P\_APP-183



Rotzetter Ethan – MID2A

Lausanne, Venne

24 P

Gaël Sonney

Table des matières

[1. conceptualisation (schéma) 2](#_Toc192241614)

[2. explications sur le code 3](#_Toc192241615)

[1.1 Profil du client 3](#_Toc192241616)

[1.2 HTTPS 4](#_Toc192241617)

[1.3 Authentification par mot de passe 5](#_Toc192241618)

[1.4 Vérification du token JWT 7](#_Toc192241619)

[1.5 Administration 8](#_Toc192241620)

[1.6 Protection contre les injections SQL 8](#_Toc192241621)

[1.7 Utilisation de bcrypt 8](#_Toc192241622)

[1.8 Versioning 9](#_Toc192241623)

[3. conclusion sur le travail fourni et sur l’attitude face au projet 9](#_Toc192241624)

[4. Webographie 9](#_Toc192241625)

# conceptualisation (schéma)

Une image contenant capture d’écran, conception

Description générée automatiquement

Figure Schéma

# explications sur le code

## Profil du client

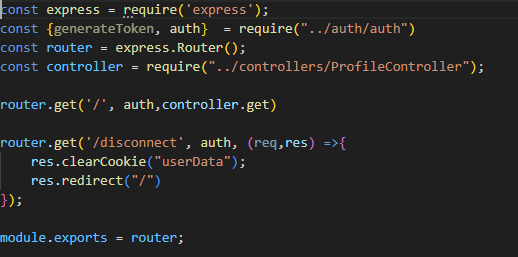


Figure Le contrôleur de profile



Figure 3 Le code du Router profile

Le code ci-dessus permet de vérifier si un utilisateur qui veut aller sur la page login est connecté, c’est vérifié à l’aide des cookies. Si l’utilisateur n’est pas connecté, ce-dernier sera redirigé vers la page de connexion. Si l’utilisateur a les droits administrateurs, alors un lien indisponible pour les autres apparaîtra. Si une erreur est attrapée, le message d’erreur apparaitra dans la console et l’utilisateur sera redirigé sur la page d’accueil.

## HTTPS

Voici les étapes de réalisation pour avoir HTTPS :

Je commence par faire cette commande :

openssl genpkey -algorithm RSA -out privkey.key

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Puis celle là:

openssl req -new -key privkey.key -out request.csr

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Et pour finir :

openssl x509 -req -in request.csr -signkey privkey.key -out certificate.crt -days 365

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

**Résumé des commandes:**

générer une clé privée :

- openssl genpkey -algorithm RSA -out privkey.key

créer une demande de signature de certificat (CSR) :

- openssl req -new -key privkey.key -out request.csr

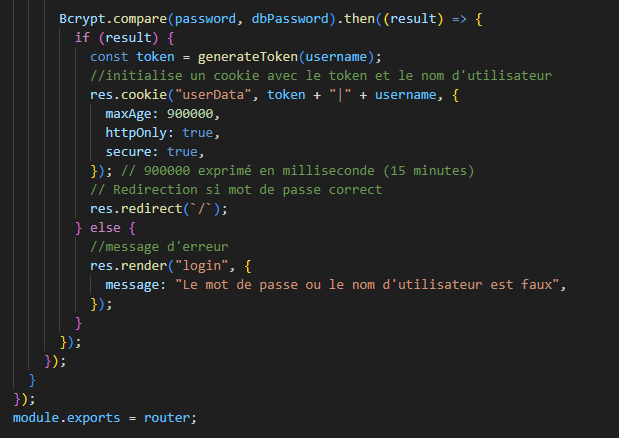
signer cette demande pour produire un certificat auto-signé :

- openssl x509 -req -in request.csr -signkey privkey.key -out certificate.crt -days 365

## Authentification par mot de passe







## Vérification du token JWT

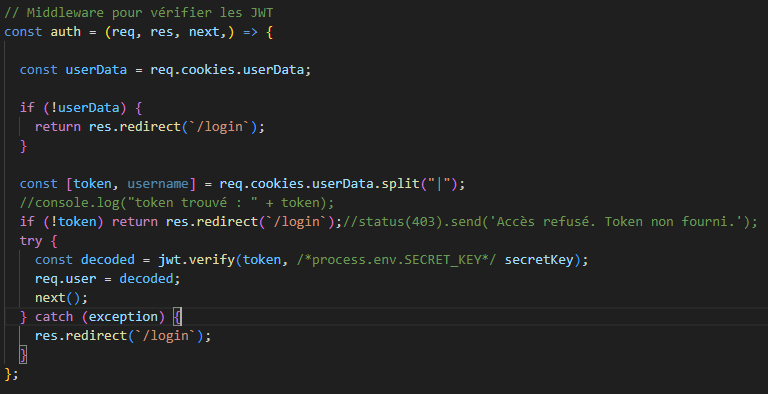


Figure 4 Middleware "auth"

Ce middleware vérifie la validité d’un token extrait d’un cookie. Ce code utilise la librairie JsonWebToken. Si le token a pu être décodé, l’utilisateur pourra aller sur la page protégée. Dans le cas contraire, une redirection sera faite pour qu’il puisse se connecter.

## Administration

## Protection contre les injections SQL

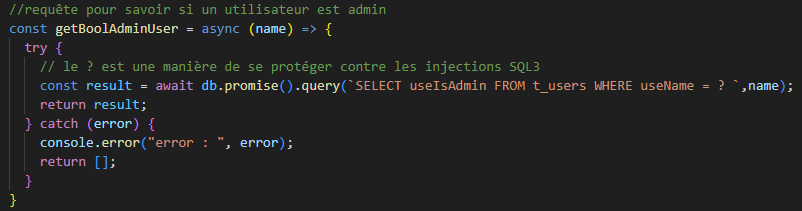


Figure 5 Exemple de protection

Dans cet exemple, la protection contre les injections est appliquée. Dans la requête SQL, il y a un point d’interrogation qui prendra la valeur de la variable name.

## Utilisation de bcrypt

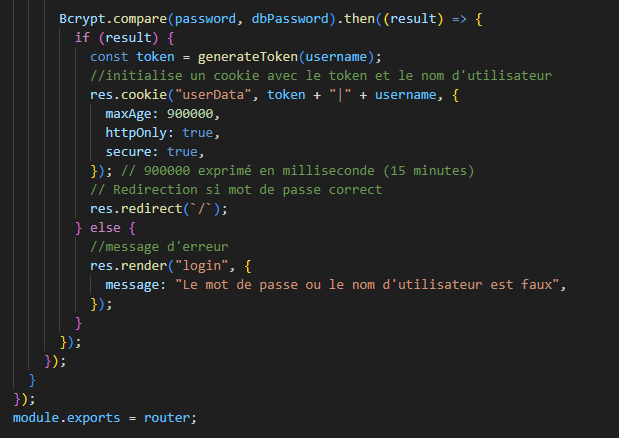


Figure Bcrypt pour le login

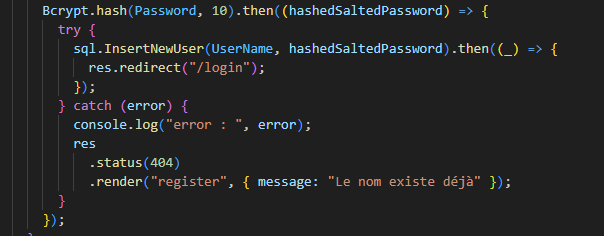


Figure Bcrypt pour le registre

## Versioning

# conclusion sur le travail fourni et sur l’attitude face au projet

# Webographie

Voici une liste approximative des sites utilisés pour le projet :

(Je mentionne des sites dans le code)

* [Créer un serveur HTTPS | NodeJS](https://node-js.fr/server/ssl.html)
* [Node JS fs.readFileSync() Method - GeeksforGeeks](https://www.geeksforgeeks.org/node-js-fs-readfilesync-method/)