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, vérifications 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 Cryptographie

2.2.1 Algorithme

Nous avions pour la partie cryptographique de Nexium, le choix entre plusieurs algorithmes de chiffrement. Nous avons étudié deux d'entre eux, qui sont les standards de la cryptographie moderne : **RSA** et **ECDSA**.

Pour des raisons d'implémentaton et de complexité algorithmique, nous avons retenu le chiffrement RSA, qui se base sur la factorisation de grands nombres premiers. Cet algorithme est plus simple à implémenter que l'ECDSA, qui se base sur les courbes elliptiques.

Le RSA est un algorithme de chiffrement asymétrique, qui utilise une paire de clés : une **clé publique** pour chiffrer les données, et une **clé privée** pour les déchiffrer. Il a été inventé en 1977 par *Ron Rivest*, *Adi Shamir* et *Leonard Adleman*, et est aujourd'hui largement utilisé pour sécuriser les communications sur Internet. C'est notamment le cas pour le **SSH**, le **HTTPS**, et les signatures numériques. Nous utiliserons donc le protocole de chiffrement **RSA** pour la cryptographie au sein du projet Nexium.

2.2.2 Implémentation

De par l'utilisation du Gitlab d'Épita pour récupérer des clés publiques, nous avons décidé d'utiliser le standard **GPG** pour la gestion des clés RSA. Cela nous permet de récupérer les clés publiques des utilisateurs de manière sécurisée, et de les utiliser pour chiffrer les données.

Le programme Nexium génère une paire de clés RSA pour chaque utilisateur, la formate en **ASN1**, convertit le résultat en **DER binaire**, l'encode en **base64**, et créée un fichier de type **PEM**. Cette manière de stocker et standardiser la clé publique est largement reconnue et compatible avec l'interface du Gitlab.

2.3 Récupération des clés publiques

Les clés publiques des utilisateurs de Nexium sont stockées sur le **Gitlab** d'Épita. Pour récupérer ces clés, nous utilisons l'**API** du Gitlab, qui nous permet de récupérer les clés publiques des utilisateurs en fonction de leur login (`prénom.nom`).

L'utilisation de cette API se fait par requêtes **HTTP**, et son utilisation est détaillée sur la **documentation officielle de Gitlab**.

Nous aurions pu utiliser le stockage des clés publiques dans la blockchain, mais préférer utiliser Épita comme autorité d'authentification garantit au réseau que seuls les étudiants (et professeurs!) d'Épita puissent l'utiliser. De plus, cette méthode permet une plus grande flexibilité dans la gestion des clés publiques : les utilisateurs peuvent **ajouter**, **supprimer** ou **révoquer** leurs clés à tout moment sans incidence sur la blockchain. Il s'agit donc d'un compromis volontaire favorisant la simplicité d'utilisation à la décentralisation totale.

2.4 Interface utilisateur

Nous avons étudié plusieurs possibilités de frameworks graphiques compatibles avec Rust pour le développement de l'interface utilisateur de Nexium. Les choix que nous avons étudiés étaient **Iced**, **Tauri** et **Slint**.

Afin d'éviter différents problèmes de stabilité et de performances, nous avons écarté l'utilisation de Tauri, qui se base sur du web pour le rendu graphique.

Enfin, et après avoir analysé sur différentes sources la réputation des deux autres frameworks, nous avons choisi d'utiliser **Slint** pour le développement de l'interface utilisateur de Nexium. Le critère principal de ce choix est la simplicité d'utilisation et la facilité de prise en main de Slint, qui est très bien documenté, possède une communauté active, et dont la **compatibilité** avec Rust est excellente.

2.5 Stockage local

2.5.1 Pour la blockchain

Chaque serveur de Nexium stocke **localement** une copie de la blockchain. Cela permet de garantir la **synchronisation** des blocs entre les différents nœuds du réseau, et de garantir l'intégrité des transactions.

Afin de stocker localement la blockchain, et pour appliquer des optimisations de traitement qui fluidifieront les transactions du réseau, nous avons choisi d'utiliser une base de données **SQLite**.

Cette base de données est légère, rapide, et facile à utiliser, et permet de stocker les blocs de la blockchain de manière structurée. Ce qui nous a également séduit dans SQLite est qu'elle ne nécessite **pas de dépendances** externes, les données étant stockées dans un **fichier unique** (et donc exportable!).

Nous utiliserons le crate `rusqlite`, qui est un wrapper Rust pour **SQLite**.

2.5.2 Pour les clés RSA

Comme mentionné plus haut, les clés **RSA** seront stockées (et transférées) en format **PEM**. Le programme Nexium (client ou serveur) stocke donc localement les clés RSA de l'utilisateur dans un **chemin relatif** au binaire.

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, et le SHA-256.

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.

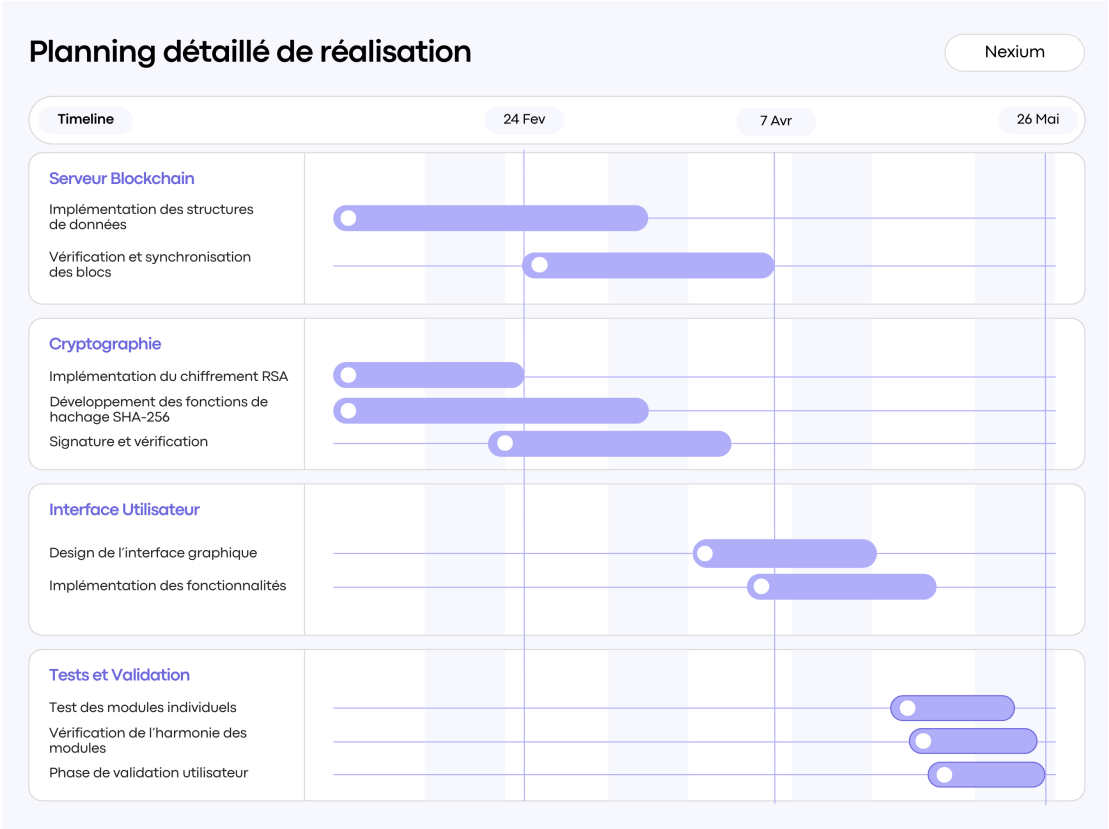
La **deuxième période** du projet Nexium concernera le développement du client. Cette période s'étendra de la deuième soutenance à la soutenance finale, et sera dédiée à l'implémentation de l'interface utilisateur, et à la gestion des communications entre le client et le serveur.

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.

Personne	Tâche	Description	Deadline
Antonin / Milo	Implémentation du RSA.	Chiffrement et déchiffrement. Signature et vérification de signature. L'implémentation du RSA se fait sous forme d'un crate contenant une librairie dans le package cargo de Nexium. Cela nous permet d'utiliser le RSA dans le client et le serveur, qui sont deux crates binaires du même package.	Première soutenance
Antonin / Milo	Implémentation de SHA-256.	Fonction de hashage SHA-256. Le hashage nous permet de générer plus facilement les signatures, et de donner un identifiant unique à chaque bloc.	Première soutenance
Jean / William	Gestion de la base de données.	Stockage, lecture et traitement de la blockchain via SQLite.	Première soutenance
Jean / William	Gestion du réseau.	Communication entre serveurs et avec les clients. Création d'une API propre à Nexium.	Première soutenance
Jean / William	Fichiers de configuration.	Enregistrement et chargement de fichiers de configurations au format TOML ou YAML (à déterminer) pour le serveur.	Première soutenance
Antonin / William / Jean / Milo	Création du client et de l'interface graphique.	Développement du GUI, de la création de requêtes API avec les serveurs, et de créations de blocs transactionnels.	Soutenance finale

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.