# WTF is happening

Server.js

C’est dans ce fichier que l’on initialise le server sur le port 5000 et que l’on dit a notre app d’utiliser certain packages tel que body-parser et cors qui nous permette de faire nos requêtes en json et morgan qui nous permet d’avoir les logs des requêtes faite sur le serveur.

On exécute aussi le fichier routes.js

Routes.js

Ce fichier permet de définir les routes de notre application.

Sur le chemin /api/users on utilisera les requêtes provenant du fichier users.controller.js

Sur le chemin /api on aura seulement un get sur la route /dashboard qui sera accessible seulement avec un token grâce au middleware requireAuth provenant du module passeport-jwt.

users.controller.js

Dans ce fichier on défini l’action a faire quand on a des posts sur les routes /authenticate et /register (préfixer par /api/users).

On a donc une fonction authenticate et une fonction register provenant du fichier users.service.js

users.service.js

Dans ce fichier on a les fonctions register et authenticate.

Register permet de crée un nouvel utilisateur, grâce a un email et un mot de passe provenant du body de la requête, ce nouvel utilisateur est ensuite envoyé à la fonction createUser du fichier db afin d’être enregistrer dans la base de données

Authenticate utilise la fonction findUser du fichier db en regardant si l’email renseigné existe dans la base de données si l’email existe on récupère les données de l’utilisateur et on compare le mot de passe renseigner et le mot de passe crypté dans la bdd grâce a la fonction compareHash du fichier crypt. S’il y a match on crée un token avec une expiration

Crypt.js

Ce fichier contient deux fonction : createhash et compareHash .

CreateHash permet de crypté une data grâce au module bcrypt,

compareHash permet de comparer une data non crypté et une data crypté

db.js

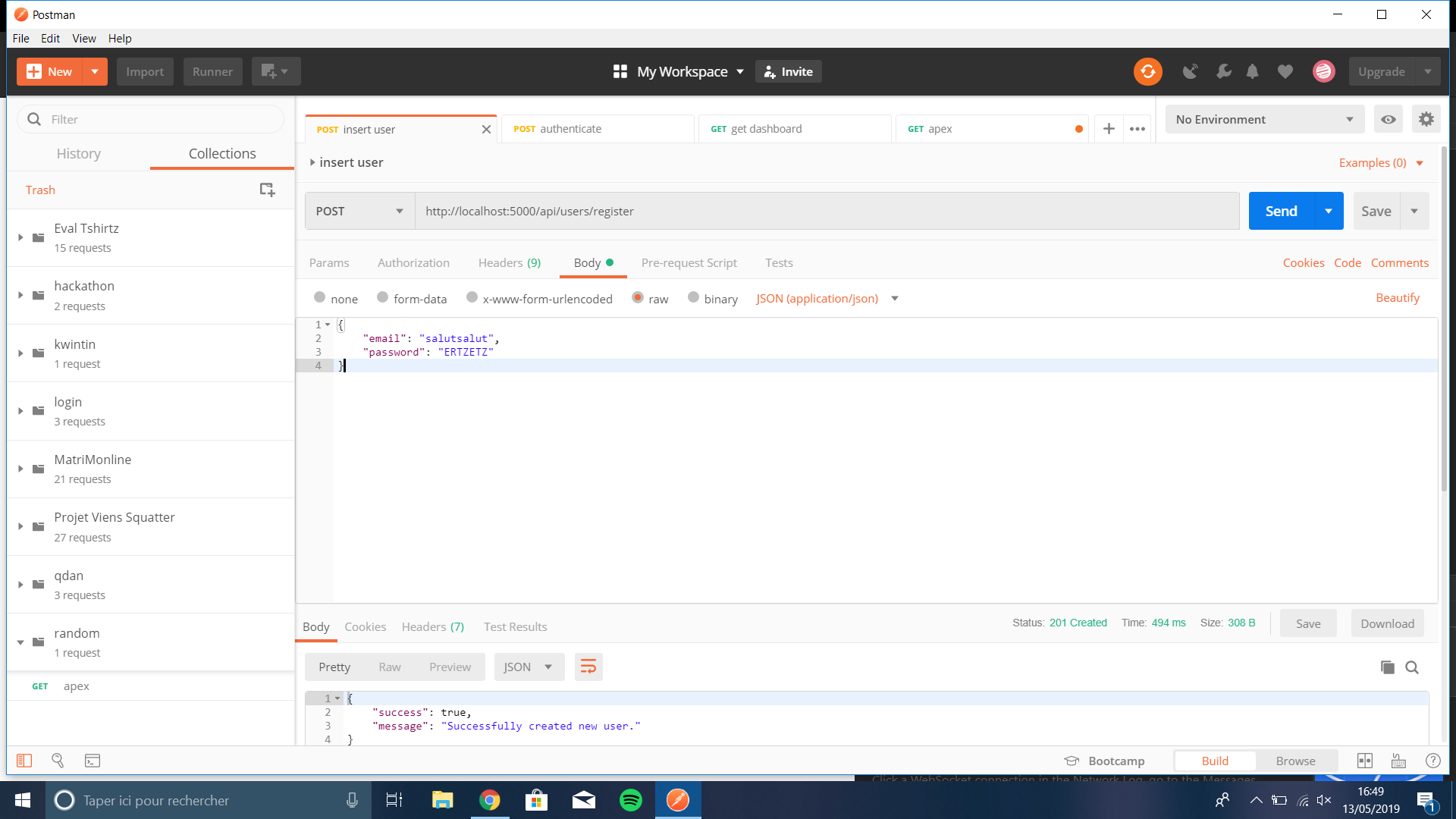
ce fichier contient la connexion a la base de donnée grâce aux identifiant provenant du fichier de config main.js, ce fichier contient aussi les fonction createUser et findUser.

CreateUser permet d’insérer le nouvel utilisateur provenant de users.service.js dans la base de données tout en cryptant son mot de passe grâce a la fonction createHash.

findUser permet de récupérer les information d’un utilisateur grâce à son email.

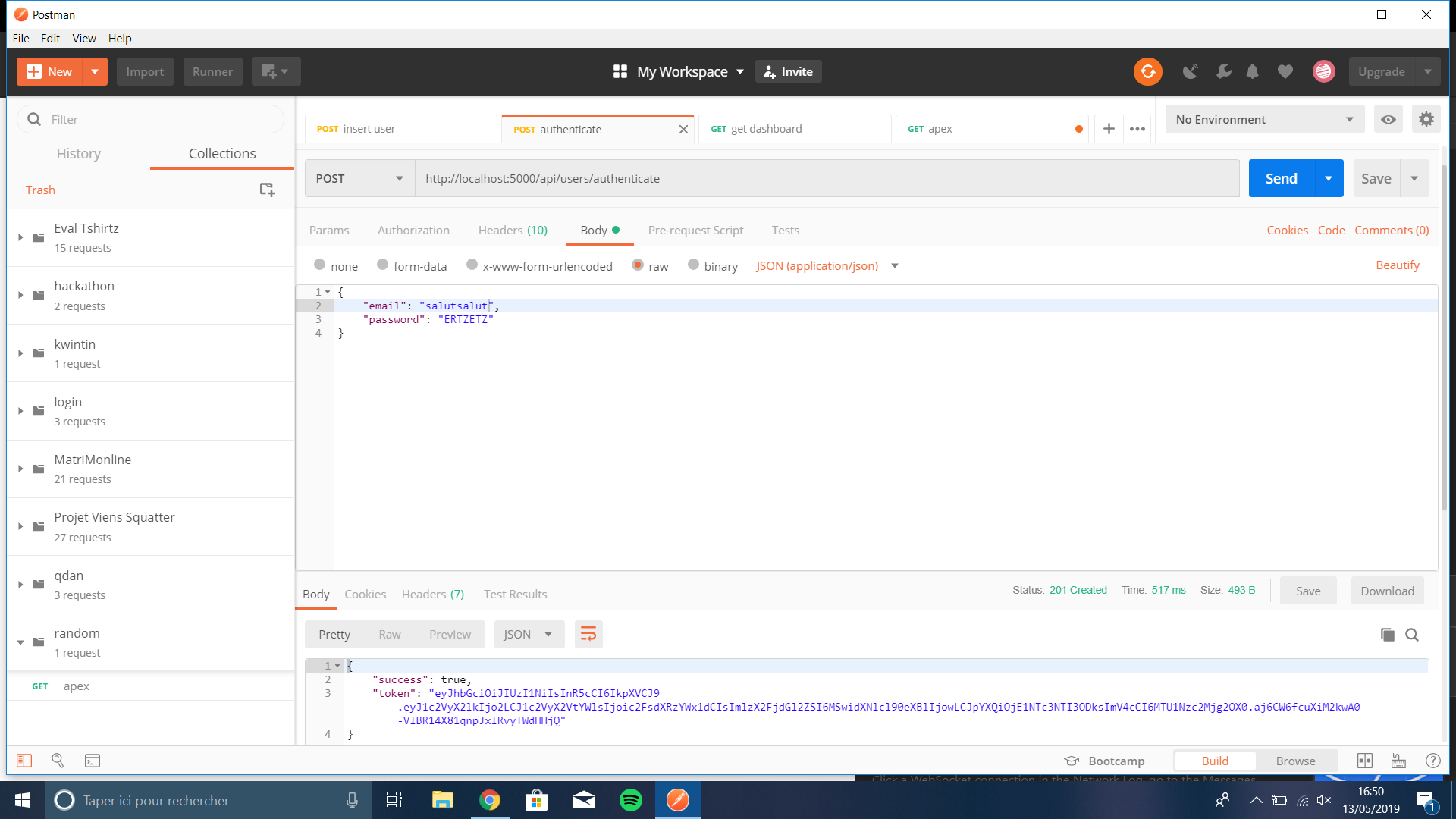
Utiliser l’api :

Pour crée un user :



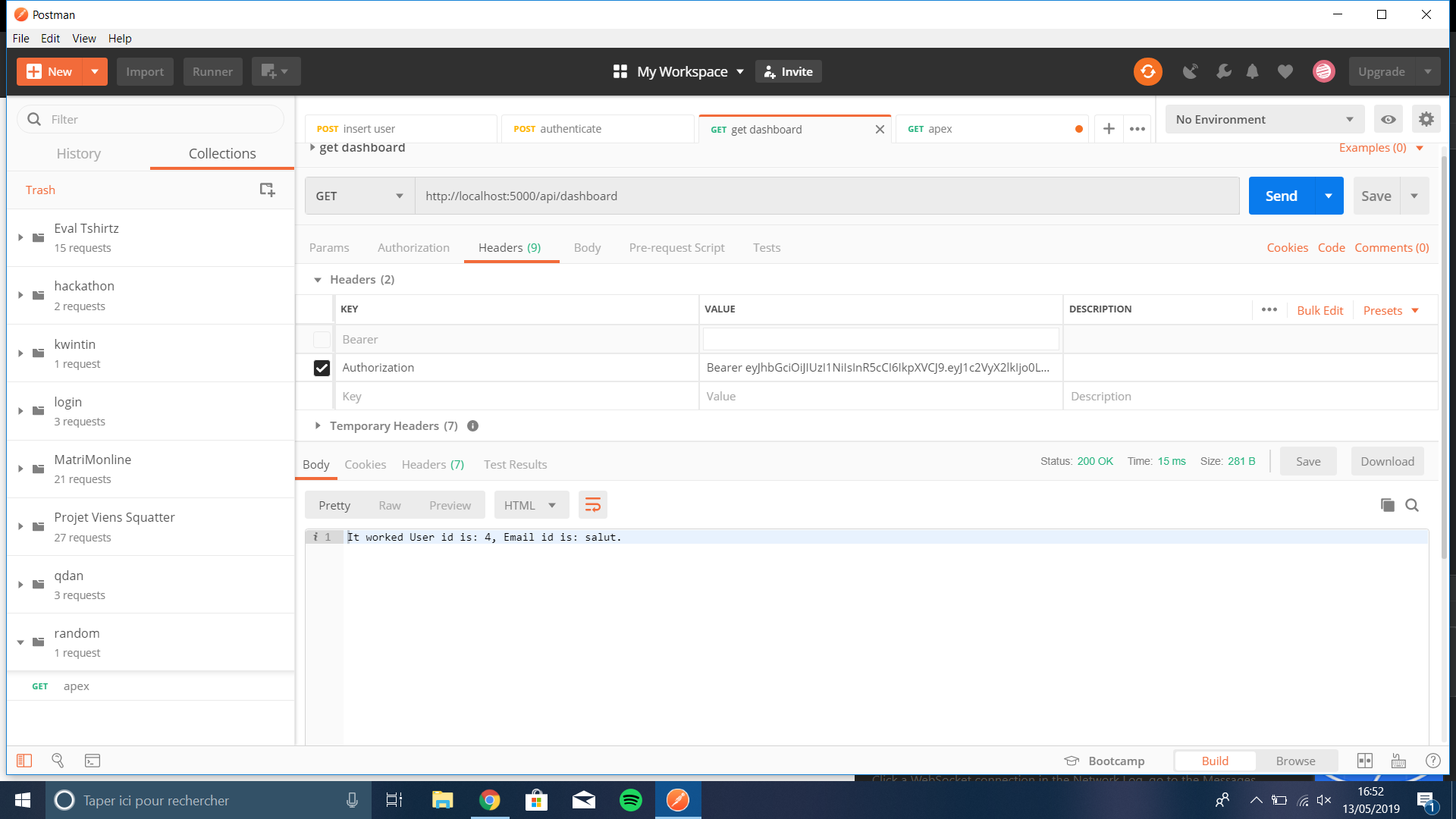
On va donc taper en POST sur http://localhost :5000/api/users/register avec en body de la requête un email et un password

Pour s’authentifier :



On va taper en POST sur http://localhost :5000/api/users/authenticate avec un email et un password en retour on obtient un token

Pour get le dashboard :



On tape en get sur <http://localhost:5000/api/dashboard> avec dans le header de la requête un clé Authorization et en valeur un premier mot clé Bearer avec ensuite le token ou directement dans l’onglet authorization on peut entrer un bearer token.