurs, de la puissance actuelle et de la charge du réseau, et puis, il change tous les 2016 blocs, afin que le temps d'extraction soit en moyenne de 10 minutes par bloc.

Malgré que ce processus de preuve de travail nécessite une grande puissance de calcul et matériel informatique très coûteux qui consomme beaucoup d'énergie, il reste l'un des méthodes les plus populaires pour parvenir à un consensus au blockchain, parce qu'il aide à protéger le réseau contre de nombreuses attaques différentes. Une attaque réussie nécessiterait une immense puissance de calcul et un temps considérable pour effectuer les calculs et serait donc inefficace, car le coût engendré par cette attaque serait supérieur aux avantages potentiels que l'on pourrait en tirer [14].

1.5.2. Preuve d'enjeu

En anglais proof-of-Stake ou (PoS) également appelé " preuve de participation ", a été développé par un inconnu sous le pseudonyme « Sunny King » en 2011 qu'il a inventé après la crypto-monnaie « Peercoin » en 2012, la première crypto-monnaie à adopter ce mécanisme.

Contrairement au preuve de travail qui exige à tous les mineurs d'exécuter un algorithme cryptographique très complexe afin de valider et créé un nouveau bloc, le protocole proof-of-Stake utilise un processus d'élection pseudo-aléatoire pour choisir le validateur du bloc [15].

Ce processus demande aux utilisateurs de prouver la possession d'une certaine quantité de jetons, c'est-à-dire plus un nœud possède une quantité importante de jetons (unités monétaires par exemple), plus il a la chance de forger un bloc et obtenir de récompenses [16].

L'algorithme de preuve d'enjeu est pratiquement moins compliqué que l'algorithme de Proof of Work et leur fonctionnement est plus simple à comprendre.

Les utilisateurs qui souhaitent participer au processus de validation (forgeage) mettent en dépôt une certaine quantité de leurs jetons et les annoncer sur le réseau comme enjeu, ensuite, l'algorithme sélectionne aléatoirement un validateur (miner) qui contient plus des unités sur leur dépôt en lui offrant le droit de créer le prochain bloc dans un intervalle de temps donné. Si celui-ci ne crée pas le bloc dans cette intervalle, l'algorithme sélectionnera automatiquement un deuxième validateur à sa place et ainsi de suite [17].

Et si le réseau détecte une transaction frauduleuse, le nœud faussaire perdra une partie de sa participation et son droit de participer en tant que faussaire à l'avenir. Ainsi, tant que la mise est supérieure à la récompense, le validateur perdra plus de pièces qu'il n'en gagnerait en cas de tentative de fraude [14].

Ce système vient pour s'attaquer à certains problèmes posés par la Proof-of-Work (l'algorithme de consensus original dans la technologie blockchain), puisque leur puissance est proportionnelle à leur fortune et non plus à la puissance informatique, mais, il reste donc manipulable que par les riches acteurs [8].

Les différences majeures entre ces deux types de consensus sont résumées dans le tableau suivant :

	Les mécanismes de consensus	
	Proof of Work	Proof of Stake
Type de blockchain	Public	Public
Nœuds validateurs	Les plus puissant en calcule	Les plus riches
Consommation d'énergie	Très consommateur	Peu consommateur
Ressources informatique	Très importantes	Moins importante
Utilisation	Plus utilisé	Moins utilisé

Tableau 1.2 : Les différences majeures entre les deux types de consensus.

1.6. Smart contract

Smart contract ou contrat intelligent en français est une nouvelle amélioration de la technologie blockchain. Ce contrat a été théorisé pour la première fois par le cryptographe « Nick Szabo » en 1994 dans un papier scientifique sobrement intitulé « Smart contracts » et rendu public en 1997. La crypto-monnaies Ethereum est le premier protocole qui a adopté cette amélioration de smart-contract, mais aujourd'hui il est déployé sur d'autres types de blockchain [18].

1.6.1. Le concept de smart contract

Un contrat intelligent ou smart contract en anglais est tout comme un contrat traditionnel en papier, c'est-à-dire, un accord entre deux ou plusieurs parties qui les lie à quelque chose à l'avenir, mais sous forme d'un code informatique.

Ce programme informatique est intégré dans la blockchain qui s'exécute automatiquement pour effectuer des transactions conditionnelles lorsque certaines conditions définies au préalable sont remplies ou certains évènements sont produis. Ces conditions sont été présentés déjà dans la blockchain soient à l'extérieur [10].

Supposons que quelqu'un nommé (A) veut acheter la voiture d'une personne nommé (B). Donc, ils vont mettre un accord entre eux sous forme d'un programme informatique qu'on appelle

smart contract. Ce dernier définit un ensemble des règles ou des conditions par exemple : « (A) paiera 1000$ le 05/03/2020, Si (B) reçoit l'argent le 05/03/2020, ALORS (A) deviendra propriétaire de la voiture, Si NON contrat annuler ».

Ensuite, le créateur du smart contract écrit celui-ci sur la blockchain sous la forme d'une transaction, exactement comme s'il effectuait un simple envoi d'une adresse à une autre. L'algorithme va donc s'exécuter automatiquement lorsque cette date soit arrivée sans aucune nécessité d'un tiers de confiance (avocat, notaire, etc.), et s'annuler automatiquement après cette date.

Alors le contrat intelligent devient impossible à modifier ou falsifier, et son contenu peut être consulté par n'importe qui, en toute transparence puisqu'il est étroitement lié à la technologie blockchain.

1.6.2. Le concept d'oracle

Enfin, s'il est possible de vérifier ce que contient un smart contract en terme de transactions (le montant, l'adresse de l'émetteur et l'expéditeur…), il est beaucoup plus difficile de vérifier une information ou un événement ayant eu lieu hors blockchain, par exemple, si on a une condition comme celle-ci : « Si la voiture tombe en panne avant le 05/03/2021, Alors (A) Reprendre ses 1000$ et (B) Reprend sa voiture ».

Il faudra dans ce cas-là un « oracle » qui permettra d'affirmer que la condition est bien réalisée.

Un oracle est une entité extérieure à la blockchain, qui peut être une personne ou machine tout dépend du code lui-même, qui gère une base de données. Dans notre exemple, l'oracle est une entreprise de réparation automobile externe automatisée ou non, pourra rechercher sur sa base de données regroupant les noms et les matricules de tous les véhicules retenus par l'entreprise pour la maintenance.

1.6.3. Les caractères du smart contract

- **Autonomie** : c'est la non nécessité d'un tiers de confiance avocat, notaire, et d'autres intermédiaires pour confirmer. Cela éloigne également le danger de manipulation par un tiers, car l'exécution est gérée automatiquement par le réseau, plutôt que par un ou plusieurs individus, qui peuvent se tromper.
- **Sécurité :** Vos documents sont cryptés et enregistrés sur un grand livre partagé (blockchain) ce qui fait qu'il n'y a aucun moyen que quelqu'un puisse modifier ou mentir.

- **Sauvegarde** : Imaginez si votre banque a perdu votre compte d'épargne. Sur la blockchain, chacun de vos amis vous soutient. Vos documents sont dupliqués plusieurs fois.
- **Rapidité** : Vous auriez normalement dû consacrer beaucoup de temps et de paperasse pour traiter manuellement les documents. Les contrats intelligents utilisent un code logiciel pour automatiser les tâches, réduisant ainsi les heures de travail sur une gamme de processus métier.
- **Économie** : Les contrats intelligents vous permettent d'économiser de l'argent car ils suppriment la présence d'un intermédiaire. Vous devrez, par exemple, payer un notaire pour assister à votre transaction.
- **Précision** : Les contrats automatisés sont non seulement plus rapides et moins chers, mais ils évitent également les erreurs résultant du remplissage manuel de tas de formulaires [19].

1.7. La crypto-monnaie

Une crypto-monnaie dite aussi crypto-actif, crypto-devise ou encore cybermonnaie, est une monnaie virtuelle alternative à la monnaie classique créée et échangée sur un réseau informatique pair à pair décentralisé et elle n'est rattachée à aucun Etat et aucune banque centrale. Elle permet de réaliser des transactions financières ou du stockage de valeur tout comme la monnaie traditionnelle mais tout est purement numérique [20].

On dit une monnaie virtuelle en ce sens qu'elle n'est pas constituée de billets et de pièces, c'est à dire, on ne peut jamais la détenir ou l'échanger physiquement, et on dit « crypto » parce qu'elle est totalement cryptée par des algorithmes cryptographiques ce qui signifie notamment qu'elle peut uniquement être utilisée par la personne qui détient le code de décryptage pour être échangée en toute sécurité sur Internet [21]. C'est pour cette raison, elle est basée essentiellement sur la technologie blockchain pour assurer la fiabilité et la traçabilité des transactions.

La crypto-monnaie née en 2008 après des années de l'innovation de la blockchain et qui représente la première application basée sur cette technologie parfois nommée blockchain version 01. Il existe plus de 1600 crypto-monnaies dans le monde comme Ethereum, Ripple, Litecoin, Peercoin, …, pour une capitalisation de marché estimée à environ 270 milliards de dollars.

1.7.1. Bitcoin

Le Bitcoin est la première crypto-monnaie et la plus célèbre jusqu'aujourd'hui. L'idée a été proposée pour la première fois en novembre 2008 par « Satoshi Nakamoto » et mis en œuvre le 3 janvier 2009.

L'idée de cette cybermonnaie est de créer une monnaie universelle et indépendante à toutes les banques et les gouvernements et échangeable sur un réseau informatique pair à pair c'est-à-dire sans aucune autorité centrale ni administrateur unique. C'est pour cela, « Satoshi Nakamoto » a choisi la blockchain comme plateforme de son protocole (open source) où les transactions sont enregistrées et visibles par tous les membres du réseau et un protocole de signature à double clefs cryptographiques dites asymétriques en parallèle pour assurer l'intégrité des données et augmenter la sécurité des transactions.

Le Bitcoin repose sur le mécanisme de consensus « la preuve de travail » (Proof of Work) des mineurs avec une fonction de hachage très complexe (SHA 256). Il est considéré comme la blockchain originale, ou de nombreuses crypto-monnaies actuellement sont basées sur ce code (Bitcoin) [22].

Voici les avantages du Bitcoin les plus couramment évoqués :

- Bitcoin est le système financier le plus ouvert, vous pouvez effectuer des paiements en tout moment (24h/24h et 7j/7) partout dans le monde sans la nécessité d'un système bancaire.
- Les virements internationaux en Bitcoin sont plus rapides et moins coûteux par rapport aux services bancaires traditionnels.
- Bitcoin introduit un concept d'argent programmable, qui permet d'autres innovations financières telles que « les smart contracts » [23].

1.7.2. Ethereum

Ethereum est un protocole de blockchain imaginé par Vitalik Buterin publié dans son white paper en décembre 2013. C'est un protocole de transfert d'actifs monétaires comme le fait le protocole Bitcoin. Cette crypto-monnaie est lancée le 30 juillet 2015 et a fait beaucoup parler d'elle depuis ce jour.

Ether est une crypto-monnaie numérique décentralisée, également comme le bitcoin mais elle possède son propre langage de programmation appelé Turing-complet, qui permettrait à

n'importe qui d'y développer ce qui signifie que les développeurs peuvent l'utiliser pour construire de nouveaux types d'applications, et créer des smart contracts. En outre, Ethereum introduit plusieurs innovations majeures, dont celle qui vise à créer un bloc idéalement, non pas toutes les 10 minutes comme Bitcoin, mais, en moyenne, toutes les 15 secondes dont le consensus repose sur Proof of Work (Preuve de Travail) et Proof of Stake [8].

La structure de la blockchain Ethereum est très similaire à celle de Bitcoin, en ce sens qu'il s'agit d'un enregistrement partagé de l'ensemble de l'historique des transactions. Chaque nœud du réseau stocke une copie de cet historique. La grande différence avec l'Ethereum est que ses nœuds stockent l'état le plus récent de chaque smart contract, en plus de toutes les transactions éther [24].

1.7.3. Ripple

Le ripple est une cryptomonnaie construite sur un réseau distribué et open source, au même titre que les autres crypto-monnaies et considérée comme la troisième plus grande cryptomonnaie après le Bitcoin (BTC) et l'Ethereum (ETH) en terme de capitalisation boursière.

Ripple est donc une start-up de San Francisco spécialisée dans les échanges interbancaires à très haute fréquence entre les institutions financières et les individus.

L'entreprise Ripple est créée en 2004 par Ryan Fugger pour concevoir la création d'un système décentralisé où n'importe qui pourrait créer de l'argent. Le réseau est définitivement lancé en 2012 et connait depuis une affluence de plus en plus conséquente. Mais alors, pourquoi est-il question de cryptomonnaie si Ripple est une entreprise ?

C'est là qu'il est important de faire la différence entre Ripple et XRP, souvent associés à une seule et même entité. Mais en réalité, le XRP est un « actif numérique indépendant avec une chaîne de blocs open source ». Ripple est donc une entreprise qui a créé le protocole de paiement Ripple, le XRP étant un jeton de cryptomonnaie utilisé par cette dernière [25].

Le réseau Ripple, en quelques premiers points, c'est :

- Concurrent direct des institutions bancaires et financières actuelles.
- Très rapide par rapport aux autres crypto-monnaies. Votre transaction sur Ripple se déroulera en moins de 4 secondes.
- Les coûts en énergie sont faibles.

1.8. Les domaines d'application

La technologie blockchain ne se limitera pas à l'administration des crypto-monnaies. Demain, la blockchain pourrait bien être la technologie phare à l'œuvre dans de nombreux autres domaines.

1.8.1. Les assurances

Les assurances commencent à utiliser la blockchain dans leur gestion quotidienne, notamment grâce aux smart contracts et le concept d'oracles (des programmes autonomes qui exécutent automatiquement les conditions et termes d'un contrat, sans nécessiter d'intervention humaine une fois démarrés).

Un exemple chez les compagnies aériennes, le retard de vol pose un grand problème pour l'entreprise lui-même et pour les voyageurs aussi (60 % des passagers assurés contre le retard de leur vol ne revendiquaient jamais leur argent), donc l'idée de créer un système d'assurance automatisé basé sur la blockchain va résoudra le problème ce qui fait que les passagers sont automatiquement indemnisés lorsque leur vol est en retard, sans avoir besoin de remplir un quelconque formulaire, et donc sans que l'entreprise ne doive traiter les demandes [5].

1.8.2. L'énergie

Le secteur de l'énergie est très sensible en plusieurs côtés, de l'extraction à la consommation, ce qui rend l'utilisation des blockchains très intéressante.

L'idée générale derrière l'exploitation de la blockchain dans ce secteur est celle d'être un tiers de confiance et permettre d'effectuer des transactions entre particuliers de manière décentralisée.

C'est le cas du projet « TransActive Grid », basé sur l'énergie renouvelable et l'économie du partage, utilise la blockchain pour acheter et vendre de l'énergie solaire. Ce réseau local et autonome a été inauguré à Brooklyn début 2016. Il s'agit de donner la possibilité aux citoyens de se réapproprier leur production énergétique, par l'établissement de mini communautés énergétiques autonomes. Pour cela, des capteurs enregistrent l'historique de la création énergétique à un point précis, et l'enregistre aussitôt sur la blockchain Ethereum. Des smart contracts peuvent ensuite régir les règles d'utilisation de cette énergie, et les tarifs des producteurs [5].

1.8.3. La santé

La contrefaçon des médicaments est beaucoup plus répandue dans le monde provocant des problèmes de santé graves et même des décès. Environ 700000 personnes décédées chaque année selon les statistiques de l'Organisation Mondiale de la Santé. L'introduction de blockchain dans ce domaine pourrait notamment servir à la traçabilité des médicaments, en tant que registre distribué, permet aux différentes entreprises pharmaceutiques, aux régulateurs et même aux particuliers d'utiliser la même base de données, sans qu'une seule entreprise ou institution n'en soit propriétaire [5].

La blockchain permet aussi de gérer de manière transparente, sûre et infalsifiable les données des patients et restreindre le partage des données (total ou partiel) aux principaux tiers de confiances : médecins, hôpitaux, laboratoires, etc.

On peut même enregistrer par la suite les étapes du parcours de soins sur la blockchain regroupant institutions de santé et assureurs. Il serait également possible d'automatiser le paiement des prestations médicales nécessaires, grâce à des smart contracts [5].

1.8.4. L'art

La musique est aussi un domaine touché par la blockchain. En effet, le but de cette utilisation est la suppression d'intermédiaires, l'assurance d'une meilleure traçabilité des œuvres, une transparence dans la gestion des droits d'auteur et la répartition des droits de paiements sans passer par des intermédiaires tel que Spotify ou Deezer où les musiciens pourraient être payés directement par ceux qui écoutent leur musique [26].

1.8.5. Gouvernement

Le cas d'utilisation de la technologie blockchain pour les gouvernements d'état va permet de sécuriser les documents gouvernementaux et de tenir les fonctionnaires responsables par le biais de contrats intelligents et d'assurer la transparence en enregistrant un dossier public de toutes les activités.

La blockchain pourrait également améliorer l'efficacité démocratique, par exemple, le cas des élections. Un vote basé sur la blockchain pourrait améliorer l'engagement civique en toute sécurité sur un appareil mobile de n'importe où dans le monde sans crainte de piratage ou de corruption de données [27].

1.9.Les avantages et les inconvénients de la blockchain

1.9.1. Les avantages

La structure décentralisée et transparente de la technologie de la blockchain comporte de nombreux avantages :

- **La sécurité du système** : les transmissions entre les utilisateurs dans le réseau blockchain reposent sur la cryptographie moderne asymétrique, ainsi la validation et le consensus commun sont effectués par un ensemble d'utilisateurs différents, qui ne se connaissent pas. Cela permet de se prémunir du risque de malveillance ou de détournement.
- **La suppression des intermédiaires :** grâce à la technologie blockchain, ce n'est plus nécessaire d'un tiers de confiance car le réseau de nœuds distribués vérifie les transactions en toute transparence. Ce système élimine alors le risque de faire confiance à une seule organisation.
- **Traçabilité et transparence :** Les informations contenues dans la blockchain ne peuvent être effacées ni modifiées. Une fois qu'une opération a été réalisée, elle restera gravée à jamais dans la blockchain ce qui permet de savoir exactement le chemin parcouru par les informations.
- **La réduction des coûts de transaction :** le système permet aussi de réduire les coûts globaux et les frais de transaction en éliminant les intermédiaires ou les tiers.
- **L'automatisation de contrats immuables :** Le registre blockchain permet la mise en place de smarts contracts dont les conditions sont fixées à l'avance entre deux personnes ou plus. Lorsqu'un événement se produit, le déclenchement de la clause associée est immédiat et transparent [22].

1.9.2. Les inconvénients

Bien que la blockchain est une technologie qui possède de nombreux avantages, elle peut aussi causer certains inconvénients :

- **L'énergie consommée :** l'énergie consommée pose un grand problème pour la technologie blockchain. En effet, ce qui consomme énormément d'énergie est le système de vérification des transactions qui fonctionne avec la PoW (Proof of Work).
- **Difficulté d'adaptation du grand public** : la blockchain demandera à la société un certain temps d'adaptation pour que le grand public se mette à l'utiliser.
- **Augmentation du chômage :** à cause de son système d'automatisation, la blockchain pourrait remplacer plusieurs services d'administration, de fonds, de compatibilité, et pourrait donc faire disparaître beaucoup de métiers et envoyer plein d'employés au chômage.
- **Une technologie lourde :** l'utilisation de la blockchain demande un support technologique plutôt important pour sa transmission d'informations. Plusieurs ordinateurs se font en compétition pour la résolution d'une série de calculs qui permettra le transfert des informations cryptées une fois qu'elle a été résolue [28].

1.10. Conclusion

La blockchain est un nouveau concept dans le monde des technologies mais c'est définitivement celui qui durera. Elle détient un potentiel élevé d'applications dans de nombreuses industries et secteurs différents où la confiance sans l'implication d'une autorité centralisée est souhaitée. Certaines industries ont déjà commencé à adopter la blockchain dans leurs entreprises.

Dans le prochain chapitre, nous allons aborder les intérêts de la blockchain dans le monde médical.

Chapitre 2

L’intérêt de la blockchain dans le monde médical

2.1. Introduction

L'un des obstacles à la transition des soins de santé vers l'espace numérique est la sécurité du transfert et du stockage des données. La disponibilité d'une base de données étendue peut être très peu fiable et fragile, si elle ne dispose pas d'une sécurité adéquate. La technologie blockchain est en mesure d'assurer l'échange sécurisé de données privées entre le patient et le médecin ou les centres de santé. De plus, la blockchain peut fournir une optimisation permettant d'économiser du temps et de l'argent et une démocratisation des données pour chaque patient.

2.2. L'opportunité de la blockchain dans le soin de santé

Il existe une variété de problèmes avec la façon dont les données de santé sont transmises et stockées aujourd'hui. Chaque médecin ou établissement médical avec lequel les patients interagissent conserve généralement sa propre copie locale de leurs données de santé. Ces méthodes rendent difficile, et impossible parfois, le partage ou la coordination des informations entre les professionnels de la santé.

La nature intrinsèquement sensible des données sur la santé, ainsi que les défis permanents de L'interopérabilité, l'appariement des dossiers des patients et l'échange d'informations sur la santé ont créé opportunités pour une approche blockchain. Les informations présentées par le dossier électronique de santé sur le grand livre d'une blockchain seraient parfaitement conciliées à l'échelle de la communauté, avec l'intégrité assurée du point de génération des données au point d'utilisation, sans intervention d'un tiers de confiance. La solution blockchain pour l'industrie de la santé réduit considérablement le temps nécessaire pour accéder aux informations du patient, améliore l'interopérabilité du système et améliore la qualité des données et permet également de réduire les frais généraux [29].

Comme nous l'avons vu dans le premier chapitre, la technologie blockchain est connue comme un système distribué qui enregistre et protège les données grâce aux méthodes cryptographiques (hachage, cryptage asymétrique), ce qui rend difficile à perturber ou à modifier les données sans l'approbation de tous les autres participants du réseau, de plus, l'architecture de pair-à-pair utilisée dans cette technologie permet de synchroniser toutes les copies de l'enregistrement au fur et à mesure de sa mise à jour, même si elles sont stockées sur différents ordinateurs pour garantir que les données sont à jour et authentiques.

Dans ce chapitre, nous aurons exploré les problèmes majeurs dans le monde médical d'aujourd'hui, et comment la blockchain peut résoudre ces problèmes.

2.3. Dossier électronique de santé (DES)

Un dossier électronique de santé (DES) ou « Electronic Health Record (EHR) en Anglais », est une version numérique du dossier papier d'un patient, contenant ses données de santé et l'ensemble d'informations sur son historique médical, qui ont été remplis par plusieurs sources professionnelles, telles que les hôpitaux, les cliniques, les pharmacies et les laboratoires.
Les DES ont des enregistrements en temps réel des informations du patient, disponibles instantanément pour les utilisateurs autorisés et n'interagissent généralement pas avec les autres systèmes en restant limités à l'information disponible dans l'établissement [30].
Ces dossiers électroniques contiennent :

- Les coordonnées du patient : le nom complet, la date de naissance, le sexe, l'âge, l'adresse, numéro de téléphone, numéro du dossier, etc.
- Les informations relatives aux soins dispensés par les autres professionnels de santé : les diagnostics, les médicaments, les plans de traitement, les images radiologiques et les résultats de laboratoire et de test.
- Les informations relatives de l'établissement qui à la prise en charge du patient : nom et l'adresse de l'établissement, le nom du médecin superviseur, le service, etc.
- Les informations formalisées établies à la fin du séjour : la date d'entrée, le compte rendu d'hospitalisation, la lettre rédigée à l'occasion de la sortie.
- Les accidents et les antécédents médicaux du patient.
- Les vaccinations et autres actions de prévention et de dépistage.
- Les allergies et les antécédents familiaux.
- Informations sur toute intervention chirurgicale ou procédure effectuée [31].

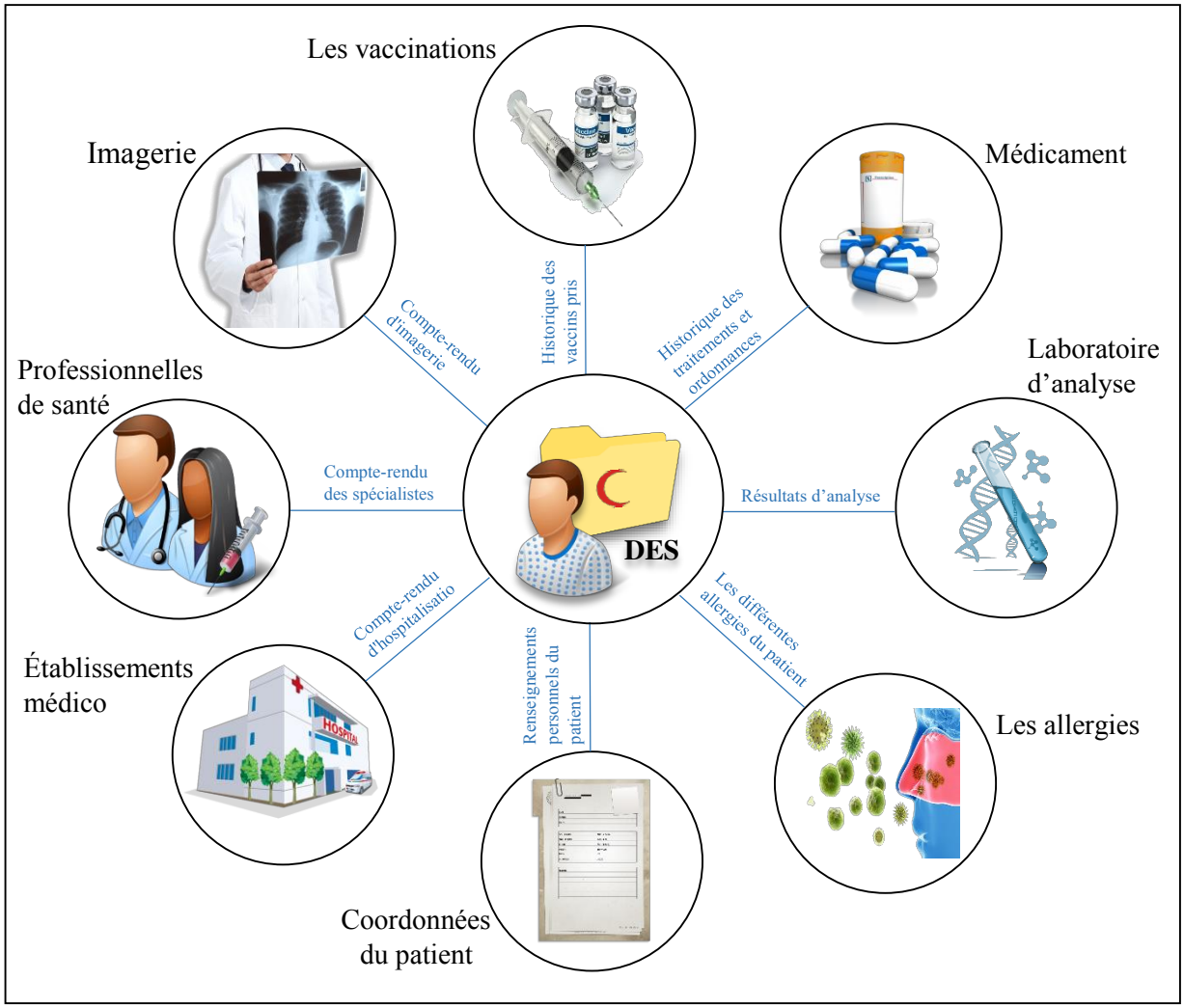

Figure 2.1 : Contenu de dossier électronique de santé (DES).

La gestion des données médicales par une association ou un groupement d'intérêt public indépendant n'est pas une bonne solution dans la mesure où la centralisation en un même lieu offre des opportunités aux hackers, certaines personnes ont peur que des personnes non autorisées aient accès au DES. Ils veulent que leur vie privée soit respectée, un souhait tout à fait légitime.

Plutôt que de vouloir organiser complètement la gestion des DES, il semblerait opportun que les pouvoirs publics s'attachent uniquement à faire respecter les droits du patient dans le domaine du dossier médical informatisé : droit à l'information, droit à l'opposition, droit à l'accès direct, droit à la sécurité et le partage sécurisé avec qui veut.

2.4. Partage sécurisé et fiable des DES à l'aide de la blockchain

Les dossiers de santé électroniques (DES) sont des informations privées essentielles et très sensibles pour le diagnostic et le traitement dans les soins de santé, qui doivent être fréquemment distribuées et partagées entre pairs tels que les prestataires de soins de santé, les compagnies d'assurance, les pharmacies, les chercheurs, les familles des patients, entre autres.

La sécurité de ces dossiers constitue un enjeu majeur pour le secteur médical. Les organismes de santé utilisent actuellement la base de donnée centralisée pour enregistrer les données personnelles des patients, en particulier, c'est à dire l'ensemble des DES où une autorité centrale a un accès complet au système, mais cette méthode de stockage n'est pas vraiment efficace et les données peuvent être utilisées à mauvais escient pour de nombreuses raisons car les patients n'ont aucun contrôle sur leurs données [32].

Selon les statistiques mondiales de l'année 2016, 450 incidents signalés ou divulgués dans les médias ou d'autre sources mais les informations n'étaient disponibles que pour 380 incidents de violation. Avec plusieurs violations de données de santé par jour pendant toute l'année, ces violations ont entraîné 27 314 647 atteintes dossiers des patients qui ont coûté près de 30 milliards de dollars, 43% de ces violations sont fait par les initiés (vol d'informations par les employés). En effet, les industries de santé subissent deux fois plus d'attaques informatiques que les autres industries [33].

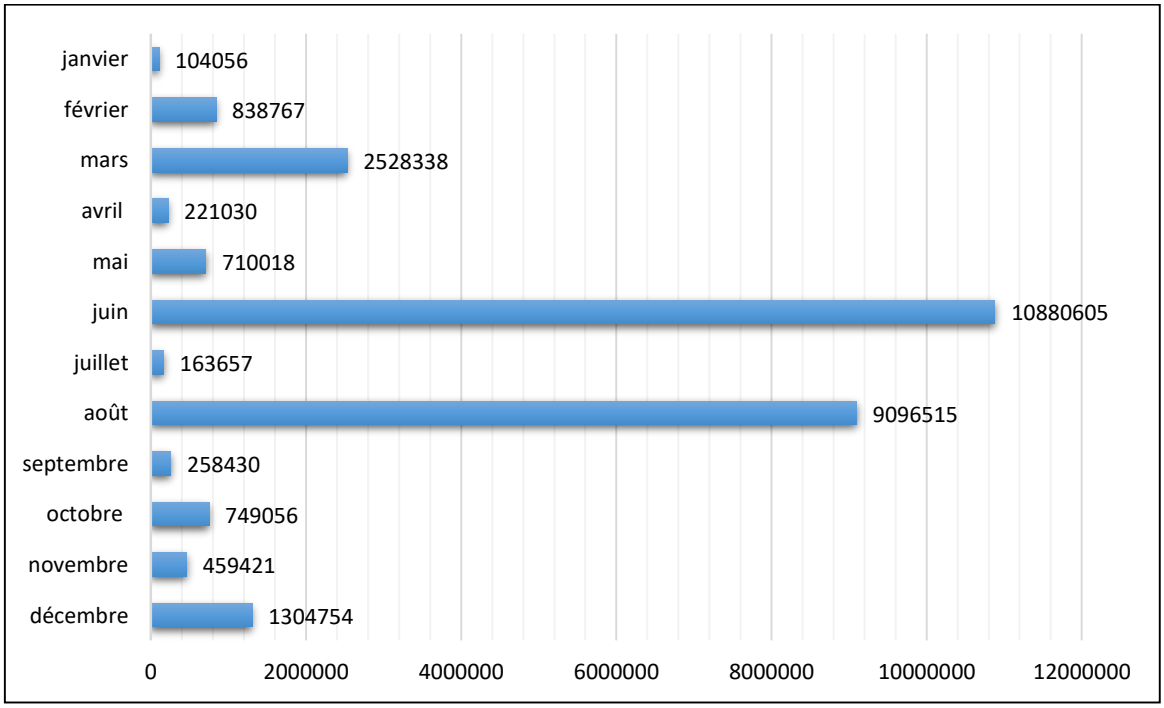

Figure 2.2 : Nombre d'enregistrements violés par mois en 2016 [33].

Ainsi devant l'explosion des coûts et le nombre de failles, il est nécessaire de trouver un moyen plus efficient de stocker les données. La technologie blockchain apparaît comme un outil intéressant pour pouvoir apporter le niveau de sécurité nécessaire.

La blockchain, et grâce à sa décentralisation et son inaltérabilité, on peut assurer l'intégrité des données, d'abord les dossiers électroniques de santé et les données personnelles du patient sont stockées dans les différents ordinateurs du réseau (médecins, pharmacies, hôpitaux, assurances, etc.), ce qui rend quasi impossible de les modifier ou falsifier car il devrait la modifier sur la majorité des nœuds du réseau ce qui est pratiquement impossible.

Ainsi, personne ne possède toutes les données à l'échelle mondiale en tant qu'autorité ce qui fait que le patient a le contrôle total de ses propres informations médicales et peut décider avec qui les partager.

Et pour améliorer beaucoup plus la sécurité et la confidentialité des données, on utilise les protocoles de cryptographie asymétrique et les contrats intelligents pour savoir pour qui et de quelle manière les informations du patient sont disponibles.

2.5. Gestion des données du patient

De nos jours, il est difficile de visualiser de manière claire toutes les données liées à un patient et accumulées au cours de son parcours de soins. Un patient visite souvent plusieurs médecins et hôpitaux ou des laboratoires d'analyses ayant des logiciels différents pour collecter et gérer les dossiers médicaux et qu'ils ne sont pas interopérables ce qui fait que le patient doit conserver l'historique de toutes ses données et maintenir les mises à jour surtout pour un patient qui souffre d'une maladie grave comme le cancer ou le VIH qu'on doit maintenir la longue histoire du processus de traitement et de réadaptation et de suivi post-traitement, cela conduit à la situation où les informations requises peuvent ne pas être disponibles. Ce problème est fréquemment rencontré lors de l'admission d'un patient à l'hôpital. Les professionnels de santé n'ont en effet pas toujours accès à son historique et n'ont pas une visibilité complète sur les traitements qu'il prend [34].

Une étude du British Medical Journal a montré qu'aux États-Unis les erreurs médicales sont la troisième cause de décès dans le pays derrière les maladies cardiaques et les cancers avec plus de 250 000 décès par an [35].

L'idéal serait donc d'avoir une liste transparente de tous les participants du réseau qui répertorie tous les données médicales d'un patient afin de pouvoir rapidement les récupérer. Cette liste serait accessible, avec l'accord du patient à tout professionnel de santé qui en ferait la demande. La technologie blockchain apporte justement cette solution sous la forme d'un registre distribué et sécurisé qui permet au patient non seulement d'avoir une visibilité sur ses données, mais aussi de les partager et de contrôler les accès [34].
Cela permet aussi de faciliter la gestion des dossiers médicaux dans les établissements de santé, d'éviter les pertes de document ou l'oubli de ces derniers par le patient lors des rendez-vous avec les professionnels de santé.

2.6. L'interopérabilité

Aujourd'hui, il existe un réel problème d'interopérabilité des systèmes informatiques rendant souvent difficile le travail entre les établissements de santé. Malgré l'importance du partage des données médicales, les systèmes de santé d'aujourd'hui exigent souvent des patients qu'ils obtiennent et partagent leurs propres dossiers médicaux avec d'autres prestataires, soit sur papier physique, soit sur des copies électroniques parfois (CD, USB, etc..). Ce processus d'obtention et de partage des dossiers médicaux est inefficace car :

- Il est lent, les copies des données doivent être préparées, livrées et récupérées par les patients.
- Il n'est pas sûr car les données peuvent être perdues ou volées lors de leur transmission par les patients d'un endroit à un autre.
- Il est incomplet, il n'y a pas de source unique qui stocke tous les dossiers médicaux d'un individu, les patients doivent donc être responsables de garder une trace de quand et où ils ont reçu des services de santé afin de demander des copies de leurs antécédents médicaux [36].

La blockchain, et lorsqu'elle est définie comme un registre distribué à un ensemble de machines, elle pourrait être une option intéressante pour résoudre ce problème via l'enregistrement de toutes les données et l'historique des patients et de certaines données médicales provenant de l'ensemble des parties prenantes du système de santé (laboratoires, médecins, hôpitaux, assurances, cliniques, patient). L'enjeu ici est de réduire drastiquement les coûts et la vitesse de transfert d'informations entre toutes les parties prenantes du système de santé, le tout en sécurisant les données et en préservant la vie privée des patients [32].

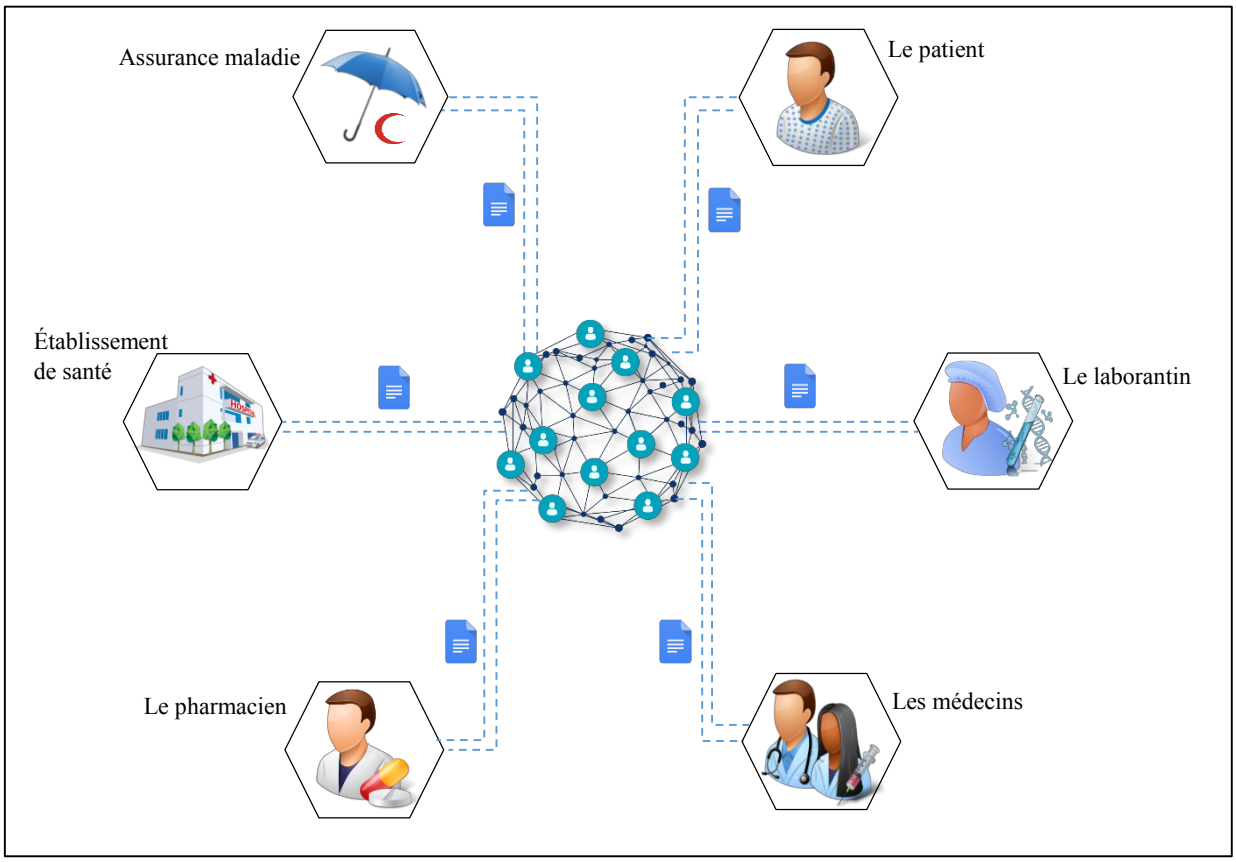

Figure 2.3 : Modèle de gestion des données dans un modèle blockchain comprenant une source de vérité unique.

2.7. Traçabilité des médicaments

Selon l'Organisation Mondiale de la Santé (OMS), plus de 120 000 personnes meurent chaque année en Afrique à cause des faux médicaments antipaludiques seuls, soit parce que les médicaments étaient de qualité inférieure ou ne contenaient tout simplement aucun ingrédient actif. On estime qu'un produit médical sur dix circulant dans les pays à revenu faible ou intermédiaire est soit de qualité inférieure, soit falsifié. La valeur estimée de ce marché des médicaments contrefaits varie de 150 à 200 milliards d'euros dans le monde [37].

Deux choses doivent à tout prix être respectées dans la chaine logistique des médicaments. D'une part, s'assurer qu'aucun médicament non légitime ne puisse s'immiscer dans la chaine logistique se faisant passer pour un produit approuvé. D'autre part, pouvoir contrôler, pendant tout le cheminement du produit les standards qualité, comme la température ou l'humidité. Il est primordial que depuis l'acheminement des matières premières au fabricant jusqu'à la

délivrance du médicament au patient les produits n'aient subi aucun dommage durant leur transport [35].

Traditionnellement, les informations relatives à un produit sont conservées par chaque entité de la chaîne logistique. Cette dispersion de l'information ne permet pas d'avoir une vue d'ensemble des opérations subies par les produits et ralentit considérablement les investigations. De ce fait, pour pouvoir avoir un historique complet de toutes les étapes logistiques, il est nécessaire de contacter chaque acteur.

La technologie blockchain possède de nombreux atouts pour traiter les problématiques de continuité et sécurité de l'information. Comme on a déjà vu dans le premier chapitre, chaque nouvelle transaction effectuée doit être signée (chiffrée) et validée par les nœuds du réseau ensuite intégrée dans un bloc qui par la suite va s'insérer dans la blockchain d'une manière irréversible à l'aide d'un processus cryptographique (le hachage).

Grâce à la transparence et l'inaltérabilité de la blockchain, les industries pharmaceutiques peuvent suivre le chemin de chaque produit médical depuis sa sortie du laboratoire vers le magasin en arrivant jusqu'au consommateur final. Les patients contrôlent l'origine de leurs médicaments et peuvent détecter un médicament qui ne satisfait pas les standards de qualité qui peut présenter des risques ou des effets indésirables pour leur santé.

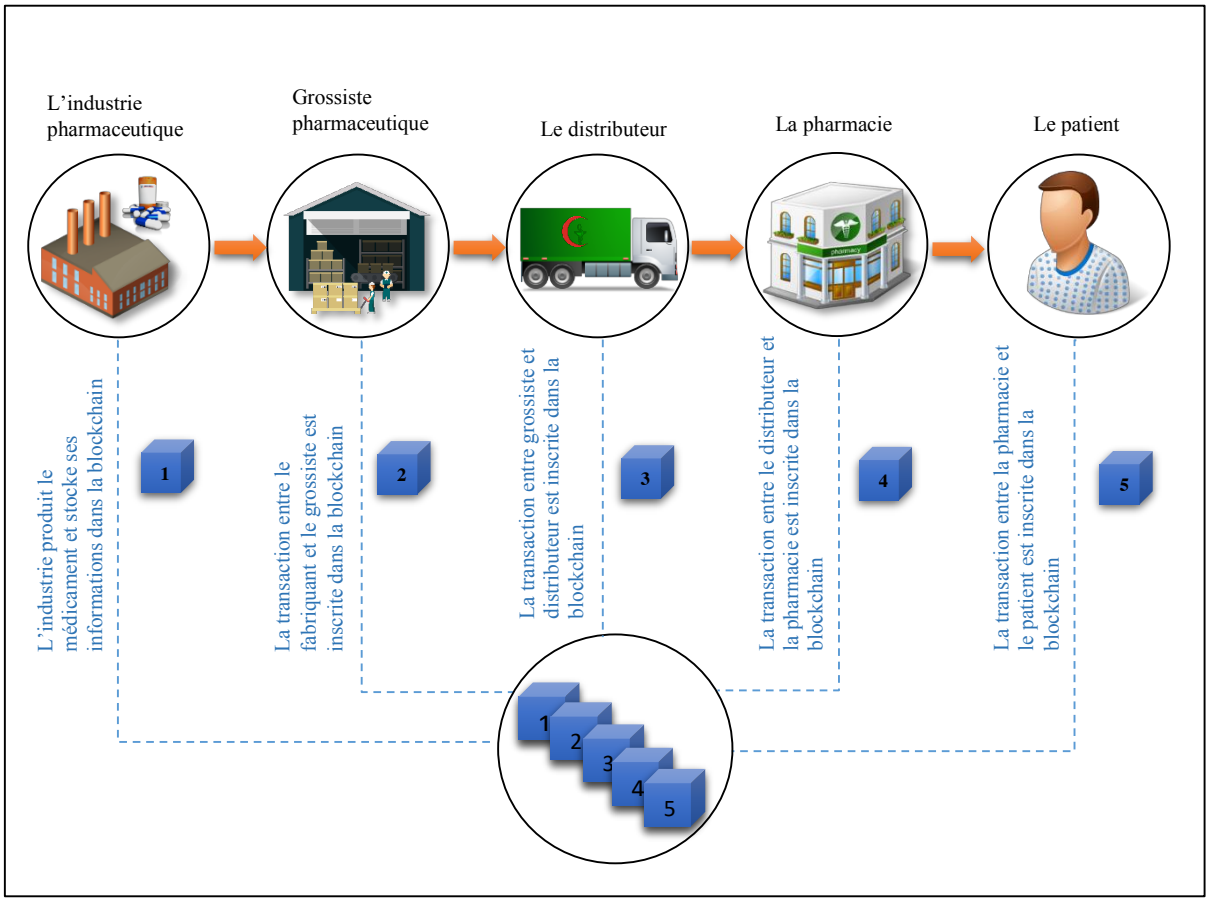

Figure 2.4 : Technologie blockchain pour la traçabilité des médicaments.

2.8. Les essais cliniques

Les laboratoires pharmaceutiques et leurs sponsors peuvent investir des milliards de dollars pour les essais cliniques afin d'identifier l'efficacité de certains médicaments. Ces tests peuvent durer plusieurs années avant de valider ou infirmer un produit, et un résultat non concluant pourrait avoir de grandes répercussions financières [38].

Au cours des essais cliniques, une quantité considérable de données est produite : rapports de sécurité et de qualité, statistiques, analyses de sang, enquêtes, imagerie médicale, etc. et de grands groupes de participants sont impliqués, ce qui rend le suivi et le contrôle assez difficile. En effet, la régulation des essais cliniques oblige à inclure de plus en plus de patients dans les essais afin d'avoir des résultats statistiquement probants. Par conséquent, des erreurs peuvent être commises en cours de route, certaines involontairement et d'autres avec intention par la modification ou la dissimulation de données qui pourraient compromettre l'avancement de

l'essai et nuire l'image d'une organisation auprès des organismes de réglementation ou des patients [38].

La modification des informations existantes dans la blockchain nécessiterait la modification de tous les enregistrements de la majorité des membres du réseau et l'ajout de nouvelles données nécessiterait également la confirmation et l'accord de tous les participants du réseau.

Ces caractéristiques sont très puissantes lorsqu'elles sont appliquées aux essais cliniques. En effet, la blockchain peut horodater et stocker les étapes et les résultats produits pendant l'essai et assurer leurs authenticités et leurs traçabilités. Cela permettrait aussi aux responsables de la recherche clinique de partager les demandes de consentement en temps réel avec les régulateurs, la société pharmaceutique et les organismes de recherche [34].

2.9. L'assurance maladie

La blockchain est un outil qui aide également les compagnies d'assurances de santé de traiter et vérifier l'historique des traitements et la facturation des ordonnances médicales pour lutter contre la fraude.

Dans le cas des remboursements des prestations médicales et pharmaceutiques par exemple, les contrats intelligents pourraient jouer un rôle salutaire. En effet, au lieu de soumettre les professionnels de santé à des tâches administratives chronophage, beaucoup de paiements pourraient être entièrement automatisés.

Il est possible d'imaginer que les assurances opèrent leurs remboursements des clients via un smart contract qui s'exécute automatiquement quand un événement assurable est vérifié (un médicament acheté, un accident de travail, opération chirurgicale, etc.), ou pour la prise en charge des allocations familiales lorsqu'une famille aura un nouveau-né.

Le prix de tous les actes médicaux et des médicaments peuvent être enregistrés dans la blockchain. L'assurance de santé et les mutuelles sont liées contractuellement avec les médecins et pharmaciens. Le contrat intelligent peut alors effectuer des virements automatiques vers les professionnels de santé lorsqu'une prestation définie est réalisée par un professionnel de santé agréé par l'assurance maladie et reconnu comme reçue par le patient [35].

À ce moment-là, si le contrat est appelé dans la blockchain, il sera vérifié que :

- Le patient et la pharmacie sont bien reconnus par l'assurance maladie (et éventuellement une mutuelle pour le patient).
- Les médicaments délivrés font bien parti de la liste des médicaments remboursables.
- Le patient certifie avoir reçu les produits de santé.

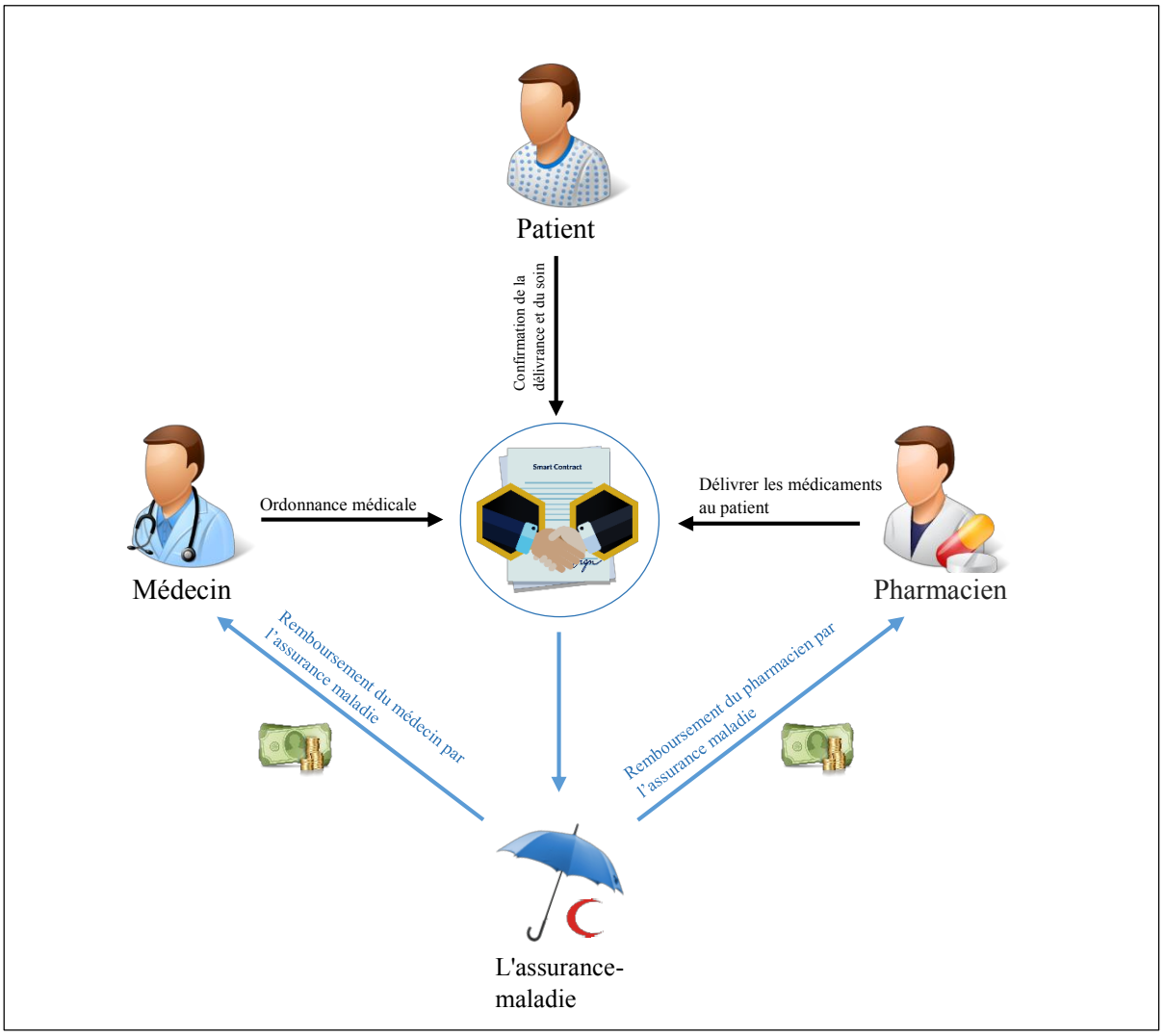

Figure 2.5 : Remboursement des frais de santé automatique grâce à un contrat intelligent.

Si ces trois critères sont remplis alors le pharmacien sera directement remboursé par l'assurance maladie ainsi que la mutuelle sans que le patient n'ait à avancer les frais et le tout sans nécessiter une surcharge administrative conséquente pour le pharmacien en plus de limiter le temps passé à des tâches administratives [35].

2.10. Lutte contre le COVID-19

Quand l'épidémie apparaît et éclaté dans un pays ou bien dans le monde entier, il est nécessaire d'abord de pouvoir le détecter de manière urgente**,** le plus précocement possible afin de pouvoir alerter les professionnels de santé, préparer des stocks nationaux de produits de santés adaptés, sensibiliser la population sur des bonnes pratiques pour réduire la propagation.

Dernièrement, le monde plonge dans une crise sanitaire en raison de l'émergence d'un nouveau virus COVID-19 surtout avec le taux de croissance effrayant d'infection qui l'accompagne. Alors que de nombreux efforts sont déployés pour faire face à cette crise par les organisations et les gouvernements, par exemple, des tests accrus pour trouver un vaccin efficace contre ce virus, des commissions présidentielles pour suivre la propagation de coronavirus, et de nombreuses campagnes de sensibilisation médical et culture, le traçage des chaînes d'infection et l'isolement des personnes potentiellement infectés ainsi que les groupes à risque.

Cette épidémie mondiale, qui est le COVID-19, peut introduire la technologie blockchain sous le microscope car elle peut jouer un rôle important dans la propagation du virus. Les laboratoires biologiques, les chercheurs et les université mondiales souhaitent visualiser, en toute sécurité, instantanément et en mode collaboratif, au niveau international les analyses et les résultats des expériences de traitement (par exemple chloroquine), développement des tests de vaccins, amélioration des tests de dépistage, accès sécurisé aux dossiers des patients contaminés et vu leurs traitements et état de santé, accès aux dernières publications scientifiques sur COVID-19 par les universités et les organisations de santé.

Le choix de cette technologie a pour but de créer un pont numérique d'information fiable entre ces différentes parties afin de garantir une haute sécurité et une haute confiance (authentification) grâce à sa décentralisation et la cryptographie asymétrique qui répond aux exigences d'authentification. C'est-à-dire une base de données décentralisée à l'échelle mondiale, où les informations peuvent être accessibles et partagées, en temps réels, au niveau international par un consortium des établissements médicaux, des professionnels de santé et des organisations mondiales tel que l'OMS (Organisation Mondial de la Santé), visant à identifier et limiter les fausses informations pour faciliter à comprendre le virus autant que possible.

Actuellement, la plupart des pays ont un système de notification des patients de Corona très passif dans lequel les laboratoires informent les hôpitaux et les cliniques diagnostiquent sur les résultats de l'analyse et ces derniers signalent les cas positifs aux autorités supérieures, qui à leur tour signalent les cas à l'autorité finale. Ce processus peut augmenter le délai de déclaration, car il existe de nombreux intermédiaires pour que le rapport passe de laboratoire à l'établissement final, ce qui peut perturber les statistiques et rendre difficile le traçage des chaînes d'infection.

En outre, l'utilisation d'un serveur central peut inévitablement entraîner des dommages plus importants si le système est exposé à une attaque de piratage. Il sera plus difficile de détecter les données modifiées après le piratage.

Si on forme un réseau basé sur la technologie blockchain entre tous les établissements impliquées dans la lutte contre l'épidémie COVID-19, on va améliorer les systèmes de notification des cas positifs et les données peuvent être automatiquement signalées à tous les établissements concernés et à l'autorité finale, en particulier, au même instant qu'elles sont stockées dans la blockchain, sans passer par un traitement intermédiaire. De plus, les données seraient totalement transparentes à toutes les parties du réseau ce qui rend impossible de les falsifier ou modifier.

Avec cette phase difficile, les échanges et les publications des informations et les actualités qui concerne ce virus sur les médias et les réseaux sociaux ont considérablement augmentés, et surtout sur les réseaux sociaux lorsqu'il s'agit de diffuser une information rapidement, mais malheureusement, cet outil n'est pas toujours utilisé correctement car ces derniers peuvent donner à n'importe qui le pouvoir de publier et partager n'importe quelle information et n'importe qui. Et cela conduit à la propagation des fausses informations.
La technologie blockchain pourrait être une des solutions sérieuses face à ce phénomène d'abord pour valider des conseils médicaux de qualité par les prestataires de santé et l'organisation mondiale de santé en particulier et reconnaître les sources de désinformation car celles-ci peuvent avoir des effets graves lors de cette crise. Cette technologie peut aussi servir un moyen de vérification des conseils de protection de qualité que le public devrait suivre, par rapport aux allégations fausses selon lesquelles le public devrait ignorer.

2.11. Défis de la blockchain dans le soin de santé

La blockchain est une technologie émergente, qui fournit de nombreux avantages et opportunités. Cependant, cette technologie s'accompagne de ses propres défis qui devraient être traités.

2.11.1. La capacité de stockage

La blockchain a été conçue pour enregistrer et traiter les données de transaction, qui ont une portée limitée, donc elle n'a pas besoin d'un stockage lourd. Mais lorsqu'elle déployait ses ailes dans le domaine des soins de santé, les défis du stockage sont devenus évidents.

Le secteur de la santé dispose d'une grande quantité de données qui doivent être traitées quotidiennement, des dossiers des patients, des antécédents médicaux et des rapports de test aux analyses IRM, aux radiographies et autres images médicales. Toutes les données, dans le scénario de la blockchain, seront disponibles pour tous les nœuds de la chaîne, ce qui nécessite un espace de stockage massif ce qui rend la vitesse de recherche et d'accès aux enregistrements faible [39].

2.11.2. La normalisation

La technologie de la blockchain est encore à ses balbutiements et, par conséquent, vers sa mise en œuvre pratique en médecine et en soins de santé, elle sera certainement confrontée à des défis de normalisation. Actuellement, il n'existe aucune norme pour le développement d'applications de santé basées particulièrement sur la blockchain. Par conséquent, les applications développées par différents fournisseurs ou sur différentes plates-formes peuvent être confrontées au problème de l'impossibilité d'interagir.
Ces normes prédéfinies seraient utiles pour évaluer la taille, la nature des données et le format des informations échangées dans les applications de blockchain. Ces normes examineront non seulement les données partagées, mais devront également servir de mesures de sécurité préventives [39].

2.11.3. Défis sociaux

La technologie de la blockchain est toujours en évolution et fait donc face à des défis sociaux, comme le changement culturel, en plus des défis techniques susmentionnés. Accepter et adopter une technologie complètement différente des méthodes de travail traditionnelles n'est jamais

facile. Bien que l'industrie médicale évolue lentement vers la numérisation, il reste encore beaucoup à faire pour passer complètement à cette technologie, en particulier à des technologies comme la blockchain, qui n'a pas encore été validée sur le plan clinique. Convaincre les médecins de passer de la paperasse à l'utilisation de la technologie prendra du temps et des efforts. En raison de son faible taux d'adoption dans le secteur de la santé, la technologie et les politiques proposées sont relativement peu fiables. En raison de tous ces défis et menaces, nous ne pouvons pas, à ce jour, le qualifier de solution viable et universelle pour tous les problèmes de santé [39].

2.12. Conclusion

La technologie blockchain est encore au stade expérimental et son potentiel de scalabilité reste encore à prouver mais elle est pleine de promesses, particulièrement dans la santé puisqu'elle répond aux enjeux majeurs (sécurité, transparence, partage des données dans le cadre d'un parcours de soin coordonné, …).

Elle peut jouer un rôle majeur dans l'instauration d'un cercle vertueux de la santé en simplifiant l'accès à nos données de santé pour améliorer prévention, suivi, recherche, etc. tout en garantissant le respect de notre vie privée.

Dans le chapitre suivant nous allons construire notre application blockchain à partir de zéros avec tous leurs fonctionnalités principales.

Chapitre 3

Conception et réalisation

3.1. Introduction

Dans ce chapitre, nous allons parler de l'environnement matériel et logiciel pour construire notre application et présenter les fonctions et les bibliothèques utilisées.

3.2. Outils de développement

3.2.1. Langage JavaScript

Le langage JavaScript (« JS » en abrégé) est un langage de programmation informatique, et plus précisément un langage de script orienté objet. Il a été inventé en 1995 par Brendan Eich, co-fondateur du projet Mozilla, de la Mozilla Foundation et de la Mozilla Corporation.

Tout d'abord, un langage de programmation est un langage qui permet aux développeurs d'écrire du code source qui sera analysé par l'ordinateur. Le code source est un ensemble d'actions, appelées instructions, qui vont permettre de donner des ordres à l'ordinateur afin de faire fonctionner le programme.

Le JavaScript est un langage qui intègre l'orienté objet dans sa définition même ce qui fait que tous les éléments du JavaScript vont soit être des objets, soit pouvoir être convertis et traités comme des objets.

Un objet, en informatique, est un ensemble cohérent de données et de fonctionnalités qui vont fonctionner ensemble. Pour le dire très simplement, un objet en JavaScript est un conteneur qui va pouvoir stocker plusieurs variables qu'on va appeler ici des propriétés. Lorsqu'une propriété contient une fonction en valeur, on appelle alors la propriété une méthode. Un objet est donc un conteneur qui va posséder un ensemble de propriétés et de méthodes qu'il est cohérent de regrouper et qui lui permet de pouvoir fonctionner et d'interagir avec d'autres objets [40].

3.2.2. Node.js

Node.js est un environnement d'exécution open source côté serveur qui exécute le code JavaScript en dehors d' un navigateur Web.

Node.js a été écrit initialement par Ryan Dahl en 2009 et son développement et sa maintenance sont effectués après par l'entreprise Joyent. Dahl a eu l'idée de prendre le moteur Javascript V8,

c'est celui qu'on trouve dans le navigateur Chrome, et de l'utiliser en dehors du navigateur, parce que tous les navigateurs sont équipés d'un moteur Javascript qui va permettre de traduire notre code Javascript en code machine [41].

Node.js peut être utilisé pour créer différents types d'applications telles que les applications de ligne de commande, les applications Web, les applications de chat en temps réel, les serveurs d'API (Application Programming Interface), etc. Cependant, il est principalement orienté vers les applications réseau événementielles hautement concurrentes qui doivent pouvoir monter en charge.

Les utilisateurs professionnels du logiciel Node.js incluent Groupon, Vivaldi, SAP, LinkedIn, Microsoft, Yahoo, Netflix, IBM, Sage et PayPal, etc [41].

3.2.3. Visual studio code

Visual Studio Code (« VSC » en abrégé) est un éditeur de code open-source gratuit, développé par Microsoft, présenté la première fois le 29 avril 2015 lors de la conférence Build 2015.

L'application, disponible à la fois pour plusieurs plateformes (Windows, Linux et mac) et se présente sous la forme d'un environnement multi langages de programmation grâce à un système d'extension bien fourni, mais il est principalement conçu pour le développement d'application avec JavaScript, TypeScript et Node.js.

VSC propose différents éléments qui peuvent être intéressants pour le développeur à tous niveau, si bien que, comparé à d'autres éditeurs de texte et représente un bon choix de départ pour les débutants parmi eux :

- IntelliSense : une technologie avancée qui propose, outre à la mise en évidence de la syntaxe et la complétion automatique du code, un système d'inférence articulé et basé directement sur la logique du code source.
- Débogueur intégré : est un logiciel qui aide le développeur à analyser les bugs (les défauts) d'un programme et exécuter le programme pas-à-pas c'est-à-dire ligne par ligne.
- Intégration native avec Git : VS Code s'interface par défaut avec le logiciel de gestion de versions Git, ce qui représente un avantage pour pouvoir effectuer les opérations de plus aisément.

- La langue de l'interface : VSC s'installe automatiquement dans la version anglaise du logiciel (car c'est plus simple de suivre la documentation officielle ou trouver de l'aide) mais il permet aux utilisateurs de la changer.
- Les utilisateurs peuvent aussi modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires [42].

3.2.4. Postman

Postman existe sous la forme d'une application (Windows/MacOS/Linux), c'est l'un des meilleurs outils qui permet d'interroger et/ou tester une API (Application Programming Interface). Il a la capacité de faire différents types de requêtes HTTP (GET, POST, PUT, PATCH) et de les stocker dans un historique afin de pouvoir les rejouer.

L'historique est certes pratique, mais n'est pas un modèle idéal pour organiser les requêtes. Postman propose pour cela une notion complémentaire appeler les "Collections qui permet d'organiser les requêtes en groupes (les collections) pour faciliter la navigation en cas de grand nombre de requêtes.

Ce logiciel aide le développeur à voir les codes d'état, le temps nécessaire pour la réponse et d'autres paramètres de performance, et il présente une interface graphique conviviale pour la construction des demandes et la lecture des réponses [43].

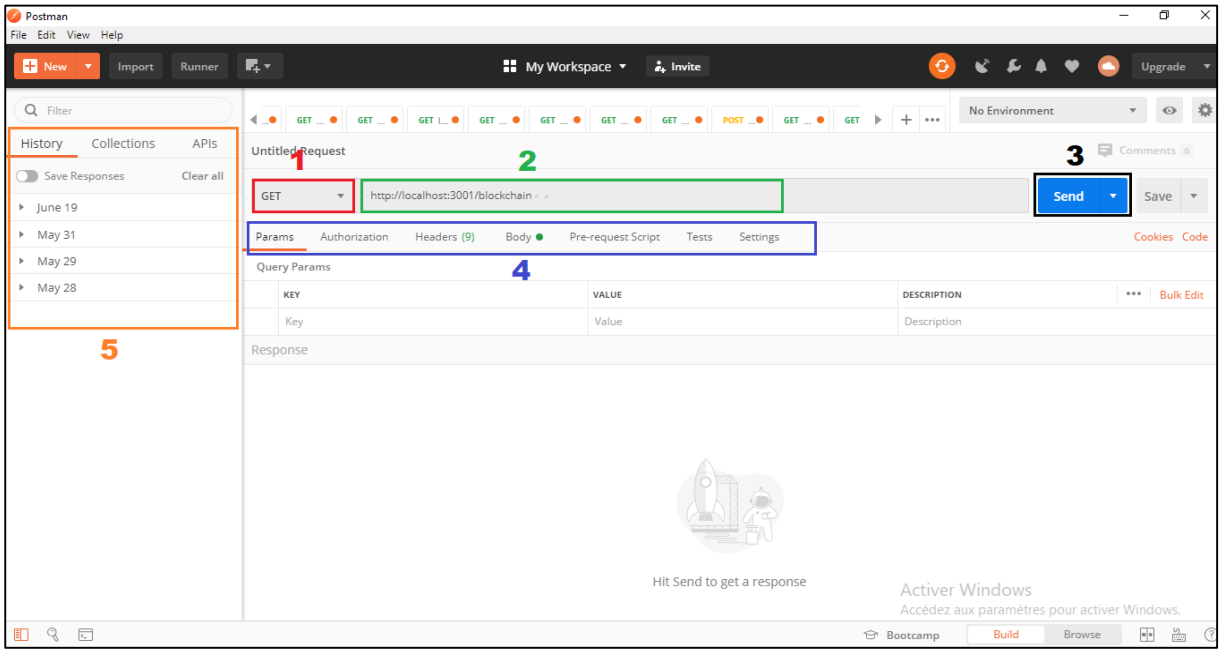

Figure 3.1 : La page de démarrage de Postman.

- **(1)** : Un bouton déroulant qui a tous les différents types de requête à envoyer : GET, POST, PUT, etc. Si vous souhaitez POSTER vers l'URL que vous avez spécifiée, sélectionnez POST.
- **(2)** : Le champ de saisie central le plus long qui ressemble à une barre de recherche est celui où l'URL de l'API.
- **(3)** : Le bouton Send qui est utilisé pour l'envoi de la demande au serveur ou à l'application dans notre cas.
- **(4)** : Ici se trouve tout ce qui constitue une requête http (header, body, etc.). Vous pouvez à partir de là créer votre requête de façon totalement personnalisée.
- **(5)** : Ici se trouve l'historique et les collections des connections que nous avons créées.

3.3. Mise en place du projet

Dans ce chapitre, on va construire notre propre blockchain et ça nous facilite à comprendre le fonctionnement réel de la technologie blockchain ainsi que le réseau décentralisé et tout ce qui concerne cette technologie (la création des blocs, la validation du blockchain, etc.).

Dans notre application, on va créer une blockchain entièrement fonctionnelle à partir de zéro en utilisant le langage de programmation JavaScript. La blockchain que nous allons créer aura des fonctionnalités similaires au niveau de la production à celles que nous trouverions dans une blockchain Bitcoin.

Nous allons commencer par construire la blockchain elle-même. À ce stade, nous allons construire une structure de données blockchain qui a les capacités suivantes :

- Prevue de travail.
- Extraction de nouveaux blocs.
- Création des données.
- Le hash des blocs.
- Validation de la blockchain et autres fonctionnalités.

Par la suite, nous créerons une API qui nous permettra d'interagir avec notre blockchain. Grâce à notre API, nous pourrons utiliser toutes les fonctionnalités que nous avons intégrées dans notre structure de données blockchain.

De plus, nous créerons un réseau décentralisé, cela signifie que nous aurons plusieurs serveurs fonctionnant et agissant comme des nœuds séparés. Nous veillerons également à ce que tous

les nœuds interagissent correctement et partagent les données les uns avec les autres correctement.

Nous allons ensuite passer à la création d'un algorithme qui sera utilisé pour synchroniser l'intégralité du réseau en assurant que tous les nouveaux nœuds ou données créés sont diffusés sur l'ensemble du réseau.

Et pour construire notre structure de code, il y a deux concepts cruciaux avec lesquels nous devons les utiliser. Ces concepts importants sont : le constructeur Function et le prototype d'objet.

3.3.1. Le constructeur Function JavaScript

Le constructeur Function est simplement une fonction qui crée une classe d'objets et vous permet de créer facilement plusieurs instances de cette classe particulière. Ce que cela signifie en réalité, c'est que le constructeur Function vous permet de créer beaucoup d'objets très rapidement. Tous ces objets créés auront les mêmes propriétés et fonctionnalités car ils font tous partie de la même classe [44].

3.3.2. Le prototype d'objet

L'objet prototype est simplement un objet auquel plusieurs autres objets peuvent se référer afin d'obtenir des informations ou fonctionnalités dont nous avons besoin [44].

3.3.3. API

En informatique, API est l'acronyme d'Application Programming Interface, que l'on traduit en français par interface de programmation applicative ou interface de programmation d'application. L'API est une interface informatique qui définit les interactions entre plusieurs intermédiaires logiciels. Elle définit les types d'appels ou de demandes qui peuvent être effectués, comment les effectuer, les formats de données à utiliser, les conventions à suivre, etc. et peut être résumée à une solution informatique qui permet aux développeurs de pouvoir interagir avec un programme sans avoir à se soucier du fonctionnement complexe d'une application [44].

3.4. Étapes de construction

Commençons par créer un dossier appelé « HealthChain ». Dans ce dossier, nous créons un répertoire appelé « dev » où nous allons faire la plupart de notre codage de programmation. C'est ici que nous allons construire nos données blockchain, structurer et créer notre API pour interagir avec notre blockchain.

À l'intérieur de ce répertoire de développement, nous allons créer deux fichiers, le premier s'appelle « blockchain.js », l'endroit où nous allons taper notre code pour créer la blockchain, et le deuxième fichier s'appelle « networkNode.js » pour l'API.

Ensuite, revenons à notre répertoire « HealthChain » en tapant la commande "npm init" dans le terminal, elle crée le fichier « package.json » pour nous. Ce fichier «. json » gardera une trace de notre projet et toutes les dépendances dont nous avons besoin, facilite l'installation par d'autres machines. Après avoir appuyé sur Entrée, on vous posera quelques questions sur notre projet : le nom du projet, la version, description, etc.

3.4.1. Structure de données Blockchain

Commençons par construire notre structure de données blockchain. Nous allons commencer par ouvrir tous les fichiers que nous avons dans notre répertoire « HealthChain » en utilisant l'éditeur "visual studio code". Ensuite, on va dans le fichier dev / blockchain.js que nous créons déjà, c'est là où nous allons coder notre blockchain.

3.4.1.1. Composition de la blockchain

```
function Blockchain () {
                this.chain = [];
                this.pendingDatas = [];
                this.createNewBlock (100, '0', '0');
                        };
```

Figure 3.2 : Mise en œuvre de la composition blokchain.

Dans le bloc de code précédent, on a déclaré une fonction constructeur nommé "Blockchain", à l'intérieur de laquelle, on a défini un tableau « this.chain = [] », l'endroit où notre blockchain sera stockée. Tous les blocs que nous extrayons seront stockés dans ce tableau particulier sous forme d'une chaîne, tandis que le tableau « this.pendingDatas = [] » constitue l'endroit où nous conserverons toutes les nouvelles données qui sont créées périodiquement avant qu'elles ne soient placées dans un bloc.

Encore une chose que nous devrions ajouter à notre structure c'est le bloc genèse (le premier bloc dans la blockchain). Pour le créer, nous allons utiliser la méthode createNewBlock qu'on va la construire par la suite, et puis nous avons passé la valeur nonce comme 100, previousBlockHash comme 0 et la valeur de hachage comme 0. Ce ne sont que des valeurs arbitraires.

3.4.1.2. Construction de la méthode createNewBlock

Après avoir défini notre fonction constructeur "Blockchain" dans la section précédente, la prochaine étape que nous voulons faire avec notre fonction constructeur est de placer une méthode dans notre fonction Blockchain. Cette méthode que nous allons créer s'appellera createNewBlock. Comme son nom l'indique, cette méthode créera un nouveau bloc pour nous.

```
Blockchain.prototype.createNewBlock = function(nonce, previousBlockHash, hash) {
        const newBlock = {
                index: this.chain.length + 1,
                timestamp: Date.now(),
                datas: this.pendingDatas,
                nonce: nonce,
                hash: hash,
                previousBlockHash: previousBlockHash
                        };
        this.pendingDatas = [];
        this.chain.push(newBlock);
        return newBlock;
    };
```

Figure 3.3 : Mise en œuvre de la méthode createNewBlock.

Dans cette méthode, on a créé un objet nommé "newBlock", qui va être un nouveau bloc à l'intérieur de notre Blockchain. Et sur cet objet, nous allons avoir des propriétés :

- **L'index** : Cette valeur d'index sera essentiellement le numéro de bloc. Il décrira le nombre de blocs du newBlock dans notre blockchain.
- **Timestamp** : cette propriété va être un horodatage, car nous voulons savoir quand le bloc a été créé.
- **Datas** : Lorsque nous créons un nouveau bloc, nous voulons mettre toutes les nouvelles données ou les données en attente qui viennent d'être créées dans le nouveau bloc afin qu'elles soient à l'intérieur de notre blockchain et ne puissent jamais être modifiées.
- **Nonce** : il s'agit simplement de n'importe quel nombre peu importe lequel. Ce nonce est à peu près la preuve que nous avons créé ce nouveau bloc de manière légitime en utilisant une méthode proofOfWork.
- **Hash** : cette propriété signifie que toutes les données de notre bloc vont être compressées en une seule chaîne de caractère, qui sera notre hachage.
- **previousBlockHash** : Cette propriété est très similaire à notre propriété de hachage, sauf qu'elle traite les données de notre bloc précédent vers le bloc actuel haché dans une chaine de caractère.

La prochaine chose que nous voulons faire dans notre méthode est de définir "this.pendingDatas " comme égal à un tableau vide. Nous faisons cela car, une fois que nous créons notre nouveau bloc, nous mettons toutes les données en attente dans le newBlock. Par conséquent, nous voulons effacer tout le nouveau tableau de datas afin de pouvoir recommencer pour le bloc suivant. Ensuite, ce que nous voulons simplement faire, c'est prendre le nouveau bloc que nous avons créé et le pousser dans notre chaîne, puis nous allons retourner le newBlock.

3.4.1.3. Construction de la méthode getLastBlock

Maintenant, la prochaine méthode que nous allons ajouter à notre fonction constructeur Blockchain sera le getLastBlock. Cette méthode nous renverra simplement le dernier bloc de notre blockchain.

```
Blockchain.prototype.getLastBlock = function () {
        return this.chain[this.chain.length - 1];   //la position du bloc dans la chaine
}
```

Figure 3.4 : Mise en œuvre de la méthode getLastBlock.

" [This.chain.length - 1] " dans le code précédent définit la position du bloc dans la chaîne, qui est, dans notre cas, le bloc précédent.

3.4.1.4. Construction de la méthode createNewData

La prochaine méthode que nous allons ajouter à notre fonction constructeur Blockchain s'appelle createNewData. Cette méthode va créer une nouvelle data pour nous.

```
Blockchain.prototype.createNewData = function (infoofpatient, infoofdoctor, infoofestablishment) {

        const newData = {

infoOfDoctor: { LastName : infoOfDoctor.LastName, FirstName : infoOfDoctor.FirstName, Gender : infoOfDoctor.Gender, Specialization :infoOfDoctor.Specialization, WorkPlace :infoOfDoctor.WorkPlace, EmailAddress : infoOfDoctor.EmailAddress,PhoneNumber : infoOfDoctor.PhoneNumber },

infoOfEstablishment: { TypesOfEstablishments : infoOfEstablishment.TypesOfEstablishments, CatégoriesOfEstablishments : infoOfEstablishment.CatégoriesOfEstablishments, OfficialName : infoOfEstablishment.OfficialName, PlaceAddress : infoOfEstablishment.PlaceAddress, EmailAddress : infoOfEstablishment.EmailAddress, PhoneNumber : infoOfEstablishment.PhoneNumber},


infoOfPatient: { LastName : infoOfPatient.LastName, FirstName : infoOfPatient.FirstName, Gender : infoOfPatient.Gender, DateOfBirth : infoOfPatient.DateOfBirth, Age : infoOfPatient.Age, Job : infoOfPatient.Job, Relationship : infoOfPatient.Relationship, PlaceAddress : infoOfPatient.PlaceAddress, PhoneNumber : infoOfPatient.PhoneNumber, BloodGroup : infoOfPatient.BloodGroup, Allergieslist : infoOfPatient.Allergieslist,ChronicDiseases : infoOfPatient.ChronicDiseases, AnalysisResults : infoOfPatient.AnalysisResults, Vaccines : infoOfPatient.Vaccines, Diagnostics : infoOfPatient.Diagnostics, Prescription : infoOfPatient.Prescription, HospitalizationReport : infoOfPatient.HospitalizationReport, Remark : infoOfPatient.Remark},

        dataId: uuid().split('-').join('')

         };

         return newData;

};
```

Figure 3.5 : Mise en œuvre de la méthode createNewData.

Dans cette méthode, on a créé un objet nommé "newData", qui va être une nouvelle donnée à l'intérieur de notre Blockchain. Toutes les données enregistrées sur notre Blockchain auront les mêmes propriétés :

- **Infoofpatient** : ce paramètre indique les informations qui concerne le patient.
- **Infoofdoctor** : ce paramètre indique les informations qui concerne le médecin qui prend en charge le patient.
- **Infoofestablishment** : ce paramètre indique les informations qui concerne l'hôpital ou la clinique où le patient fait sa visite.
- **dataId** : un identifiant est ajouté à chaque data créé. Pour créer cet identifiant, une chaîne de caractères unique est utilisée avec l'utilisation de la bibliothèque uuid.

```
npm install uuid
```

Par conséquent, au début du fichier dev / blockchain.js, où toutes les constantes sont définies, on ajoute la ligne de code suivante afin d'utiliser la bibliothèque uuid :

```
const uuid = require('uuid/v1');
```

3.4.1.5. Construction de la méthode addDataToPendingDatas

```
Blockchain.prototype.addDataToPendingDatas = function(dataObj) {
        this.pendingDatas.push(dataObj);
        return this.getLastBlock()['index'] + 1;

};
```

Figure 3.6 : Mise en œuvre de la méthode addDataToPendingDatas.

Prenons ici le "dataObj" et poussons-le dans le tableau "pendingDatas" de la blockchain. Ensuite, nous voulons simplement renvoyer l'index du bloc auquel le data est ajoutée.

3.4.1.6. Construction de la méthode hashBlock

La méthode suivante que nous allons examiner et ajouter à notre structure de données blockchain est appelée hashBlock. La méthode hashBlock prend un bloc de notre blockchain et hache ses données dans une chaîne de caractères de longueur fixe.

```
Blockchain.prototype.hashBlock = function(previousBlockHash, currentBlockData, nonce) {
    const dataAsString = previousBlockHash + nonce.toString() + JSON.stringify(currentBlockData);
    const hash = sha256(dataAsString);
    return hash;
};
```

Figure 3.7 : Mise en œuvre de la méthode hashBlock.

À l'intérieur de cette méthode, nous voulons utiliser le hachage SHA256 pour hacher nos données de bloc. Et pour utiliser cette fonction de hachage SHA256, on va l'importez en tant que bibliothèque npm. Pour ce faire, nous devrons taper la commande suivante dans notre terminal :

```
npm install  sha 256 --save
```

Maintenant, à l'intérieur de notre structure de fichiers « HealthChain », nous pouvons voir que le dossier node_modules est apparu. A l'intérieur de ce dossier se trouve notre bibliothèque SHA256 et toutes les autres dépendances ont été téléchargées.

Pour utiliser cette bibliothèque, nous devrons importer-la dans notre code afin de pouvoir l'utiliser. Au tout début de notre code, tapez la ligne suivante :

```
const sha256 = require ('sha256');
```

Les données que nous allons hacher à l'intérieur de notre méthode hashBlock sont : previousBlockHash, currentBlockData et nonce. Et avant les hacher, nous voulons changer tous ces éléments de données en une seule chaîne et renvoyer le hachage au final.

3.4.1.7. Construction de la méthode proofOfWork

Construisons la méthode proofOfWork, cette méthode est très importante et essentielle pour la sécurité de notre blockchain.

```
Blockchain.prototype.proofOfWork = function(previousBlockHash, currentBlockData) {
        let nonce = 0;
        let hash = this.hashBlock(previousBlockHash, currentBlockData, nonce);
        while (hash.substring(0, 4) !== '0000') {
                nonce++;
                hash = this.hashBlock(previousBlockHash, currentBlockData, nonce);
        }
        return nonce;
};
```

Figure 3.8 : Mise en œuvre de la méthode proofOfWork.

La première chose que nous voulons faire à l'intérieur de cette méthode est de définir un nonce et de l'initialiser par zéros. Ensuite, nous voulons hacher toutes nos données pour la première fois, en appelant la méthode " hashBlock ". La prochaine étape consiste à répéter cette opération à l'aide d'une boucle while jusqu'à ce que nous obtenions un hachage commençant par quatre zéros.

3.4.2. Accéder à la Blockchain via une API

Maintenant, on va construire une API qui nous permettra d'interagir avec notre structure de données blockchain. Nous allons construire notre API dans le fichier « networkNode.js » que nous placerons dans notre dossier dev.

3.4.2.1. Configuration d'Express.js

Express.js est un framework Node.js qui implémente une couche de fonctionnalités nécessaires pour créer notre API. Nous devons l'installer en tant que bibliothèque dépendance, donc on tape la commande suivante dans notre terminal :

```
npm install express --save
```

Express.js a des fonctions prêtes à l'emploi pour le traitement des requêtes HTTP, et chaque méthode HTTP a sa propre fonction, ce qui est particulièrement pratique lors de la création d'une API. Et c'est loin d'être la seule raison d'utiliser Express [45].

Pour utiliser cette bibliothèque, nous devrons importer-la dans notre code afin de pouvoir l'utiliser. Au tout début de notre code, tapez la ligne suivante :

```
const express = require('express');
const app = express ();
```

Figure 3.9 : Implémentation de la bibliothèque Epress.js.

3.4.2.2. Installation de package nodemon

Dans cette section, nous allons installer un nouveau paquet appelé "nodemon". Dans notre répertoire « HealthChain », nous écrirons la commande suivante :

```
npm install nodemon --save
```

Cette bibliothèque nodemon redémarrera automatiquement notre serveur pour nous afin que nous apportons une modification à l'un de nos fichiers et n'ayons pas à aller et venir du terminal à notre code pour redémarrer le serveur à chaque fois que nous faisons un changement.

3.4.2.3. Installation de package body-parser

Maintenant, nous devons installer une autre bibliothèque appelée "body-parser". Revenons dans notre terminal et tapons la commande suivante :

```
npm install body-parser --save
```

Ce body-parser est un analyseur de corps des données codées JSON, soumises à l'aide de la demande HTTP POST. Pour lire les données HTTP POST, nous devons utiliser le module

"body-parser", qui est un élément partie d'express.js et qui lit l'entrée d'un formulaire et le stocke sous forme d'objet javascript accessible via req.body.

Quand il s'agit d'utiliser body-parser, nous voulons simplement l'importer en haut de notre fichier :

```
const express = require('express');
const app = express ();
const bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
```

Figure 3.10 : Implémentation de l'analyseur body-parser.

3.4.2.4. Construction du point de terminaison / blockchain

Dans cette section, nous allons interagir avec notre point de terminaison / blockchain, cela signifie que nous devrons importer notre blockchain depuis notre fichier « blockchain.js ». Ensuite, nous voulons créer une instance de notre blockchain appelée "healthblockchain". Nous pouvons le faire comme suit :

```
const healthblockchain = new Blockchain ();
```

Tout ce que ce point de terminaison va faire est de renvoyer l'intégralité de notre blockchain à celui qui a appelé.

```
app.get ('/blockchain', function(req, res) {
    res.send(healthblockchain);
    });
```

Figure 3.11: Mise en œuvre du point de terminaison / blockchain.

3.4.2.5. Construction du point de terminaison / data

Dans cette section, nous allons construire notre point de terminaison data, pour créer une nouvelle donnée dans notre blockchain. Nous allons utiliser notre méthode " addDataToPendingDatas " dans le fichier blockchain.js que nous avons déjà créé.

```
    app.post ('/data', function (req, res) {
        const newData = req.body;
        const blockIndex = healthblockchain.addDataToPendingDatas(newData);
        res.json({ note: `Data will be added in block ${blockIndex}.` });
});
```

Figure 3.12: mise en œuvre du point de terminaison / data.

On ajoute une nouvelle donnée au tableau "pendingDatas" du nœud qui reçoit l'appel. Pour ce faire, la méthode "addDataToPendingDatas" sera utilisée. Maintenant, à partir de cette dernière méthode, nous obtenons l'index du bloc dans lequel le data sera ajouté.

3.4.2.6. Construction du point de terminaison / mine

Construisons le point de terminaison final pour notre API blockchain, le point de terminaison de la mine, cela va extraire et créer un nouveau bloc.

Pour créer un nouveau bloc, nous allons utiliser notre méthode "createNewBlock", que nous avons déjà définie dans notre blockchain.js ainsi que la méthode " getLastBlock ", " proofOfWork "et " hashBlock ".

```
app.get ('/mine', function(req, res) {
  const lastBlock = healthblockchain.getLastBlock();
  const previousBlockHash = lastBlock['hash'];
  const currentBlockData = {
      datas: healthblockchain.pendingDatas,
       index: lastBlock['index'] + 1
      };
  const nonce = healthblockchain.proofOfWork(previousBlockHash, currentBlockData);
  const blockHash = healthblockchain.hashBlock(previousBlockHash, currentBlockData, nonce);
  const newBlock = healthblockchain.createNewBlock(nonce, previousBlockHash, blockHash);


  });
```

Figure 3.13: Mise en œuvre du point de terminaison / blockchain.

Commençons par créer "lastBlock" afin de récupérer le bloc précédent pour pouvoir obtenir son hach. Ensuite, prenons notre nonce. Pour produire un nonce pour notre bloc, nous devons générer un proofOfWork.

Nous avons notre currentBlockData comme objet, et cela se compose des données. Ces données seront simplement constituées des données de ce bloc, ainsi que d'un index, qui est l'index du nouveau bloc que nous allons créer.

Ce nouveau bloc doit prendre un hach, donc nous allons appeler la méthode hashBlock pour calculer et enregistrer les résultats dans une variable appelée blockHash.

3.4.3. Création d'un réseau Blockchain décentralisé

Dans le monde réel, toute la technologie blockchain est hébergée sur un réseau décentralisé. Dans cette partie, c'est sur ce concept que nous allons nous concentrer. Nous allons construire un réseau de blockchain décentralisé en créant diverses instances de l'API. Chacune de ces instances de l'API va être un nœud de réseau dans notre réseau blockchain. Tous ces nœuds fonctionneront ensemble pour héberger notre blockchain.

3.4.3.1. Création de plusieurs nœuds

Pour configurer le réseau décentralisé, nous devrons exécuter le fichier « networkNode.js » plusieurs fois. Chaque fois que nous l'exécutons, nous voulons qu'il agisse comme un nœud de réseau différent. Faisons-le en exécutant le fichier sur différents ports et pour avoir une valeur de ports différente à chaque fois, il faut faire du port une variable. Pour ce faire, ajoutons la ligne suivante au début du code dans notre dev / networkNode.js :

```
const port = process.argv [2] ;
```

Ensuite, nous allons en bas, où nous avons mentionné le code suivant :

```
app.listen (port, function() {
    console.log (`Listening on port ${port}...`);
});
```

Figure 3.14 : Mise en œuvre du la variable de port.

Accédons maintenant au fichier « package.json » et apportons des modifications à la commande "start". Premièrement, on change la commande "start" en "node_1", ensuite nous allons à la fin de notre commande et passons une variable pour le numéro de port sur lequel nous voulons qu'un nœud de réseau fonctionne. Dans notre exemple, nous voulons exécuter notre nœud de réseau pour qu'il s'exécute sur le port numéro 3001.

```
"node_1": "nodemon --watch dev -e js dev/networkNode.js 3001"
```

Pour notre réseau décentralisé, nous voulons exécuter cinq nœuds en même temps. Donc, on ajoute d'autres commandes similaires à "node_1". Pour ce faire, dupliquez la commande précédente quatre fois de plus, puis apportez des modifications à ces commandes.

3.4.3.2. Ajout de currentNodeUrl

La prochaine chose que nous allons faire est de modifier légèrement les commandes de notre package.json. La raison pour laquelle nous allons faire cela est que nous voulons que chacun

de nos nœuds de réseau sache sur quelle URL il se trouve actuellement. Par conséquent, nous voulons que chaque nœud soit conscient de l'URL sur laquelle il est hébergé.

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "node_1": "nodemon --watch dev -e js dev/networkNode.js 3001 http://localhost:3001",
  "node_2": "nodemon --watch dev -e js dev/networkNode.js 3002 http://localhost:3002",
  "node_3": "nodemon --watch dev -e js dev/networkNode.js 3003 http://localhost:3003",
  "node_4": "nodemon --watch dev -e js dev/networkNode.js 3004 http://localhost:3004",
  "node_5": "nodemon --watch dev -e js dev/networkNode.js 3005 http://localhost:3005"
},
```

Figure 3.15 : Capture indique l'URL des nœuds.

Nous devons maintenant attribuer la méthode "currentNodeUrl" à notre structure de données Blockchain. Ensuite, nous voulons également que notre blockchain soit consciente de tous les autres nœuds qui se trouvent à l'intérieur de notre réseau. Par conséquent, nous ajouterons un tableau nommé "networkNodes" dans notre fonction Blockchain.

```
function Blockchain () {
        this.chain = [];
        this.pendingDatas = [];
        this.currentNodeUrl = currentNodeUrl;
        this.networkNodes = [];
        this.createNewBlock (100, '0', '0');
```

Figure 3.16: Méthodes ajoutées à la fonction Blockchain.

3.4.3.3. Construction du point de terminaison /register-and-broadcast-node

La fonction de ce point d'extrémité sera d'enregistrer l'URL du nouveau nœud sur son propre nœud puis de diffuser le vers tous les autres nœuds déjà présents dans le réseau.

```
app.post ('/register-and-broadcast-node', function (req, res) {
    const newNodeUrl = req.body.newNodeUrl;
    if (healthblockchain.networkNodes.indexOf(newNodeUrl) == -1)
healthblockchain.networkNodes.push(newNodeUrl);
    const regNodesPromises = [];
    healthblockchain.networkNodes.forEach(networkNodeUrl => {
    const requestOptions = {
        uri: networkNodeUrl + '/register-node',
        method: 'POST',
        body: { newNodeUrl: newNodeUrl },
        json: true
    };
        regNodesPromises.push(rp(requestOptions));
});
Promise.all(regNodesPromises)
.then(data => {
        const bulkRegisterOptions = {
        uri: newNodeUrl + '/register-nodes-bulk',
        method: 'POST',
        body: {allNetworkNodes:
[...healthblockchain.networkNodes,healthblockchain.currentNodeUrl]},
        json:true
        };
return rp (bulkRegisterOptions);
})
.then (data => {
res.json({ note: 'New node registered with network successfully.' });
});
});
```

Figure 3.17: mise en œuvre du point de terminaison /register-and-broadcast-node.

La première chose que nous faisons à l'intérieur de ce point de terminaison est de définir une variable appelée "newNodeUrl", et ces données "newNodeUrl" seront transmises au corps de la demande, après, on prend le "newNodeUrl " et on l'enregistre avec le nœud actuel en le poussant dans notre tableau "networkNodes ". Nous ne voulons faire cela que si le "newNodeUrl " n'est pas déjà présent dans le tableau. Vérifions cela à l'aide de l'instruction if.

La prochaine étape que nous devons faire est de diffuser ce "newNodeUrl " au reste des nœuds de notre réseau. Nous faisons cela à l'intérieur de la boucle "forEach ". Tout ce qui se passe à l'intérieur de cette boucle for, c'est que nous faisons une demande à chacun des autres nœuds de notre réseau. Cela se fait en important une nouvelle bibliothèque appelée "request-promise" :

```
npm install request-promise --save
```

Et importons la bibliothèque que nous venons de télécharger à notre fichier dev / networkNode.js en tapant la ligne de code suivante au début :

```
const rp = require('request-promise');
```

La demande précédente va nous retourner une promesse, et nous voulons récupérer toutes ces promesses dans un seul tableau. Alors on pousse toutes ces demandes dans notre tableau de promesses "regNodesPromises" de nœud de registre, puis nous exécutons simplement toutes ces demandes.

Une fois que toutes ces demandes sont terminées sans aucune erreur, nous pouvons supposer que le newNodeUrl a été enregistré avec succès avec tous nos autres nœuds de réseau.

3.4.3.4. Construction du point de terminaison /register-node

Dans cette section, commençons-nous à créer le point de terminaison "register-node". Ce point de terminaison est l'endroit où chaque nœud du réseau va recevoir la diffusion qui est envoyée par notre point de terminaison "register-andbroadcast-node". La seule chose que ce point de terminaison de nœud de registre doit faire est d'enregistrer l'URL de nouveau nœud avec le nœud qui en reçoit la demande.

```
app.post('/register-node', function (req,res) {
    const newNodeUrl = req.body.newNodeUrl;
    const nodeNotAlreadyPresent = healthblockchain.networkNodes.indexOf(newNodeUrl) == -1;
    const notCurrentNode = healthblockchain.currentNodeUrl !== newNodeUrl;
   if (nodeNotAlreadyPresent && notCurrentNode )
healthblockchain.networkNodes.push(newNodeUrl);
    res.json({ note: 'New noderegistered successfully.' });
    });
```

Figure 3.18: Mise en œuvre du point de terminaison /register-node.

La première chose que nous faisons à l'intérieur de ce point de terminaison est de définir la variable "newNodeUrl" qui est envoyée à req.body. Ce sont les données que nous envoyons au point de terminaison / register-node. Ensuite, tout ce que nous ferons ici est poussez simplement le "newNodeUrl" dans le tableau "networkNodes" du nœud actuel à l'aide de la commande "push".

Ensuite, nous voulons effectuer une gestion des erreurs à l'intérieur de ce point de terminaison, c'est à dire ajouter le nouveau URL à notre tableau "networkNodes", s'il n'existe pas déjà dans ce tableau et aussi si le "newNodeUrl" est en fait l'URL du nœud actuel sur lequel nous sommes.

3.4.3.5. Construction du point de terminaison /register-nodes-bulk

Le prochain point de terminaison que nous allons construire est notre point de terminaison register-nodes-bulk. Chaque fois qu'un nouveau nœud est diffusé vers tous les autres nœuds à l'intérieur du réseau, nous voulons prendre tous les nœuds qui sont déjà à l'intérieur du réseau et renvoyons ces données à notre nouveau nœud afin que le nouveau nœud puisse s'enregistrer et reconnaître tous des nœuds déjà présents à l'intérieur du réseau.

```
app.post ('/register-nodes-bulk', function (req, res) {
        const allNetworkNodes = req.body.allNetworkNodes;
        allNetworkNodes.forEach(networkNodeUrl => {
        const nodeNotAlreadyPresent =
healthblockchain.networkNodes.indexOf(networkNodeUrl) == -1;
        const notCurrentNode = healthblockchain.currentNodeUrl !== networkNodeUrl;
                if (nodeNotAlreadyPresent && notCurrentNode)
healthblockchain.networkNodes.push(networkNodeUrl);
        });
```

Figure 3.19: Mise en œuvre du point de terminaison /register-node-bulk.

Pour construire le point de terminaison register-nodes-bulk, nous devrons supposer que toutes les URL de nœuds qui sont actuellement dans notre réseau sont transmises en tant que données et que nous pouvons y accéder sur le req.body.allNetworkNodes propriété.

Maintenant, tout ce que nous allons faire à l'intérieur de la boucle "forEach" est d'enregistrer chaque URL de nœud de réseau avec le nœud actuel sur lequel nous sommes, qui est le nouveau nœud ajouté au réseau.

Il existe maintenant quelques conditions dans lesquelles nous ne souhaitons pas ajouter de networkNodeUrl à notre tableau networkNodes. Pour gérer ces instances, nous allons utiliser une instruction if. Une raison pour laquelle nous ne voudrions pas ajouter un networkNodeUrl à notre tableau networkNodes est si ce networkNodeUrl existe déjà dans notre tableau networkNodes, la deuxième condition, est si ce nœud de réseau à la même URL que le nœud de réseau sur lequel nous sommes actuellement.

3.4.4. Synchronisation du réseau

Jusqu'à maintenant, nous avons construit un réseau composé de cinq nœuds. Chaque nœud était conscient de tous les autres nœuds du réseau, ce qui a créé un réseau blockchain décentralisé. Nous devons maintenant créer un réseau synchronisé, de sorte que la blockchain sur chaque nœud soit la même et que les données soient cohérentes partout. Nous ne pouvons pas nous

permettre d'avoir différentes versions des blockchains fonctionnant sur différents nœuds, car cela détruirait totalement l'objectif d'avoir une blockchain.

3.4.4.1. Construction du point de terminaison / data / broadcast

Dans cette partie, construisons un nouveau point de terminaison appelé / data / broadcast dans notre fichier dev/networkNode.js. Chaque fois que nous voulons créer une nouvelle donnée à partir de maintenant, nous allons atteindre ce point de terminaison. Ce point de terminaison fera deux choses, créera une nouvelle data et diffusera ensuite à tous les autres nœuds du réseau.

```
app.post ('/data/broadcast', function(req, res) {
        const newData = healthblockchain.createNewData(req.body.infoofpatient,
req.body.infoofdoctor, req.body.infoofestablishment);
        healthblockchain.addDataToPendingDatas(newData);
        const requestPromises = [];
        healthblockchain.networkNodes.forEach(networkNodeUrl => {
                const requestOptions = {
                        uri: networkNodeUrl + '/data',
                        method: 'POST',
                        body: newData,
                        json: true
                };
                requestPromises.push(rp(requestOptions));
        });
        Promise.all(requestPromises)
        .then(data => {
                res.json({ note: 'Data created and broadcast successfully.' });
        });
});
```

Figure 3.20: Mise en œuvre du point de terminaison /data/broadcast.

La méthode "createNewData" prend en compte ces paramètres. Pour notre point de terminaison, supposons que toutes ces données sont envoyées sur le req.body. Par conséquent, ces paramètres seront définis comme indiquer dans le code. Ensuite, ajoutons la variable "newData" au tableau "pendingDatas" sur le nœud à l'aide de la méthode "addDataToPendingDatas".

Maintenant, diffuson les nouvelles données à tous les autres nœuds du réseau. Cela fait dans la boucle "forEach" après la définition du tableau " requestPromises".

3.4.4.2. Refactoriser le point de terminaison / mine

Accédons-nous au fichier dev / networkNode.js. Dans le point de terminaison / mine, sous la partie où nous avions défini la variable "newBlock", ajoutons la fonctionnalité pour diffuser le nouveau bloc créé à tous les autres nœuds du réseau.

```
const requestPromises = [];
   healthblockchain.networkNodes.forEach(networkNodeUrl => {
      const requestOptions = {
         uri: networkNodeUrl + '/receive-new-block',
         method: 'POST',
         body: { newBlock: newBlock },
         json: true
         };
    requestPromises.push(rp(requestOptions));
     })
    Promise.all(requestPromises)
  .then(data => {
         res.json({
            note: "New block mined & broadcast successfully",
            block: newBlock
         });
      });
```

Figure 3.21 : mise à jour du point de terminaison /mine.

Le code bloc précédent mentionne que pour chacun des "networkNodes", nous allons faire une requête et envoyer le "newBlock" grâce à la boucle "forEach" qu'elle a certaines options de demande et le nouveau point de terminaison, qui sera / receive-new-block. Nous travaillerons sur ce point de terminaison dans la section suivante.

Chaque fois qu'une de ces demandes est faite, elle retourne une promesse. Faisons donc un tableau de toutes ces promesses en ajoutant avant la boucle "forEach".

3.4.4.3. Construire le point de terminaison / receive-new-block

La prochaine chose que nous allons faire est de créer le point de terminaison / receive-new-block que nous utilisons dans le point de terminaison / mine.

```
app.post ('/receive-new-block', function(req, res) {
    const newBlock = req.body.newBlock;
    const lastBlock = healthblockchain.getLastBlock();
    const correctHash = lastBlock.hash === newBlock.previousBlockHash;
    const correctIndex = lastBlock['index'] + 1 === newBlock['index'];

    if (correctHash && correctIndex) {
        healthblockchain.chain.push(newBlock);
        healthblockchain.pendingDatas = [];
        res.json({
            note: 'New block received and accepted.',
            newBlock: newBlock
        });
    } else {
        res.json({
            note: 'New block rejected.',
            newBlock: newBlock
        });
    }
});
```

Figure 3.22 : Mise en œuvre du point de terminaison / receive-new-block.

À l'intérieur de ce point de terminaison, le code s'attend à recevoir un nouveau bloc en cours de diffusion. Sauvegardons ce nouveau bloc dans une variable "newBlock". Lorsque tous les autres nœuds reçoivent ce nouveau bloc, ils doivent vérifier s'il s'agit réellement d'un bloc réel et s'il s'intègre correctement dans la chaîne. Pour vérifier cela, le previousBlockHash sur le newBlock est vérifié pour s'assurer qu'il est égal au hachage sur le dernier bloc de la chaîne, on utilise ici la methode "getLastBlock".

Après la vérification précédente, nous voulons également nous assurer que le newBlock a l'index correct. Cela signifie que le newBlock doit être un index au-dessus du LastBlock de la chaîne.

Nous utilisons donc une instruction if-else qu'elle doit être acceptée et ajouté le bloc à la chaîne, si non, il devrait tout simplement être rejeté. Et quand le newBlock a été ajouté à la chaîne, le tableau pendingDatas doit être effacé, car les données en attente sont maintenant à l'intérieur du nouveau bloc.

3.4.5. Construire l'algorithme de consensus

Dans cette section, nous allons construire un algorithme de consensus pour notre réseau blockchain. Un algorithme de consensus est un moyen pour tous les nœuds à l'intérieur du réseau de s'entendre sur les données correctes et doivent être conservées à l'intérieur de la blockchain. Afin de construire l'algorithme de consensus, nous allons d'abord construire une nouvelle méthode, appelée "chainIsValid". Cette méthode validera simplement une blockchain en comparant tous les hachages de tous les blocs à l'intérieur de la chaîne. Après cela, nous allons construire un point de terminaison / consensus que nous atteindrons chaque fois que nous voudrons utiliser l'algorithme de consensus.

L'algorithme de consensus nous fournira un moyen pour confirmer que nous avons les données correctes sur un nœud spécifique. Il existe actuellement de nombreux algorithmes de consensus différents utilisés pour différents réseaux de blockchain. Pour notre réseau, nous allons créer un algorithme de consensus qui implémente la règle de la plus longue chaîne.

La théorie derrière l'utilisation de cela est que nous devrions être en mesure de faire confiance à la plus longue chaîne pour tenir les données correctes, parce que le plus de travail a été mis dans la création de cette chaîne. La plus longue chaîne a le plus de blocs en elle et chacun de ces blocs a été miné en utilisant une preuve de travail. Par conséquent, nous pouvons supposer que tout le réseau a contribué à la plus longue chaîne en raison de la quantité de travail qui a été consacrée à cette chaîne. Pour cette raison, nous allons utiliser un algorithme de consensus qui met en œuvre la règle de la chaîne la plus longue.

3.4.5.1. Construire la méthode chainIsValid

Commençons par construire l'algorithme de consensus en créant une nouvelle méthode a l'intérieur de notre fichier « blockchain.js » appelée "chainIsValid". Cette méthode validera si une chaîne est légitime ou non.

```
Blockchain.prototype.chainIsValid = function(blockchain) {
   let validChain = true;
        for (var i = 1; i < blockchain.length; i++) {
         const currentBlock = blockchain[i];
         const prevBlock = blockchain[i - 1];
         const blockHash = this.hashBlock(prevBlock['hash'], { datas: currentBlock['datas'],
 index: currentBlock['index'] }, currentBlock['nonce']);
       if (blockHash.substring(0, 4) !== '0000') validChain = false;
       if (currentBlock['previousBlockHash'] !== prevBlock['hash']) validChain = false;
};
   //vérifier le genesis block
       const genesisBlock = blockchain [0];
       const correctNonce = genesisBlock['nonce'] === 100;
       const correctPreviousBlockHash = genesisBlock ['previousBlockHash'] === '0';
       const correctHash = genesisBlock ['hash'] === '0';
       const correctDatas = genesisBlock ['datas'].length === 0;

 if (!correctNonce || !correctPreviousBlockHash || !correctHash || !correctDatas) validChain = false
;
     return validChain;
};
```

Figure 3.23 : Mise en œuvre de la méthode chainIsValid.

Afin de valider que la blockchain est légitime, nous allons simplement parcourir chaque bloc à l'intérieur de la blockchain et vérifier si tous les hachages s'alignent correctement ou non, et pour parcourir chaque bloc à l'intérieur de la blockchain, nous utiliserons une boucle for. À l'intérieur de cette boucle for, comparons le bloc actuel au bloc précédent, le currentBlock aura la valeur de i et le prevBlock sera la valeur de i - 1.

Ensuite, tout ce que nous voulons faire est de comparer la propriété previousBlockHash sur le currentBlock avec la propriété de hachage sur le bloc précédent. Pour ce faire, on a défini la

condition if. Au départ, nous avons mentionné la variable "validChain" égale à true Pour satisfaire la condition de vérification. Lorsque nous parcourons la blockchain et voyons que les hachages ne s'alignent pas correctement, nous voudrions alors définir la variable validChain sur false pour signifier que la chaîne n'est pas valide.

Il y a encore une autre chose que nous voulons faire, pour s'assurer que la chaîne a les données correctes. Nous pouvons le faire en reformulant le currentBlock à l'aide de la méthode hashBlock. Si le hachage généré commence par quatre zéros comme nous l'avons vu déjà, alors nous savons que toutes les données sont valides. Cependant, si cela ne commence pas par les quatre zéros, nous savons que les données à l'intérieur du bloc ne sont certainement pas valides. Donc, à l'intérieur de la boucle for, on a mentionné d'autre condition afin de vérifier cela.

Maintenant, en dehors de la boucle for, nous mentionnerons une autre condition qui vérifier une chose spéciale, c'est de vérifier le genesis bloc. Car nous l'avons fait nous-mêmes sans faire de preuve de travail. Et nous voulons juste vérifier que toutes les propriétés de ce bloc que nous avons déjà défini sont correctes, commençons par le nonce, avec une valeur de 100, previousBlockHash, avec une valeur 0, et le hachage avec 0 également et au final de vérifier qu'il ne doit contenir aucune data.

3.4.5.2. Construire le point de terminaison / consensus

Maintenant, construisons le point de terminaison / consensus, qui utilisera la méthode "chainIsValid" que nous avons construite dans la section précédente.

```
app.get('/consensus', function(req, res) {
	const requestPromises = [];
	healthblockchain.networkNodes.forEach(networkNodeUrl => {
		const requestOptions = {
			uri: networkNodeUrl + '/blockchain',
			method: 'GET',
			json: true
		};

		requestPromises.push(rp(requestOptions));
	});

	Promise.all(requestPromises)
	.then(blockchains => {
		const currentChainLength = healthblockchain.chain.length;
		let maxChainLength = currentChainLength;
		let newLongestChain = null;
		let newPendingDatas = null;

		blockchains.forEach(blockchain => {
			if (blockchain.chain.length > maxChainLength) {
				maxChainLength = blockchain.chain.length;
				newLongestChain = blockchain.chain;
				newPendingDatas = blockchain.pendingDatas;
			};
		});
```

```
if (!newLongestChain || (newLongestChain &&
!healthblockchain.chainIsValid(newLongestChain))) {
                        res.json({
                                note: 'Current chain has not been replaced.',
                                chain: healthblockchain.chain
                        });
                }
                else {
         healthblockchain.chain = newLongestChain;
        healthblockchain.pendingDatas = newPendingDatas;
                        res.json({
                        note: 'This chain has been replaced.',
                        chain: healthblockchain.chain
                        });
                }
        });
});
```

Figure 3.24 : Mise en œuvre du point de terminaison consensus.

À l'intérieur du point de terminaison / consensus, faisons une demande à tous les autres nœuds du réseau blockchain grâce à la boucle "forEach" pour obtenir leurs copies de la blockchain et comparons-les à la copie de la blockchain hébergée sur le nœud actuel sur lequel nous sommes actuellement. Après avoir défini les options de la boucle, nous devons pousser toutes ces demandes dans un tableau de promesse. Une fois la boucle forEach exécutée, nous aurons le tableau rempli de toutes les requêtes. Les données que nous recevons seront un tableau de la blockchain de chaque nœud à l'intérieur du réseau.

Ensuite, il faut voir s'il existe une blockchain à l'intérieur du réseau blockchain que celle actuellement sur le nœud sur lequel nous nous trouvons. Pour ce faire, on insère une autre boucle "forEach" et nous allons définir quelques variables pour déterminer si nous avons besoin

de remplacer la chaîne hébergée sur ce nœud actuel ou non. Une variable appelée "newLongestChain" au départ, que nous allons la définir comme égale à null et puis la variable

"maxChainLength". Ensuite, la dernière variable que nous définirons sera appelée "newPendingTransactions" définie également égale à null au départ.

Ensuite, après la boucle, nous allons définir quelques conditions. Fondamentalement, ce que nous déclarons dans cette instruction if est que s'il n'y a pas de signification "newLongestChain", alors la chaîne actuelle est la plus longue. Alternativement, s'il existe une nouvelle chaîne la plus longue mais que cette nouvelle chaîne n'est pas valide, dans ces deux cas, nous ne voulons pas remplacer la blockchain hébergée sur le nœud actuel. Nous allons donc renvoyer la note qui dit « La chaîne actuelle n'a pas été remplacée ». Sinon, s'il y a une "newLongestChain" et que cette chaîne est valide, c'est maintenant que nous voulons remplacer la blockchain hébergée sur le nœud actuel par la chaîne la plus longue du réseau. Nous définirons tout cela à l'intérieur de la condition "else".

3.4.6. Le code d'exploitation

Dans cette section, nous construisons un code bloc qui nous permettra d'interagir avec la blockchain. Tout simplement, nous permettrons d'explorer les données à l'intérieur de la blockchain en recherchant un bloc spécifique.

3.4.6.1. Construire la méthode getBlock

Construisons une nouvelle méthode appelée getBlock qui prendra le blockHash donné et recherchera dans toute la blockchain.

```
Blockchain.prototype.getBlock = function (blockHash) {
        let correctBlock = null;
        this.chain.forEach(block => {
                if (block.hash === blockHash)correctBlock = block;
        });
 return correctBlock;


};
```

Figure 3.25: Mise en œuvre de la méthode getBlock.

Dans cette méthode, nous voulons parcourir l'ensemble de la blockchain et rechercher le bloc qui a une valeur blockHash particulière. Ensuite, cette méthode nous retournera ce bloc spécifique. Nous allons faire tout cela à l'aide d'une boucle for, ensuite, à l'intérieur de la boucle, mentionnons les conditions à l'aide d'instructions if.

Au fur et à mesure que nous parcourons tous les blocs de la chaîne, si nous rencontrons le bon bloc, nous l'affectons à correctBlock.

3.4.6.2. Construction du point de terminaison / block /: blockHash

Utilisons la méthode getBlock à l'intérieur du point de terminaison / block /: blockHash pour récupérer un bloc spécifique par son blockHash.

```
app.get('/block/:blockHash', function(req, res) {
        const blockHash = req.params.blockHash;
        const correctBlock = healthblockchain.getBlock(blockHash);
        res.json({
                block: correctBlock
        });
});
};
```

Figure 3.26 : Mise en œuvre du point de terminaison / block /: blockHash.

La première chose que nous voulons faire dans ce point de terminaison est d'utiliser la valeur blockHash qui est envoyée avec la requête / block /: blockHash. Nous pouvons accéder à ce blockHash sur l'objet req.params. Ensuite, nous voulons utiliser la méthode getBlock créée précédemment. Enfin, renvoyez la variable correctBlock en tant que réponse.

3.4.6.3. Construire la méthode getData

Ajoutons une nouvelle méthode sur la structure de données blockchain appelée getData. Cela nous permettra d'obtenir une data spécifique en passant dataId. Nous utiliserons cette nouvelle méthode à l'intérieur du point de terminaison / data /: dataId.

```
Blockchain.prototype.getData = function(dataId) {
        let correctData = null;
        let correctBlock = null;

        this.chain.forEach(block => {
                block.datas.forEach(data => {
                        if (data.dataId === dataId) {
                                correctData = data;
                                correctBlock = block;
                        };
                });
        });

        return {
                data: correctData,
                block: correctBlock
        };
};
```

Figure 3.27 : Mise en œuvre de la méthode getData.

Cette méthode est très similaire à la méthode getBlock. Ici, nous allons parcourir toute la blockchain et définir un indicateur égal à la data correcte que nous recherchons. Pour cela, utilisons la boucle forEach. De plus, dans cette méthode, nous recherchons des data, nous devons parcourir chaque data sur chaque bloc de la blockchain. Par conséquent, nous devons ajouter une autre boucle for à l'intérieur de la boucle for précédente.

Maintenant que nous avons accès à chaque data sur la blockchain, nous devons simplement comparer le dataId de chaque data avec le dataId que nous recherchons. Lorsque les deux correspondent, nous savons que nous avons trouvé la bonne transaction. Définissons donc la condition if à l'intérieur de la boucle. Ensuite, nous voulons définir un indicateur pour indiquer que nous avons trouvé la transaction correcte dans la méthode getData. Et enfin, juste pour

rendre un peu cette méthode plus utile, nous allons également envoyer le bloc dans lequel nous avons trouvé la data que nous recherchions.

3.4.6.4. Construction du point de terminaison / data /: dataId

Construisons le point de terminaison / data /: dataId en utilisant la méthode getData créée précédemment.

```
app.get('/data/:dataId', function(req, res) {
    const dataId = req.params.dataId;
    const trasactionData = healthblockchain.getData(dataId);
    res.json({
        data: trasactionData.data,
        block: trasactionData.block
    });
});
```

Figure 3.28 : Mise en œuvre du point de terminaison / data /: dataId.

La première chose à faire à l'intérieur de ce point de terminaison est de stocker l'ID de data envoyé en tant que paramètre de demande. Stockons cela dans une variable dataId. La prochaine chose à faire est d'utiliser la méthode getData. À partir de la méthode getData, nous obtenons un objet qui nous est renvoyé contenant la data que nous recherchons et le bloc dans lequel se trouve. Nous souhaitons stocker ces données et les renvoyer comme réponse dans une variable appelée reqData.

3.5. Lancement du programme

Dans cette section, nous allons tester nos points de terminaison pour nous assurer que tout fonctionne bien ensemble.

- **Etape 01**

Exécutons le fichier « networkNode.js ». Dans la fenêtre du terminal en tapant la commande npm run node_1", le serveur devrait commencer à écouter le port 3001 du node_1, comme nous pouvons l'observer dans la capture d'écran suivante :

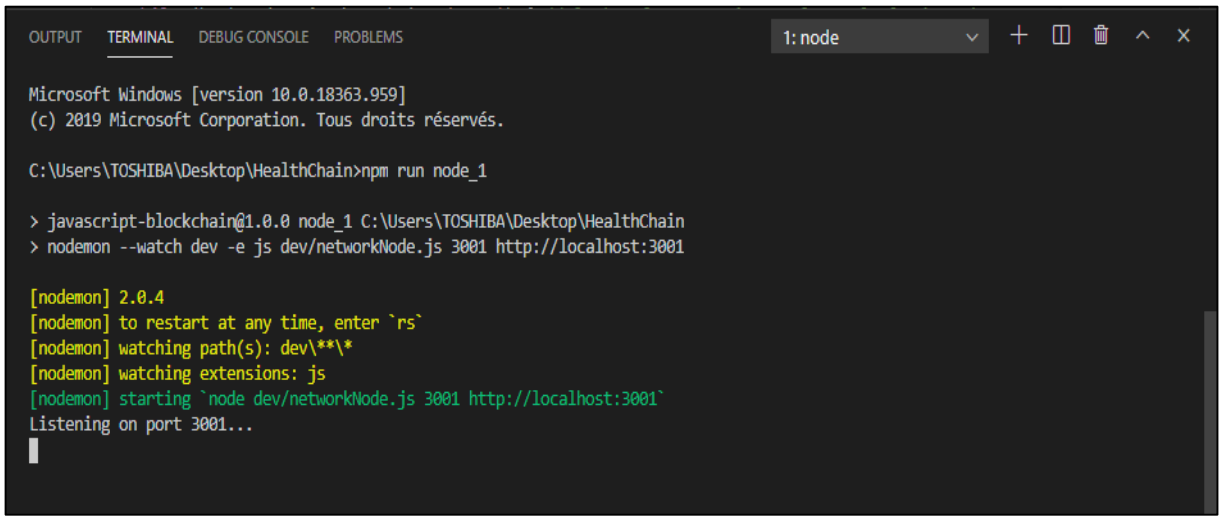

Figure 3.29 : Exécution du node_1 sur le port 3001.

Nous voudrons exécuter le deuxième nœud, node_2, sur le port 3002. Par conséquent, tapons simplement "npm run node_2", puis appuyons sur Entrée. On fera la même chose pour tous les autres nœuds.

Nous créons, en fait, cinq instances différentes dans notre fichier "networkNode.js". Donc, essentiellement, nous avons cinq nœuds de réseau différents en cours d'exécution.

Nous allons vers Postman et essayons de consulter notre réseau en touchant le point de terminaison / blockchain avec la méthode "GET" sur les différents nœuds de réseau que nous avons en cours d'exécution comme nous pouvons l'observer dans la capture d'écran suivante :

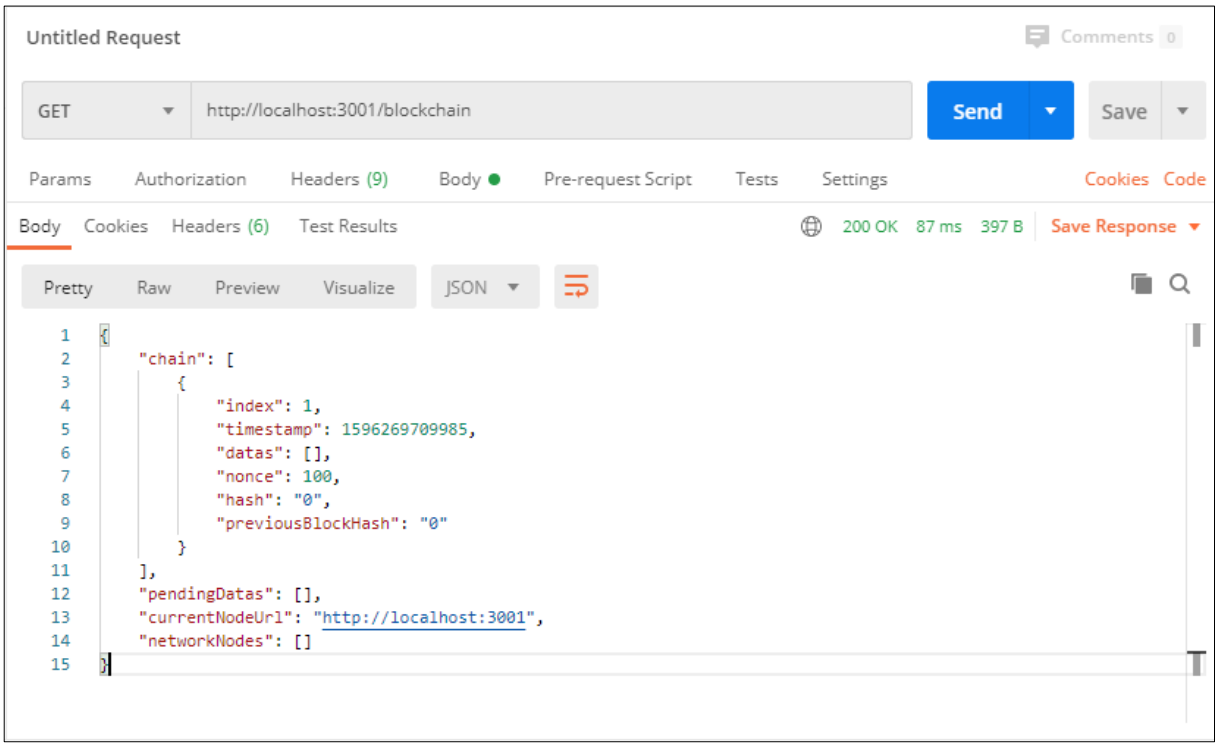

Figure 3.30 : Récupération de la blockchain du premier nœud.

Nous pourrons observer une réponse similaire pour tous les nœuds, comme indiqué le premier nœud dans la capture d'écran précédent. On remarque qu'on a juste le bloc genesis dans le tableau "chain", et le tableau "networkNode" est vide. Cela signifie que nous avons cinq nœuds en cours d'exécution mais ils ne sont en aucun cas connectés. Tout ce que nous avons, ce sont cinq nœuds distincts ou cinq instances distinctes de notre API.

- **Etape 02**

Le prochain point de terminaison que nous allons toucher est notre point de terminaison / register-and-broadcast-node pour créer notre réseau de blockchain décentralisé.

Dans l'application Postman, nous voulons faire une demande "POST" pour enregistrer et diffuser le nœud sur localhost : 3001. Le premier appel que nous allons faire va simplement connecter deux nœuds ensemble pour former les débuts de notre réseau. Lorsque nous atteignons le point de terminaison register-and-broadcast-node, nous devons envoyer un newNodeUrl que nous voulons enregistrer.

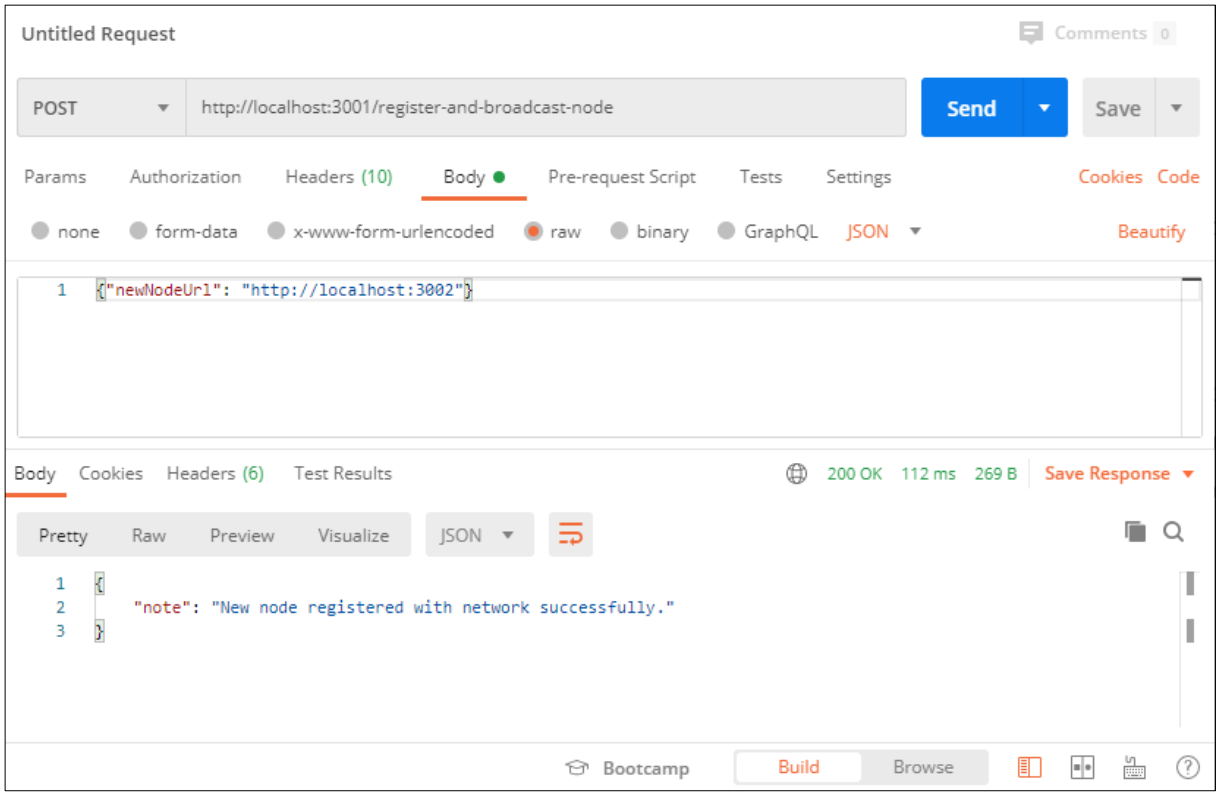

Figure 3.31 : Connection du deuxième nœud avec le premier nœud.

Maintenant, lorsque nous faisons cette demande, il devrait enregistrer notre nœud hébergé sur localhost: 3002 avec notre nœud hébergé sur localhost: 3001. À partir de la capture d'écran précédente, nous pouvons voir que le nouveau nœud a été enregistré avec succès sur le réseau.

Ensuite, ajoutons un autre nœud à ce réseau, revenons à Postman et changeons le localhost: 3002 en localhost: 3003. Nous allons faire une demande au nœud qui se trouve sur 3001. Ce que cela devrait faire est d'enregistrer notre nœud hébergé sur localhost: 3003 avec tous les autres nœuds du réseau. Donc, 3003 devrait s'inscrire avec 3001 et 3002. Suivant les mêmes procédures que nous avons suivies pour enregistrer le quatrième nœud et on laisse le cinquième nœud déconnecté pour les prochaines étapes. Vérifions cela dans notre application Postman.

Lorsque nous actualisons dans localhost: 3001, nous devrions avoir tous les URL des autres nœuds à l'intérieur du tableau networkNodes comme l'indique la capture d'écran suivante :

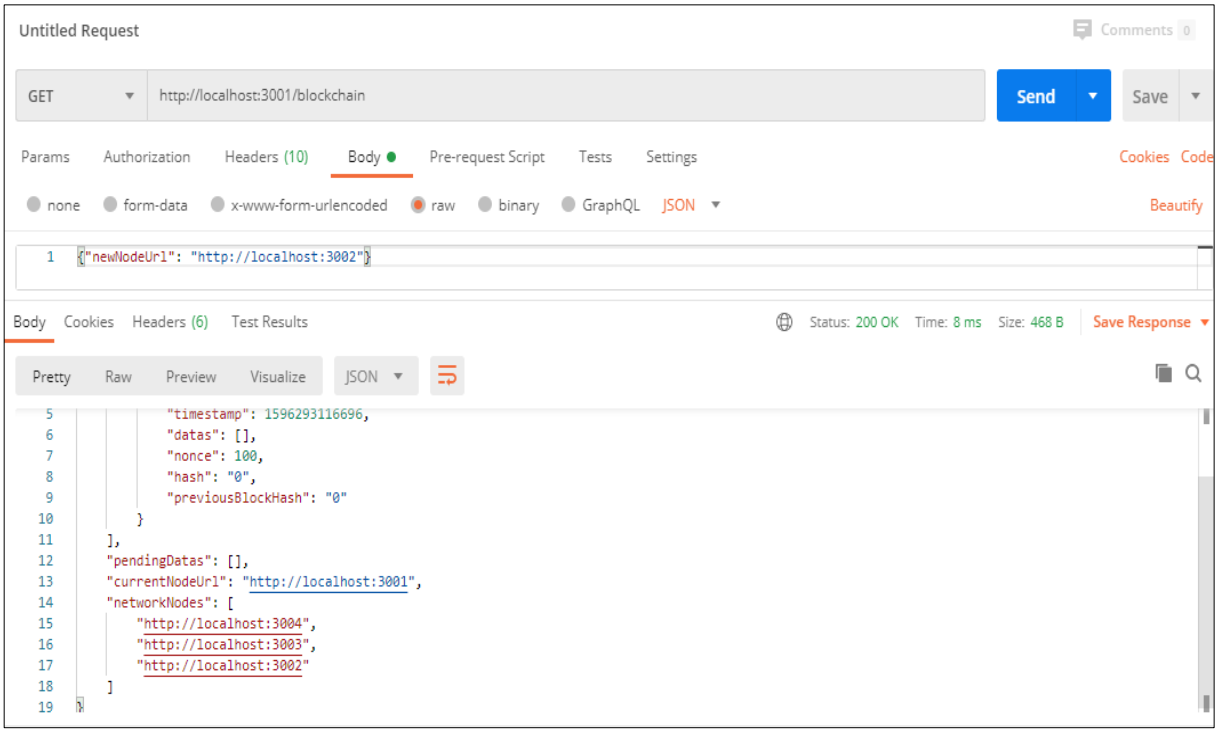

Figure 3.32 : Actualisation de la blockchain du premier nœud.

Et si nous actualisons dans localhost: 3004 par exemple, nous devrions avoir aussi tous les URL des autres nœuds à l'intérieur du tableau networkNodes comme l'indique la capture d'écran suivante :

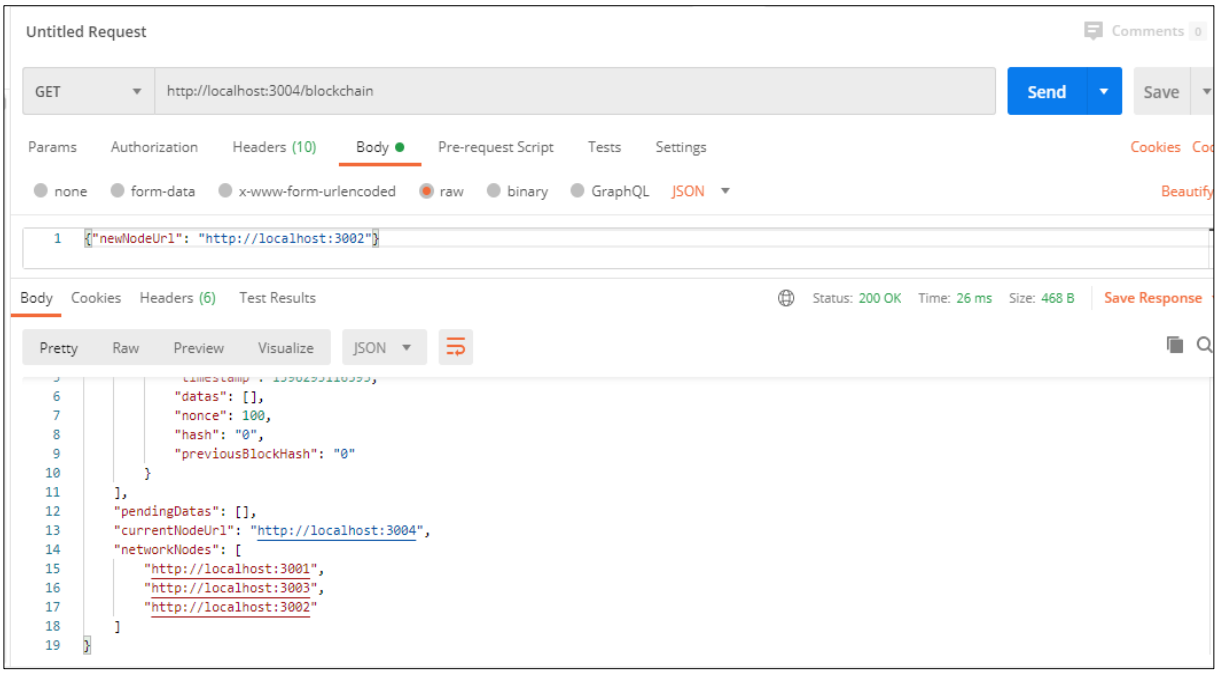

Figure 3.33 : Actualisation de la blockchain du nœud quatre.

Par conséquent, nous avons construit un réseau composé de quatre nœuds. Chaque nœud était au courant de tous les autres nœuds du réseau, ce qui a créé un réseau blockchain décentralisé.

- **Etape 03**

Maintenant que le réseau blockchain est configuré, frappons le point de terminaison / data / broadcast. Ce point de terminaison nous permettra de diffuser des données sur l'ensemble du réseau blockchain, afin que chaque nœud ait les mêmes données que l'ensemble des nœuds.

Revenons à Postman et appuyons sur le point de terminaison / data / broadcast sur le nœud qui est hébergé sur le port 3002 par exemple, on crée notre structure data dans le body et on définit la méthode "POST" comme indiqué dans la capture d'écran suivante :

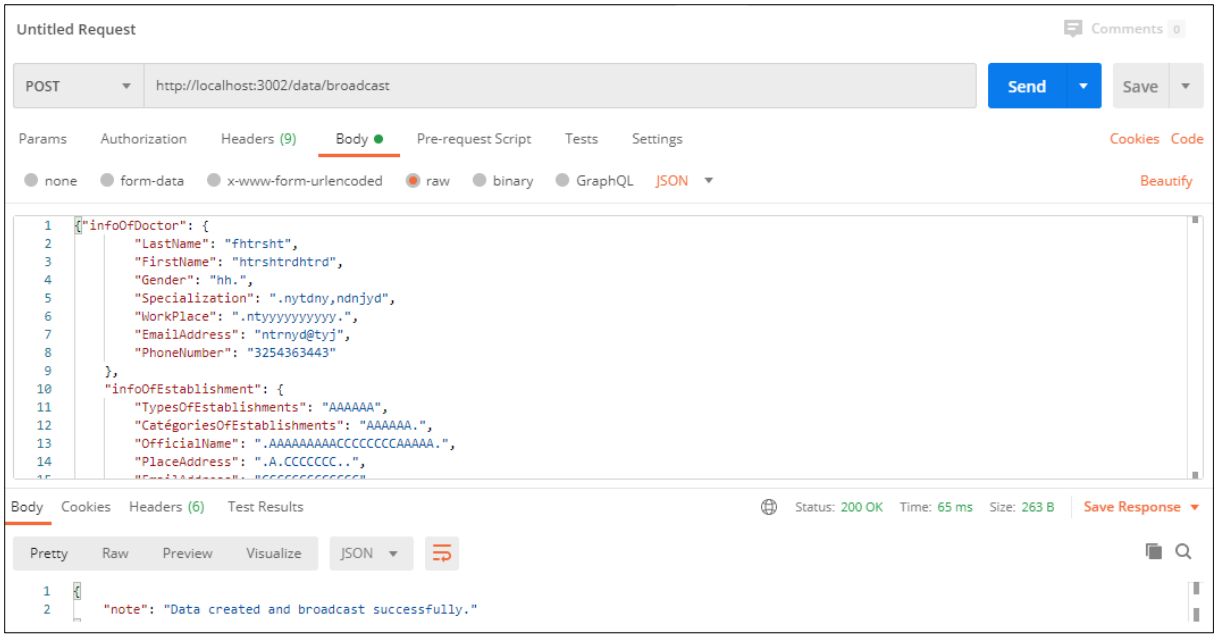

Figure 3.34 : Création et diffusion de la nouvelle data.

Envoyons cette demande en cliquant sur le bouton Envoyer. Si la data est envoyée avec succès, une réponse sera reçue indiquant, Data créée et diffusée avec succès.

Maintenant, allons voir la data que nous avons créée sur le nœud du réseau. Vérifions si cela a fonctionné. Dans la barre d'adresse de Postman, saisissons localhost: 3002 / blockchain, puis appuyons sur Entrée. Nous devrions voir les données dans le tableau pendingDatas, comme illustré dans la capture d'écran suivante :

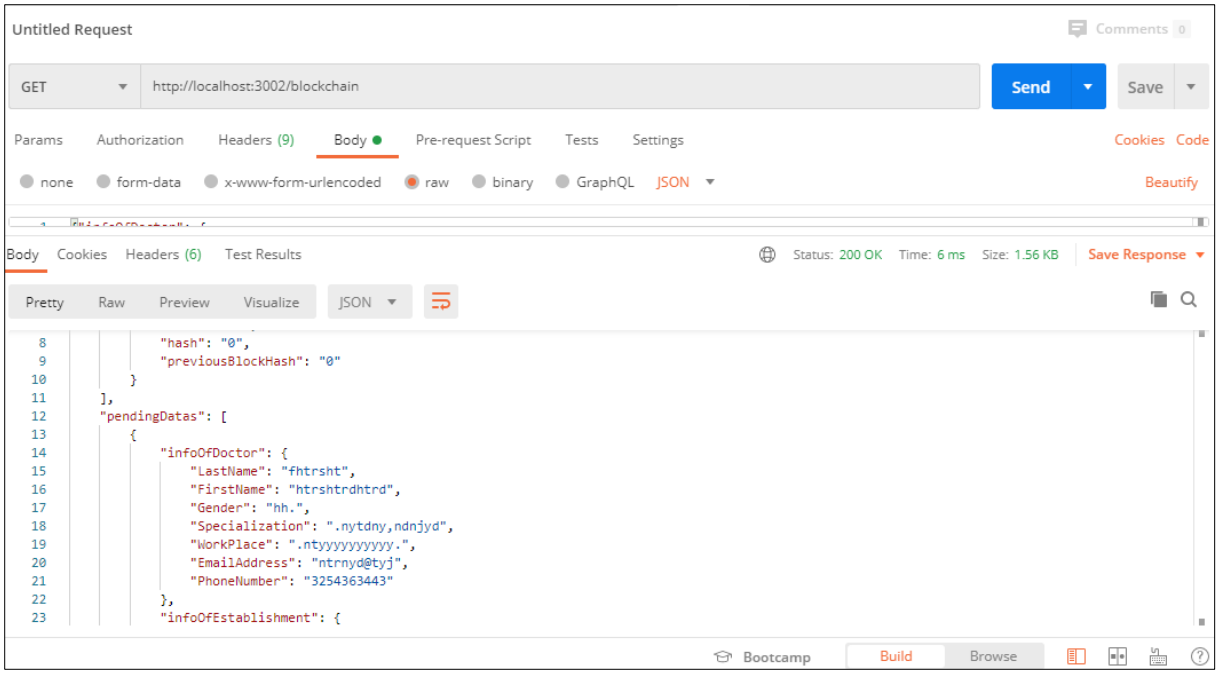

Figure 3.35 : Les données attendent du deuxième nœud.

Nous passons aux autres nœuds du réseau pour effectuer une vérification similaire pour tous les nœuds restants. Nous pouvons observer les mêmes données dans le tableau pendingDatas de chaque nœud. Chaque nœud à l'intérieur du réseau blockchain est maintenant conscient qu'une nouvelle data a été créée.

- **Etape 04**

Une fois nous créons des données, nous voudrons maintenant les insérer dans un nouveau bloc. Donc, en allant dans l'application Postman toujours, la première chose à faire ici est de choisir un nœud pour miner le nouveau bloc. Nous avons le choix entre quatre nœuds. Dans notre cas, nous nous choisissons le troisième nœud. Par conséquent, tapons localhost: 3003 / mine dans la barre d'adresse, puis appuyez sur Entrée. Vous obtiendrez une sortie comme celle-ci :

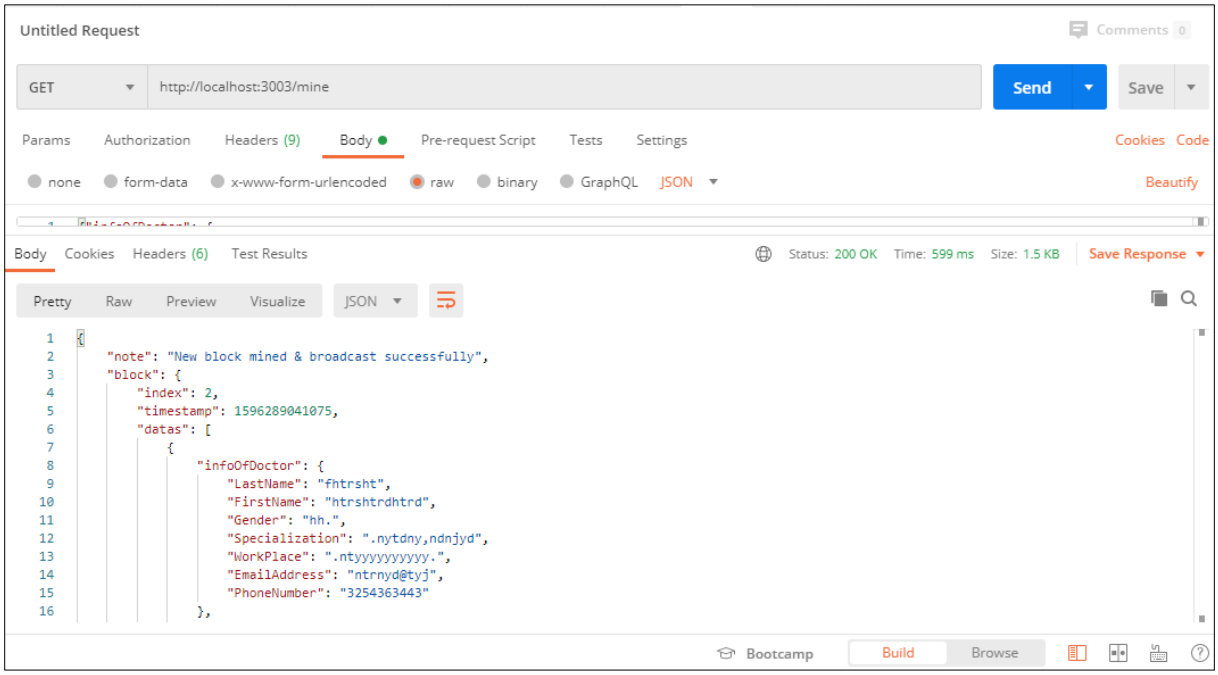

Figure 3.36 : Création d'un nouveau bloc.

La réponse indique que le nouveau bloc a été extrait et diffusé avec succès. On peut également voir le nouveau bloc dans le capture d'écran précédente avec son index.

Alors, si on vérifie si le nouveau bloc a été ajouté au réseau. Dans Postman, ouvrons un autre onglet, tapons localhost: 3003 / blockchain ou n'importe quel nœud connecté dans le réseau dans la barre d'adresse, puis appuyez sur Entrée. Nous pouvons voir que le nouveau bloc a été inséré dans la blockchin de tous les nœuds du réseau :

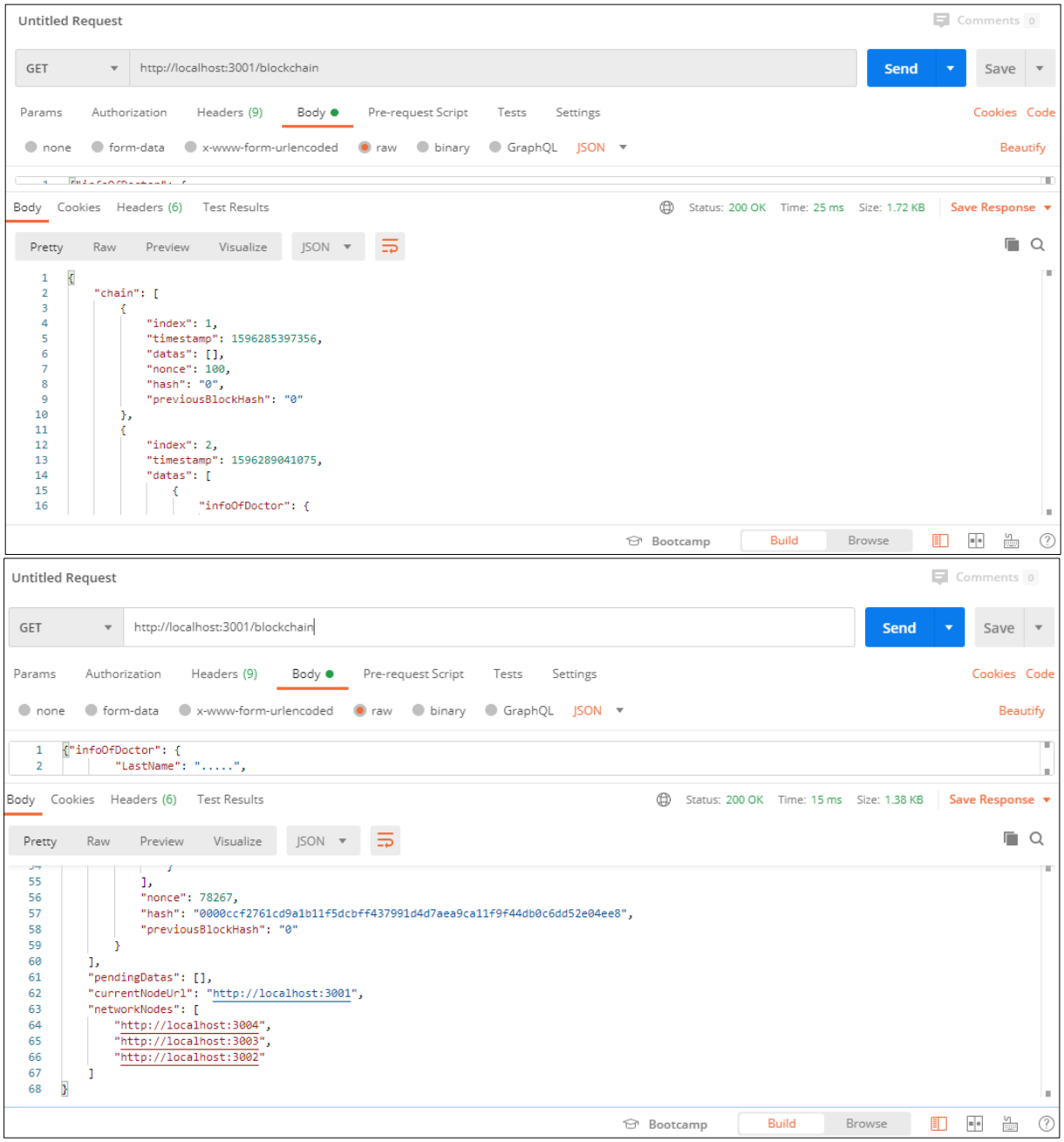

Figure 3.37 : Mise à jour de la blockchain.

Nous remarquons également que les données qu'elles étaient présentes dans le tableau pendingDatas sont insérées dans le bloc que nous venons d'exploiter, et le tableau pendingDatas sera vide.

- **Etape 05**

Jusqu'à maintenant, on a un réseau blockchain décentralisé, composé de quatre nœuds bien synchronisés. Alors, si on ajoute un nouveau nœud au réseau, il devrait nous confirmer que ce nœud particulier contient les données de blockchain correctes et que le nœud est synchronisé avec le reste du réseau.

D'abord, on va l'ajouter au réseau en suivant les procédures des étapes précédentes, ensuite, on va le vérifier en utilisant notre application Postman comme l'indique la capture d'écran suivante :

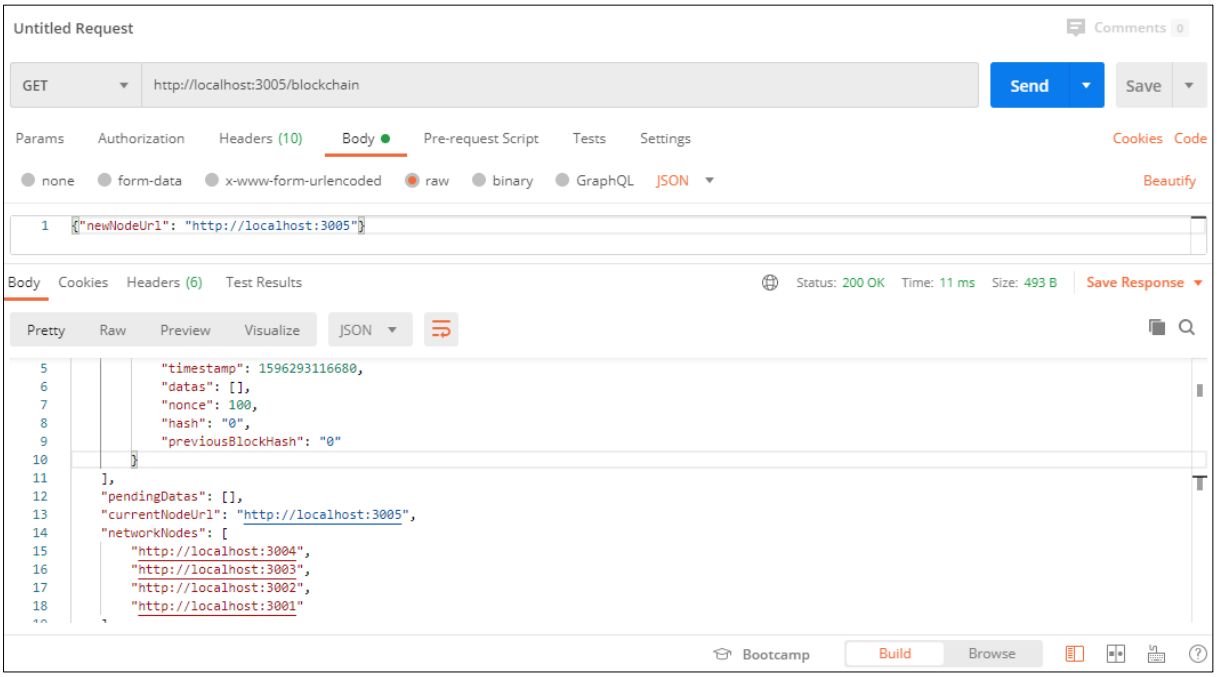

Figure 3.38 : Récupération de la blockchain du cinquième noeud.

Maintenant que le cinquième nœud fait partie du réseau, mais le problème que ce dernier n'a pas les bonnes données blockchain. A l'intérieur de la blockchain, il devrait avoir toutes les données et les blocs qu'auront les autres nœuds.

C'est là que le point de terminaison / consensus entre en jeu. Nous devrions être en mesure d'atteindre le point final / consensus et de résoudre ce problème. Après cela, nous devrions nous attendre à ce que la blockchain sur le nœud cinq ait les mêmes données que tous les autres nœuds à l'intérieur du réseau.

Ouvrons alors un autre onglet dans Postman, et dans la barre d'adresse, tapons localhost: 3005 / consensus, puis exécutez-le par en appuyant sur Entrée. Nous devrons observer des sorties montrées par la capture d'écran :

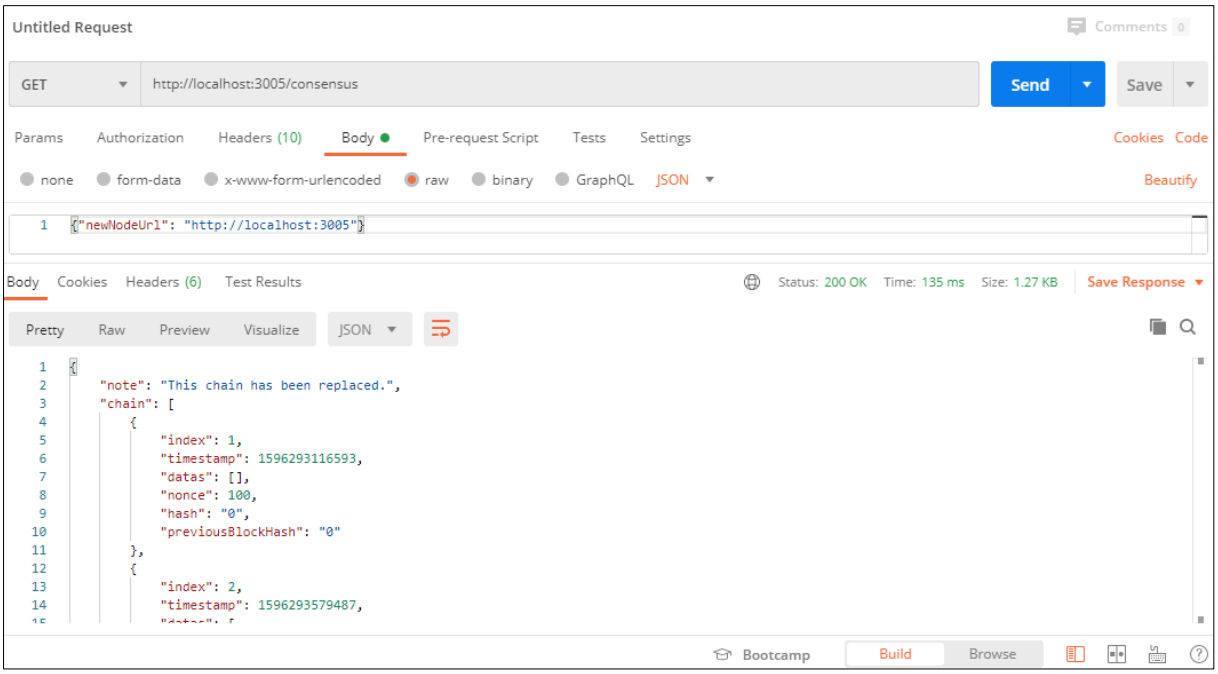

Figure 3.39 : Synchronisation des données du cinquième nœud.

Dans la capture d'écran précédente, nous obtenons la réponse "cette chaîne a été remplacée", et les nouvelles données blockchain remplacent alors les anciennes données sur ce nœud. Vérifions ce nœud en allant dans un autre onglet du Postman et frappons localhost: 3005 / blockchain. Nous verrons que tous les blocs qui étaient présents dans le réseau sont ajoutés au nœud sur le port 3005.

Maintenant, si nous essayons à nouveau d'atteindre le point de terminaison / consensus sur le nœud 3005, nous obtiendrons la réponse "cette chaîne n'a pas été remplacée".

- **Etape 06**

On va ajouter maintenant d'autre data et d'autre blocs dans notre blockchain. On suit les mêmes étapes précédentes. On a ajouté quatre autres blocs dans notre exemple et chacun a son data.

Afin de récupérer un bloc spécifique dans notre blockchain, prenons simplement la valeur de hachage de ce bloc et utilisons le point de terminaison / block /: blockHash. Copions la valeur

de hachage du troisième bloc de la blockchain, Ensuite, allons dans un autre onglet du Postman, tapons localhost: 3001 / block dans la barre d'adresse, puis collons la valeur de hachage que nous avons copiée directement après cette URL. Jetez un œil à la capture d'écran suivante pour une meilleure compréhension :

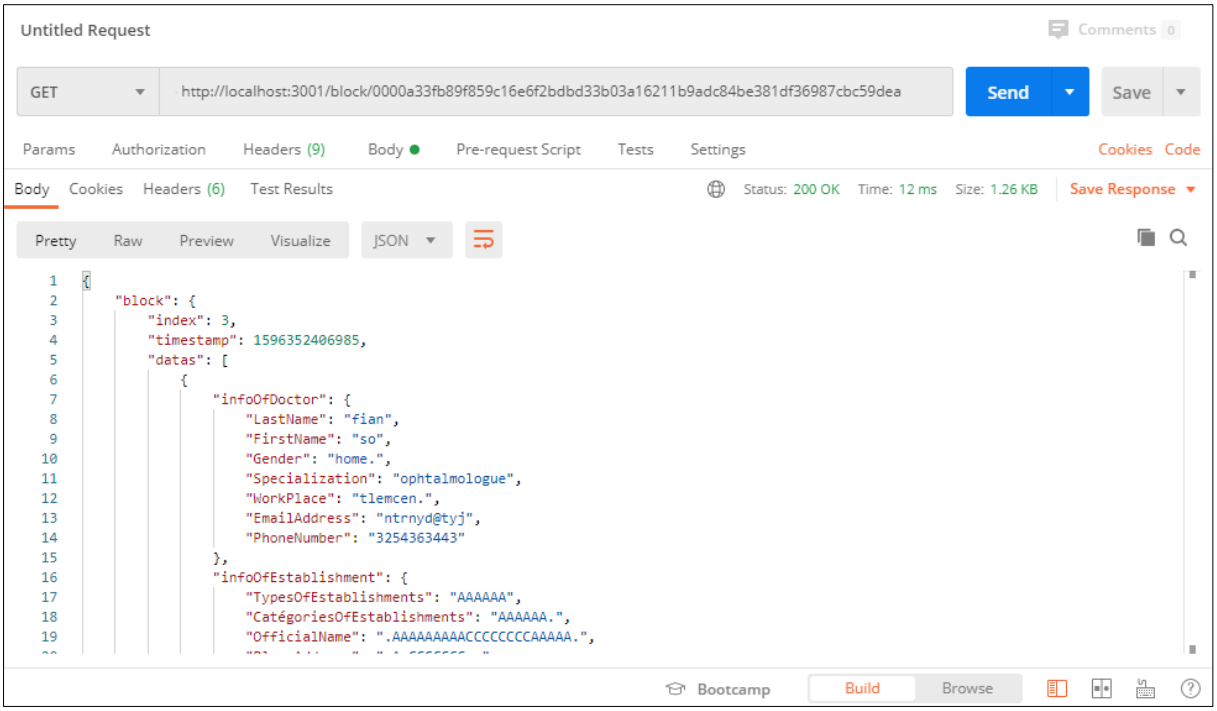

Figure 3.40 : Récupération de troisième bloc.

À partir de la capture d'écran précédente, nous pouvons observer que le bloc correct nous est retourné. Le bloc retourné se compose de la valeur de hachage que nous avons utilisée dans le point de terminaison / block /: blockHash pour rechercher le bloc.

Maintenant, si nous devions envoyer un mauvais hachage ou un hachage qui n'existe pas avec le point de terminaison, nous devrions nous attendre à ce que null nous soit renvoyé en sortie, au lieu que le bloc soit renvoyé.

Semblable à ce que nous avons fait pour récupérer un bloc spécifique, on peut aussi récupérer un data spécifique prenons simplement la valeur ID du data et utilisons le point de terminaison / data /: . Copions la valeur dataId du cinquième bloc de la blockchain, Ensuite, allons dans un autre onglet du Postman. Tapons localhost: 3001 / data dans la barre d'adresse, puis collons la

valeur dataId que nous avons copiée directement après cette URL. Jetez un œil à la capture d'écran suivante pour une meilleure compréhension :

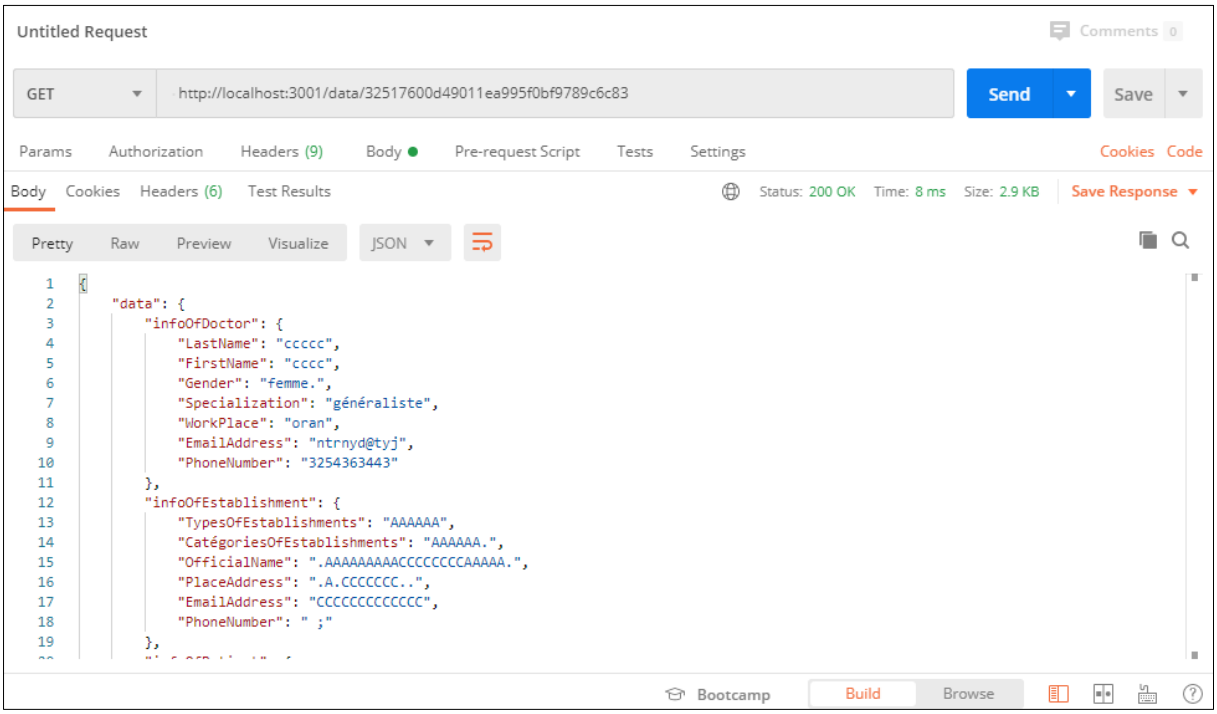

Figure 3.41 : Récupération de d'une data de cinquième bloc.

Dans la capture d'écran précédente, nous pouvons observer que la data, associée à l'identifiant de data que nous avons passé avec le point de terminaison, est retournée en sortie. Nous avons également renvoyé le bloc, qui consistait en le dataId particulier que nous recherchions.

Maintenant, si nous devions envoyer un mauvais ID ou un ID qui n'existe pas avec le point de terminaison, nous obtiendrons la valeur null renvoyée en sortie.

3.6. Conclusion

Dans ce chapitre, nous avons présenté les programmes qui permettent de réaliser la blockchain. Nous avons commencé par présenter l'environnement de développement suivi par les fonctions et les bibliothèques nécessaires utilisées. Enfin, nous avons présenté le résultat obtenu.

Conclusion générale

Conclusion générale

La blockchain est un nouvel entrant dans l'écosystème technologique. Deux facteurs ont été déterminants dans la genèse de cette technologie. D'une part, les innovations incrémentales réalisées dans les domaines de l'informatique, de la cryptographie et des communications. D'autre part, l'émergence d'une idéologie plébiscitant le besoin de disposer de technologies garantissant la vie privée des individus, et une possible indépendance vis à vis des systèmes centralisés étatiques ou privés.

Le déploiement de la technologie blockchain dans le secteur de la santé a ses nombreux avantages comme elle répond essentiellement aux défis de l'interopérabilité et sera donc largement acceptée et déployée dans toutes les organisations, les établissements et les industries médicaux.

L'utilisation de la blockchain médicale proposée décrite dans ce mémoire a le potentiel d'engager des millions d'individus, de prestataires de soins de santé, d'entités de soins de santé et de chercheurs médicaux à partager de grandes quantités de données de santé avec une sécurité garantie et une protection de la vie privée des patients.

Cette technologie améliorera la sécurité, la gestion et l'analyse des données de santé en générale. Étant donné que les données de santé sont sensibles et nécessitent un traitement en temps réel, en particulier en cas d'urgence, une autre préoccupation est de s'assurer que le bon accès à la blockchain est correctement développé pour éviter la fuite des informations personnelles des patients. En outre, les responsables de la santé qui planifient les implémentations de la blockchain devraient envisager cette technologie qui peux accélérer l'adoption et le délai de rentabilisation

Références bibliographique

Bibliographie

[1] binance, https://www.binance.vision/fr/blockchain/history-of-blockchain, consulté le : 20/02/2020.

[2] LELOUP, Laurent. Blockchain : La révolution de la confiance. Editions Eyrolles, 2017.

[3] PIGNEL, Marion et STOKKINK, Denis. LA TECHNOLOGIE BLOCKCHAIN Une opportunité pour l'économie sociale ?

[4] cryptoast, https://cryptoast.fr/bloc-blockchain-crypto-explication/, consulté le :25/02/2020.

[5] FRANCE, Blockchain. La blockchain décryptée. Les clefs d'une révolution. Paris, Netexplo, 2016.

[6] cryptobjectif, https://info.cryptobjectif.fr/?p=79 , consulté le : 27/02/2020.

[7] binance, https://www.binance.vision/fr/blockchain/private-public-and-consortium-blockchains-whats-the-difference#consortium-blockchains , consulté le 27/02/2020.

[8] PARIS, Jimmy. Apports des Smart Contracts aux Blockchains et comment créer une nouvelle crypto-monnaie. 2017. Thèse de doctorat. Haute école de gestion de Genève.

[9] PIGNEL, Marion et STOKKINK, Denis. LA TECHNOLOGIE BLOCKCHAIN Une opportunité pour l'économie sociale ?

[10] GODEBARGE, Ferréol et ROSSAT, Romain. Principes clés d'une application Blockchain. EM Lyon Business School, 2016.

[11] lehub.bpifrance, https://lehub.bpifrance.fr/comment-fonctionne-blockchain/ , consulté le :02/03/2020.

[12] cryptoast, https://cryptoast.fr/bloc-blockchain-crypto-explication/, consulté le : 02/03/2020.

[13] bitcoin, https://bitcoin.fr/minage/, consulté le : 04/03/2020.

[14] binance, https://www.binance.vision/fr/blockchain/proof-of-work-explained, consulté le 04/03/2020.

[15] hackernoon, https://hackernoon.com/is-spos-the-pos-killer-hw2l3364,consulté le 04/03/2020.

[16] bitconseil, https://bitconseil.fr/proof-of-stake-definition-explication/, consulté le 05/03/2020.

[17] ethereum-France, https://www.ethereum-france.com/quest-ce-que-la-preuve-denjeu-proof-of-stake-faq-par-v-buterin-traduction-francaise/, consulté le 05/03/2020.

[18] bitconseil, https://bitconseil.fr/smart-contract-ethereum/, consulté le 06/03/2020.

[19] blockgeeks, https://blockgeeks.com/guides/smart-contracts/, consulté le 06/03/2020.

[20] novethic, https://www.novethic.fr/lexique/detail/cryptomonnaie.html, consulté le 07/03/2020.

[21] journalducm, https://www.journalducm.com/dictionnaire-marketing/blockchain-cryptomonnaie, consulté le 08/03/2020.

[22] cryptoencyclopedie, https://www.cryptoencyclopedie.com/single-post/bitcoin, consulté le 08/03/2020.

[23] cryptonews, https://fr.cryptonews.com/guides/bitcoin-pros-and-cons.htm, consulté le 10/03/2020.

[24] courscryptomonnaies, https://courscryptomonnaies.com/ethereum, consulté le 11/03/2020.

[25] thecointribune, https://www.thecointribune.com/guides-crypto/guide-des-monnaies/ripple-definition-quest-ce-que-le-ripple-xrp/, consulté le 11/03/2020.

[26] masterassasfinance, https://www.masterassasfinance.com/single-post/2018/03/08/Les-domaines-dapplication-de-la-Blockchain, consulté le 13/03/2020.

[27] builtin, https://builtin.com/blockchain/blockchain-applications, consulté le 14/03/2020.

[28] perlefinance, https://perlefinance.fr/les-avantages-et-les-inconvenients-de-la-blockchain/, consulté le 16/03/2020.

[29] hitconsultant, https://hitconsultant.net/2018/10/26/healthcare-data-blockchain/?fbclid=IwAR0-HQWGBbppZb2uwuUGsVBRGWVXy9ZYLphDfowQGNLKUfv_N1ZrS1pZy2M#.Xo4AdcgzbIV, consulté le : 15/03/2020.

[30] LA CLÉ, DE LA CONTINUITÉ, CENTRÉS, POUR DES SOINS, et LE PATIENT, S. U. R. ANALYSE.

[31] caducee, https://www.caducee.net/DossierSpecialises/systeme-information-sante/dmi.asp, consulté le : 17/03/2020.

[32]blockchainpartner,https://blockchainpartner.fr/wp-content/uploads/2017/05/Etude-Sant%C3%A9-industrie-pharmaceutique-Blockchain-Partner.pdf, consulté le : 20/03/2020.

[33]databreaches,https://cdn2.hubspot.net/hubfs/2331613/Breach_Barometer/2016/2016%20Year%20in%20Review/Protenus%20Breach%20Barometer-2016%20Year%20in%20Review-%20final%20version.pdfconsulté le : 20/03/2020.

[34] PETRE, Anca et HAÏ, Nassima. Opportunités et enjeux de la technologie blockchain dans le secteur de la santé. Médecine/sciences, 2018, vol. 34, no 10, p. 852-856.

[35] TESSIER, Sylvain. Fonctionnement de la blockchain et son intérêt pour le monde pharmaceutique. 2019.

[36] ZHANG, Peng, SCHMIDT, Douglas C., WHITE, Jules, et al. Blockchain technology use cases in healthcare. In: Advances in computers. Elsevier, 2018. p. 1-41.

[37] healtheuropa, https://www.healtheuropa.eu/who-guideline-recommendations-on-digital-health/96422/, consulté le : 22/03/2020.

[38] ibm, https://www.ibm.com/blogs/blockchain/2018/12/what-are-the-use-cases-for-blockchain-tech-in-healthcare/, consulté le : 23/03/2020.

[39] SIYAL, Asad Ali, JUNEJO, Aisha Zahid, ZAWISH, Muhammad, *et al*. Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. Cryptography, 2019, vol. 3, no 1, p. 3.

[40] wikipedia.org, https://fr.wikipedia.org/wiki/JavaScript, consulté le : 21/05/2020.

[41] makina-corpus, https://makina-corpus.com/blog/metier/2014/introduction-a-nodejs, consulté le : 22/05/2020.

[42] journaldunet, https://www.journaldunet.com/web-tech/developpeur/1204957-visual-studio-code-l-ide-open-source-que-les-devs-s-arrachent/, consulté le : 22/05/2020.

[43] geeksforgeeks, https://www.geeksforgeeks.org/introduction-postman-api-development/, consulté le : 30/05/2020.

[44] TRAUB, Eric. Learn Blockchain Programming with JavaScript: Build your very own Blockchain and decentralized network with JavaScript and Node. js. Packt Publishing Ltd, 2018.

[45] github, https://github.com/lhartikk/naivechain , consulté le 08/07/2020.