Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées
et de Technologies

Rapport de Projet de Fin d'Année

Filière

Réseaux Informatiques et Télécommunications 4

Développement d'un DNS décentralisé basé sur Ethereum

Réalisé par

Hazem MDIMEGH **Houssine AYEDI** **Hakam KOUBAA**

Encadré par

Mr Souheib YOUSSFI

Année Universitaire 2019 - 2020

Remerciements

Nous tenons à exprimer notres sincère gratitude

à ***Mr Souheib YOUSSFI****, notre tuteur de projet de fin d'année*

à l'INSAT qui nous a suivi tout au long de cette période

et nous a conseillé sur l'orientation que celui-ci devrait prendre.

Par ailleurs,

nous remercions ***Mr Ali ABDENNADHER****, chef du département génie informatique et mathématiques et les honorables membres du jury pour avoir accepter d'évaluer notre projet de fin d'année.*

Table des matières

Table des figures

Liste des abréviations

— **ABI** = **A**pplication **B**inary **I**nterface

— **CSS** = **C**ascading **S**tyle **S**heets

— **DAPP** = **D**ecentralized **APP**lication

— **DDNS** = **D**ecentralized **D**omain **N**ame **S**ystem

— **DDOS** = **D**istributed **D**enial **O**f **S**ervice

— **DLT** = **D**istributed **L**edger **T**echnology

— **DNS** = **D**omain **N**ame **S**ystem

— **DNSSEC** = **D**omain **N**ame **S**ystem **SEC**urity extensions

— **ABI** = **D**ocument **O**bject **M**odel

— **DoH** = **D**NS **O**ver **H**TTPS

— **DoT** = **D**NS **O**ver **T**LS

— **ERC** = **E**thereum **R**equest for **C**omment

— **EVM** = **E**thereum **V**irtual **M**achine

— **HTML** = **H**yper**T**ext **M**arkup **L**anguage

— **HTTPS** = **H**yper**T**ext **T**ransfer **P**rotocol **S**ecure

— **ICANN** = **I**nternet **C**orporation for **A**ssigned **N**ames and **N**umbers

— **IP** = **I**nternet **P**rotocol

— **JS** = **J**ava**S**cript

— **JSON** = **J**ava **S**cript **O**bject **N**otation

— **MAJ** = **M**ise **A** **J**our

— **PoS** = **P**roof **o**f **S**take

— **PoW** = **P**roof **o**f **W**ork

— **RPC** = **R**emote **P**rocedure **C**all

— **SSL** = **S**ecure **S**ockets **L**ayer

— **TCP** = **T**ransmission **C**ontrol **P**rotocol

— **TLD** = **T**op **L**evel **D**omain

— **TLS** = **T**ransport **L**ayer **S**ecurity

— **UDP** = **U**ser **D**atagram **P**rotocol

Introduction générale

Pour que les systèmes TCP/IP utilisent ses noms d'hôtes, ils doivent avoir un moyen pour découvrir l'adresse IP associée au nom. Au début de la mise en réseau TCP/IP, chaque ordinateur possédait une liste de noms et leurs adresses IP équivalentes, appelée table d'hôtes. À cette époque, le petit nombre d'ordinateurs sur Internet rend pratique la maintenance et la distribution d'une seule table d'hôtes. Aujourd'hui, il y a des millions d'ordinateurs sur Internet, et l'idée de maintenir et de distribuer un seul fichier contenant des noms pour chacun d'eux est absurde. Au lieu d'utiliser la table d'hôtes stockée sur chaque ordinateur, les réseaux TCP/IP utilisent aujourd'hui des serveurs DNS pour convertir les noms d'hôtes en adresses IP.
La gestion des serveurs DNS détient l'ensemble d'internet tel qu'il est et est considéré comme l'une des découvertes les plus importantes qui a propulsé le « World Wide Web » pendant plus de trois décennies. Mais c'est une arme à double tranchant !
Le présent rapport décrivant les différentes étapes de notre travail s'articule autour de quatre principaux chapitres. Nous commencerons, dans le premier chapitre, par une introduction au DNS suivie par les problèmes auxquels il est confronté puis une brève explication des solutions proposées pour finir par expliquer le choix adopté.
Le deuxième chapitre comportera la description de la technologie blockchain et la spécification du choix technologique fait à propos du type de blockchain implémenté dans notre solution.
Dans le troisième chapitre et à partir de l'analyse et la spécification des besoins, nous présenterons la conception de notre application décrite à l'aide d'une vue statique et d'une vue dynamique.
Le dernier chapitre sera consacré à la description de la phase de la réalisation et l'implémentation du projet. Nous traiterons en premier lieu l'architecture de notre solution. Nous présenterons par la suite une illustration de notre application à travers un ensemble d'interfaces graphiques et une explication du choix technologique.

Chapitre 1

Cadre général du projet

Plan

Introduction

Dans ce premier chapitre, nous allons introduire le DNS, suivi d'une étude de ses faiblesses et nous détaillons par la suite la problématique de notre projet et ses principaux objectifs.

1.1 Introduction au DNS

À la base, le DNS est une correspondance entre les noms de domaines et leurs adresses IP, mais les méthodes de création, de stockage et de récupération de ces noms sont très différentes de celles d'une table d'hôtes.

Dans cette section, nous commençons par présenter les différents composants du DNS, par la suite nous expliquons le fonctionnement du DNS et pour finir nous décrivons les deux problématiques de cette technologie en ces jours.

1.1.1 Les composants du DNS

Le DNS se compose principalement de trois éléments. La compréhension de ces éléments, nous aidera dans le développement de notre application plus tard.

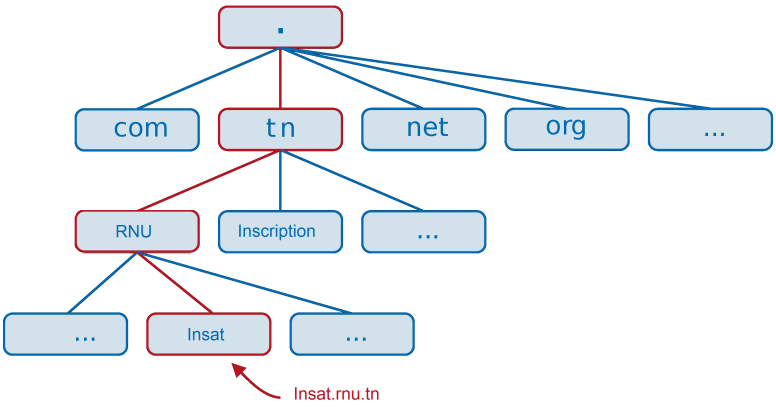

Figure 1.1: Hiérarchie du DNS

L'espace de noms : Les normes DNS définissent un espace de noms arborescent dans lequel chaque branche de l'arbre identifie un domaine. La racine de cet arbre est notée «.», ce point n'apparaît pas dans les adresses de sites web que nous utilisons sur Internet mais c'est bien le niveau le plus haut de la hiérarchie DNS. Les nœuds situés à la profondeur 1 de l'arbre sont appelés les domaines top-level (TLD), et les noeuds au dessous de chaque TLD sont des sous domaines ainsi qu'une délégation c'est-à-dire une indication que les informations de ces sous domaines sont enregistrées sur un autre serveur.

On représente un nom de domaine en indiquant les domaines successifs séparés par un

point, les noms de domaines supérieurs se trouvant à droite.

Il est possible aussi d'avoir plusieurs sous domaines. Nous illustrons au dessous un exemple de l'architecture arborescent du domaine de l'INSAT (insat.rnu.tn) dans la figure.1.1

Les serveurs de noms : Un serveur DNS est un service exécuté sur un ordinateur serveur qui conserve des informations sur la structure de l'arborescence de domaine et parfois contient des informations faisant autorité sur un ou plusieurs domaines spécifiques de cette structure. L'application est capable de répondre aux requêtes d'informations sur les domaines pour lesquels elle est l'autorité et également de transmettre des requêtes sur d'autres domaines à d'autres serveurs de noms. Cela permet à n'importe quel serveur DNS d'accéder aux informations sur n'importe quel domaine de l'arborescence.

Les résolveurs : Un résolveur est un programme client qui génère des requêtes DNS et les envoie à un serveur DNS pour exécution. Un résolveur a un accès direct à au moins un serveur DNS et il peut également traiter les renvois pour diriger ses requêtes vers d'autres serveurs si c'est nécessaire.

1.1.2 Le fonctionnement du DNS

Dans sa forme la plus élémentaire, le processus de résolution du nom DNS consiste en un résolveur soumettant une demande de résolution de nom à son serveur DNS désigné.
Lorsque le serveur ne possède pas d'informations sur le nom demandé, il transmet la demande à un autre serveur DNS du réseau. Le deuxième serveur génère une réponse contenant l'adresse IP du nom demandé et la renvoie au premier serveur, qui relaie les informations au résolveur. La résolution des noms de domaine est réalisée en parcourant la hiérarchie depuis le sommet en suivant les délégations successives.

Les étapes détaillées : Dans cette partie, nous allons étudier le cas d'un utilisateur qui a tapé "insat.rnu.tn" sur son navigateur.

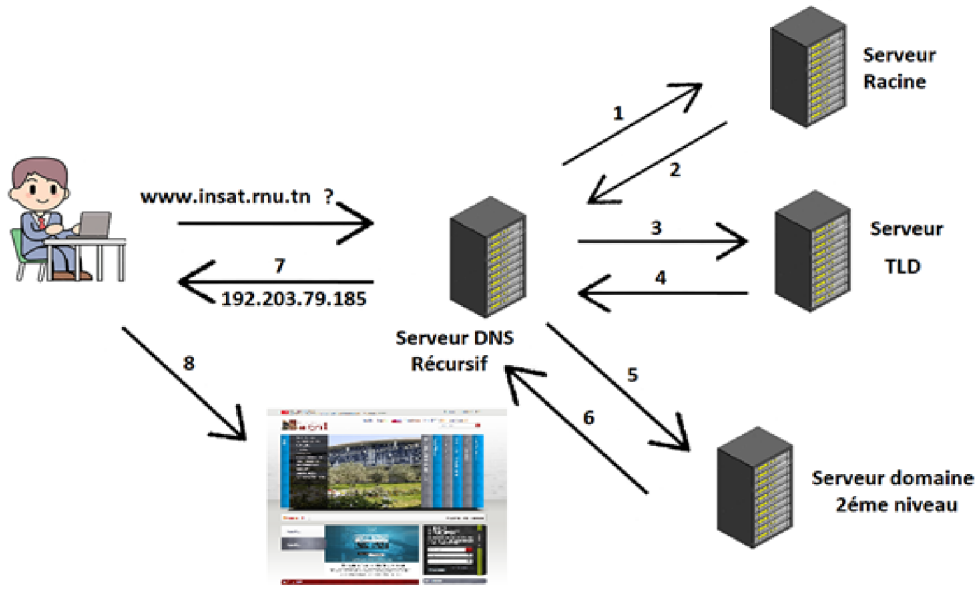

Figure 1.2: Schéma simplifié du fonctionnement du DNS

La figure 1.2 illustre les étapes du fonctionnement du DNS, le navigateur commence par chercher l'adresse IP du nom tapé tout simplement parce qu'il a besoin d'une adresse IP pour savoir à quel serveur se connecter et pour savoir à quelle adresse IP est associé le nom de ce domaine

1. Le serveur DNS de l'utilisateur, après avoir reçu la requête, vérifie ses enregistrements de ressources (ses registres) pour voir s'il s'agit de la source faisant autorité pour la zone contenant le nom de serveur demandé. Si ce n'est pas le cas, le serveur DNS génère une requête itérative et la soumet à l'un des serveurs de noms racine.
   Le serveur racine examine le nom demandé par le serveur DNS «.tn» et consulte ses registres pour identifier les serveurs faisant autorité pour le TLD.
2. Le serveur racine transmet ensuite une réponse au serveur DNS de l'utilisateur qui contient une référence aux adresses IP du serveur du TLD (.tn).
3. Le serveur DNS de l'utilisateur, désormais en possession de l'adresse du serveur du TLD pour le nom demandé, génère une nouvelle requête itérative et la transmet au serveur TLD.
4. Le serveur examine le domaine du 2éme niveau dans le nom demandé (insat.rnu.tn) et transmet une référence contenant les adresses des serveurs faisant autorité pour ce domaine du deuxième niveau au serveur DNS de l'utilisateur.
5. Le serveur DNS de l'utilisateur génère une autre requête itérative et la transmet au serveur de domaine du 2éme niveau.

6. Si le serveur de domaine du 2éme niveau est l'autorité de la zone contenante le nom demandé il consulte ces registres pour déterminer l'adresse IP de l'adresse demandé (insat.rnu.tn) et la retransmet dans un message de réponse au serveur DNS de l'utilisateur.
7. Le serveur DNS de l'utilisateur reçoit la réponse du serveur faisant autorité et retransmet l'adresse IP (192.203.79.185) au résolveur sur le système de l'utilisateur.
8. Le résolveur relaie l'adresse à l'application, qui peut alors initier des communications IP avec l'adresse spécifié par l'utilisateur.

1.1.3 Deux problématiques récurrentes

Dans cette partie , nous traitons les deux problématiques auxquels est confronté le DNS

Les vulnérabilités du DNS : Le DNS constitue un élément clé de l'infrastructure, ainsi il est devenu un élément ciblé par les pirates informatiques comme nous pouvons le voir sur la figure 1.3 où est représentée une architecture simplifiée du DNS d'une zone.

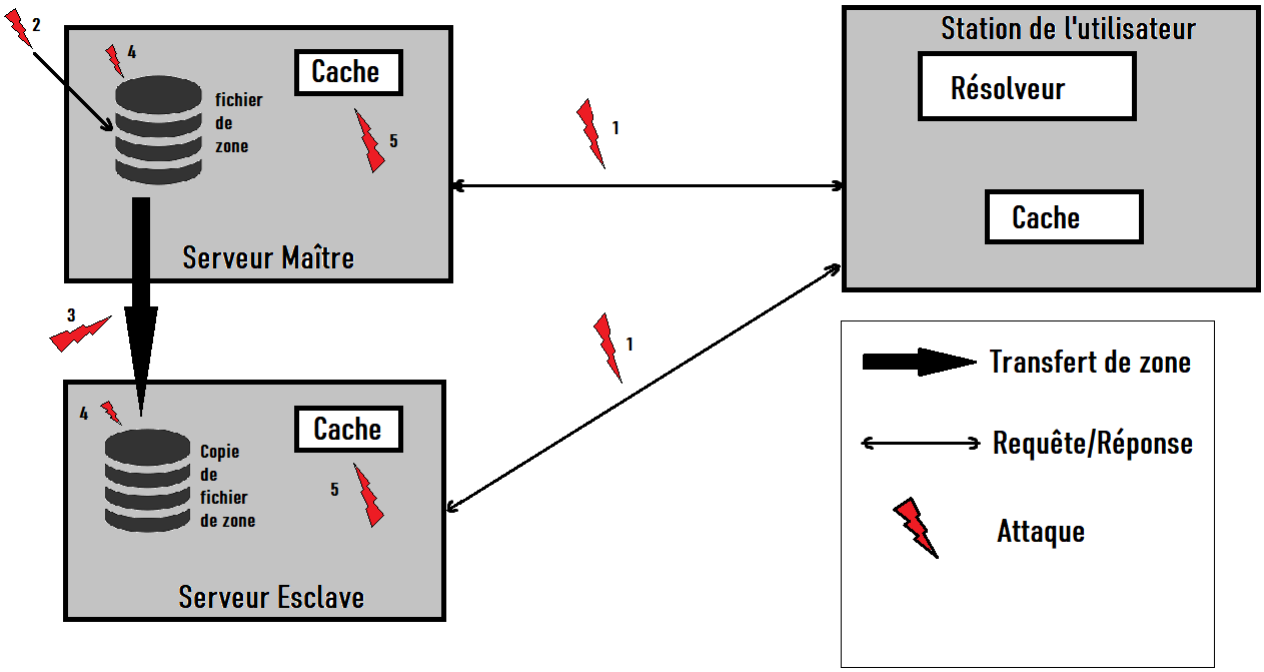

Figure 1.3: Les vulnérabilités du DNS

Selon Monsieur Gilles Guette [1], nous remarquons qu'il existe deux types d'attaques :

— Attaques visant à corrompre les transactions DNS :

Il existe deux types de transactions dans le DNS. Celles qui servent à la résolution des noms (requêtes et réponses) et les transactions de maintenance de la base de données (messages de mise à jour et de transfert de zone)

1) Attaques visant à corrompre les requêtes DNS :

   Les requêtes DNS sont susceptibles aux attaques par n'importe quelle personne ayant la performance adéquate lui permettant ainsi d'avoir accès à la modification des messages DNS

2) Attaques visant à corrompre les messages de mise à jour (MAJ) DNS :

   Quand une nouvelle adresse attribuée, par exemple, subit une modification de topologie, des messages de MAJ, qui sont bien vulnérables, s'envoient systématiquement au serveur primaire dans le but de lui passer l'information. Intercepter un message de MAJ ou créer un message de MAJ complets seraient deux façons pratiques pour exploiter la vulnérabilité des messages de mise à jour.

3) Attaques visant à corrompre les messages de transfert de zone :

   Le fichier de zone original est seulement à la portée du serveur primaire, les serveurs secondaires doivent récupérer ces informations dont le but est d'échapper à la vieillesse de la copie en question. Les méthodes d'attaques sont les mêmes que celles présentées ci-dessus pour les messages de mise à jour : l'interception de message et la création d'un message de transfert de zone.

— Attaques visant les enregistrements stockés sur les serveurs

Cette partie s'intéresse aux données stockées sur les serveurs.

4) Corruption du fichier de zone :

   La zone est considérée comme étant complètement corrompue si nous disposons d'un serveur primaire. Ce dernier est chargé de l'envoi d'une copie du fichier de zone aux serveurs secondaires. Les serveurs de noms hébergeant le fichier de zone ou une copie de celui-ci sont des cibles privilégiées.

   Un autre type d'attaque peut être mené lorsque l'attaquant possède les droits nécessaires sur un serveur de noms primaire, soit parce qu'il les a usurpé ou soit tout simplement parce qu'il en est l'administrateur.

5) Pollution de cache :

   Afin d'optimiser les performances du DNS, l'utilisation d'un cache dans lequel seraient placés les enregistrements DNS auxquels les serveurs ont eu accès récemment, serait une solution bien applicable, c'est justement là où naît un autre type d'attaque appelé pollution de cache, dont l'existence serait grâce à la confiance que donne le serveur aux donnés qu'il a placé en cache.

Problème de centralisation : La liste d'arguments est assez longue que nous pouvons pas tout développer mais juste en prendre quelques-uns :

D'une part, les serveurs racines ne peuvent être contrôlés que par l'ICANN, une organisation à but non lucratif dont les racines sont liées à un seul pays. Cela remet en question le concept de la centralisation du net et a été un argument largement diffusé au cours des trois dernières décennies. D'autre part, nous faisons confiance à un outil ou un service que nous ne connaissons pas, nous ne savons pas aussi ce qu'il peut faire de nos données, ni à qui peut il les confier lui aussi, en effet, selon une étude, conduite par un bureau de statistique appelé « 99Firms » en 2020, montre que le nombre de personnes utilisant Google quotidiennement est supérieur à 1 milliard. Malgré ses nombreuses déclarations, Google est une agence de publicité et la collecte des données des utilisateurs constituent son principal atout, bien que les données collectées à partir des serveurs DNS soient théoriquement impersonnelles, l'entreprise les collecte néanmoins.

1.2 Problématique du projet

Pour résumer ce qui précède, un DNS centralisé est un service obscur, géré par un tiers à qui nous ne devrions pas faire confiance, où nous connaissons l'architecture globale est cible de plusieurs types d'attaques comme :

- DDOS qui rend le service indisponible.
- Spoofing qui redirige le visiteur vers un autre site en changeant l'adresse IP correspondante au nom du domaine.
- Phishing qui permet à une personne de voler les informations sensibles de l'utilisateur, comme les détails de son compte bancaire.

Notre projet, intitulé "Développement d'un DNS décentralisé (DDNS) basée sur Ethereum" se focalise sur l'étude d'une solution qui est à la fois sécurisée et décentralisée, mais avant de contourner ces problèmes avec notre solution, nous devons mentionner les solutions actuellement disponibles et leurs points faibles, c'est le sujet principal de la section suivante.

1.3 Solutions existantes

En Cryptographie, lorsque nous parlons de “sécuriser un échange”, nous souhaitons prêter attention aux 3 services suivants : la confidentialité, l'intégrité et l'authentification.[2] Dans

notre cas, L'authentification et l'intégrité des données sont deux services absolument nécessaires au DNS pour fournir un service sûr. Assurément, nous devons pouvoir vérifier que les données reçues du serveur de noms n'ont pas été modifiées et que le serveur interrogé a envoyé les données. De plus, il faut garantir la disponibilité pour maintenir le bon fonctionnement du système, et enfin il faut avoir un service transparent et décentralisé.

1.3.1 DNS Over HTTPS

L'industrie et les chercheurs ont réagi ces dernières années aux risques inhérents à la sécurité des informations DNS, en se concentrant sur la tunnelisation du trafic DNS sur des protocoles de transport et d'application chiffrés. Un de ces mécanismes, DNS over HTTPS (DoH) place la fonctionnalité DNS directement dans le navigateur Web lui-même pour diriger les requêtes DNS vers un résolveur récursif de confiance via des connexions HTTPS chiffrées.[3]

Les désavantages du DoH L'architecture DoH résout les risques informatiques (par exemple, l'écoute clandestine) mais introduit de nouvelles préoccupations, y compris celles associées à la centralisation des requêtes DNS à l'opérateur d'un seul résolveur récursif qui est sélectionné par le vendeur du navigateur

1.3.2 DNS Over TLS

« DNS over TLS », ou DoT, est une norme pour crypter les requêtes DNS afin de les garder sécurisées et privées. DoT utilise le même protocole de sécurité, TLS, que les sites Web HTTPS utilisent pour crypter et authentifier les communications. TLS est également appelé « SSL ». DoT ajoute le chiffrement TLS au-dessus du protocole de datagramme utilisateur (UDP), qui est utilisé pour les requêtes DNS. De plus, il garantit que les demandes et réponses DNS ne sont pas falsifiées.[4]

Les désavantages du DoT : Du point de vue de la sécurité du réseau, DoT est sans doute excellent. Il donne aux administrateurs réseau la possibilité de surveiller et de bloquer les requêtes DNS, ce qui est important pour identifier et arrêter le trafic malveillant. Mais en vue de garantir la transparence, DoT contredit la philosophie du DNS indiquée ci-dessus.

1.3.3 DNSSEC

DNSSEC renforce l'authentification dans DNS à l'aide de signatures numériques basées sur la cryptographie à clé publique. Avec DNSSEC, ce ne sont pas les requêtes DNS et les réponses elles-mêmes qui sont signées cryptographiquement, mais plutôt les données DNS elles-mêmes sont signées par le propriétaire des données. DNSSEC ajoute deux fonctionnalités importantes

au protocole DNS[5] :

- L'authentification de l'origine des données permet à un résolveur de vérifier cryptographiquement que les données qu'il a reçues proviennent bien de la zone souhaitée.
- La protection de l'intégrité des données permet au résolveur de savoir que les données n'ont pas été modifiées en transit depuis qu'elles ont été initialement signées par le propriétaire de la zone avec la clé privée de la zone.

Les désavantages du DNSSEC : Cette technologie a été inventé par ICANN qui pose un problème en elle-même, la centralisation du DNS.

De plus, elle présente plus de complexité du côté technique ce qui implique plus d'opportunités pour que les choses tournent mal, et enfin pas assez d'utilisateur aujourd'hui, il est peu probable que nous aurons beaucoup plus de protection jusqu'à ce que les noms de domaines populaires décident de signer leurs zones.[5]

Conclusion

Après avoir introduit le DNS, mentionné les problématiques face à cette technologie et précisé la démarche à suivre pour satisfaire nos besoins, nous allons définir, dans le chapitre suivant, la blockchain et prouver que l'ethereum représente un bon candidat pour l'implémentation de notre solution.

Chapitre 2

État de l'art

Plan

Introduction

Étant parlant de l'importance du passage d'un serveur DNS centralisé à un autre décentralisé, la technologie de la blockchain s'impose fortement dans cette étude et une explication dans ce contexte serait évidente pour l'introduire.

2.1 La technologie Blockchain

L'objectif de cette partie est de définir la Blockchain et ses principes fondamentaux.

2.1.1 Définition et explication

Actuellement, la blockchain [1] est un phénomène à la pointe de l'innovation, En fait, la blockchain est une technologie de stockage contenant l'historique complet des échanges effectués entre ses utilisateurs depuis sa création et qui ne fait pas appel à un tiers de confiance d'où son caractère **décentralisé**.

Les informations sont transférées et enregistrées d'une manière sécurisée à travers des procédés cryptographiques qui empêchent les utilisateurs de les modifier ou de les supprimer ce qui nous donne un système immuable. La sécurité est également assurée par le fait que chaque nœud (Node) de la blockchain possède une copie synchronisée de l'historique des transactions ce qui garantit **la disponibilité** et **la transparence** ainsi qu'une forte protection contre la corruption des données. La blockchain se caractérise également par la baisse du coût de transfert comme elle fonctionne sans autorité centrale de contrôle.

Comme son nom l'indique, une blockchain est une chaîne de blocs contenant des informations. Cette technique a été décrite à l'origine dans 1991 par un groupe de chercheurs et était à l'origine destiné à horodater les documents numériques afin qu'il soit impossible de les antidater.

Cependant, elle est restée pratiquement inutilisé jusqu'à ce qu'elle ait été adaptée par "Satoshi Nakamoto" en 2009 pour créer la crypto-monnaie numérique "Bitcoin".

2.1.2 Fonctionnement d'une Blockchain

La blockchain comme écrite indique l'existence de deux termes qui sont indispensable à la compréhension de la manière de fonctionnement de cette technologie commençons par le mot clé **Bloc** comme modélisé ci-dessous dans la figure 2.1 :

1. La blockchain est un registre distribué

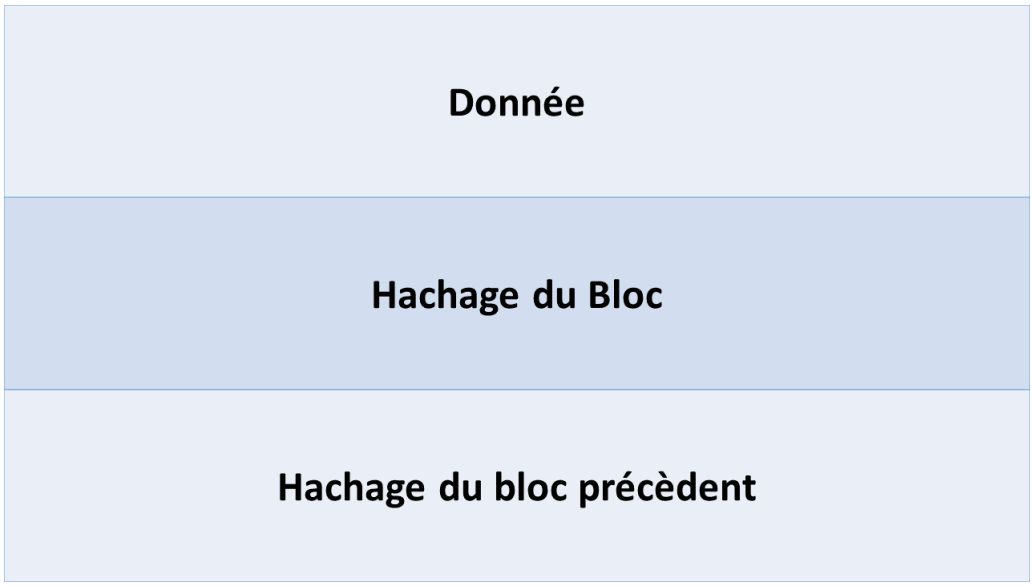

Figure 2.1: Détails du bloc

Regardons de plus près un bloc, chaque bloc contient 3 choses à savoir :

1) **Les données :**

   Les données stockées dans un bloc dépendent du type de blockchain par exemple, on peut stocker les détails d'une transaction tels que l'expéditeur, le destinataire et le montant des pièces.

2) **Le hachage :**

   Un bloc a également un hachage qui lui identifie avec son contenu et c'est toujours unique, tout comme une empreinte digitale. Une fois qu'un bloc est créé, son hachage est calculé. Si on change quelque chose à l'intérieur du bloc cela entraîne le changement du hachage de ce bloc en d'autres termes, les hachages sont très utiles lorsque nous souhaitons détecter les modifications apportées aux blocs.

3) **Le hachage du bloc précédent :**

   Le troisième élément à l'intérieur de chaque bloc est le hachage du bloc précédent, un chaînage des différents blocs va engendrer effectivement une chaîne de blocs qui nous renseigne plus sur le deuxième mot à savoir **chain** et explique la nomination de cette technologie "**Blockchain**".

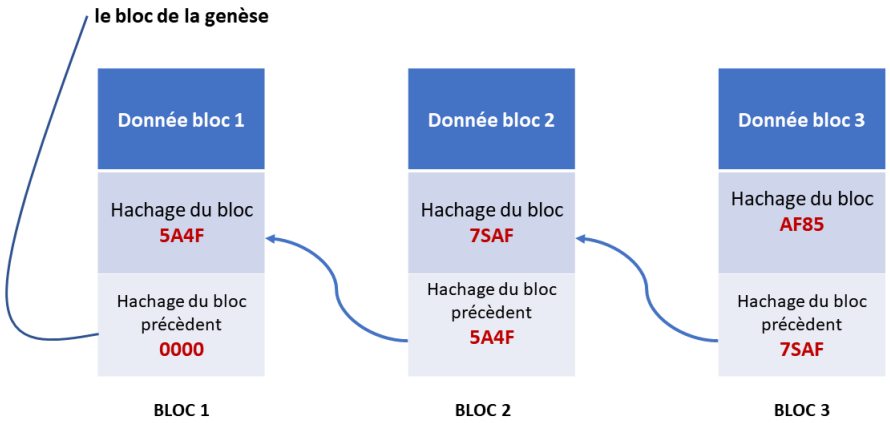

Figure 2.2: Une blockchain composée de trois blocs

Prenons un exemple décrit dans la figure 2.2 ci-dessus. Ici, nous avons une chaîne de 3 blocs et chaque bloc a un hachage et le hachage du bloc précédent donc, le bloc numéro 3 pointe vers le bloc numéro 2 et le bloc numéro 2 pointe vers le bloc numéro 1. Maintenant, le premier bloc est un peu spécial, il ne peut pas pointer vers les blocs précédents parce que c'est le Premier. Nous appelons cela le bloc de la genèse. Disons que nous trafiquons le second bloc. Cela va provoquer le hachage du bloc à changer ainsi qu'à son tour, cela fera le bloc 3 et tous les suivants blocs non valides car ils ne stockent plus un hachage valide du bloc précédent comme indiqué dans la figure 2.3 ci-dessous.

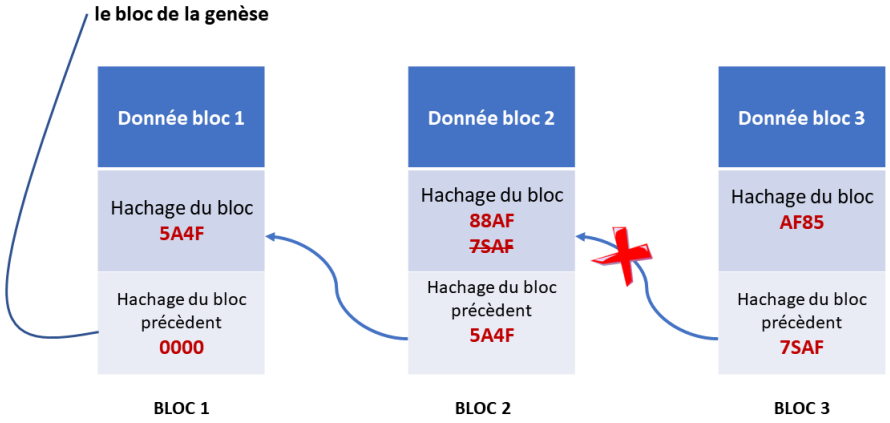

Figure 2.3: Falsification d'un bloc

Donc, changer un seul bloc fera tout ce qui suit blocs invalides. Mais utiliser des hachages ne suffit pas pour empêcher la falsification. De nos jours, les ordinateurs sont très rapides et peuvent calculer des centaines de milliers de hachages par seconde. Nous pouvons efficacement altérer un bloc et recalculer tous les hachages d'autres blocs pour rendre notre blockchain valide à nouveau. Pour atténuer cela, les blockchains ont quelque chose appelé "Proof of Work" [2] (PoW). Dans la section suivante nous abordons le sujet de l'ethereum qui a apporter des améliorations par rapport au technologie "Bitcoin".

2.2 Ethereum

La blockchain Ethereum a été lancé par un programmeur russe-canadien Vitalik Buterin en 2015 qui a déclaré « La blockchain de Bitcoin a été conçue spécifiquement pour des applications monétaires, alors qu'Ethereum permet de créer tout type d'applications »[6].

2.2.1 Smart Contract

L'ethereum permet d'exécuter du code provenant d'applications décentralisées. Ce code permet en particulier la mise en place de Smart Contracts, qui constituent le cœur du potentiel d'Ethereum. Les smart contrats sont des programmes qui exécutent automatiquement des conditions définies au préalable[7]. Leurs implémentions dans une blockchain garantit que les termes du contrat ne pourront pas être modifiés en cours d'exécution. Ceci garantit également la rapidité d'exécution automatique, l'évitement des erreurs et la réduction des coûts et problèmes liés aux intermédiaires. Les smart contracts sont exécutés d'une façon décentralisée à travers une entité appelée EVM (Ethereum Virtual Machine) et c'est en consommant la cryptomonnaie [3] de l'ethereum, l'Ether, avant leur déploiement dans la blockchain. En effet, la force du code interne d'Ethereum, par rapport à celui du bitcoin, est qu'il est Turingcomplet, c'est à dire capable de tout calculer avec suffisamment de temps. Ce code permet d'installer dans chaque nœud du réseau Ethereum une EVM.

2.2.2 Jetons ERC20

Le jeton standard de l'Ethereum (ERC20) est utilisé pour les Smart Contracts sur le réseau Ethereum. ERC20 définit une liste commune des règles qu'un jeton sur ethereum doit suivre.

2. C'est un mécanisme qui ralentit la création d'un nouveau bloc. Dans le cas du "Bitcoin" un bloc prend environ 10 minutes

3. Monnaie numérique en usage sur Internet, indépendante des réseaux bancaires et liée à un système de cryptage

En effet, Ethereum offre aux développeurs la possibilité de programmer la circulation de nouveau jeton dans l'écosystème Ethereum. Les jetons ERC20 utilisent du "gas" [4] pour les frais d'opérations.

2.2.3 La méthode de consensus

Pour assurer cette sécurité et valider les transactions, le système repose sur une "méthode de consensus" (consensus mechanism), à savoir, un nombre de règles qui fait qu'il soit difficile à ajouter de l'information à la blockchain et encore plus difficile de changer ou de pirater ces informations[8]. Il y a plusieurs types de méthodes de consensus, mais la méthode dite "Proof Of Work" (PoW) est jusqu'ici la plus utilisée, quoique Ethereum ait déjà annoncé un changement vers une méthode dite "Proof Of Stake" (PoS) progressivement dans les prochaines années. La méthode de "PoW" consiste à vérifier que chaque bloc contient les bonnes transactions et le bon hash. A chaque bloc un nombre, appelé "nonce" est ajouté à la fin, qui change le hash et rend le travail plus difficile. Les nœuds qui font ce travail doivent essayer un nonce différent jusqu'à trouver le hash qui va avec le bloc pour valider la transaction. Ce travail est rémunéré par des nouvelles cryptomonnaies nouvellement créées. Pour cette raison, les participants à la blockchain qui font ce travail sont appelés les mineurs [5].

2.3 Solution proposée

Après que nous avons détaillé les exigences de notre application dans le premier chapitre et énuméré les défaillances de chaque solution existante. Nous avons pris recours à la technologie Blockchain afin de remédier à l'ensemble des problèmes et plus précisément nous avons choisi d'utiliser la Blockchain Ethereum puisqu'elle nous donne la main à développer notre DNS décentralisé en utilisant les Smarts Contracts. L'ethereum possède aussi de nombreux outils qui nous aideront dans le processus de déploiement de notre Smart Contract. Notre application se compose de deux parties :

— Le Smart Contract qui va être déployé sur la blockchain Ethereum et qui gère tous les opérations d'un serveur de nom de domaine que nous détaillons dans le chapitre suivant dans les besoins fonctionnels.

— Et l'application web qui représente l'intermédiaire entre l'utilisateur et le Smart contract.

4. Une unité de gaz est le plus petit type de travail traité sur le réseau, la validation et la confirmation des transactions sur la blockchain nécessitent une certaine quantité de gaz, selon la taille et le type de chaque transaction

5. Le rôle des mineurs est de sécuriser le réseau Blockchain et de traiter chaque transaction

Conclusion

Nous avons définir les notions de base de la technologie utilisée pour le développement de notre application, cette étude facilitera la tâche de la conception que nous détaillerons dans le chapitre suivant.

Chapitre 3

Conception

Plan

Introduction

Après avoir présenté le projet et compris la technologie qui le sous-tend. Ce chapitre est entièrement dédié à la conception, nous analysons en premier lieu les besoins fonctionnels et non fonctionnels. Nous spécifions ensuite les différents cas d'utilisation de notre application et à la fin, nous illustrons les vues statique et dynamique du système .

3.1 Analyse et spécification des besoins

3.1.1 Analyse des besoins

L'analyse fonctionnelle est une démarche qui consiste à rechercher et à caractériser les fonctions offertes par un produit pour satisfaire les besoins de son utilisateur.

Les besoins fonctionnels : Les besoins fonctionnels sont les fonctionnalités et les actions que le système doit obligatoirement effectuer, ils sont issus du cahier de charges du projet .

L'application à réaliser doit permettre aux éventuels utilisateurs :

- Ajouter un domaine
- Résoudre un nom de domaine
- Modifier l'adresse IP de son propre domaine
- Transférer la propriété de son domaine
- Renouveler le nom du domaine

Les besoins non fonctionnels : Un besoin non fonctionnel est un besoin spécifiant des priorités du système, telles les contraintes liées à l'environnement, à l'implémentation et les exigences en matière de performance, de facilité de maintenance, d'extensibilité et de fiabilité. Ainsi cette application doit répondre à un ensemble de fonctionnalités qui ne sont pas explicitement demandées par l'utilisateur mais qui contribuent à l'amélioration de l'application, à savoir :

- **Le déploiement facile :** L'application doit être facile à déployer. Elle ne doit pas nécessiter l'installation d'outils autre que l'utilisateur typique de blockchain utilise déjà.
- **L'accès rapide :** Il est impérativement nécessaire que le temps de réponse de l'application s'approche le plus possible du temps réel.

- **La performance :** Il est indispensable de prévoir une application performante qui, à travers ses fonctionnalités, répond à toutes les exigences des usagers d'une manière optimale.
- **L'ergonomie et la convivialité :** L'application doit être facile à accéder et ne demande pas un temps d'apprentissage. Elle aura des interfaces simples, ergonomiques et faciles à utiliser.
- **L'extensibilité :** L'application doit permettre la possibilité d'ajout ou de modification de nouvelles fonctionnalités.

3.1.2 Spécification des besoins

Les acteurs et les cas d'utilisation sont les concepts fondamentaux de l'UML pour la spécification des exigences du système. Dans cette section, nous les identifions et les représentons graphiquement à l'aide du diagramme de cas d'utilisation général et ses raffinements.

Identification des acteurs : Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système. Nous avons détecté l'existence de deux acteurs qui interviennent dans les processus de notre application : l'utilisateur et la Blockchain Ethereum.

Identification des cas d'utilisation : Les attentes des utilisateurs ont été spécifiées et pour mieux comprendre les besoins fonctionnels nous présentons les fonctionnalités principales du système sous forme de cas d'utilisation qui regroupent une famille de scénarios possibles selon un critère fonctionnel. Un scénario correspond au flot de messages échangés entre l'acteur et le système. Dans cette section, nous modéliserons les exigences spécifiées précédemment à l'aide d'un diagramme de cas d'utilisation global et une description textuelle.

- **Diagramme des cas d'utilisation général**

La figure 3.1 décrit le diagramme des cas d'utilisation global de notre solution.

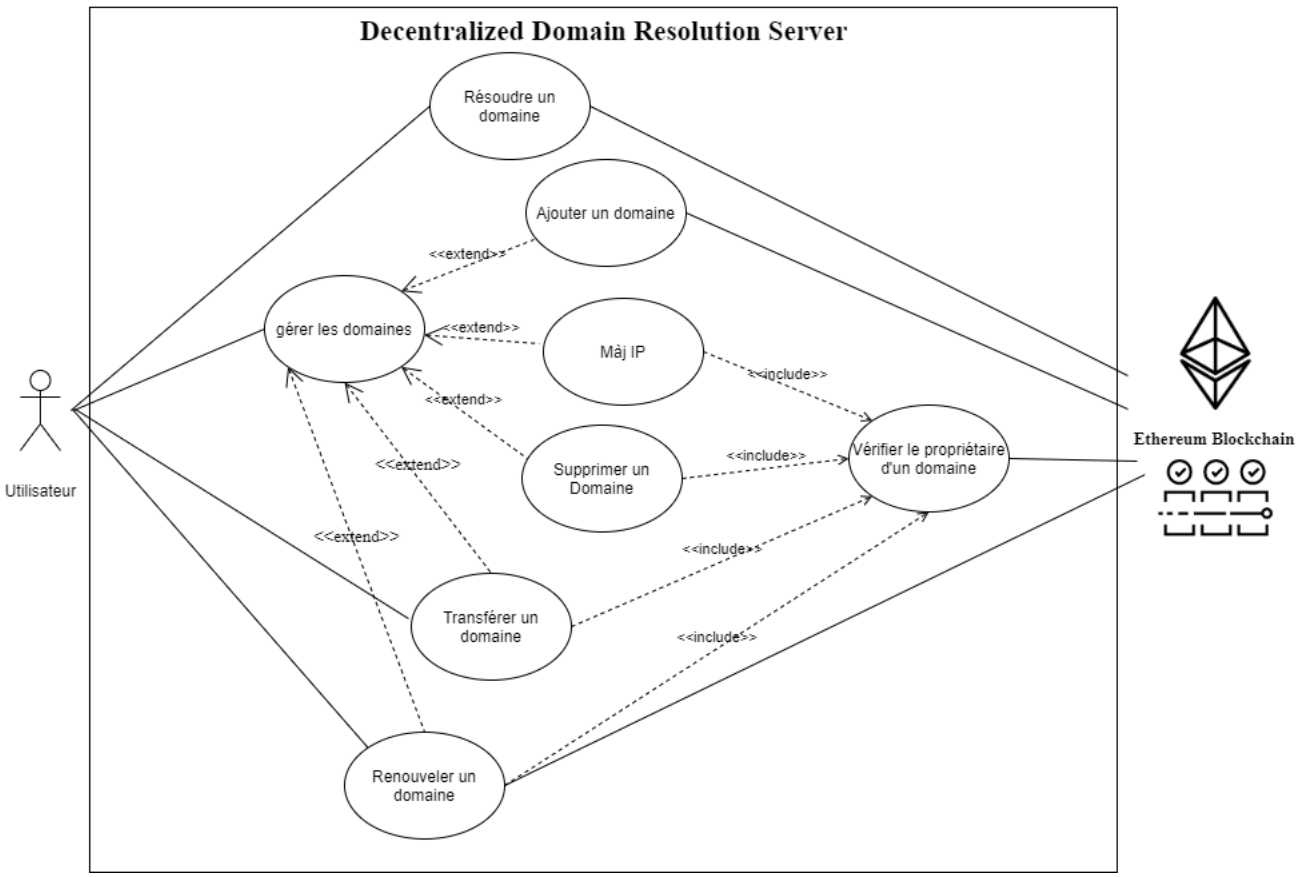

Figure 3.1: Diagramme des cas d'utilisation général

Nous avons traduit à travers le diagramme des cas d'utilisation général divers tâches réalisées par l'utilisateur :

— Le cas d'utilisation "Résoudre un domaine" permet à l'utilisateur de résoudre un nom de domaine sur l'interface de l'application.

— Le cas d'utilisation "Gérer les domaines" permet à l'utilisateur d'ajouter, modifier potentiellement les paramètres des noms de domaine ou de les transférer a un autre propriétaire.

— Le cas d'utilisation "Renouveler domaine" permet à l'utilisateur d'augmenter la période de propriété.

3.2 Vue statique : Diagramme de classes

Le terme vue statique s'applique aux aspects d'un modèle donné ayant un rapport avec les types d'objets qui existent dans le modèle, leur structure interne et leurs relations qui existent. Pour décrire la vue statique de notre application, nous avons choisi le diagramme de classes.Un diagramme de classes UML décrit les structures d'objets et d'informations utilisées sur notre application. Il décrit les informations sans faire référence à une implémentation particulière.

Ses classes et relations peuvent être implémentées de nombreuses manières, dans notre cas nous avons besoin de connaître les fonctions principales et les structures utilisées dans le développement du contrat.

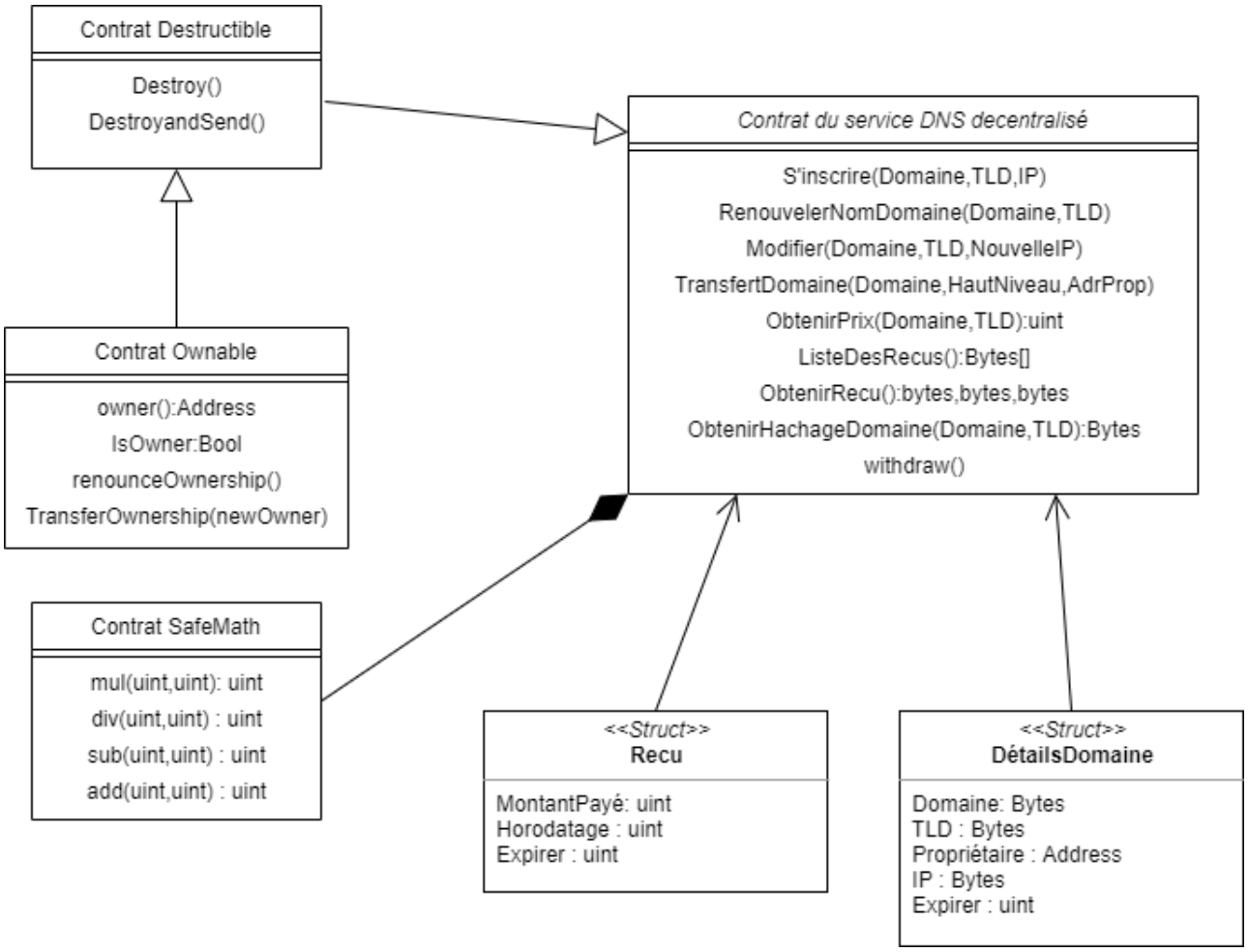

Figure 3.2: Diagramme de Classes

Nous remarquons que la seule classe dont nous aurons besoin dans la figure 3.2 est le contrat du service DNS décentralisé qui nous permet d'exécuter les différentes fonctions de notre application.Il existe déjà des contrats pour les opérations mathématiques, la logique de propriété et la logique de destruction fournis par "OpenZeppelin", qui ont passé plusieurs audits de sécurité, nous n'avons donc pas besoin de les réinventer.

3.3 Vue dynamique : Diagramme d'activité

Le but de la conception dynamique est de donner des scénarios de déroulement des différentes tâches de l'application. Nous avons opté, pour la conception dynamique de notre projet, aux diagrammes d'activité.

3.3.1 Diagramme d'activité d'ajout d'un domaine

La figure 3.3 décrit le processus d'ajout d'un domaine par un utilisateur.

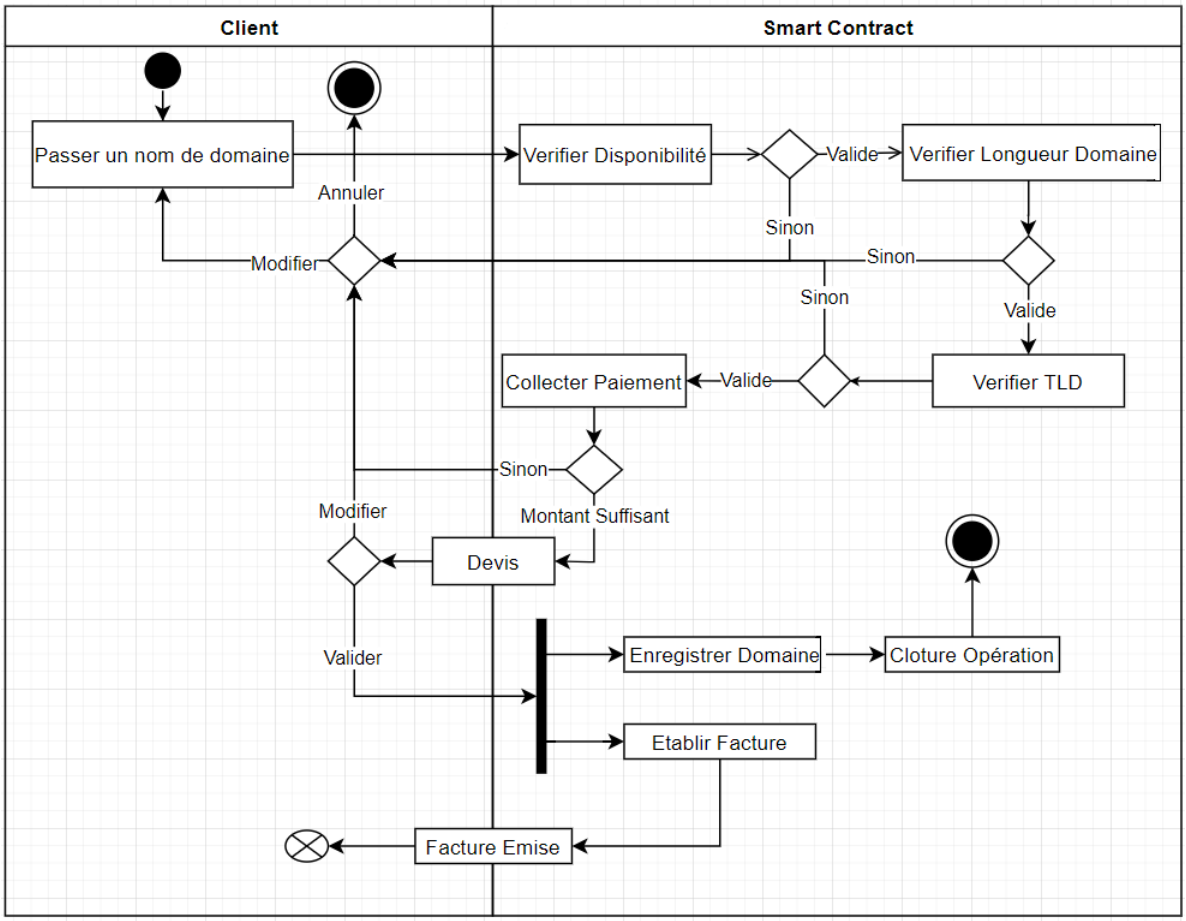

Figure 3.3: Diagramme d'activité d'ajout d'un domaine

Pour organiser le travail et donner la meilleure expérience possible aux différents utilisateurs, la gestion des utilisateurs est primordiale. L'utilisateur doit suivre les séquences suivantes pour accomplir cette opération :

- L'utilisateur essaie un nom de domaine.
- Le "Smart Contract" vérifie la disponibilité,le solde de l'utilisateur, la longueur du domaine et du TLD.
- Si tout va bien, L'utilisateur a le choix entre accepter ou refuser le montant proposé par notre contrat.
- Si l'utilisateur accepte, Le "Smart Contract" enregistre le domaine et établir une facture
- L'utilisateur reçoit la facture et obtient son domaine.

3.3.2 Diagramme d'activité du transfert de la propriété d'un domaine

La figure 3.4 décrit le processus du transfert la propriété d'un domaine a un autre utilisateur.

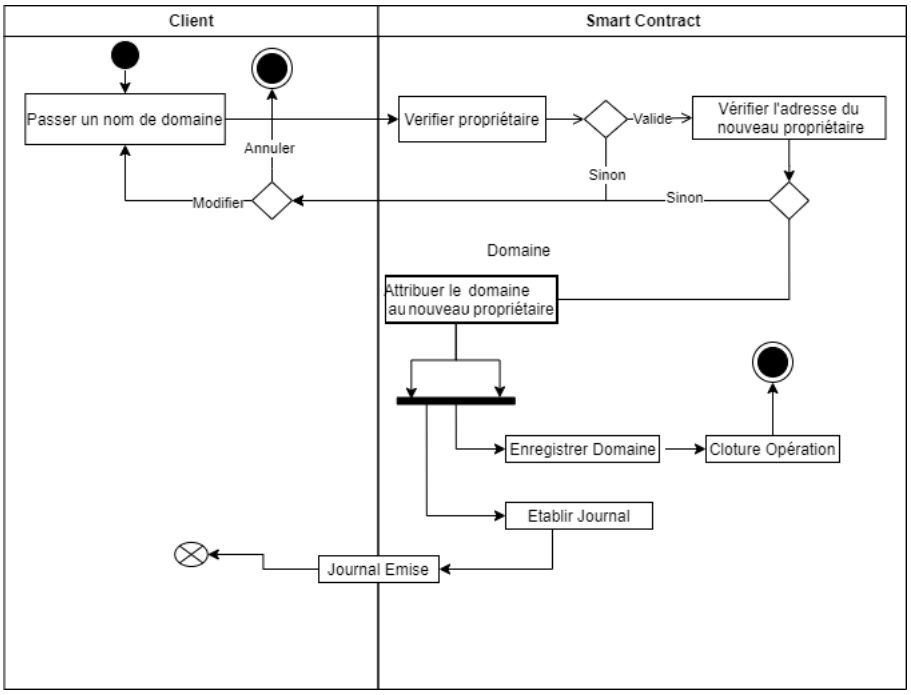

Figure 3.4: Diagramme d'activité du transfert de la propriété d'un domaine

Pour transférer la propriété d'un domaine a un autre utilisateur, l'utilisateur doit suivre les étapes suivantes :

- L'utilisateur passe le nom du domaine et l'adresse du nouveau propriétaire
- Le "Smart Contract" vérifie le propriétaire du domaine et l'adresse du nouveau propriétaire
- Si l'utilisateur est le propriétaire du domaine et l'adresse du nouveau propriétaire existe et elle n'est pas vide, Le "Smart Contract" enregistre les nouveaux changements et établir un journal
- L'utilisateur reçoit le preuve de cet changement.

Conclusion

Au cours de ce chapitre nous avons pu dégager une vue conceptuelle qui englobe toutes les fonctionnalités majeures à réaliser. La nécessité d'une telle étude se révèle très utiles lors de la phase d'implémentation.

Chapitre 4

Réalisation

Plan

Introduction

Ce chapitre traite les aspects techniques liés à l'implémentation et la mise en œuvre de notre système. Nous commençons donc par la présentation de l'architecture globale de notre solution et nous enchaînons par la spécification des différents outils, logiciels, technologies et matériels utilisés et adoptés durant le processus de développement et mise en place de notre système. Nous aborderons ensuite les principales fonctionnalités offertes par notre application à travers l'illustration des différentes interfaces qui vont résumer le fonctionnement global de notre application.

4.1 Architecture de la solution

Nous entamons cette section par l'architecture globale de notre système à travers laquelle nous exposons les différentes parties de notre solution. à la suite de ce chapitre, nous détaillerons chaque partie et les choix technologiques correspondants .

4.1.1 Architecture globale

Comme illustré dans la figure ci dessous, l'architecture de notre système repose essentiellement sur deux parties :

Front-end : Cette partie constitue l'application web qui englobe tous les interfaces graphiques fournis à l'utilisateur afin d'interagir avec notre DAPP. Ces interfaces sont synchronisées avec la blockchain en utilisant le "Redux Store" du Drizzle qui est toujours en écoute pour les changements faits sur le reseau blockchain. Drizzle utilise de sa part la librairie Web3 qui nous donne la possibilité de communiquer avec le réseau blockchain et plus précissement avec notre "Smart contract".

Back-end : Cette partie est bien évidement le réseau blockchain et notre "Smart Contract" qui stocke les données et communique avec notre DAPP en utilisant des interfaces ABI . Mais afin que l'utilisateur puisse s'identifier, il doit impérativement utilise le plugin MetaMask qui lui permet de connecter à son compte personnel et interagir avec notre "Smart Contract".

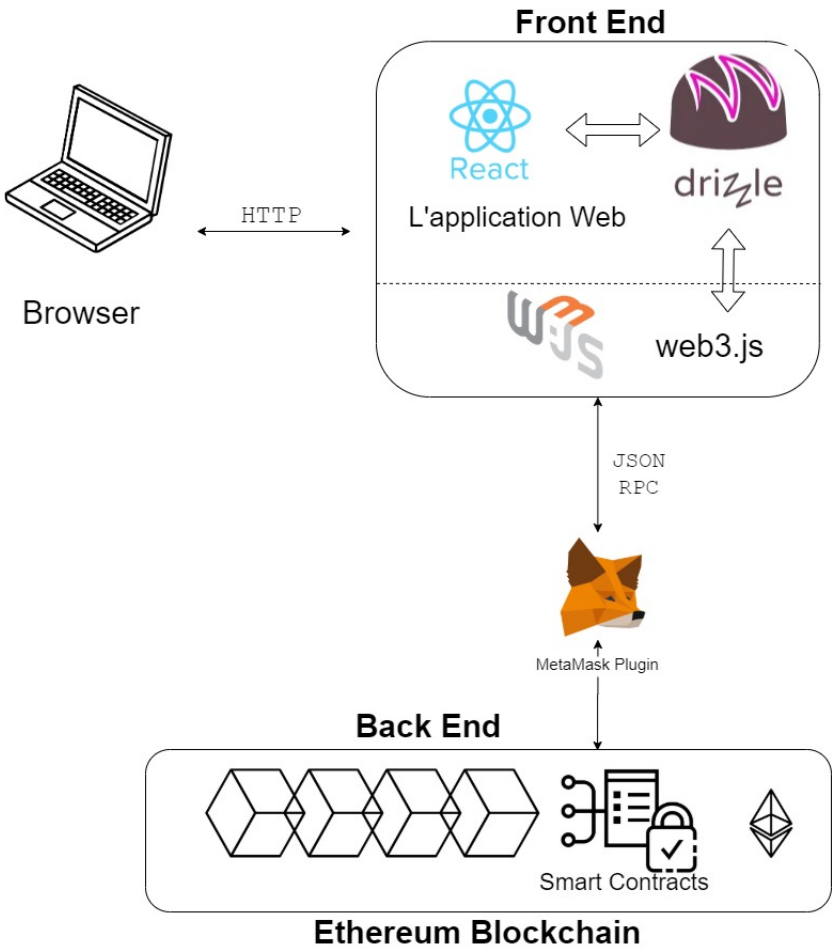

Figure 4.1: Architecture de la solution

4.1.2 Architecture physique

L'architecture que nous avons adapté pour la réalisation de notre solution est de type 3 tiers, ou nous illustrons dans la figure ci dessus les composants de chaque couche.

- Couche 1 : qui est le navigateur web permettant à l'utilisateur d'accéder à notre application hebergée sur le cloud.
- Couche 2 : l'ensemble des interfaces affichées à l'utilisateur connectées avec la librairie web3.js qui synchronise les données avec le réseau blockchain.
- Couche 3 : elle consiste du "Smart Contract" et ses interfaces ABI qui livrent les données à l'application web.

4.2 Choix technologique

Pour le développement de notre solution, nous avons utilisé un certain nombre de technologies et frameworks qui facilitent l'implémentation des différentes parties de notre application.

4.2.1 Ganache

Figure 4.2: Logo de ganache

Le premier outil que nous avons utilisé est Ganache. Ganache est un environnement de la blockchain privé Ethereum qui nous permet d'émuler la blockchain Ethereum afin que nous puissions interagir avec des contrats intelligents dans notre propre blockchain privée. [9]
Voici quelques fonctionnalités que Ganache fournit :

- Affiche la sortie du journal de la chaîne de blocs
- Fournit un contrôle du "mining" avancé
- Explorateur de blocs intégré
- Environnement Ethereum blockchain
- Ganache a une application de bureau ainsi qu'un outil en ligne de commande

4.2.2 Truffle Framework

Figure 4.3: Logo de Truffle

C'est un framework qui nous permet de construire des applications décentralisées sur la blockchain Ethereum. Voici une liste de fonctionnalités qui font de Truffle un outil puissant pour créer des applications décentralisées basés sur Ethereum [10] :

- Prise en charge intégrée pour compiler, déployer et lier des contrats intelligents
- Test de contrat automatisé
- Prend en charge les applications de console ainsi que les applications Web
- Gestion de réseau et gestion de packages
- Console Truffle pour communiquer directement avec les contrats intelligents
- Prend en charge une intégration étroite

4.2.3 MetaMask

Figure 4.4: Logo de MetaMask

MetaMask est un plugin de navigateur facile à utiliser (pour les navigateurs Google-Chrome, Firefox et Brave), qui fournit une interface utilisateur graphique pour effectuer des transactions Ethereum. Il nous permet d'exécuter Ethereum DApps sur notre navigateur sans exécuter un nœud Ethereum complet sur notre système. Fondamentalement, MetaMask agit comme un pont entre Ethereum Blockchain et le navigateur.

4.2.4 OpenZippelin

Figure 4.5: Logo OpenZippelin

OpenZeppelin construit des outils de développement et effectue des audits de sécurité pour les systèmes distribués. Elle a construit la première bibliothèque Open Source au monde pour le développement de contrats intelligents. Il existe déjà des contrats intelligents pour les opérations mathématiques sécurisées, la logique de propriété et la logique de destruction fournis par OpenZeppelin, qui ont passé plusieurs audits de sécurité, nous n'avons donc pas besoin de les réinventer.[11]

4.2.5 Drizzle

Figure 4.6: Logo Drizzle

Drizzle est une collection de bibliothèques utilisées pour créer un front-end facile et meilleur pour Ethereum DApps. L'architecture de Drizzle est complètement modulaire – arrosez autant ou aussi peu que nous le souhaitons. Il existe deux paquets à utiliser avec React, mais la fonctionnalité principale est contenue dans le drizzle module de base lui-même.[12] Les bibliothèques :

- Drizzle : La bibliothèque principale responsable de l'instanciation du Web3, des comptes et des contrats. Faire les synchronisations nécessaires et fournir des fonctionnalités contractuelles supplémentaires.
- drizzle-react : Fournit un composant DrizzleProvider et une méthode d'assistance drizzleConnect pour faciliter la connexion de Drizzle à notre application React.
- drizzle-react-components : Une bibliothèque de composants utiles pour les fonctions dapp courantes. Inclut actuellement ContractData, ContractForm et LoadingContainer.

4.2.6 React JS

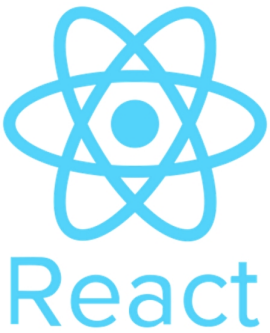

Figure 4.7: Logo React

React se présente comme une bibliothèque JavaScript pour créer des interfaces utilisateurs. [13]

- Declarative : React facilite la création d'interfaces utilisateur interactives. Concevez des vues simples pour chaque état de notre application et React mettra à jour et restituera efficacement les composants appropriés lorsque nos données changent. Les vues déclaratives rendent notre code plus prévisible, plus simple à comprendre et plus facile à déboguer
- Basé sur virtual-DOM : Nous créons des composants encapsulés qui gèrent leur propre état, puis composons-les pour créer des interfaces utilisateur complexes. Étant donné que la logique des composants est écrite en JavaScript au lieu de modèles, nous pouvons facilement transmettre des données riches via notre application et garder l'état hors du DOM

4.3 Présentation de la solution

Dans cette section, nous présentons les différentes interfaces de notre application.

4.3.1 L'interface d'ajout d'un nom de domaine

Le processus d'ajout d'un domaine à notre DDNS se compose de 3 étapes :

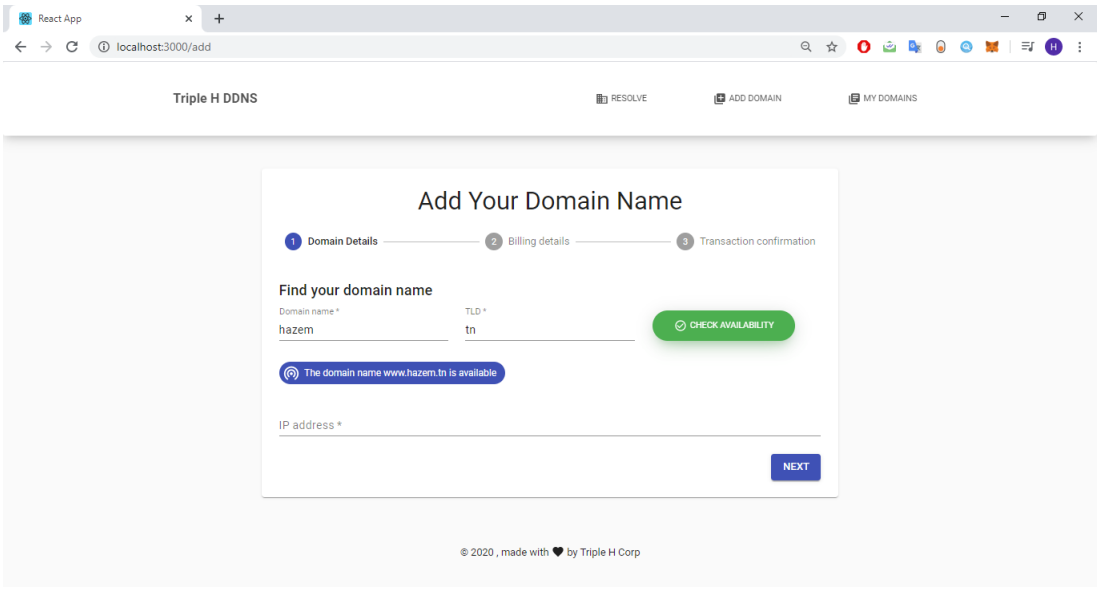

Figure 4.8: Interface de choix du domaine

La première étape est la vérification de la disponibilité du nom de domaine choisi par l'utilisateur et s'il est disponible l'utilisateur saisie l'adresse IP et passe à l'étape suivante.

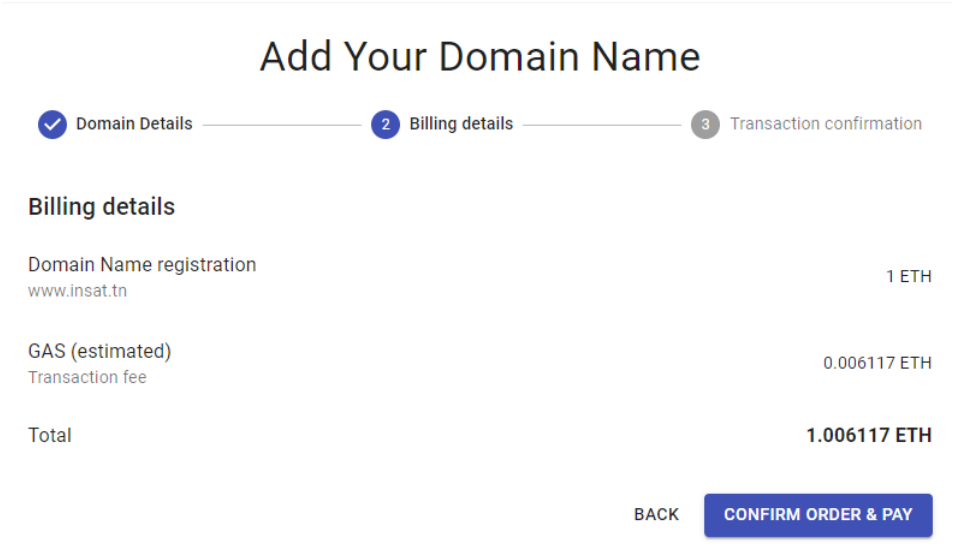

Figure 4.9: Interface de facturation du domaine

Durant cette étape, l'application demande le prix du domaine d'après le Smart Contract et l'affiche à l'utilisateur avec une estimation du coût de Gas et attend l'approbation.

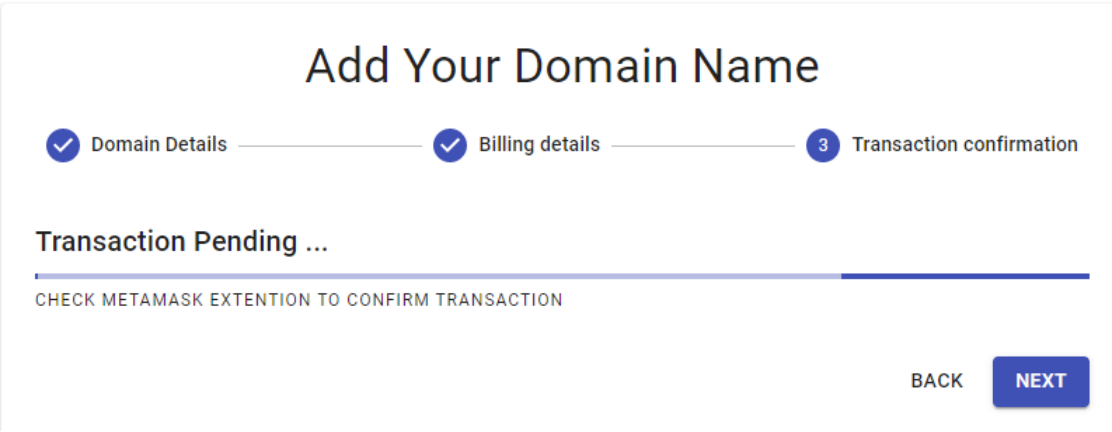

Figure 4.10: Attente de la confirmation de la transaction

Cette étape est la partie du paiement, elle commence par l'envoi de la transaction et l'utilisateur doit la confirmer avec le plugin MetaMask. Après que la transaction soit confirmée, on affiche le reçu.

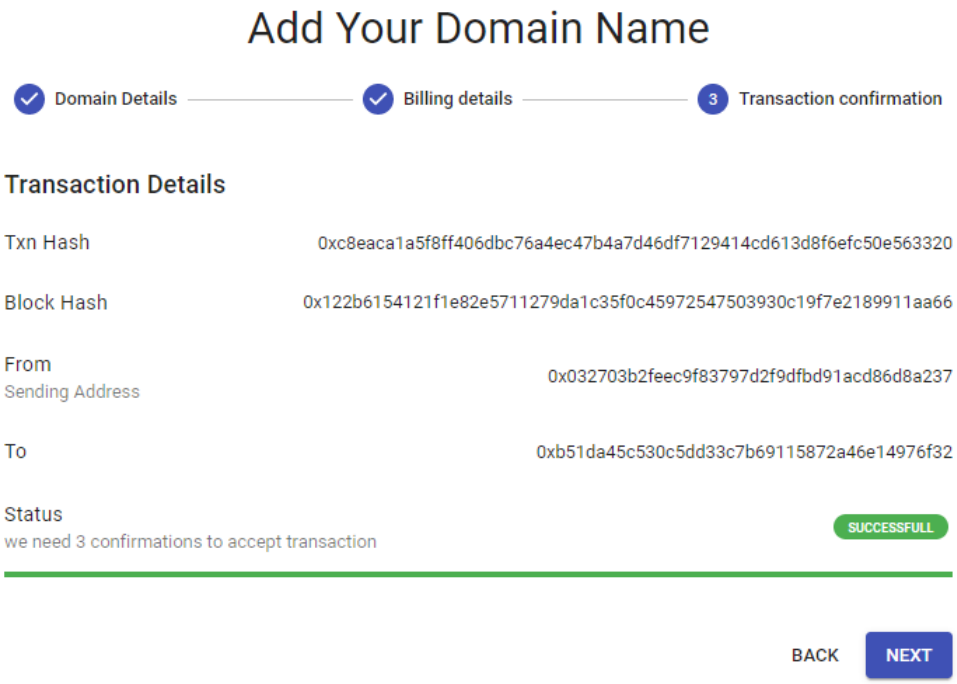

Figure 4.11: Affichage du reçu de la transaction

La figure 4.11 représente le succès de la transaction et l'affichage du reçu de la transaction.

4.3.2 L'interface de résolution d'un nom de domaine

Figure 4.12: Résolution d'un nom de domaine

La figure 4.12 représente l'interface de résolution d'un nom de domaine. Où l'utilisateur saisie un nom de domaine et l'application affiche l'adresse IP correspondante au domaine s'il est enregistré sur le réseau blockchain. Sinon il affiche un message d'erreur.

4.3.3 L'interface de gestion des noms de domaines

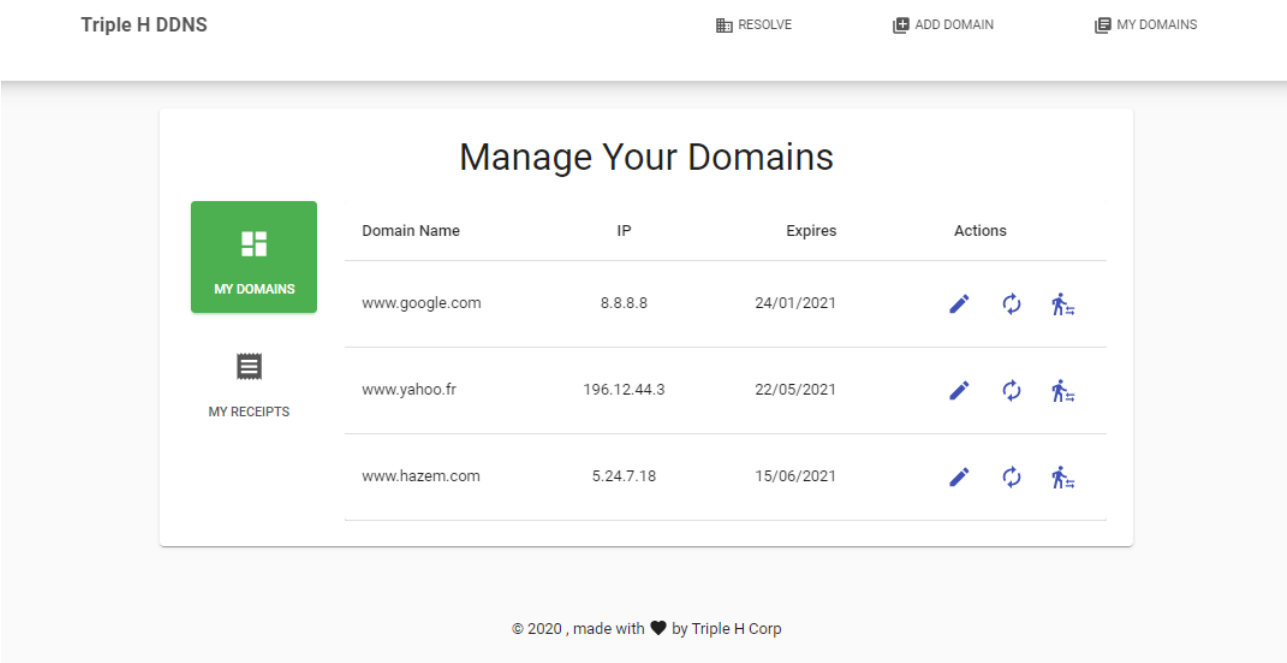

Figure 4.13: Interface de gestion des noms de domaines

La figure 4.13 représente l'interface de gestion des noms de domaines où l'utilisateur peut modifier l'adresse IP, renouveler le domaine et transférer la propriété des noms de domaines appartenant à l'utilisateur. D'autre part, cette section de l'application permet de consulter l'historique des reçus du paiement des domaines.

Conclusion

Dans ce chapitre, nous avons complété la phase de mise en œuvre de notre application. En effet, nous avons présenté l'architecture générale de l'application ainsi que les différents technologies et outils de développement utilisés. Ensuite, nous avons détaillé finalement la réalisation des différentes fonctionnalités telles que conçues dans le chapitre précédant tout en présentant des captures de notre application.

Conclusion générale

La compréhension des nouvelles technologies et de leur impact est une condition préalable pour pouvoir innover et faire évoluer. Dans ce contexte, nous avons fait une étude de l'existant sur les systèmes DNS et leurs problèmes ensuite nous avons enchaîné par une étude sur les différentes solutions existantes et les axes dont elles cherchent à améliorer. En se basant sur ces études théoriques, nous avons développé notre solution d'un système DNS décentralisé en utilisant la technologie Ethereum de la blockchain afin de remédier aux problèmes étudiés. Nous avons dégagé en premier lieu les acteurs qui vont agir sur notre système ainsi que les différents besoins fonctionnels et non fonctionnels de notre solution. Ensuite nous avons entamé la phase de la conception pour modéliser ces besoins en élaborant le diagramme de classe suivie par les diagrammes d'activité. Dans la dernière phase, nous avons évoqué les différentes technologies utilisées ainsi que l'implémentation de notre système. Même si, le nombre d'utilisateur du réseau Ethereum et de la blockchain en générale ne cesse d'augmenter, il reste inférieur au nombre ciblé par les serveurs DNS.

En guise de conclusion, on dirait que notre projet se situe dans une voie en pleine expansion vu le progrès assez rapide et continu de la technologie blockchain.

Bibliographie

[1] Les faiblesses du dns. [En ligne]
Disponible sur : http://people.irisa.fr/Gilles.Guette/pdf/sar2003.pdf [Accès le 18 Avril 2020].

[2] R. Dumont, *Cryptographie et Sécurité informatique Start Guide*, 2009.

[3] Dns over https. [En ligne]
Disponible sur : https://www.cloudflare.com/learning/dns/dns-over-tls/ [Accès le 19 Avril 2020].

[4] Dns over tls. [En ligne]
Disponible sur : https://www.cloudflare.com/learning/dns/dns-over-tls/ [Accès le 19 Avril 2020].

[5] Dnssec. [En ligne]
Disponible sur : https://dnsinstitute.com/documentation/dnssec-guide/ch06s06.html [Accès le 20 Avril 2020].

[6] Histoire ethereum. [En ligne]
Disponible sur : https://www.blockchainfrance.net/2016/03/04/comprendre-ethereum/ [Accès le 5 Juin 2020].

[7] Smart contract. [En ligne]
Disponible sur : https://www.numerama.com/business/272641-ethereum-tout-savoir-sur-lacrypto-monnaie-et-ses-contrats-intel/ [Accès le 6 Juin 2020].

[8] Smart contract. [En ligne]
Disponible sur : https://www.academia.edu/39915497/Les_smart_contracts_et_lex%C3%A9cution_des_contrats_daffaires_internationaux [Accès le 6 Juin 2020].

[9] W. S. Xun Wu, *Blockchain Quick Start Guide*, 2018.

[10] Truffle. [En ligne]
Disponible sur : https://www.trufflesuite.com/docs/truffle/overview [Accès le 20 Mars 2020].

[11] Openzeppelin. [En ligne]
Disponible sur : https://github.com/OpenZeppelin/openzeppelin-contracts [Accès le 20 Mars 2020].

[12] Drizzle. [En ligne]

Disponible sur : https://www.trufflesuite.com/docs/drizzle/overview

[13] React js. [En ligne]

Disponible sur : https://github.com/facebook/react/

ملخّص

في إطار مشروع ختم السنة في المعهد الوطني للعلوم التطبيقيّة و التكنولوجيا، نقدم هذا التقرير الذي يمثل خلاصة عملنا الذي يهدف إلى تصميم و تطوير نظام أسماء نطاقات لامركزي يرتكز على تكنولوجية البلوكتشين و على شبكة الإيثيريوم

كلمات مفاتيح : نظام أسماء نطاقات لامركزي ، بلوكتشين، إيثيريوم

Résumé

Dans le cadre de notre projet de fin d'année à l'Institut National des sciences appliquées et technologie, nous présentons ce document qui constitue la synthèse de notre travail dont l'objectif est de concevoir et développer un DNS décentralisé basé sur la technologie Blockchain et le réseau Ethereum .

Mots clés : DDNS, Blockchain, Ethereum, Smart Contract

Abstract

As part of our end-of-year project at the National Institute of Applied Sciences and Technology, we present this document which constitutes the synthesis of our work whose objective is to design and develop a decentralized DNS based on the Blockchain technology and the Ethereum network.

Keywords : DDNS, Blockchain, Ethereum, Smart Contract