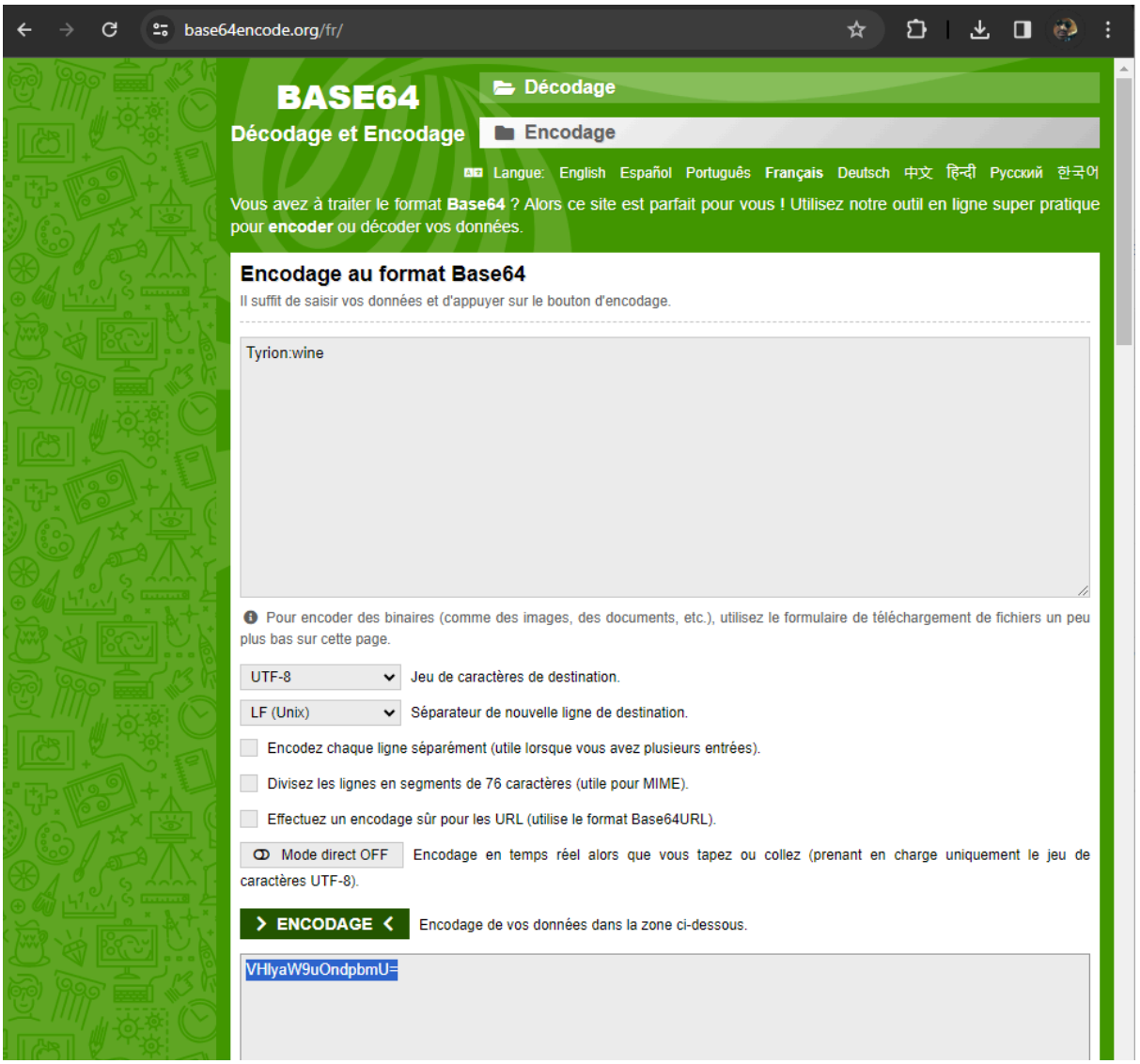
Développement Avancé : TP3

Etape 1 : “De base… ”

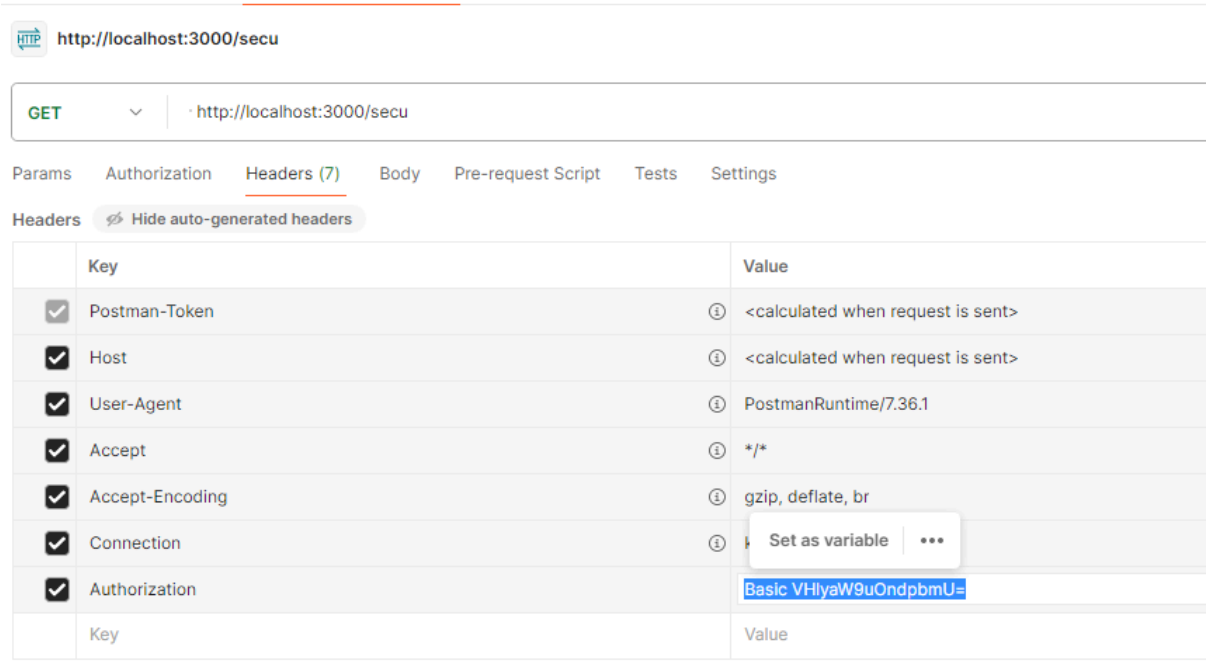
Dans un premier temps on récupère ici les identifiants tyrion:wine

```
1 usage   Laurent Giustignano
async function validate(username, password, req, reply) : Promise<...> {
    if (username !== 'Tyrion' || password !== 'wine') {
        return new Error('Winter is coming')
    }
}
```

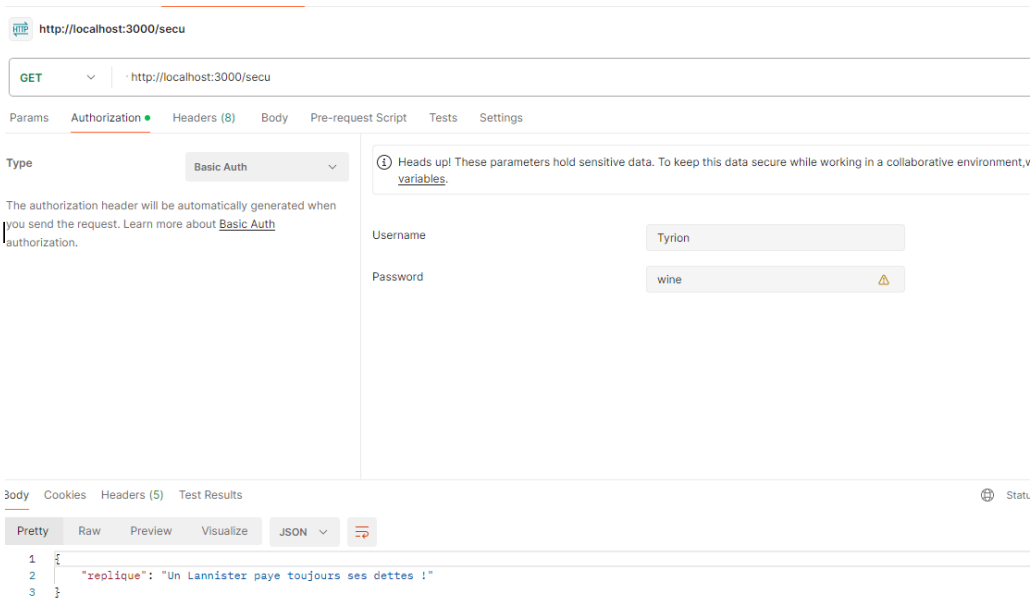
Ensuite on copie colle dans le site d’encodage puis on récupère les identifiants encodé

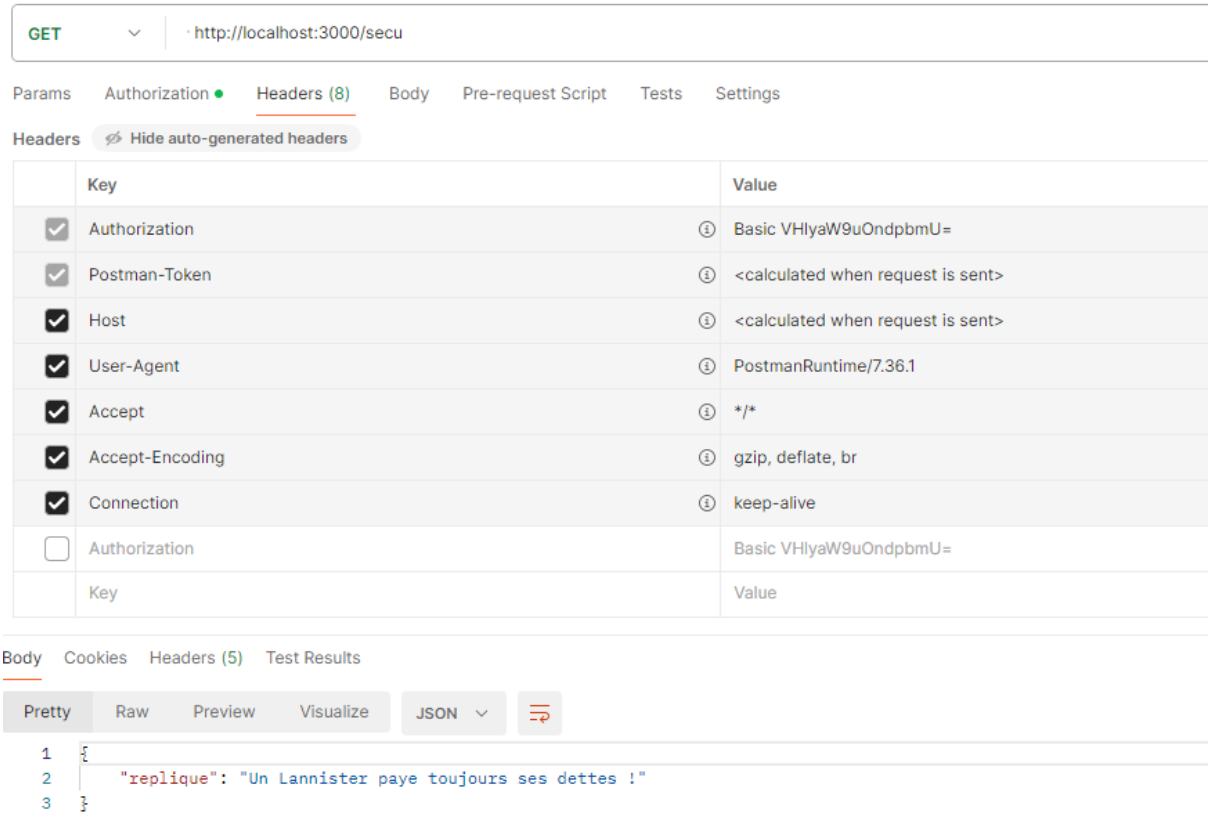


Ensuite, j'ajoute la clé d'autorisation et je mets l'identifiant encodé. Ensuite, la réplique change en "Un Lannister paie toujours ses dettes"

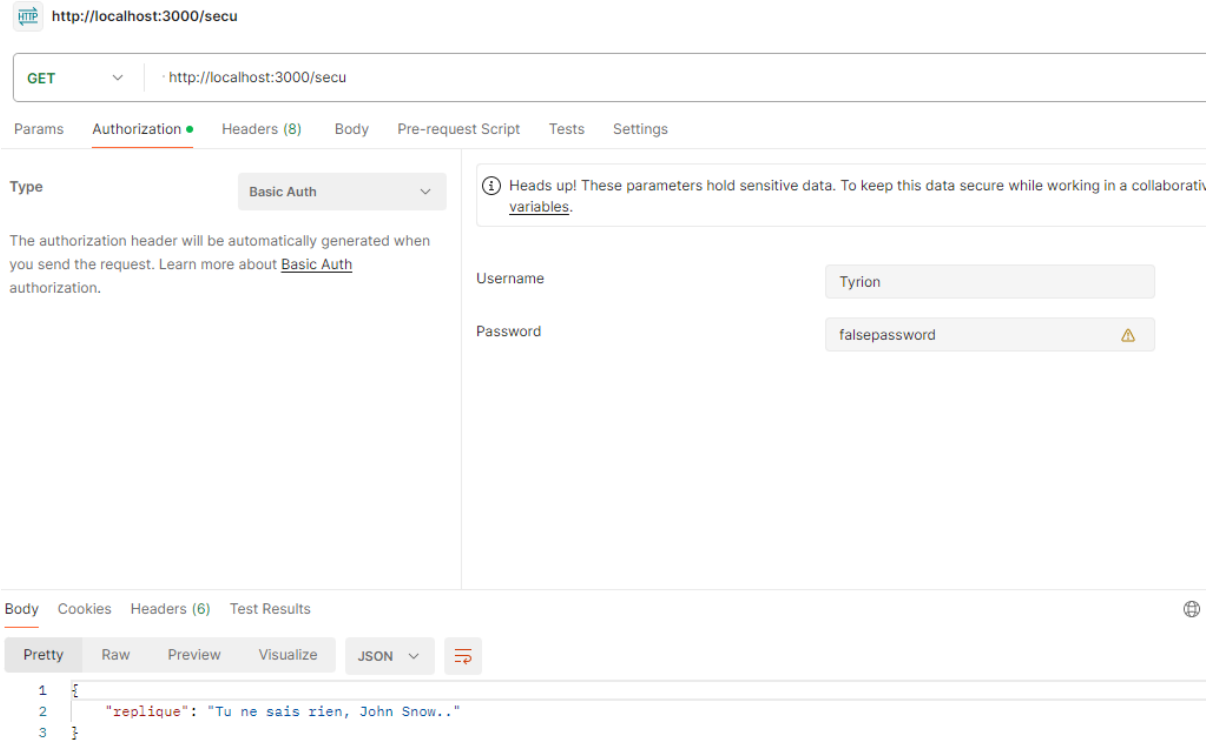


Lorsque nous saisissons l'identifiant en clair dans l'onglet 'Type d'autorisation' comme 'authentification de base', nous pouvons constater que l'en-tête est automatiquement complété.





Et lorsque l'on se trompe dans le mot de passe, la réplique change en 'Tu ne sais rien, Jon Snow'.



La fonction after sert à afficher la réplique 'Un Lannister paie toujours ses dettes !' lorsque

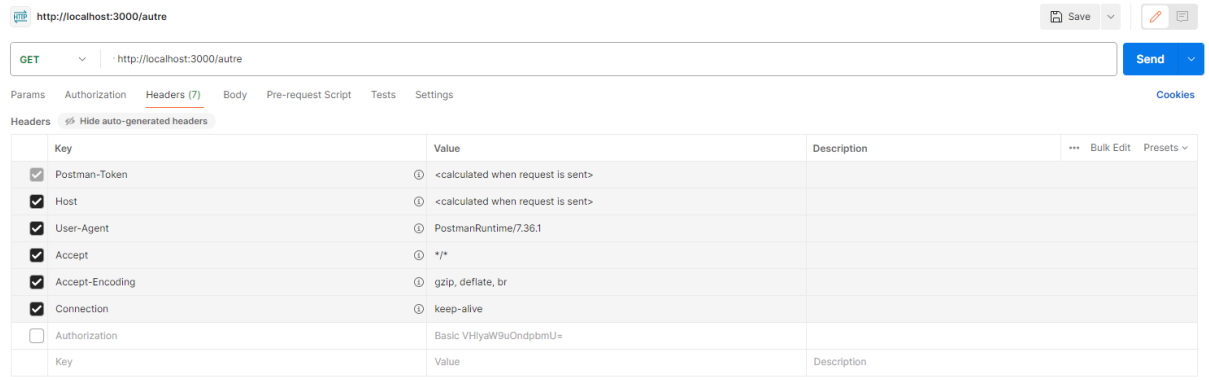
les identifiants d'authentification sont corrects. Lorsque les identifiants sont faux, la fonction setErrorHandler est appelée et la réplique devient alors 'Tu ne sais rien, Jon Snow'.

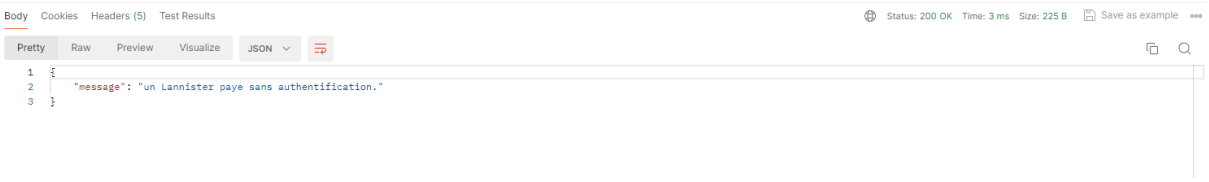
```
Laurent Giustignano
fastify.after(() : void  => {
    fastify.route( opts: {
        method: 'GET',
        url: '/secu',
        onRequest: fastify.basicAuth,
        handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
            return {
                replique: 'Un Lannister paye toujours ses dettes !'
            }
        }
    })
})

Laurent Giustignano
fastify.setErrorHandler( handler: function (err : TError , req : FastifyRequest<RouteGeneric, http.Server, RawRequestDefaultExpression<...>, SchemaCompiler, TypeProvider> , reply : FastifyReply<http.Server, RawRequestDefaultExpre

    if (err.statusCode === 401) {
        console.log(err)
        reply.code( statusCode: 401).send( payload: {replique: 'Tu ne sais rien, John Snow..'})
    }
    reply.send(err)
})
```

Ici, c'est la création de la route /autre :

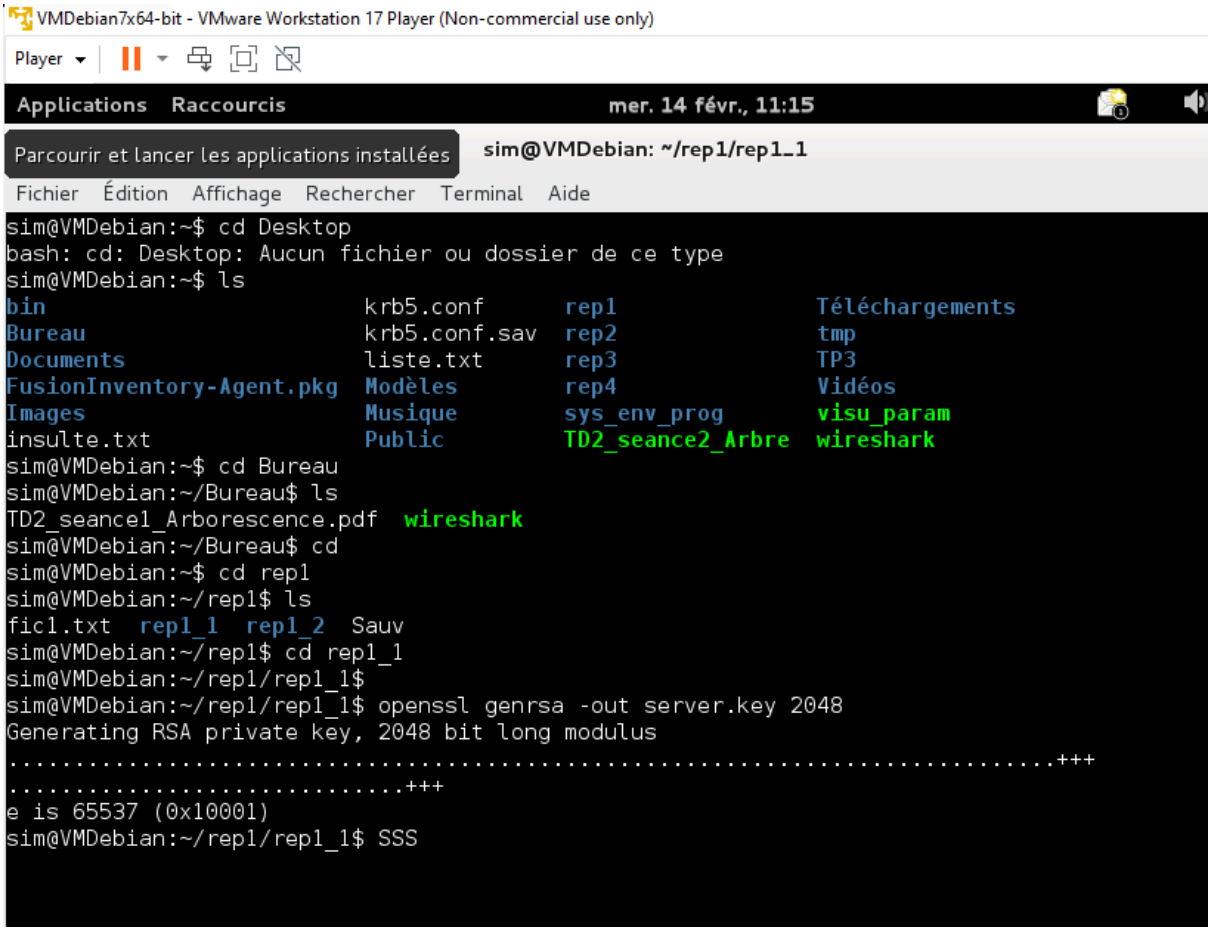





```
Laurent Giustignano *
fastify.after(() : void  => {
    // Route '/secu' avec authentification
    fastify.route( opts: {
        method: 'GET',
        url: '/secu',
        onRequest: fastify.basicAuth,
        handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
            return {
                replique: 'Un Lannister paye toujours ses dettes !'
            }
        }
    });

    // Route '/autre' sans authentification
    fastify.route( opts: {
        method: 'GET',
        url: '/autre',
        handler: async (req : FastifyRequest<RouteGeneric, http.Server, http.IncomingMessage, SchemaCompiler, FastifyTypeProviderDefault, ContextConfig, boolean> , reply : FastifyReply<http.Server, http.IncomingMessage, http.ServerR
            return {
                message: 'un Lannister paye sans authentification.'
            }
        }
    });
});
```

J'ai créé une nouvelle clé RSA de 2048 bits en utilisant la commande du TD de la semaine 4 'openssl genrsa -out server.key' en ajoutant '2048' à la fin.



Ensuite je rentre la commande openssl req -new -key server.key -out server.csr :

```
sim@VMDebian:~$ cd Desktop
bash: cd: Desktop: Aucun fichier ou dossier de ce type
sim@VMDebian:~$ ls
bin                       krb5.conf      rep1              Téléchargements
Bureau                    krb5.conf.sav  rep2              tmp
Documents                 liste.txt      rep3              TP3
FusionInventory-Agent.pkg Modèles        rep4              Vidéos
Images                    Musique        sys_env_prog      visu_param
insulte.txt               Public         TD2_seance2_Arbre wireshark
sim@VMDebian:~$ cd Bureau
sim@VMDebian:~/Bureau$ ls
TD2_seance1_Arborescence.pdf  wireshark
sim@VMDebian:~/Bureau$ cd
sim@VMDebian:~$ cd rep1
sim@VMDebian:~/rep1$ ls
fic1.txt  rep1_1  rep1_2  Sauv
sim@VMDebian:~/rep1$ cd rep1_1
sim@VMDebian:~/rep1/rep1_1$
sim@VMDebian:~/rep1/rep1_1$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................................+++
.............................+++
e is 65537 (0x10001)
sim@VMDebian:~/rep1/rep1_1$ openssl req -new -key server.key -out server.csr
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]:Paris
Locality Name (eg, city) []:Paris Descartes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IUT
Organizational Unit Name (eg, section) []:IUT PARIS
Common Name (e.g. server FQDN or YOUR name) []:Mehdi
Email Address []:mehdi.zaoui@etu.u-paris.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:mehdi.zaoui
An optional company name []:IUT
sim@VMDebian:~/rep1/rep1_1$
```

Puis la commande openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt :

```
bin                       krb5.conf      rep1              Téléchargements
Bureau                    krb5.conf.sav  rep2              tmp
Documents                 liste.txt      rep3              TP3
FusionInventory-Agent.pkg Modèles        rep4              Vidéos
Images                    Musique        sys_env_prog      visu_param
insulte.txt               Public         TD2_seance2_Arbre wireshark
sim@VMDebian:~$ cd Bureau
sim@VMDebian:~/Bureau$ ls
TD2_seance1_Arborescence.pdf  wireshark
sim@VMDebian:~/Bureau$ cd
sim@VMDebian:~$ cd rep1
sim@VMDebian:~/rep1$ ls
fic1.txt  rep1_1  rep1_2  Sauv
sim@VMDebian:~/rep1$ cd rep1_1
sim@VMDebian:~/rep1/rep1_1$
sim@VMDebian:~/rep1/rep1_1$ openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.............................+++
e is 65537 (0x10001)
sim@VMDebian:~/rep1/rep1_1$ openssl req -new -key server.key -out server.csr
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]:Paris
Locality Name (eg, city) []:Paris Descartes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IUT
Organizational Unit Name (eg, section) []:IUT PARIS
Common Name (e.g. server FQDN or YOUR name) []:Mehdi
Email Address []:mehdi.zaoui@etu.u-paris.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:mehdi.zaoui
An optional company name []:IUT
sim@VMDebian:~/rep1/rep1_1$ openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.c
rt
Signature ok
subject=/C=FR/ST=Paris/L=Paris Descartes/O=IUT/OU=IUT PARIS/CN=Mehdi/emailAddress=mehdi.zaoui@etu.u-pari
s.fr
Getting Private key
sim@VMDebian:~/rep1/rep1_1$
```

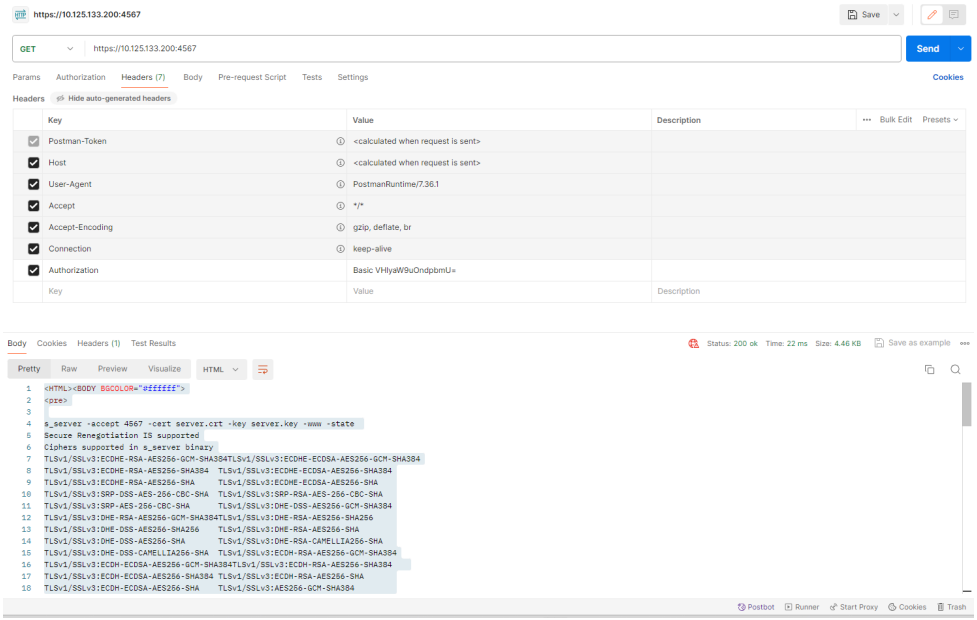
Et enfin la commande openssl s_server -accept 4567 -cert server.crt -key server.key :

```
sim@VMDebian:~/rep1/rep1_1$ openssl s_server -accept 4567 -cert server.crt -key server.key
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

Pour tester le certificat généré, on peut utiliser la commande suivante : openssl s_server -accept 4567 -cert server.crt -key server.key -www -state :

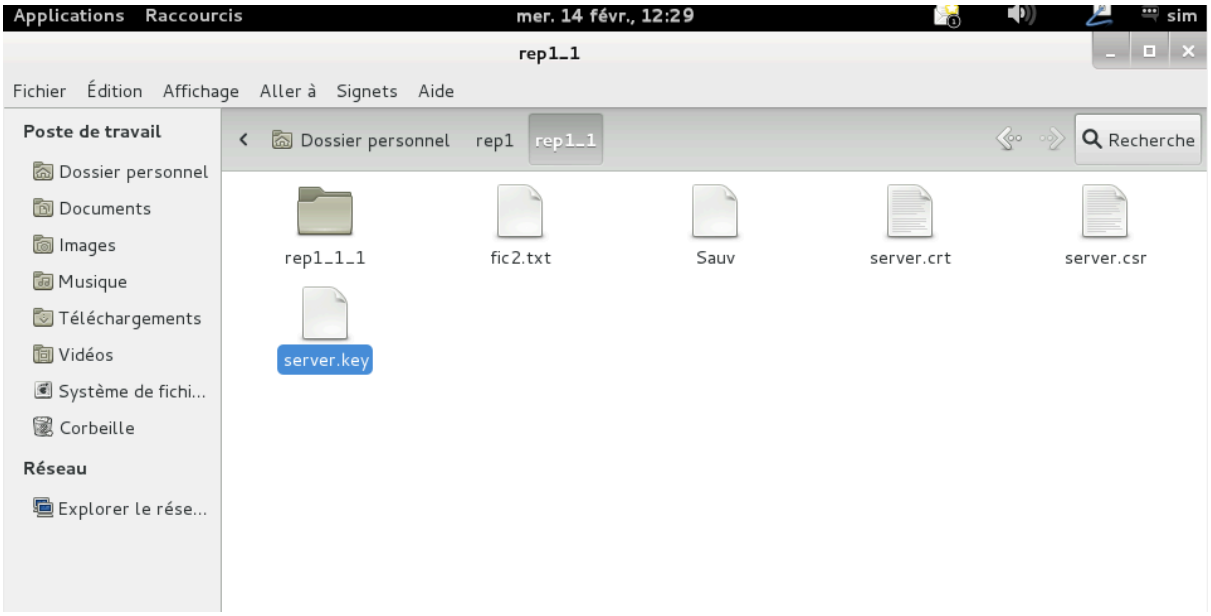
```
sim@VMDebian:~/rep1/rep1_1$ openssl s_server -accept 4567 -cert server.crt -key server.key -www -state
Using default temp DH parameters
Using default temp ECDH parameters
ACCEPT
```

Puis on test avec Postman https://localhost:4567/



La réponse que j'obtiens montre les options de configuration de mon serveur OpenSSL, les types de chiffrement pris en charge, et les détails de la connexion sécurisée établie entre mon serveur et Postman, comme le protocole SSL/TLS utilisé et le succès de la vérification de la connexion SSL. Cela confirme que la communication entre mon serveur et Postman est sécurisée et fonctionne correctement.

J’ai importé les fichiers :



Puis avec la commande :

```
const myFastify : FastifyInstance<...> & PromiseLike<...>  = Fastify( opts: { https: {
        key: readFileSync('server.key'),
        cert: readFileSync('server.crt')
    }});
```

Et j'obtiens :