Nexium

Cahier des charges

Jean HERAIL
jean.herail@epita.fr

Milo DELBOS
milo.delbos@epita.fr

Antonin BESSIÈRES
antonin.bessieres@epita.fr

William VALENDUC
william.valenduc@epita.fr

Sommaire

1 Introduction

Le projet **Nexium** vise à développer une monnaie virtuelle décentralisée spécifiquement conçue pour les étudiants d'Épita. Chaque utilisateur disposera d'une paire de clés cryptographiques **RSA** (privée et publique) associée à son adresse *@epita.fr*, garantissant l'authentification et la sécurité des transactions de crédits entre les participants. Grâce à une architecture distribuée, Nexium assurera la transparence et l'intégrité des transactions tout en offrant une interface graphique simple et accessible.

La monnaie du réseau Nexium, le **NEX**, sera utilisée pour effectuer des transactions entre les utilisateurs, qui pourront consulter leur solde et leur historique de transactions de manière publique. Le réseau Nexium fonctionne comme une blockchain ouverte liée directement à Épita, où chaque utilisateur peut constituer un nœud du réseau.

Le projet se décompose en trois axes interdépendants : la gestion des communications, vérification et synchronisation des blocs pour la blockchain Nexium (le réseau), le développement des fonctionnalités graphiques, gestion des comptes, virements et consultation des blocs pour l'interface utilisateur, et le développement des fonctionnalités de chiffrement, déchiffrement, signature et vérification des données pour les implémentations cryptographiques, en utilisant le chiffrement **RSA** selon le standard **GPG** pour la cryptographie asymétrique et le standard **SHA-256** pour les fonctions de hachage, toutes ces fonctions étant implémentées en interne sans recours à des bibliothèques externes.

2 État de l'art et choix techniques

2.1 Architecture du réseau

Afin de garantir la distribution du réseau, nous avons opté pour un protocole de type **blockchain**. Cette architecture, basée sur un registre immuable de blocs incrémentalement liés, nous permet d'assurer une large distribution, et donc une décentralisation plus flexible des informations du réseau. Le projet Nexium se décline en deux binaires : un **client** et un **serveur**.

Ce dernier permet à n'importe quel utilisateur du réseau de participer à sa robustesse, en apportant sa capacité de stocker, diffuser, et valider les blocs. C'est cette force communautaire qui nous a fait choisir cette méthode décentralisée : un tel fonctionnement par réseau fédéré permettra à Nexium d'être toujours viable, même si des serveurs venaient à être hors-ligne.

Enfin, plus le nombre de nœuds (serveurs) du réseau sera élevé, plus ce dernier sera sûr et stable, grâce à la fonction validatrice que prend chaque serveur.

2.2 Sécurisation cryptographique

Il existe, pour l'implémentation du RSA, un crate dédié en rust. Il met à disposition les fonctions de génération à commencer par la création des clés, qui utilise des formules mathématiques précises suivies par le mécanisme de chiffrement du message et de son déchiffrement (principe d'utilisation du modulo des mathématiques). Nous avons implémenté ces fonctions nous-même, tout d'abord par principe de logique et connaissances retenues, autrement dit, pour comprendre l'essence du chiffrement et exactement pouvoir utiliser ses fonctionnalités en les assimilant au maximum. Toute l'équipe du projet est motivée par ce défi, celui de réimplémenter de zéro le standard RSA. Ce défi s'annonce complexe et formateur; Aussi devrons-nous implémenter le hashage SHA256 et le chiffrement AES. Les fondamentaux de Nexium, sur lesquels nous nous concentrons majoritairement pour l'instant, s'annoncent donc comme de longs parcours des différentes méthodes cryptographiques standards.

Ensuite, parce que le RSA propose une certaine simplicité d'implémentation qui traite avec des nombres premiers, tandis que l'ECDSA nécessite d'utiliser le DLP (problèmes logarithmiques discrets) et des courbes elliptiques pour la signature digitale. Pour comparer les deux méthodes citées, RSA et ECDSA, on peut parler de plusieurs points; La taille des clés RSA est nettement supérieure à la taille des clés de la méthode ECDSA pour atteindre le même palier de sécurité de 128-bit, impactant la rapidité qu'ont les programmes à s'effectuer par rapport à leur complexité mathématique. En conclusion sur leur "rivalité" la RSA se démarque par sa simplicité à implémenter parmi les autres méthodes de cryptographie asymétrique, ce qui cause beaucoup moins de problèmes de sécurité dans le cas où il n'est pas parfaitement implémenté.

L'utilisation la plus répandue et forte du RSA est principalement dans les protocoles comme les clés SSH (Secure Shell) utilisées pour un transfert d'informations et/ou de fichiers entre un receveur et un envoyeur, elles sont présentes dans beaucoup de cas dans les réseaux comme GitHub ou similaires et permettent une transmission efficace et facile d'accès au plus grand nombre. On retrouve aussi le RSA dans l'OpenPGP (signatures des e-mails) ou bien dans un usage auquel ont recours une majorité de sites internet, les SSL (Secure Socket Layer) qu'on retrouve dans HTTP / HTTPS soit le chiffrement et la validation de l'intégrité des données, il sera suivi par le protocole TLS, plus moderne qui recouvre également le domaine de la sécurité sur internet mais couvrant un plus grand rôle (emails, sécurité des webapps, VoIP, messages et relations entre webapps et serveurs).

Le chiffrement RSA implique donc une utilisation considérable sur la globalité d'internet et des communications mondiales, le ECDSA est dans son cas nettement plus utilisé dans les transactions de Bitcoin, chaque transaction doit être "signée" lorsqu'elle est effectuée pour vérifier l'autorité de l'utilisateur sur les fonds. Mais aussi pour la sécurité des communications en ligne comme les données échangées entre utilisateurs et sites web et enfin dans les systèmes embarqués et les dispositifs

IoT comme dans les capteurs où les contrôleurs, son utilisation est principalement dans la recherche de mise à jour des firmwares et vérification de leur sûreté.

2.3 Récupération des clés publiques

Pour la récupération des clés publiques, nous avons envisagé deux méthodes principales. La première méthode, via la blockchain, est couramment utilisée par de nombreuses cryptomonnaies. Par exemple, Bitcoin, Ethereum et Litecoin utilisent toutes des clés publiques et privées pour sécuriser les transactions. Ces systèmes permettent de garantir l'authenticité et la sécurité des échanges grâce à des mécanismes cryptographiques robustes.

En utilisant la blockchain, chaque transaction est enregistrée de manière transparente et immuable, ce qui assure une traçabilité complète et une protection contre les fraudes. De plus, la nature décentralisée de la blockchain élimine le besoin d'une autorité centrale, ce qui renforce la sécurité et la résilience du réseau. Les utilisateurs peuvent ainsi avoir confiance dans l'intégrité des transactions sans avoir à se fier à une entité unique.

Les clés publiques et privées sont des éléments essentiels de la cryptographie à clé publique, qui repose sur des fonctions mathématiques complexes. Par exemple, dans le cas de Bitcoin, la clé publique est calculée à partir de la clé privée à l'aide de l'algorithme ECDSA (Elliptic Curve Digital Signature Algorithm). Cette méthode assure que les transactions ne peuvent être falsifiées, car il est pratiquement impossible de dériver la clé privée à partir de la clé publique.

Cependant, nous avons finalement retenu la méthode de récupération des clés publiques via Gitlab Epita. Cette approche présente plusieurs avantages. Contrairement aux blockchains publiques, elle ne nécessite pas de mécanismes de preuve de travail ou de preuve d'enjeu , ce qui simplifie grandement le processus. De plus, les utilisateurs peuvent facilement mettre à jour leurs adresses sans nécessiter de consensus global, offrant ainsi une plus grande flexibilité.

Cette méthode est également exclusive aux étudiants d'Epita, garantissant une utilisation contrôlée et sécurisée. En utilisant Gitlab Epita comme entité d'autorité, nous bénéficions d'une gestion des clés simplifiée tout en maintenant un haut niveau de sécurité.

Gitlab Epita permet une gestion centralisée des clés, facilitant ainsi la maintenance et la mise à jour des systèmes de sécurité. Cette centralisation permet également de simplifier les processus administratifs et de garantir une meilleure conformité aux politiques de sécurité de l'institution. Par exemple, les étudiants peuvent générer une paire de clés SSH et publier leur clé publique sur leur profil Gitlab, ce qui permet une authentification sécurisée et une gestion efficace des accès.

2.4 Interface utilisateur

Pour le développement de l'interface utilisateur dans le projet, nous avons regardé plusieurs frameworks possibles d'utilisation avant de faire un choix sur celui que nous utiliserons. Iced est un framework Rust pour le développement d'interfaces graphiques. Bien qu'il soit utilisé dans des projets comme Druid et Relm, nous avons estimé qu'il ne répondait pas parfaitement à nos besoins. Iced est flexible et est souvent apprécié pour sa simplicité d'utilisation, mais il ne correspondait pas aux exigences spécifiques de notre projet à cause de ses performances et de compatibilité.

Iced est connu pour sa capacité à créer des interfaces utilisateur très créatives et modernes avec l'utilisation de la méthode déclarative. Cependant, ses limitations en matière de performance et de compatibilité avec certains systèmes d'exploitation ont été des facteurs déterminants dans notre décision. De plus, la communauté autour d'Iced, bien que croissante, n'est pas aussi vaste que celle d'autres frameworks, ce qui peut poser des défis en termes de support et de ressources.

Pour ce qui concerne Tauri, nous l'avons écarté aussi car bien que permettant de créer des applications de bureau légères et sécurisées mais aussi utilisé dans certains grands projets comme LogRocket et Awesome. Nous avons préféré nous orienter vers une solution plus adaptée à notre projet. Tauri se distingue par sa capacité à créer des applications de petite taille. Cependant, l'obligation d'utiliser des technologies web comme HTML, CSS et JavaScript est un souci pour la réalisation du projet devant être réalisé avec Rust. Il est aussi possible d'ajouter que l'intégration de Tauri avec certains outils de développement peut être complexe et nécessiter des configurations supplémentaires.

Finalement, nous avons orienté notre choix vers Slint pour plusieurs raisons. Slint fonctionne directement avec Rust sans besoin de l'utilisation de HTML ou autres, ce qui simplifie son utilisation. Il est, de plus, compatible avec plusieurs systèmes d'exploitation (Windows / Linux), et possède une facilité de déploiement. Il est aussi nécessaire de rajouter que Slint est en grande partie reconnu et utilisé dans l'industrie utilisant Rust, ce qui assure une certaine sérénité sur son utilisation ainsi que de la fiabilité deux arguments non négligeable dans ce projet. Slint permet de développer des interfaces utilisateur avec une méthode déclarative, ce qui permet de simplifier le développement et améliore la maintenabilité du code.

Slint se distingue grâce à ses performances élevées et sa grande flexibilité, tout en restant simple à utiliser. Son utilisation dans l'industrie témoigne de sa robustesse et de sa flexibilité, des qualités essentielles pour le succès du projet Nexium. De plus, la communauté active autour de Slint offre un support précieux et une documentation fournie pour les développeurs.

2.5 Chiffrement des clés privées

Le chiffrement des clés privées repose sur plusieurs algorithmes conçus pour assurer une protection efficace des données sensibles. AES (Advanced Encryption Standard) est le plus utilisé, adopté comme standard pour sa rapidité et sa robustesse face aux attaques modernes. Il fonctionne sur des blocs de 128 bits avec des clés allant jusqu'à 256 bits, garantissant un chiffrement fiable et performant.

Twofish propose une alternative flexible avec une architecture pensée pour la vitesse et une adaptation aux processeurs modernes, tandis que Serpent privilégie une approche plus conservatrice avec 32 rounds de transformation, le rendant plus sécurisé mais aussi plus lent à exécuter. Blowfish, quant à lui, repose sur un chiffrement à clé variable pouvant aller jusqu'à 448 bits, offrant une bonne résistance aux attaques malgré son âge. Ces algorithmes sont intégrés dans plusieurs logiciels utilisés pour sécuriser les fichiers, les communications et les données stockées.

GPG (GNU Privacy Guard) exploite ces technologies pour chiffrer et signer des fichiers, des emails et des communications sensibles. Il repose sur un système hybride combinant AES, Twofish ou Serpent pour le chiffrement des fichiers et RSA ou ECC pour la gestion des clés. Son format OpenPGP le rend compatible avec de nombreux systèmes, mais sa gestion des clés reste complexe pour les non-initiés. OpenSSL, de son côté, est largement utilisé pour la sécurisation des communications sur Internet via les protocoles SSL/TLS, qui garantissent la confidentialité des échanges grâce à AES, Blowfish ou ChaCha20.

Il est la référence en matière de cryptographie réseau, bien qu'il soit plus orienté vers les développeurs et administrateurs. D'autres logiciels se concentrent sur la protection des données locales. VeraCrypt et LUKS assurent le chiffrement des disques et partitions en utilisant AES, Twofish et Serpent, empêchant tout accès non autorisé aux fichiers stockés. VeraCrypt se distingue par ses volumes cachés, permettant une couche supplémentaire de protection, tandis que LUKS est privilégié dans les environnements Linux pour sa gestion avancée des clés et sa compatibilité avec plusieurs utilisateurs.

BitLocker, intégré à Windows, mise sur AES pour chiffrer les disques de manière transparente en exploitant TPM, offrant une solution simple mais moins flexible. Enfin, Age propose une approche plus moderne avec ChaCha20 et X25519, simplifiant le chiffrement des fichiers pour ceux qui recherchent une alternative plus accessible à GPG. Ces solutions répondent à des besoins différents, qu'il s'agisse de protéger des fichiers stockés, sécuriser des échanges en ligne ou garantir l'intégrité des communications.

2.6 Architecture réseau

Concernant l'architecture réseau, certaines blockchains se basent sur une architecture centralisée, ce qui apporte à la fois des avantages et des inconvénients. En effet, un nombre restreint de noeud de confiance facilite la validation des blocs et ainsi fluidifie le réseau en permettant une haute vitesse de transaction.

Un des inconvénients se trouve justement dans le principe d'un réseau limité en terme de noeuds favorisant un potentiel risque de censure ou de prise de contrôle par un faible nombre d'acteurs. On trouve notamment XRP (Ripple) qui utilise un mécanisme de ce type, où les transactions sont vérifiées par un nombre restreint de noeuds.

Au contraire, une architecture décentralisée favorise une grande sûreté contre les pannes. Dans le cas où un serveur ne répondrait plus, les autres noeuds du réseau peuvent prendre le relais immédiatement sans problème. Néanmoins, le coût de cette redondance se trouve au niveau des transactions. Comme chaque noeud indépendant, au moment d'ajouter un bloc, tous les noeuds communiquent entre pour valider de manière unanime avant d'effectuer une quelconque action.

De même, les noeuds doivent pouvoir se synchroniser et réagir correctement en cas de modification volontaire qui aurait pour but de déstabiliser le réseau. Également un autre point qui n'est pas négligeable est la consommation énergétique bien plus élevée induite par le nombre de serveurs.

Notre choix s'est basé sur la sûreté de la décentralisation, pour ne pas dépendre d'une base unique et éviter par la même occasion les attaques ciblées. Nous avons également étudié les contraintes liée au grossissement de réseau, et avons trouvé des pistes d'amélioration en termes de scalabilité comme les Directed Acyclic Graphs (DAG).

Afin de garantir la distribution du réseau, nous avons opté pour un protocole de type blockchain. Cette architecture, basée sur un registre immuable de blocs incrémentalement liés, nous permet d'assurer une large distribution, et donc une décentralisation plus flexible des informations du réseau. Le projet Nexium se décline en deux binaires : un client et un serveur. Ce dernier permet à n'importe quel utilisateur du réseau de participer à sa robustesse, en apportant sa capacité de stocker, diffuser, et valider les blocs. C'est cette force communautaire qui nous a fait choisir cette méthode décentralisée : un tel fonctionnement par réseau fédéré permettra à Nexium d'être toujours viable, même si des serveurs venaient à être hors-ligne. Enfin, plus le nombre de nœuds (serveurs) du réseau sera élevé, plus ce dernier sera sûr et stable, grâce à la fonction validatrice que prend chaque serveur de Nexium.

2.7 Stockage de la blockchain

Quant au stockage de la blockchain, plusieurs choix se sont présentés à nous. Nous avons d'abord pensé à des fichiers simples du genre json, yaml, ou encore toml.

Cependant, un problème majeur nous est apparu qui est la lecture et l'ajout de données lorsque le fichier commence à devenir volumineux. Ces fichiers étant structurés, il est souvent nécessaire de les parcourir entièrement pour les modifier. Une alternative que nous avions trouvé pour résoudre ce problème était csv, car chaque ligne représentant une entrée, il n'est pas nécessaire de lire tout le fichier pour ajouter des entrées.

Un autre problème se trouvait dans la scalabilité, c'est pourquoi nous avons choisi une approche SQL, plus précisément SQLite qui ne nécessite pas de serveur SQL, mais qui permettrait toutefois une transition facile vers un serveur SQL. De plus, les relations SQL nous offrent un bon nombre de possibilités d'optimisation de stockage, ce qui en fait une solution particulièrement efficace.

Bitcoin utilise un stockage hybride, stockant les blocs brutes de la blockchain et utilisant levelDB pour les données d'indexation, permettant de retrouver facilement les blocs. Chaque fichier brute a une taille maximale de 128Mo, après cela un nouveau fichier est créé. De plus, les transactions en attente, sont stockées dans un fichier temporaire pour une restauration rapide en cas de redémarrage du nœud.

2.8 Site Web

La réalisation d'un site web pour notre projet est essentielle, afin de fournir aux utilisateurs une documentation claire et facilement accessible. Ce site permettrait également une redirection vers le repository Git de Nexium, afin que n'importe qui puisse récupérer des binaires et compiler, contribuer et vérifier les sources. Nous avons envisagé deux approches pour développer ce site, chacune avec ses avantages et inconvénients.

La première méthode aurait été d'écrire manuellement, *"from scratch"*, un site web statique en HTML et CSS. Cette façon de faire apporte une simplicité et de meilleures performances une fois le site déployé. Souvent utilisée pour des sites vitrines ou des pages d'informations statiques, cette approche impose toutefois des limites importantes;

De par notre besoin de maintenabilité, d'évolutivité et de scalabilité, l'ajout manuel de contenus sous forme d'HTML représente un trop gros sacrifice de productivité et de cohérence dans le processus de développement de notre site.

Nous souhaitons en effet écrire la documentation du projet au fur et à mesure du développement, ce qui implique de disposer d'une manière simple et rapide pour

modifier le contenu du site : c'est tout l'avantage d'Astro.
Astro, que nous avons retenu comme solution et donc choisi pour le développement du site de Nexium, est un framework moderne permettant de déployer des sites web statiques (ou hybrides) en utilisant le format MDX (un dérivé du Markdown). Ce format étant très facile à écrire et l'ensemble des membres du groupe ayant déjà une expérience avec cette syntaxe, Astro s'impose donc comme un choix évident pour le développement du site. Ce framework dit *"content-driven"*, c'est à dire basé sur le contenu (ici de la documentation au format Markdown) est de plus orienté pour les performances : le code statique généré par Astro peut, selon la configuration utilisée, être exporté et utilisé de manière statique, en simple HTML.

La principale limitation d'Astro est liée à sa force : moins adapté pour des sites avec interactions dynamiques, il est une solution évidente pour une grande partie des projets Open-Source sur internet. Cela explique sûrement pourquoi des projets et entreprise de grande renommée utilisent Astro pour exposer leur site vitrine / documentation : on trouve parmi eux CloudFlare, roadmap.sh, ou encore Namesake.

Cette génération statique efficace, qui garantit pour nous des performances optimales sans une charge de serveur excessive, et le support natif du Markdown favorisant l'aisance d'écriture, ont été entre autres les raisons qui nous ont mené à choisir Astro comme solution pour notre site web.

3 Répartition des tâches

Afin de mener à bien le projet Nexium, nous avons pris le parti de fonctionner en deux temps.

la **première période** s'étend jusqu'à la semaine du 7 avril, soit la deuxième soutenance. Cette première période est dédiée à la création des fondamentaux du projet, c'est-à-dire l'implémentation des algorithmes cryptographiques, et la gestion de la blockchain par le serveur.

Durant cette première période, notre équipe sera donc divisée en deux groupes:

- **Groupe 1** : **Jean** et **William** développeront le serveur Nexium (gestion de la base de données, logique de la blockchain, fichiers de configuration...)
- **Groupe 2** : **Milo** et **Antonin** implémenteront les fonctionnalités cryptographiques de Nexium, donc le RSA, le GPG, le SHA-256 et le AES.

Afin de fluidifier le travail de chaque équipe en parallèle, des réunions hebdomadaires ont déjà été planifiées. De plus, les prototypes de toutes les fonctions cryptographiques nécessaires au développement du serveur Nexium seront écrites en amont, avec des valeurs de retours par défaut, pour quand même permettre de

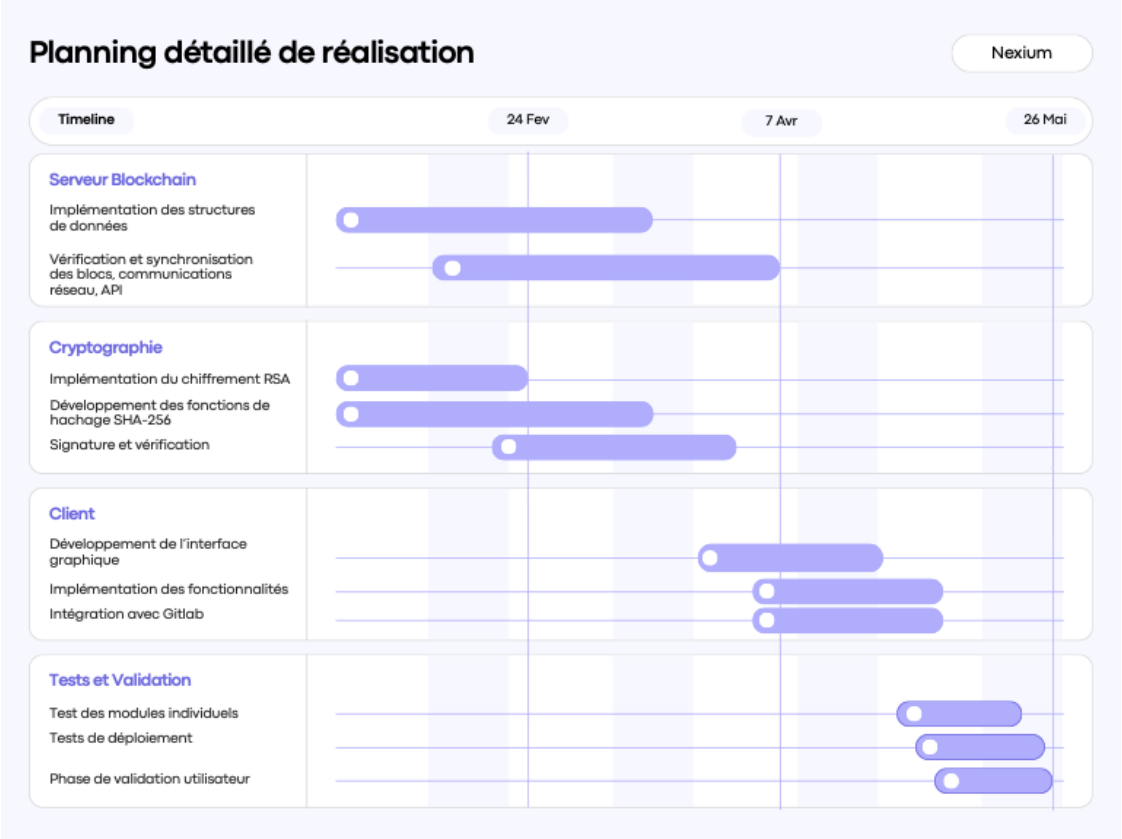
tester le serveur.

La **deuxième période** du projet Nexium concernera le développement du client. Cette période s'étendra de la deuxième soutenance à la soutenance finale, et sera dédiée à l'implémentation de l'interface utilisateur, et à la gestion des communications entre le client et le serveur.

Tous les membres de l'équipe travailleront sur le client, et les tâches seront réparties en fonction des compétences de chacun. Nous prévoyons de créer ultérieurement, en fonction de l'avancement du projet, un sous-groupe pour le backend du client, et un autre pour le frontend.

4 Planification détaillée de réalisation

Voici le diagramme de Gantt de la réalisation du projet Nexium :



5 Conclusion

Pour conclure, le projet Nexium représente une initiative innovante visant à instaurer un écosystème interactif, sécurisé et décentralisé pour les étudiants Épita tout en s'appuyant sur une approche moderne avec des éléments comme la blockchain et la cryptographie.

Ce projet constitue également une opportunité éducative précieuse, il offre une application concrète des systèmes les plus en vogue dans le monde actuellement tout en développant notre panel de connaissances en gestion de réseaux et en cryptographie.

Il sera également une solution facile d'accès et sur mesure pour la communauté étudiante pour se renseigner et/ou s'impliquer dans le domaine de la cryptographie.

Les prochaines étapes de ce projet consisteront à réaliser au mieux l'ensemble de ce projet tout en ajustant le système en fonction des retours recueillis par les utilisateurs, tout en espérant réussir à concrétiser notre vision d'un écosystème sérieux et pratique pour notre école.