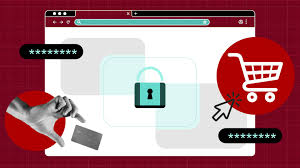
P\_App 183 – Secured WebShop



Mathis Olaya

24 périodes

Lausannes – Vennes

Table des matières

[1) Schéma du projet 3](#_Toc192505425)

[2) Explications du code 3](#_Toc192505426)

[a) Login 4](#_Toc192505427)

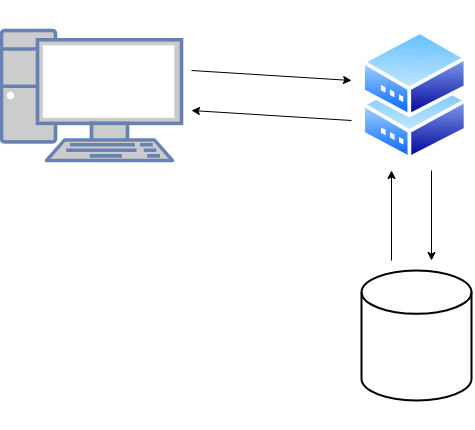
[b) Génération d’un Token JWT 5](#_Toc192505428)

[c) Middleware d’authentification 6](#_Toc192505429)

[d) Hachage sans librairie 7](#_Toc192505430)

# Schéma du projet

Ci-dessous se trouvera le schéma représentant l’infrastructure matériel du projet.



Comme vous pouvez le remarquer, le premier élément de cette infrastructure, est un PC ordinaire. Autrement dit, celui-ci représente un utilisateur classique visitant le site Web. Que se soit sur mobile, portable, tablette, etc. Ensuite, le serveur, celui-ci s’occupe de communiquer avec l’utilisateur et la base de données. Il est responsable de renvoyer les données, et les pages HTML. Et finalement, le base de données, celle-ci communique donc uniquement avec l’utilisateur, pas le biais du serveur. Elle contient toutes les informations relatives aux utilisateurs.

# Explications du code

Ci-dessous se trouvera diverses explications relatives au code du site web, plus précisément des fonctionnalités précises.

## Login



Cette fonction est utilisée pour vérifier les entrées d’un utilisateur lors de la tentative de connexion à son compte. Voici comment elle fonctionne :

1. Les entrées de l’utilisateur sont récupérées dans le Body de la requête.
2. Le serveur demande à la base de données les informations dont le nom d’utilisateur est celui récupéré juste avant.
3. Il se peut qu’il y ait une erreur du côté du serveur, si c’est le cas, on retourne une erreur.
4. S’il n’y pas de résultat, alors on retourne une erreur disant qu’aucun utilisateur n’existe sous ce pseudonyme.
5. Sinon, on récupère diverses informations, comme le mot de passe hashé, ou encore son rôle.
6. Si la base de données n’a pas trouvé de sel pour cet utilisateur, cela signifie que son mot de passe a été hashé avec la librairie ‘bcrypt’. Sinon, il a été hashé avec une fonction faite maison. A noter qu’il est possible de choisir la méthode de hashage depuis les variables d’environnements (.env)
7. En fonction du résultat du point n6, on compare le mot de passe avec celui de la base de données. Si les deux entrées ne correspondent pas, on retourne une erreur.
8. Si les deux entrées sont valides, alors l’utilisateur est autorisé à accéder à son profile. Pour ce faire, on créer un jeton JWT, que nous enregistrons ensuite sur les cookies du serveur.
9. Puis, le serveur retourne un message de succès.

## Génération d’un Token JWT



Cette fonction permet de générer un jeton JWT. Ceux-ci contiennent des données relatives à l’utilisateur, et une date d’expiration. Ils permettent de confirmer l’identité d’un utilisateur durant la navigation sur le site. Pour ce faire, à chaque fois que l’utilisateur tente d’accéder à une ressource (page), le serveur vérifiera son jeton.

En premier temps nous récupérons une clé secrète stockées dans les variables d’environnement. Celle-ci doit être gardée secrète, car elle permet de signer le jeton.

Ensuite, nous définissons des options, dans ce cas, la durée du jeton, soit 1 heure.

Et finalement nous créons et renvoyons un jeton, avec pour données, celles transmises lors de l’appelle de la fonction (data).

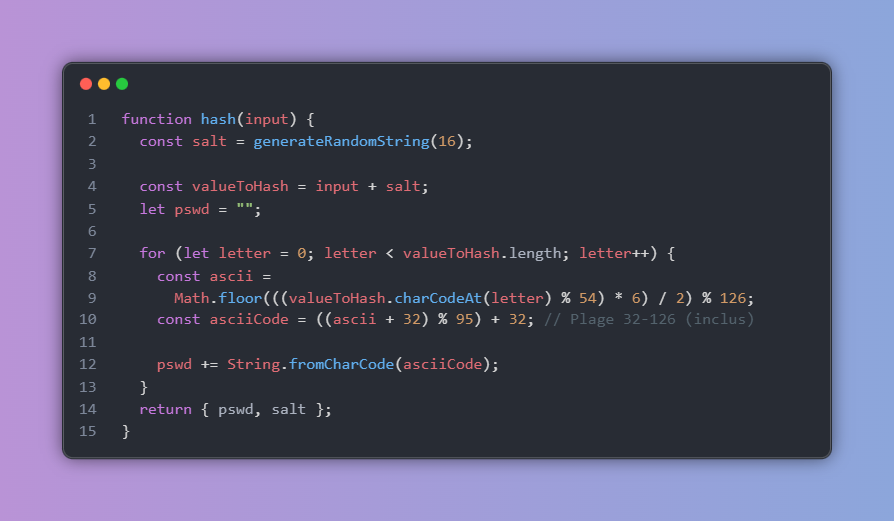
## Middleware d’authentification



Cette fonction nommée ‘verifyToken’, a pour but de vérifier l’identité de l’utilisateur à chaque fois qu’il essaie d’accéder à une route (page).

Pour ce faire, comme défini dans les points précédents, le jeton est récupéré dans les cookies. Si le serveur ne le trouve pas, on renvoie une erreur, qui signifie surement que l’utilisateur n’est pas connecté. Ensuite, on utilise une fonction fournie par ‘jsonwebtoken’, celle-ci permet de vérifier le jeton récupéré. S’il y a une erreur, on la renvoie, celle-ci signifie généralement que le jeton de l’utilisateur à expiré. Et finalement, on compare le nom d’utilisateur enregistré avec celui de la page que l’utilisateur tente d’accéder. Cela permet d’éviter l’accès d’un utilisateur ‘A’ à la page d’un utilisateur ‘B’. Si tout est valide, on utilise la fonction next(), qui permet de passer à la suite du code.

## Hachage sans librairie



Cette fonction est utilisée pour générer un hash à partir d’une entrée, généralement un mot de passe. A noter que par la suite du projet, on à également inclus la possibilité d’utiliser bcrypt à la place.

En premier temps, nous générons un sel aléatoire. Celui-ci est ajouté à la fin des mots de passes afin de rendre les attaques par rainbow tables plus compliqué. Et ensuite, on effectue une suite d’opérations mathématique pour générer des lettres hachées.

Et finalement on retourne le mot de passe, et le sel, qui seront stockés sur la base de données.