ARCHITECTURE DE SOLUTIONS

Le développement d'un mini serveur demeure un processus complexe donc la réalisation dépend de la bonne approche conceptuelle. Le projet 2 à élaborer rend bien compte de la difficulté de la manœuvre tant il fait appel à plusieurs données du développement Web, de la cryptographie et autres domaines non moins importants du web. Notre solution implémente ce mini serveur web, capable de gérer la collecte et le stockage des différents messages venant de tous les utilisateurs du service. Le serveur web est conçu pour répondre à 2 types de requêtes : **les demandes des clients pour récupérer les messages et les demandes des pairs pour synchroniser les messages.** L'idée de base est de créer ce serveur auquel le client web se connecte pour recevoir et envoyer des messages.

I. Les différentes interactions entre les composants du réseau

L'envoi d'un message

La première étape de l'envoi d'un message par un client Web, c'est l'encryptage. Par cette étape; le client encrypte le message avec la clé publique du destinataire. Ensuite, le client web envoie une requête POST de la forme "/addLetter", ;au mini-serveur local avec le message encrypté. Aucune opération de cryptage et d'encryptage n'est faite par les serveurs car ils ne possèdent aucune clé. Une synchronisation naturelle est faite entre les serveurs locaux afin que ces derniers aient accès eu tous les messages du service. Au final, le message est stocké dans la collection des messages du destinataire.

Réception d'un message

Pour chercher un message : l'utilisateur envoi une requête Get de la forme "/getLetters" vers le serveur. Ayant le même niveau d'information que le serveur distant, le serveur local intercepte la requête de l'utilisateur et renvoie les messages qui lui sont destinées. Étant donné que ceux sont cryptés. Il utilisera sa clé privée pour décrypter les messages afin de les lire.

La synchronisation des pairs

C'est une étape clé du processus de communication du mini serveur car elle permet à plusieurs pairs de partager des informations, telles que les messages envoyés et reçus. Certes, cela crée une répétition des données. Au-delà, cela a un avantage

particulier de rendre l'information accessible même en cas d'indisponibilité d'un pair spécifiquement associé à un utilisateur. La requête Get/ Peers renvoi une liste de pairs pour récupérer la liste de messages en attente pour chaque utilisateur.

II. Les outils utilisés

Dans le cadre de notre solution, nous sommes appelés à utiliser des outils logiciels pour 2 pans entiers de notre application à savoir **les requêtes et la cryptographie.** Dans le premier cas, nous utiliserons **Node** tandis que pour le second, nous utiliserons API REST.

API REST : Le mini-serveur doit fournir une API REST pour permettre au client web de communiquer avec lui. L'éventail des solutions offertes par API REST a été évoqué ci haut. Il inclut comme précédemment expliqué :

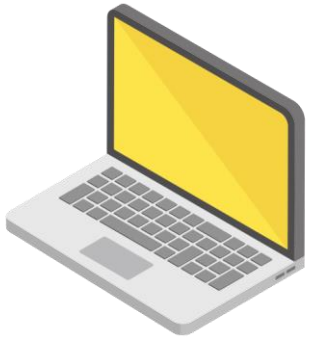
- GET /getLetters : Pour récupérer tous les messages d'un utilisateur
- POST /addLetter : Pour envoyer un nouveau message
- GET /peers : Pour récupérer la liste des pairs

En utilisant l'API REST, le client web pourra communiquer avec le mini-serveur de manière sécurisée et de manière efficace, et les pairs pourront synchroniser leurs messages en utilisant une méthode standardisée et facile à utiliser.

NODE : Node Forge est une bibliothèque qui fournit des outils de cryptographie pour les navigateurs et les serveurs Node.js. Pour utiliser Node Forge dans le cadre de ce projet, nous devons d'abord inclure la bibliothèque dans notre application. Cela peut se faire en téléchargeant la bibliothèque depuis le site Web de Node Forge et en l'incluant dans notre application via une balise script dans notre fichier HTML.
Une fois que nous avons inclus la bibliothèque Node Forge, nous pouvons utiliser ses fonctions de cryptographie pour chiffrer et déchiffrer des messages. Nous pouvons également utiliser Node Forge pour générer des clés publiques et privées RSA pour chaque utilisateur de notre application de courrier électronique.
Dans notre application, lorsque nous envoyons un message, nous devons d'abord chiffrer le message avec la clé publique de l'utilisateur destinataire avant de l'envoyer au serveur. Pour ce faire, nous pouvons utiliser la fonction de chiffrement RSA de Node Forge.

Lorsque nous recevons des messages du serveur, nous devons décrypter le message avec la clé privée de l'utilisateur destinataire. Pour ce faire, nous pouvons utiliser la fonction de déchiffrement RSA de Node Forge.

En conclusion, la création d'un projet de mini serveur courriel est un projet complexe car elle fait appel à plusieurs domaines de la science informatique. Il nécessite des connaissances approfondies en cryptographie asymétrique et de communication peer to,peer. Mais il s'agit d'un projet stimulant qui effleure ces domaines dans l'optique d'offrir une solution sécurisée de communications entre utilisateurs.
Cependant ce projet est à l'épreuve des nouvelles difficultés que rencontre le monde informatique a travers la difficulté de gestion de la sécurité dans l'espace virtuel.
D'où la nécessité de développer de nouvelles stratégies de protection des environnements informatiques

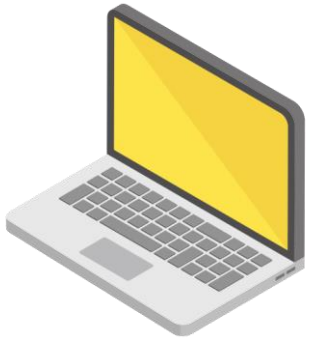


CLIENT WEB

Envoi d'un nouveau message avec la requête Post/AddLetter

SERVEUR LOCAL 1

Échange entre les deux serveurs pour la synchronisation des données : contacts, messages



UTILISATEUR

Envoi des requêtes GET/ Letters pour chercher des messages

Réception et décryptage du message avec sa clé privée

SERVEUR LOCAL 2