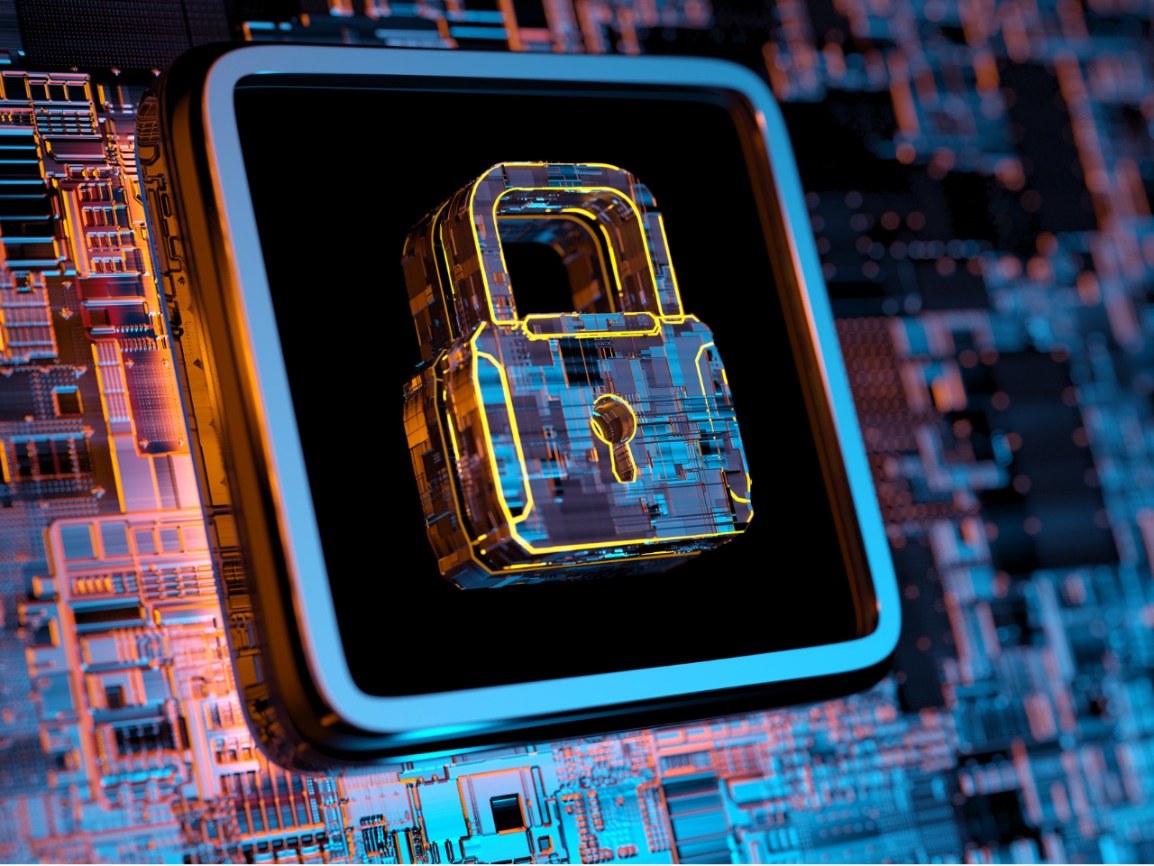
P\_Web 183



Romain Denis – CID2B

ETML - Vennes

8 Semaines

Cédric Schaffter

Table des matières

Table des matières

[1 Spécifications 3](#_Toc192834237)

[Titre 3](#_Toc192834238)

[Matériel et logiciels à disposition 3](#_Toc192834239)

[Les points suivants seront évalués 3](#_Toc192834240)

[Validation et conditions de réussite 3](#_Toc192834241)

[2 Réalisation 3](#_Toc192834242)

[Dockerisation 3](#_Toc192834243)

[Profil du client 3](#_Toc192834244)

[HTTPS 4](#_Toc192834245)

[Authentification par mot de passe 4](#_Toc192834246)

[Vérification du token JWT 4](#_Toc192834247)

[Administration 5](#_Toc192834248)

[Protection contre les injections SQL 5](#_Toc192834249)

[Utilisation de bcrypt 5](#_Toc192834250)

[Versioning 5](#_Toc192834251)

[Documentation 6](#_Toc192834252)

[Utilisation de l’IA 6](#_Toc192834253)

[Schéma 6](#_Toc192834254)

[3 Conclusion 8](#_Toc192834255)

# Spécifications

## Titre

P\_APP\_183 : Secured WebShop

## Matériel et logiciels à disposition

Un ordinateur standard de la section informatique avec Docker Desktop

## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

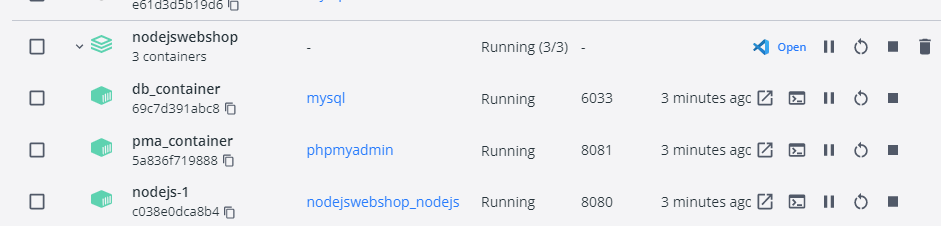
## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Réalisation

## Dockerisation

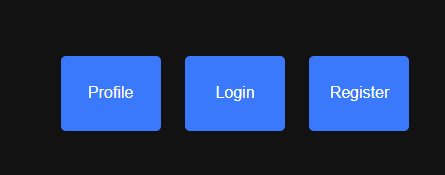
*L’ensemble des services web sera conteneurisé.*

**

## Profil du client

*Le client peut accéder à son propre profil depuis la page de bienvenue. Seul son profil lui sera rendu visible et les routes du backend seront protégées.*

Fait, pas le temp de faire la doc

**

## HTTPS

*Il doit être possible d’accéder à votre site de e-commerce de manière sécurisée (https://localhost). Le port utilisé sera le 443. Le certificat sera auto-signé par OpenSSL*

Je n’ai pas eu le temp de faire ceci.

## Authentification par mot de passe

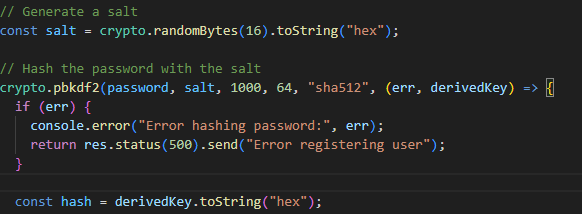
*L’utilisateur devra s’authentifier par la page https://localhost/login Le mot de passe sera haché et salé avant d’être stocké dans la base de données (table t\_users).*

Le code hache et sale le mot de passe avant de le placer dans la base de données. Cela se fait dans la route POST pour /register, avec le code ci-dessous.

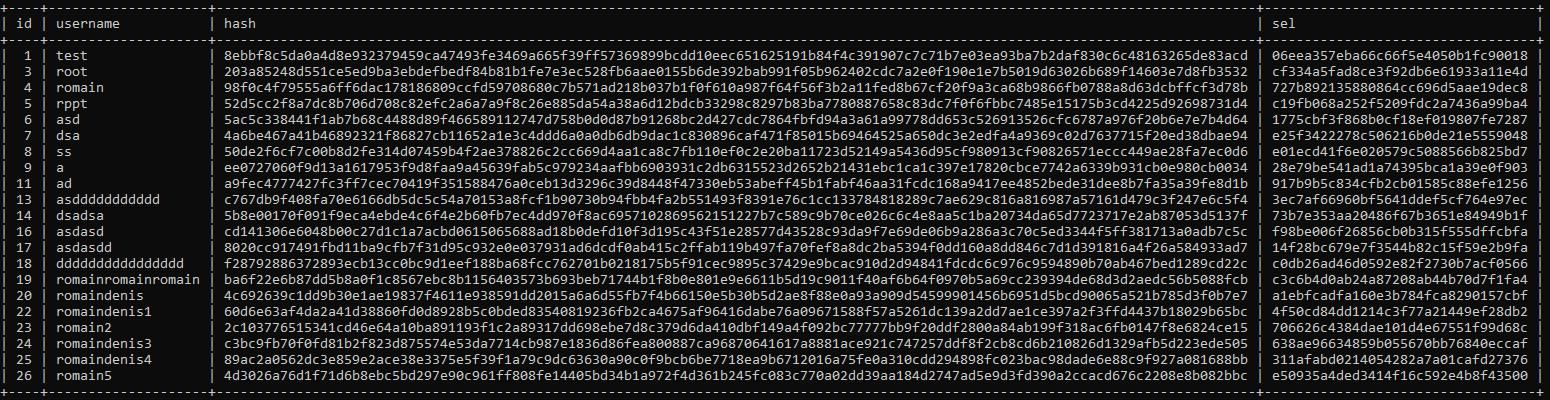
Le code génère d’abord un sel (chaîne aléatoire de caractères) avec le paramètre 16 pour indiquer la longueur de bytes.

Ensuite, le code utilise la fonction pbkdf2 (Password-Based Key Derivation Fonction 2) de la bibliothèque crypto pour obtenir le mot de passe.

Les arguments sont : *password,* qui est le mot de passe défini par l'utilisateur, *salt*, le sel généré aléatoirement par crypto au-dessus, *1000*, le nombre d'itérations pendant lesquelles la clé est hachée, *64*, la longueur d'une clé et *sha512,* l'algorithme utilisé.



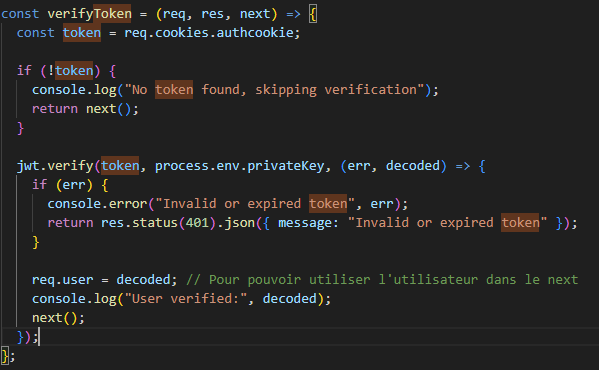
En conséquence, la base de données ressemble à ceci, avec le mot de passe haché et le sel avec l'utilisateur correspondant :

**

## Vérification du token JWT

*Le token devra être vérifié lorsque l’utilisateur vous l’envoie et la signature du jeton devra aussi être utilisée pour vérifier que le token n’a pas été modifié.*

Le code vérifie automatiquement les JWT à l'aide du middleware verifyToken ci-dessous :



Ce code récupère le jeton JWT des cookies du site à l'aide de l'analyseur de cookies. S'il n'y a pas de jeton, la vérification est ignorée. Le programme qui utilise la fonction de la librairie jsonwebtoken, verify pour vérifier si le JWT est valide ou non. Si c'est le cas, l'utilisateur est autorisé à aller où il le souhaite.

## Administration

*Une page d’administration devra avoir un champ de recherche (Nom du visiteur) et permettre d’afficher tous les utilisateurs ayant tout ou partie de ce nom.*

Je n’ai pas eu le temp de faire ceci.

## Protection contre les injections SQL

*Votre page d’administration devra être protégée contre les injections SQL. Sans utiliser sequelize ou tout autre ORM, votre site devra être robuste face aux injections. L’utilisation de sequelize (ou autre ORM) sera autorisée dans l’approfondissement du projet.*

Le code est protégé des injections SQL en utilisant des instructions préparées qui présentent une requête légèrement différente afin de représenter les entrées utilisateur et d'indiquer à la base de données que l'utilisateur ne peut pas quitter la requête et injecter du code malveillant.

Les entrées utilisateur sont représentées par les (?, ?, ?) et la ligne en dessous qui met les valeurs que l'utilisateur a saisies, pour éviter les injections SQL. Cela signifie également que nous n'avons pas besoin d'utiliser un ORM tel que sequelize pour ce projet

## Utilisation de bcrypt

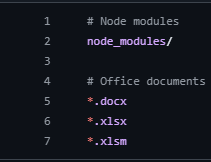
*Une fois l’authentification faite avec votre propre mécanisme (mot de passe en base de données, salé et haché). Vous modifierez votre code pour utiliser bcrypt. Votre code doit prévoir une condition pour utiliser soit bcrypt soit votre propre implémentation.*

Je n’ai pas eu le temp de faire ceci.

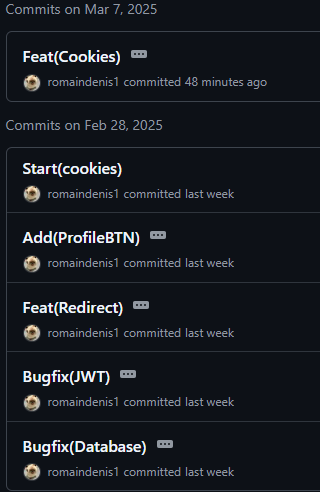
## Versioning

*Votre code sera versionné sur Github et un .gitignore empêchera de versionner les binaires npm (dossiers node\_modules, documents word/excel). Votre dépôt sera partagé avec votre chef de projet dès le début du projet.*

Lien du repo GitHub : [romaindenis1/P\_APP](https://github.com/romaindenis1/P_APP)

Voici mon .gitignore, qui ignore les fichiers office et les binaires npm 

Tous les changements de code sont faits public avec des commits GitHub bien nommés et claires.



## Documentation

• Journal de travail (1 ligne par quart d’heure de travail)

• Rapport, contenant :

Votre conceptualisation (schéma)

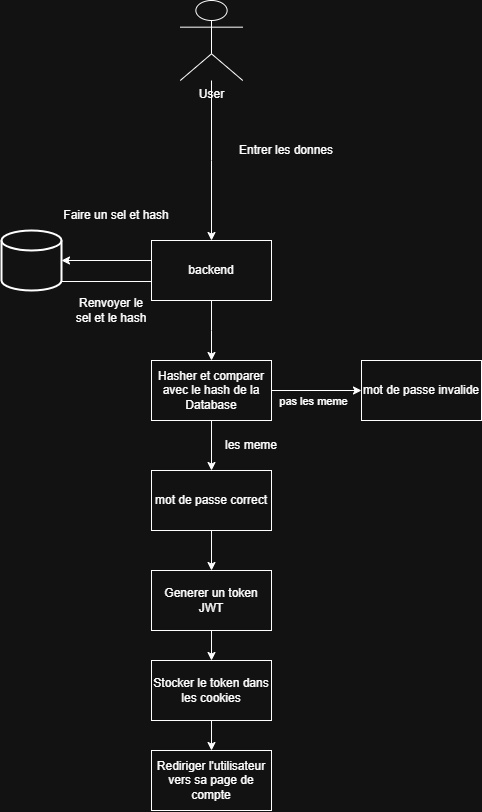
Des explications sur votre code o Une conclusion sur le travail fourni et sur l’attitude face au projet

## Utilisation de l’IA

Une utilisation injustifiée de l’IA ou une absence de maîtrise du code, induira un non-acquis dans les compétences techniques du projet.

## Schéma

Voici le schéma de la procédure de login



# Conclusion

En conclusion, j'ai beaucoup aimé ce projet. Je pense qu'il a offert deux perspectives différentes sur la sécurité : celle du développeur avec de la sécurisation d'un site web, et la perspective de l'attaquant qui tente de rentrer dans le site. C'était très intéressant et j'aurais aimé que d'autres projets fassent de même. Je pense que le CDC avait des aspects que j'aurais pu améliorer, par exemple la page d'administration. Cela n'aurait pas pris beaucoup de temps et je pense que je continuerai à y travailler pendant mon temps libre. Je trouve ce projet excellent et j'aurais aimé que le module soit plus long pour que nous puissions adopter une approche plus axée sur les tests d'intrusion.

# Annexes

[romaindenis1/P\_APP](https://github.com/romaindenis1/P_APP)

(Il est possible que le dernier commit ne marche pas, dans ce cas, utilisez l’avant dernier)