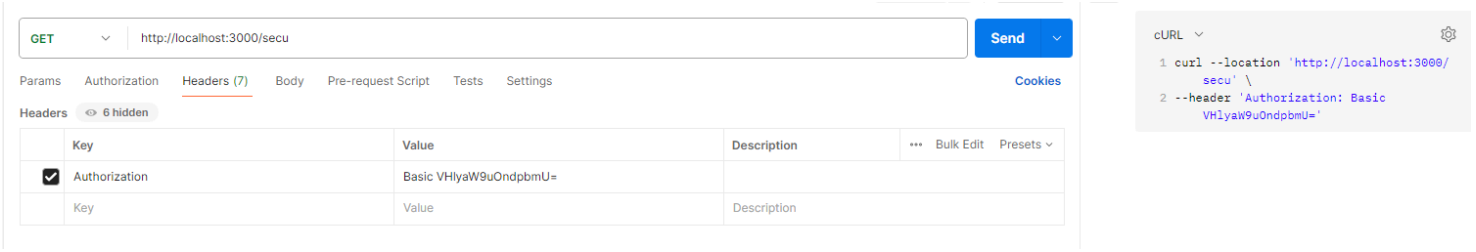
Rapport TP4 : SSL Certificate

Babylone ISSHAK - 303

Etape 1 :

Sur Postman, comme indiqué dans l'énoncé, j'ai d'abord ajouté une clé d'authentification manuellement dans l'onglet Headers :



J'ai obtenu la clé en base 64 à l'aide du site donné dans l'énoncé en écrivant Tyrion:wine (au début j'avais oublié d'ajouter ":" entre le nom et le mot de passe, ce qui ne fonctionnait pas, alors après avoir relu le cours, j'ai vu qu'il fallait ajouter ce caractère entre les deux). Ce qui renvoie bien ce que l'on souhaitait.

Ensuite, pour l'autre manière de faire, il fallait simplement aller dans l'onglet Authorization et cliquer sur Basic auth, puis entrer le nom et le mot de passe.

Enfin, le rôle de la fonction after() présente dans le code source, est d'ajouter une route supplémentaire après l'initialisation du serveur (ici c'est la route "/secu"). A présent pour ajouter une autre route comme "/autre", il faut dupliquer ce code, mais la seule différence est de supprimer la ligne `onRequest: fastify.basicAuth` . En effet, en supprimant celle-ci, nous enlevons l'authentification obligatoire. Au début, je pensais que c'était plus compliqué que cela. Je pensais qu'il fallait trouver dans la documentation, une fonction fastify permettant d'accéder sans authentification, mais par défaut un navigateur ne demande pas forcément d'authentification, donc logiquement si l'on enlevait cette ligne, cela enlevait cette dernière.

Etape 2 :

A l'aide du cours, j'ai écrit les commandes et complété comme dans l'exemple. Voici ma demande de signature d'un certificat dans mon invite de commande sur ma VM de la clé USB :

```
...+...+...........+..+.....+...+....+...+...+.................+....+..........+.+...+........
.........+.+..+.......+...+..+...+.....+........+........+..+.++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
linuxetu@linuxetu:~$ openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:tata.fr
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
linuxetu@linuxetu:~$
```

Ensuite nous devons signer ce certificat votre avec notre clé privée pour obtenir un certificat auto-signé, à l'aide d'une commande :

```
An optional company name []:
linuxetu@linuxetu:~$ openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Enter pass phrase for server.key:
Certificate request self-signature ok
subject=C = FR, ST = Some-State, L = Paris, O = Internet Widgits Pty Ltd, CN = tata.fr
linuxetu@linuxetu:~$
```

Pour tester si la signature du certificat a bien été faite, j'ai utilisé la commande de l'énoncé, ce qui m'a donné en retour "ACCEPT", donc tout devrait être bon. Sauf que, lorsque je teste l'URL donné dans l'énoncé aussi sur Postman, j'obtiens cette erreur : Error: connect ECONNREFUSED 127.0.0.1:4567. J'ai longuement réfléchi, fait des modifications pour savoir d'où vient le problème. Il s'avère que je n'avais pas modifié le numéro de port auquel le serveur Fastify du code se connecte, il est donc passé de 3000, au port 4567 : `const port = 4567;`

Cependant, une nouvelle erreur apparaît : Error: write EPROTO 77845512:error:100000f7:SSL routines:OPENSSL_internal:WRONG_VERSION_NUMBER:../../../../src/third_party/boringssl/src/ssl/tls_record.cc:242:

Ne comprenant pas l'erreur, de quelle version elle parlait ? etc. J'ai simplement demandé à ChatGPT une explication de l'erreur et une solution.

Il fallait que je modifie le code, de sorte à ce qu'il puisse lire les fichiers server.key et server.crt crées. Voici les modifications nécessaires dans le code :

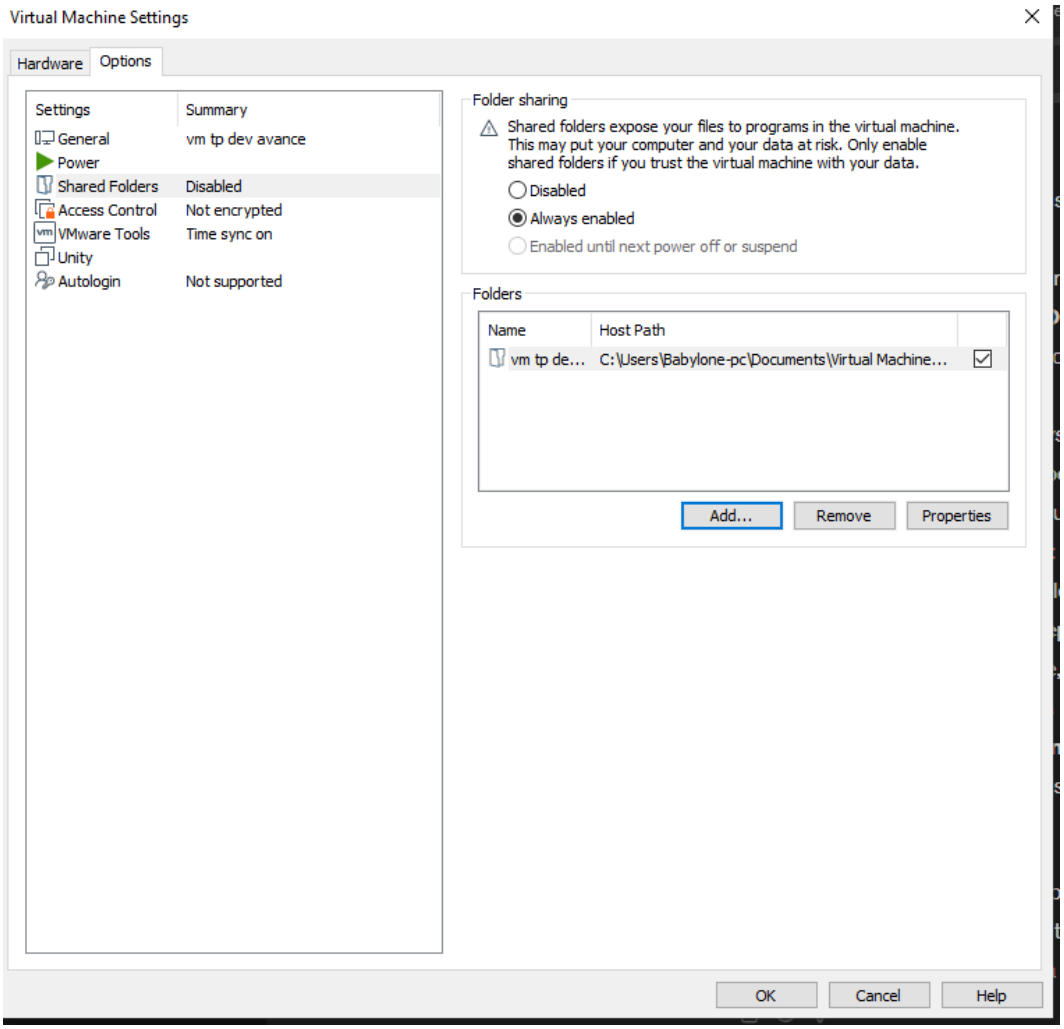
```
13  fastify.register(fastifyBasicAuth, {
14      https: {
15          key: fs.readFileSync('server.key'),
16          cert: fs.readFileSync('server.crt'),
17      },
18  })
```

Cependant, je n'arrive pas à récupérer les fichiers server.key et .crt, je n'arrive pas à savoir pourquoi. Lorsque j'ouvre dans mon explorateur de fichiers, ma VM, je m'aperçoit qu'il n'y a

pas ces fichiers. Pourtant lorsque que j'exécute la commande ‘ls’ dans ma vm, je vois que les fichiers ont bien été créés (comme on le voit dans la capture d’écran ci-dessous).

```
linuxetu@linuxetu:~$ find / -name "server.key" -or -name "server.crt" -or -name "server.der" 2>/dev/
null
/home/linuxetu/server.key
/home/linuxetu/server.crt
/home/linuxetu/server.der
linuxetu@linuxetu:~$ cd /mnt/hgfs
-bash: cd: /mnt/hgfs: No such file or directory
linuxetu@linuxetu:~$ vmware-hgfsclient
vm tp dev avance
linuxetu@linuxetu:~$ cd vm tp dev avance
-bash: cd: too many arguments
linuxetu@linuxetu:~$ ls
server.crt  server.csr  server.der  server.key
linuxetu@linuxetu:~$ cd "vm tp dev avance"
-bash: cd: vm tp dev avance: No such file or directory
```

J’ai essayé plusieurs solutions possibles pour partager des fichiers entre la machine virtuelle et l'hôte. En effet, VMware, par défaut n’autorise pas ce partage, j’ai donc essayer de modifier cela (capture d’écran ci-dessous), mais cela n’a toujours pas fonctionné.



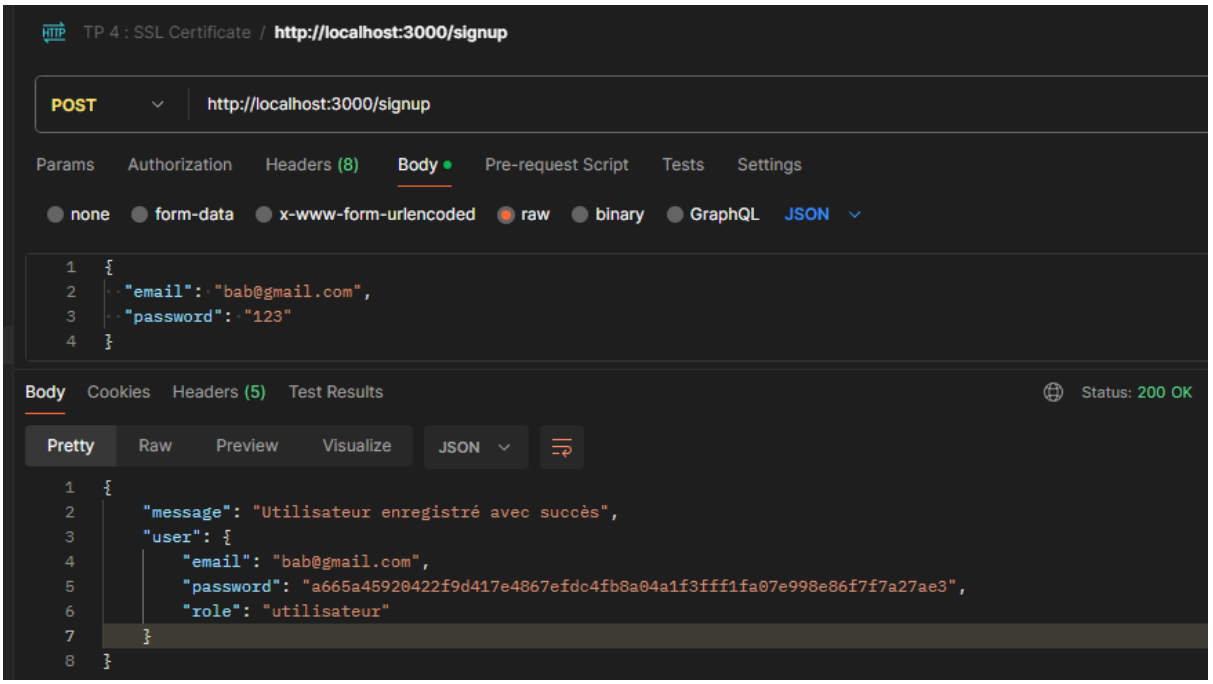
Etape 3 :

Pour la certification avec jeton JWT, nous devions aussi créer une clé privée et publique,

puisque dans l'étape précédente les commandes de copie de fichiers virtuels sur la machine de l'hôte ne fonctionnaient pas, j'ai préféré opter pour la solution se trouvant dans le read.me du dossier .ssl (serveur auth). J'ai donc créé un fichier keys.js où j'ai copié ce code permettant de créer les clés privée et publique qui seront utilisées pour signer les jetons JWT, si j'ai bien compris.
J'ai compris qu'il fallait lire ces fichiers dans jwt.js puisque c'est aussi ce que l'on avait fait dans l'étape précédente (où j'avais pas réussi à transférer les fichiers clés dans mon projet). Ensuite, lorsque j'ai vu que je bloquait longtemps sur le fichier login.js, ne savant pas par où commencer etc, sachant que l'algorithmique et les tableaux ne sont pas mon fort, j'ai demandé de l'aide à ChatGPT pour compléter les fonctions addUser() et loginUser().

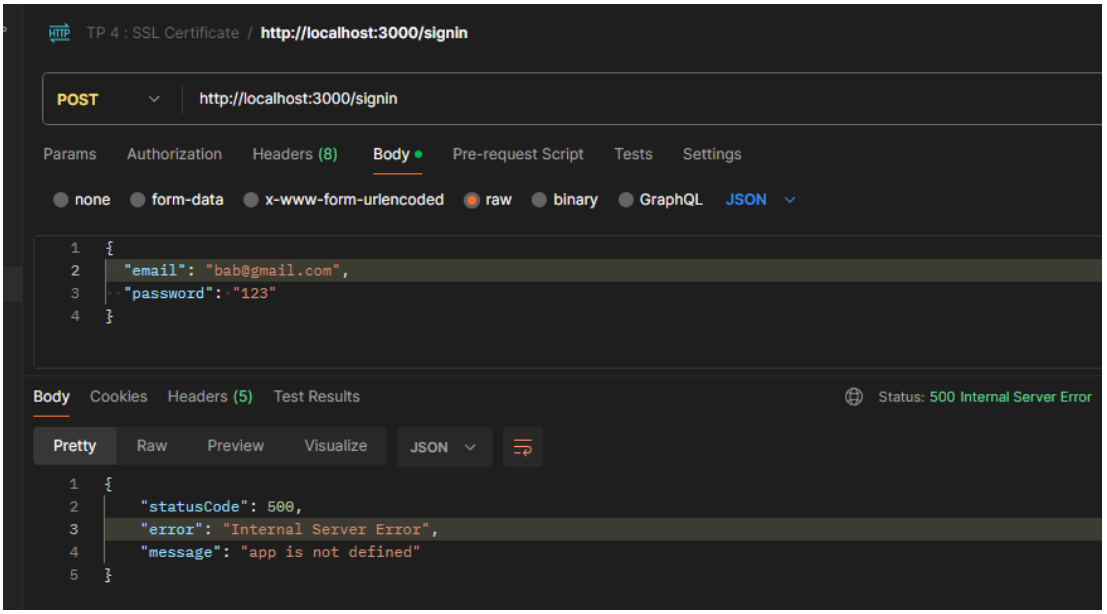
Après avoir pris le dépôt du serveur data, et analysé le rôle de chaque fichiers présent, j'ai commencé à compléter ce qu'il manquait dans la fonction `getAuthenticate` par exemple : il fallait simplement utiliser l'objet req entré en paramètres (notre requête), et lui appliquer la méthode jwtVerify() qui est donnée dans la documentation de @fastify/jwt que l'on a utilisé dans le fichier jwt.js.

Pour tester l'application, on démarre les deux serveurs ainsi que Postman, et on teste toutes les URLs. Pour que l'utilisateur s'enregistre dans la base de données (ici un tableau users), dans le body de la requête POST, j'y ai entré ses coordonnées. On peut voir dans l'image ci-dessous que l'inscription à bien fonctionné.

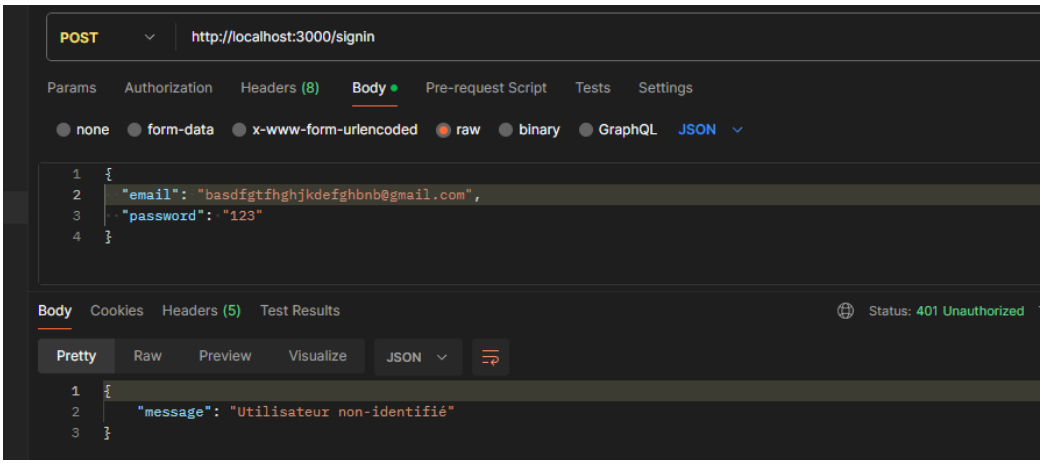


Pour ce qui est de la connexion de l'utilisateur, je ne comprend pas pourquoi cela passe bien dans le else, mais pas dans le if(user) existe. Donc il y a une erreur dans ma fonction loginUser ! Je pense que c'est parce que je recrée un email et le mot de passe haché peut-être ?

Voici ce que j'ai lorsque j'essaye de me connecter à bab@gmail.com (utilisateur inscrit) :



Et lorsque l'utilisateur n'existe pas :



Pour ce qui est de l'authentification GET http://localhost:4000/auth, je ne sais pas ce qu'il faut remplir dans payload, j'ai pris ce qui était dans le diapo du cours dans l'espoir de ne pas obtenir cette erreur qui me dit que le jeton n'a pas pu être signé, mais en vain :