Compte rendu

Développement Avancé : TP 4

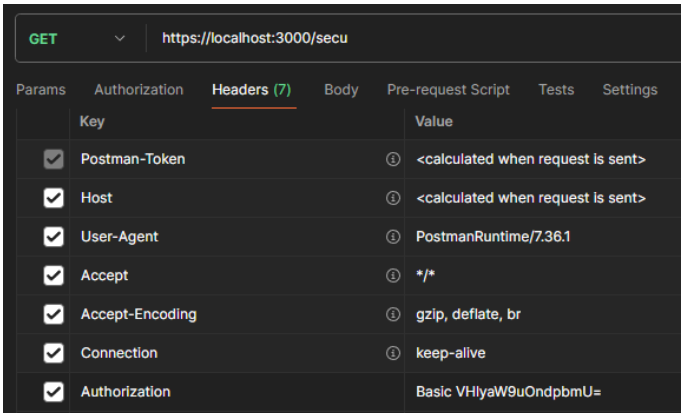
Laurent Giustignano

Par Steven TEA 303

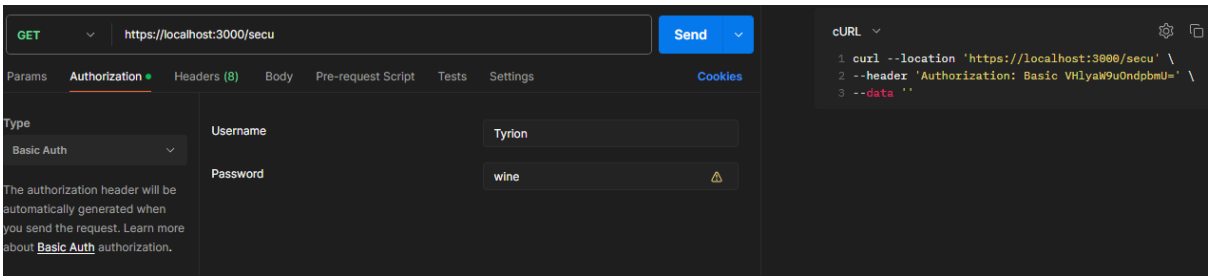
Étape 1

Pour la première étape du TP, nous essayons d'intégrer un système d'authentification avec une basic auth. J'ai donc essayé d'aller à la route /secu, qui nécessite une authentification basique.

Je m'y suis pris de différentes manières, tout d'abord, j'ai utilisé le site https://www.base64encode.org/fr/, pour convertir en base64 avec le username/password, donc convertir "Tyrion:wine" qui donne "VHlyaW9uOndpbmU=", pour mettre dans le header Authorization Basic VHlyaW9uOndpbmU=.



J'ai ensuite fait avec une manière moins compliqué, en utilisant le basic auth dans la catégorie Authorization, en mettant mon username "Tyrion" et le password "wine", la conversion est faite automatiquement par Postman.



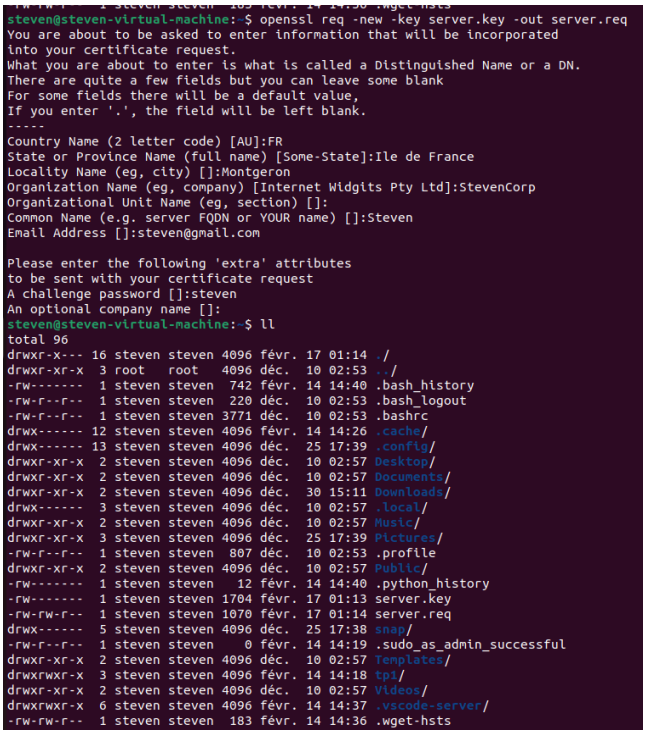
Ces deux méthodes fonctionnent, mais la deuxième est privilégiée pour sa simplicité.

La méthode after de Fastify, permet que le contenu dans la méthode soit exécuté après les méthodes appelées auparavant. Dans notre cas, l'utilisation de la méthode after sert à ce que l'enregistrement de fastifyBasicAuth s'effectue avant l'ajout de la route /secu car, cette route utilise le basic auth, et nécessite donc son chargement au préalable.

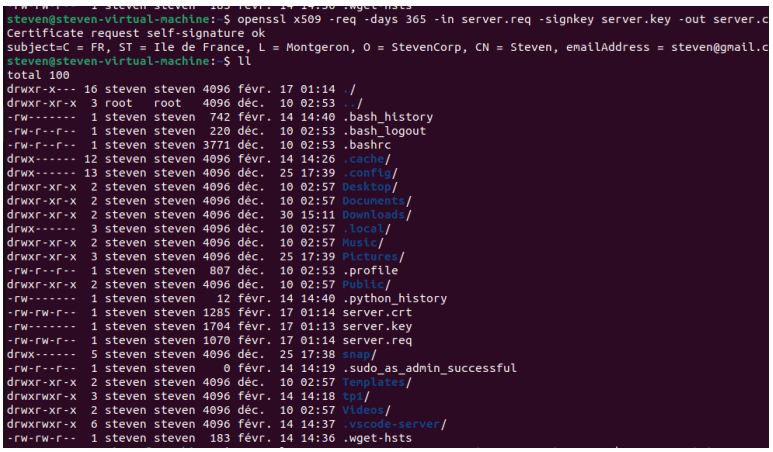
Étape 2

Maintenant nous allons mettre le protocole https sur notre serveur web. Pour créer un certificat, nous utilisons dans un premier temps, générer une clé privée 2048bits, puis

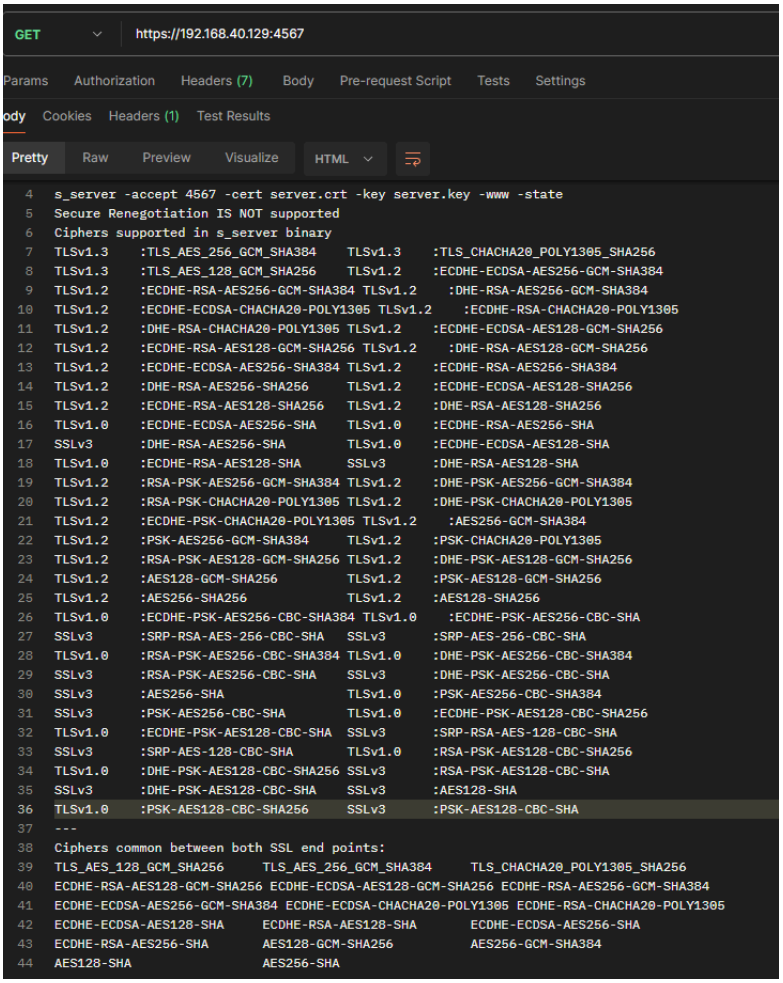
créer un fichier de demande de signature de certificat, dans mon cas, le fichier s'appelle server.req.



Une fois la clé privée et la demande de signature créées, nous allons auto signer un certificat. Donc, en entrant la clé privée, la demande et une durée d'expiration, un fichier de sortie est généré, server.crt. Je peux désormais tester le certificat, en ouvrant un serveur SSL/TLS sur mon ordinateur.



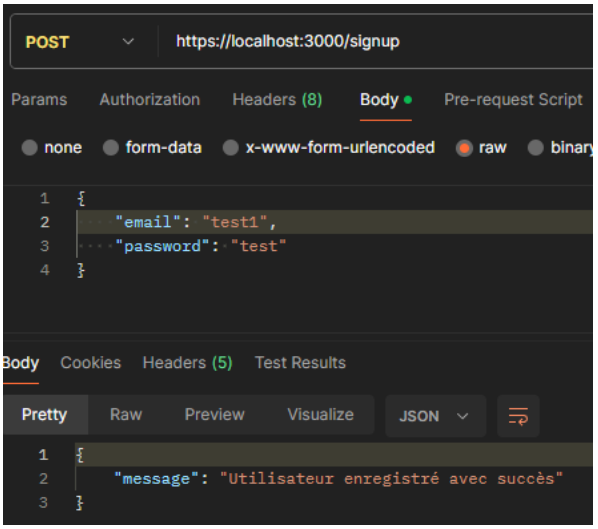
En utilisant Postman, nous pouvons afficher de nombreuses informations sur le certificat. Il y a tout d'abord, les méthodes de chiffrage supportées par le serveur, ici, TLS version 1.3, 1.2, SSL version 3, etc…



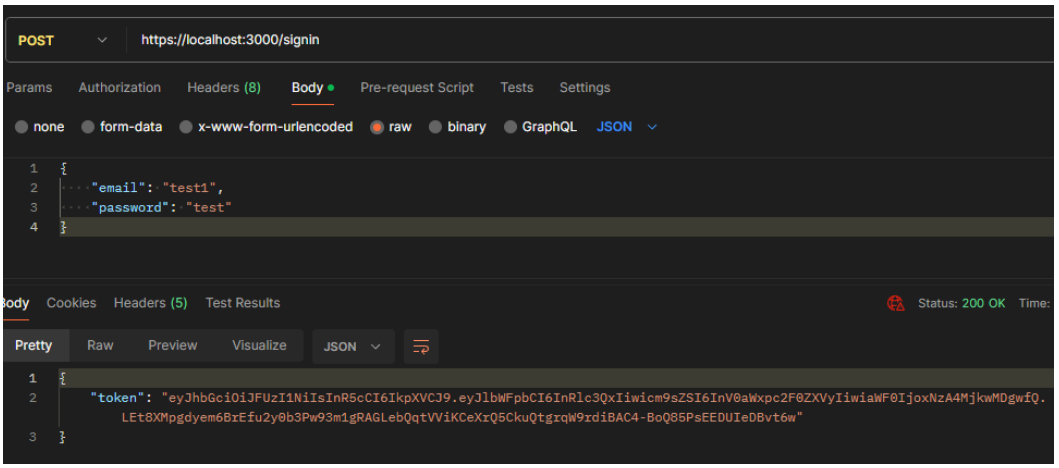
Étape 3

Pour la dernière étape de ce TP, j'ai tout d'abord commencé par générer les clés de chiffrement en utilisant la fonction présente dans le fichier .Readme du projet. J'ai complété le fichier jwt.js pour y ajouter les clés privée et publique dans le serveur Fastify.

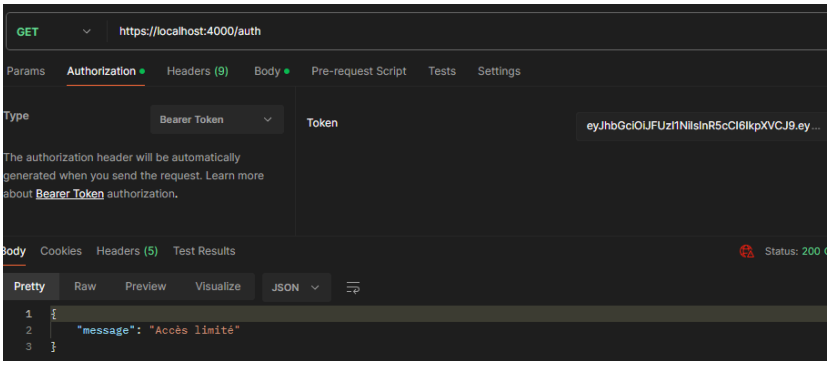
Étant donné qu'il y a deux serveurs à lancer, un sur le port 3000 pour l'authentification et le 4000 pour la page d'accueil, j'ai eu des difficultés concernant le partage des données, comme la liste des utilisateurs, j'ai notamment pensé à faire un serveur redis que les deux serveurs lisent pour récupérer les nouveaux utilisateurs, cependant, en regardant mieux les consignes, j'ai saisie que le serveur data au port 4000 ne nécessitait pas de connaître la liste des utilisateurs.



J'ai ensuite développé la fonction addUser pour ajouter un utilisateur à la liste et la fonction loginUser pour qu'un utilisateur inscrit puisse se connecter, j'ai également porté attention d'enlever le mot de passe des informations à chiffrer avant de créer le jeton JWT.



Du côté du serveur data au port 4000, j'ai eu des difficultés à déchiffrer les jetons JWT, comme une méthode existe pour chiffrer un objet JSON en jeton, j'ai alors pensé qu'une méthode permettrait de faire l'opération inverse, cependant, après de nombreux essaies et de différentes méthodes, je suis finalement arrivé à faire une fonction qui décode le jeton "à la main".



Finalement, nous affichons le message adapté en fonction du champ “role” dans l’objet JSON chiffré en JWT.