

Rapport Développement Avancé – TP4

Mes démarches et choix

J'ai décidé de faire les étapes du TP les unes après les autres, sans sauter d'étapes pour passer à une plus simple ou plus rapide par exemple. Il était important pour moi de comprendre chaque partie, avec ses nouvelles technologies et méthodes, afin de passer à la suivante.

Ce qui a marché (facilement)

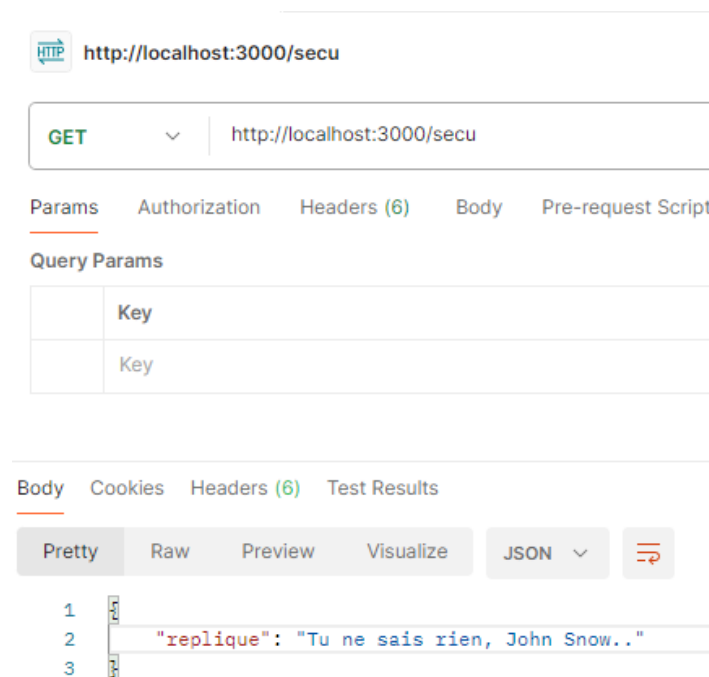
J'ai réussi facilement à faire les étapes 1 et 2, sans rencontrer aucun, ou presque, problème. Je vais donc ici les détailler pour expliquer mon cheminement.

Étape 1

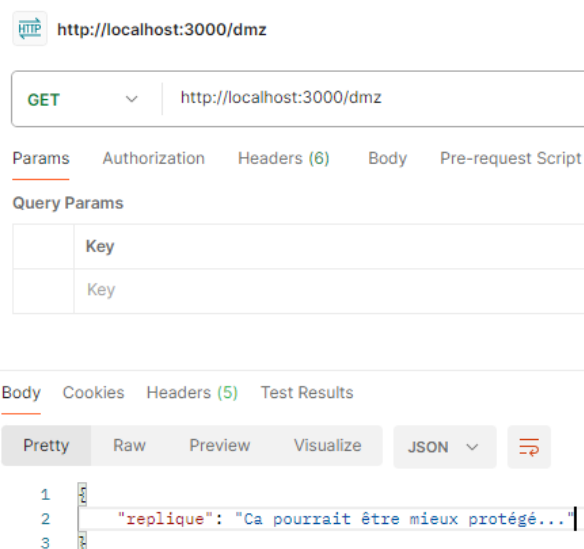
J'ouvre PHPStorm, je clone le repo GitHub dans mon TP4. Ensuite je prends connaissance des fichiers. J'ai un peu du mal à comprendre les méthodes du fichier server.js car c'est beaucoup de Fastify et je ne suis pas très familier avec ça.

Je télécharge Postman et l'installe sur mon dossier personnel. Je lance la configuration « dev » sur PHPStorm, afin de lancer le sujet.

J'entre la 1ere requête sur Postman soit l'adresse suivante : <http://localhost:3000/secu>
Je reçois bien le bon retour :



Ensuite j'essaie la 2ème requête : <http://localhost:3000/dmz>
Elle fonctionne également, car je reçois le bon retour :



Ensuite, je fais l'encodage des identifiants comme ils sont attendus, c'est à dire `username:mdp`.

Encodage au format Base64

Il suffit de saisir vos données et d'appuyer sur le bouton d'encodage.

Tyrion:wine

Ce qui me donne le résultat suivant :

> ENCODAGE <

Encodage de vos données dans la zone ci-dessous.

VHlyaW9uOndpbmU=

Ensuite, dans Postman je rajoute dans l'onglet Headers, la clé Authorization avec la valeur obtenue, précédé de la mention "Basic".

Headers 6 hidden		
	Key	Value
<input checked="" type="checkbox"/>	Authorization	Basic VHlyaW9uOndpbmU=


Après cela, je reteste donc les deux endpoints précédemment rentrés et les deux marchent.
Puis, je décoche la case de cette clé, et je vais dans l'onglet Authorization, je choisis "Basic Auth" et je rentre les valeurs en dur.

Params **Authorization** Headers (8) Body Pre-request Script Tests Settings

Type **Basic Auth** ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username


Password 

Puis je reteste, et tout marche toujours. Je teste quand même une mauvaise valeur de mot de passe pour m'assurer que le code d'erreur fonctionne, et cela renvoie bien le message d'erreur choisi.


Type **Basic Auth** ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Username

Password 

Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ▼ 

```

1  {
2    "replique": "Tu ne sais rien, John Snow.."
3  }

```

Je pense que la fonction `after()` a pour rôle dans la déclaration de la route `/secu`, de vérifier les entrées.

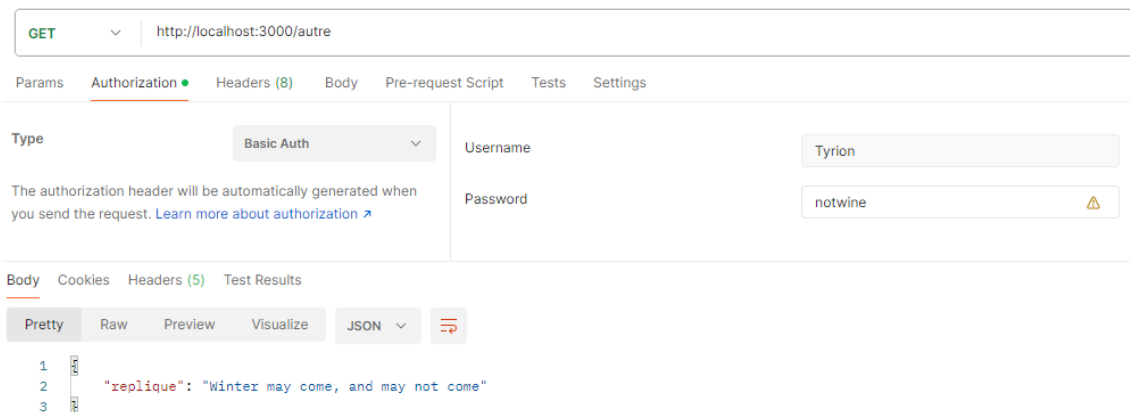
Dans la fonction `after()`, je rajoute une route que j'appelle `/autre` (en copiant celle d'au-dessus), mais j'enlève la partie authentification pour qu'on en ait pas besoin à la connexion.

```

//route autre
fastify.route({opts: {
  method: 'GET',
  url: '/autre',
  handler: async (req : FastifyRequest<RouteGeneric, http.Server,
    return {
      replique: 'Winter may come, and may not come'
    }
  }
})
})

```

En testant sur Postman, on peut voir que l'authentification n'est pas nécessaire, grâce à ma modification. En effet, même avec un mauvais mot de passe, on obtient le message classique.



Étape 2

Pour cette étape je lance la VM donnée sur clé USB, car il faut utiliser des commandes notamment `openssl`, etc.

Je crée une clé RSA de 2048 bits (qui est la valeur par défaut) que j'appelle `server.key` avec la commande `openssl genrsa -out server.key`

```
linuxetu@linuxetu:~/dev-avance-tp4$ openssl genrsa -out server.key
linuxetu@linuxetu:~/dev-avance-tp4$ ls
server.key
linuxetu@linuxetu:~/dev-avance-tp4$ _
```

D'abord, je crée le fichier de demande de signature de certificat (CSR) avec la commande `openssl req -new -key server.key -out server.req`

Après exécution, cela me demande de remplir plusieurs champs, que j'ignore pour la plupart, et ça génère le fichier.

```
linuxetu@linuxetu:~/dev-avance-tp4$ openssl req -new -key server.key -out server.req
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) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234
An optional company name []:
linuxetu@linuxetu:~/dev-avance-tp4$ ls
server.key  server.req
```

Ensuite, j'effectue la signature de ce certificat avec ma clé privée `server.key` et à l'aide de la commande : `openssl x509 -req -days 365 -in server.req -signkey server.key -out server.crt`

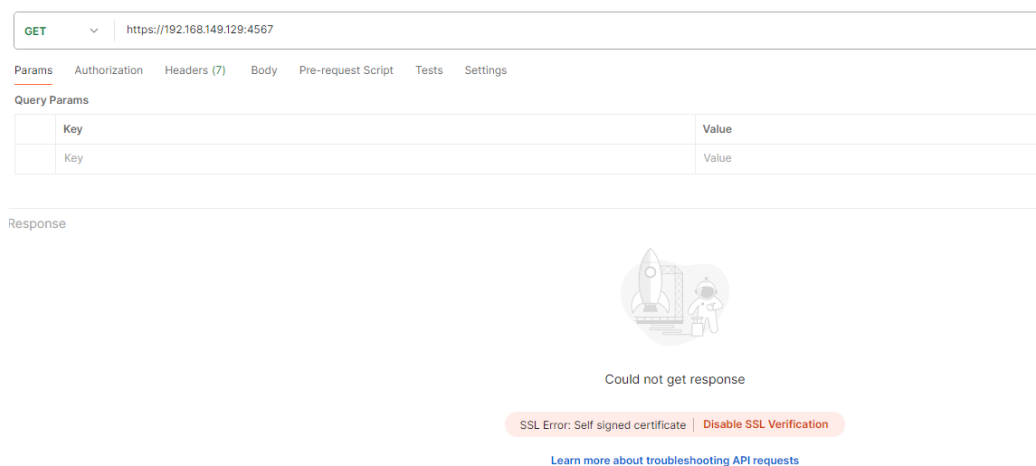
```
linuxetu@linuxetu:~/dev-avance-tp4$ openssl x509 -req -days 365 -in server.req -signkey server.key -out server.crt
Certificate request self-signature ok
subject=C = FR, ST = Some-State, O = Internet Widgits Pty Ltd
linuxetu@linuxetu:~/dev-avance-tp4$ ls
server.crt server.key server.req
linuxetu@linuxetu:~/dev-avance-tp4$ _
```

Cela fonctionne.

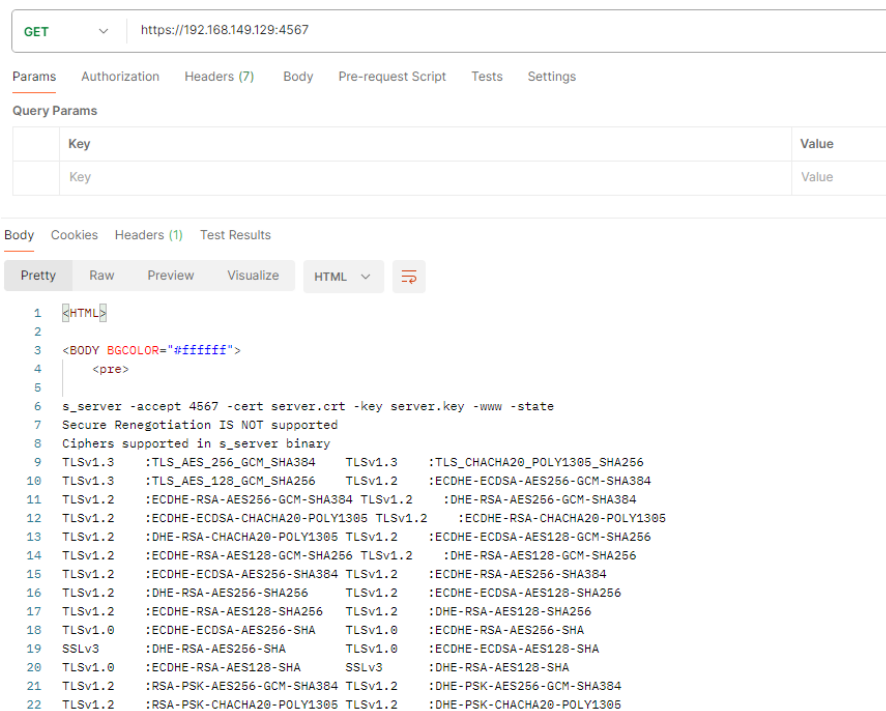
Afin de tester le certificat généré, j'utilise la commande `openssl s_server -accept 4567 -cert server.crt -key server.key -www -state`

Ce qui semble lancer l'exécution d'un processus. En laissant le processus lancé, je vais ensuite sur Postman pour essayer de contacter l'adresse `https://localhost:4567`. Mais sur Postman ça ne marche pas car je ne dois pas mettre localhost mais je dois mettre l'IP de ma VM, car c'est dessus que se lance le processus. Pour cela, j'utilise la commande `ip a` dans ma VM. Ce qui me donne l'adresse IP suivante : `inet 192.168.149.129` qui est donc sur le port 4567.

Donc je lance Postman sur l'adresse : <https://192.168.149.129:4567>



Et cela me fait une erreur qui dit "SSL Error : Self signed certificate". Cela était prévu car, mon certificat est effectivement signé par moi-même. Mais je désactive la vérification SSL sur Postman et je ressaye, et cela donne bien un autre résultat.



Enfin, afin de faire évoluer le service web de mon projet, je vais aller sur la documentation Fastify, prendre le code présent pour changer mon propre code. Ensuite, je vais avoir besoin des fichiers `server.key` et `server.crt` sur mon projet PHPStorm. Donc pour que ce soit le cas, je les transfère de ma VM au PC à l'aide de la commande : `scp linuxetu@192.168.149.129:dev-avance-tp/server.key server.key`

Et je fais la même chose pour le fichier `server.crt`. J'ai donc maintenant ces deux fichiers sur mon PC et je les ajoute au projet sur PHPStorm.

```
const fastify : FastifyInstance<...> & PromiseLike<...> = Fastify( opts: {
  logger: true,
  http2: true,
  https: {
    allowHTTP1: true, // fallback support for HTTP1
    key: fs.readFileSync('server.key'),
    cert: fs.readFileSync('server.crt')
  }
})
```

Mes difficultés et comment je les ai surmontés

Au début du projet, j'ai eu quelques difficultés à prendre en main le code et le projet donné. Cela est dû notamment au fait qu'il s'agit de technologies et méthodes avec lesquelles je ne suis pas familier, comme Fastify par exemple.

Néanmoins, la réalisation de toutes ces étapes m'a pris du temps et je n'ai pas pu faire l'étape 3. Pour autant, j'avais commencé à lire et essayer de comprendre ce qui était demandé et les technologies utilisées, mais j'ai eu du mal pour cela.

Ce qui pourrait être amélioré

Faire l'étape 3

Par mon manque de temps et de compréhension du début de l'étape 3, je n'ai pas pu la faire. En gérant mieux mon temps la prochaine et avec de meilleures connaissances des technologies à utiliser dans l'étape 3, comme les clés de chiffrements ECDSA et les Tokens JWT par exemple, je pourrais sûrement réussir à réaliser cette étape.