

INF4533 - Technologies internet

Devoir 2

Mini-serveur HTTP, courriel, pour le partage de messages

Par

Foli Amah

Maude Godbout

Marie Desaulniers

Mélanie Bélisle-Leclerc

Riadh Said

Vincent Crispin Emond

Remis à Fraczak Wojciech

18 avril 2016

Contenu

[1. Instructions d’utilisation 3](#_Toc448667243)

[1.1. Exécuter le programme 3](#_Toc448667244)

[2. Spécifications fonctionnelles 3](#_Toc448667245)

[2.1. Se connecter au serveur avec le navigateur 3](#_Toc448667246)

[2.2. Envoyer un message 3](#_Toc448667247)

[2.3. Voir les messages reçus 4](#_Toc448667248)

[2.4. Voir les messages envoyés 4](#_Toc448667249)

[2.5. Voir la liste des contacts et ajouter un contact 4](#_Toc448667250)

[2.6. Ajouter un utilisateur 4](#_Toc448667251)

[3. Organisation des fichiers 5](#_Toc448667252)

[3.1. Les dossiers du projet sont structurés comme suit : 5](#_Toc448667253)

[3.2. La base de données est structurée avec les tables et les vues suivantes : 5](#_Toc448667254)

[4. Communication entre pairs 7](#_Toc448667255)

[5. Fonctionnement des clés 7](#_Toc448667256)

[6. Considérations sécuritaires : chiffrement 7](#_Toc448667257)

[7. Améliorations possibles 8](#_Toc448667258)

[7.1. Sécurité 8](#_Toc448667259)

[7.2. Communication entre pairs 8](#_Toc448667260)

[7.3. Ajout de nouveaux contacts 9](#_Toc448667261)

[7.4. Ajout d’utilisateurs 9](#_Toc448667262)

## Instructions d’utilisation

### Exécuter le programme

* + 1. Faire un Clone de <https://github.com/Mariedez/INF4533-Projet1> .
    2. Faire npm install à partir de /courriel.
    3. Faire node www à partir de /courriel/bin.
    4. Une fois le serveur en marche, l’utilisateur doit se connecter à l’adresse <http://localhost:3000/> – le serveur redirigera l’utilisateur vers la page de connexion (/login).

Note : Le code fonctionne dans Firefox et Google Chrome

Le code ne fonctionne pas dans Internet Explorer

* + 1. Cliquer sur « Choose file » et choisir l’une des clés privées (.pem) contenues dans le dossier \INF4533-Projet1\
    2. Se connecter avec le nom d’utilisateur approprié – c’est-à-dire que si on a choisi la clé marie.pem, on se connecte avec le nom d’utilisateur Marie, si on a choisi vincent.pem, le nom d’utilisateur est Vincent, etc. À noter qu’idéalement, ces clés ne seraient pas gardées dans un seul fichier, mais chaque utilisateur sauvegarderait plutôt sa clé localement. Toutes les clés ont été incluses dans un fichier pour des raisons de clarté et de simplicité.
    3. Entrer le mot de passe approprié – pour des raisons de simplicité, tous les mots de passe sont les mêmes, soit 123456.
    4. Une fois connecté, l’utilisateur devrait être redirigé vers la boîte de réception.

## Spécifications fonctionnelles

### Se connecter au serveur avec le navigateur

* + 1. La page d’identification permet à l’utilisateur de choisir le fichier contenant sa clé privée qui est sauvegardée localement.
    2. La page d’identification permet à l’utilisateur d’entrer son nom ainsi que son mot de passe.
    3. Lorsque l’utilisateur est connecté, l’utilisateur est redirigé vers sa page de boîte de réception (inbox).

### Envoyer un message

* + 1. L’utilisateur qui s’est connecté peut envoyer un message à une personne qui est dans sa liste de contacts.
    2. Si elle désire envoyer un message à une personne qui n’est pas dans ses contacts, elle doit d’abord aller à la page de ses contacts et ajouter cette personne.
    3. Un message envoyé est stocké dans la base de données centrale.

### Voir les messages reçus

* + 1. L’utilisateur qui s’est connecté peut voir la liste des messages qu’il a reçue ainsi que la date et le nom de la personne qui lui a envoyé le message en se dirigeant à la boîte de réception. Cette liste ne contient qu’un aperçu du texte du message.
    2. L’utilisateur peut cliquer sur un message pour voir le texte du message au complet.

### Voir les messages envoyés

* + 1. L’utilisateur qui s’est connecté peut voir la liste des messages qu’il a envoyé ainsi que la date et le nom de la personne à qui il a envoyé le message en se dirigeant vers la boîte d’envoi.
    2. L’utilisateur peut cliquer sur un message pour voir le texte du message au complet.

### Voir la liste des contacts et ajouter un contact

* + 1. L’utilisateur peut voir la liste des noms de ses contacts en se dirigeant à la page « Liste des contacts ».
    2. L’utilisateur peut ajouter une personne dans ses contacts à partir de la liste de tous les utilisateurs qui sont dans la base de données centrale. Les clés publiques et privées des utilisateurs ont été générées préalablement.

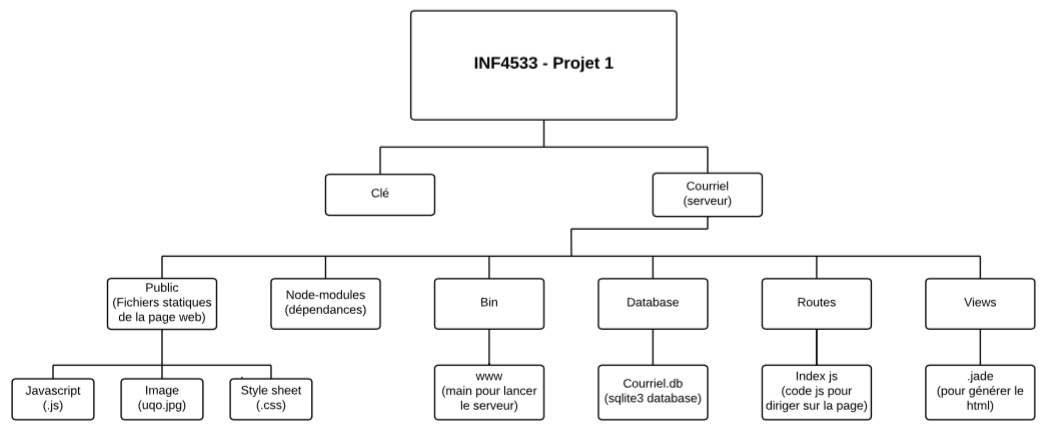
### Ajouter un utilisateur

* + 1. Il est possible d’ajouter un utilisateur à partir de la page de connexion (/login). Un nouvel utilisateur peut ajouter son nom et son mot de passe.

Note : pour l’instant, cette fonctionnalité ne permet pas de créer la paire de clés privée et publique. Voir le fichier /public/javascript/generateCle.js

## Organisation des fichiers

### Les dossiers du projet sont structurés comme suit :



### La base de données est structurée avec les tables et les vues suivantes :



**Tables**

*Utilisateurs*

* *ClePublique* : Base64
* *Password* : Haché avec SHA1 ou MD5

*Contacts* : associe un utilisateur à un autre

* *CleUtilisateur* : Clé publique de l’utilisateur
* *CleContact* : Clé publique de son contact

*Messages*

*CleDe* : Clé publique de l’utilisateur qui a envoyé le message

*CleA* : Clé publique de l’utilisateur qui a reçu le message

*Texte* : Encrypté avec une clé symétrique et en Base64

*cleSymDe* : Clé symétrique utilisée pour chiffrer le message

*cleSymA* :

**Vues**

*Boîte\_Courriel* : liste de tous les messages envoyés par tous les utilisateurs. Les champs *De* et *A* contiennent les noms des utilisateurs au lieu de la clé publique comme dans la table *Messages.*

*Liste\_Contact* : liste de toutes les paires de contacts. Les champs *Utilisateur* et *Contact* contiennent les noms des utilisateurs au lieu de la clé publique comme dans la table *Contacts*

Clé symétrique=Encryptée avec clé publique de « A » et Base64. Sera sauvegardée ailleurs que dans la base de données.

**Exemples de données**

Table *Utilisateurs*

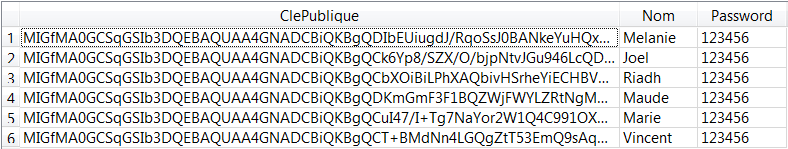


Table *Messages*

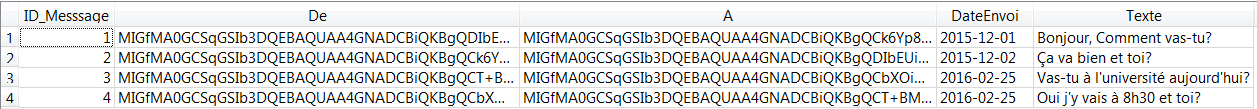
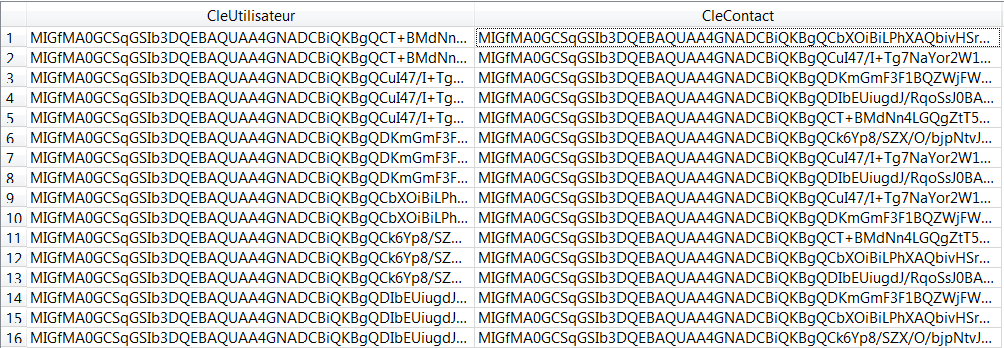
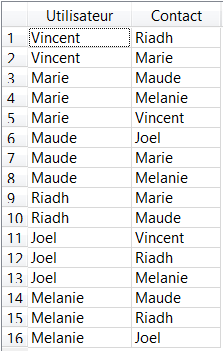
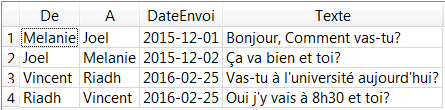


Table *Contacts*



Vue *Liste\_Contact* Vue *Boite\_Courriel*



## Communication entre pairs

Dans le cadre de ce projet, la communication entre pairs se limite à ceci :

La base de données locale se mettant à jour suite à tout envoie de message, nous proposons que les utilisateurs remettent sur GitHub la base de données locale à la fin de chaque session, puis, qu’il récupère la version la plus récente à partir de GitHub au début de chaque session. De cette façon, ils peuvent accéder à tous les messages qui leur sont envoyées entre chaque session d’utilisation.

## Fonctionnement des clés

Les clés privées et publiques des utilisateurs ont été générés préalablement. Les clés privées des utilisateurs sont sauvegardées localement (fichier clé dans le projet), et les clés publiques sont contenus dans la base de données centrale. Pour se connecter, l’utilisateur doit choisir sa clé privée à partir de l’endroit qu’il a choisi, puis entrer le nom et le mot de passe correspondant.

## Considérations sécuritaires : chiffrement

Les messages sont tous chiffrés avec une clé symétrique générée au moment de l’envoie d’un message et l’ensemble chiffré avec la clé publique du destinataire. La clé symétrique de chaque message est sauvegardée dans la base de données et la clé privée du destinataire est sauvegardée localement.

## Améliorations possibles

### Sécurité

Les mots de passe qui sont présentement sauvegardés en clair devraient être chiffrés.

### Communication entre pairs

Ayant eu plus de temps, nous aurions voulu que les noms et les adresse IP des utilisateurs soient stockés sur serveur externe (GitHub, Twitter, Facebook, ou autre) dans un fichier javascript sous la forme d’un objet JSON. Les paires contiendraient la clé publique de chaque utilisateur qui servira d’identificateur ainsi que l’adresse IP de l’utilisateur lors de la dernière connexion.

ClePubliqueIP = {

"AF22111212232211122": "192.168.1.122",  
"90221F212A4200001AA": "192.168.42.1",

"BF5464F845356273486": "10.0.1.65",

"BA2674F84578427C784": "192.168.92.5",

"CA2674F845784273476": "192.168.55.2",  
}

Les instructions suivantes seraient exécutées lorsqu’un utilisateur se connecte à internet et ouvrirait son application web de courriel :

1. Communiquer avec le serveur externe pour lire l’objet JSON.
2. Envoyer son adresse IP pour mettre à jour la variable JSON avec sa nouvelle adresse IP.
3. Découvrir les contacts qui sont en ligne en même temps à l’aide des autres adresses IP de l’objet JSON (ping?)
4. Établir une communication avec les contacts qui sont en ligne.
5. Pour chaque contact qui est en ligne, lire les messages du contact. Le transfert des messages entre l’utilisateur et ses contacts se ferait à l’aide d’un autre objet JSON contenant les messages provenant de la base de données de ses contacts.
6. Comparer les messages du contact avec les messages de la base de données de l’utilisateur. La comparaison pourrait se faire en comparant les empreintes des messages (fonction de hachage) pour accélérer le processus.
7. Pour tous les nouveaux messages trouvés, ajouter les messages dans la base de données locale.

### Ajout de nouveaux contacts

Présentement, il est possible d’ajouter des nouveaux contacts à partir de la liste d’utilisateurs existante dans la base de données, comme s’il s’agissait d’un courriel interne. Une amélioration possible serait de permettre l’envoi de messages à des utilisateurs externes.

### Ajout d’utilisateurs

Présentement, seul le nom et le mot de passe de l’utilisateur sont ajoutés à la base de données. Le champ de clé publique contient la date du jour. Une amélioration serait de générer la paire de clés privé-publique, de sauvegarder localement la clé privée et d’envoyer la clé publique au serveur.