P\_Sec



Emma Blanchoud – MID2A

Vennes

24 périodes

Gaël Sonney

Table des matières

[1 Conceptualisation 3](#_Toc188274215)

[2 Réalisation 3](#_Toc188274216)

[2.1 Dockerisation 3](#_Toc188274217)

[2.2 Profil du client 3](#_Toc188274218)

[2.3 HTTPS 3](#_Toc188274219)

[2.4 Authentification par mot de passe 3](#_Toc188274220)

[2.5 Vérification du token JWT 4](#_Toc188274221)

[2.6 Administration 4](#_Toc188274222)

[2.7 Protection contre les injections SQL 4](#_Toc188274223)

[2.8 Utilisation de bcrypt 4](#_Toc188274224)

[2.9 Versioning 4](#_Toc188274225)

[3 Conclusion 4](#_Toc188274226)

[3.1 Bilan personnel 4](#_Toc188274227)

# Conceptualisation

# Réalisation

## Dockerisation

L’ensemble des services web sera conteneurisé.

Dans ce projet il est nécessaire de créer un docker conteneurisé. Il sera nommé webshop\_183 et contient pma\_container et db\_container qui sont les containers par défaut ainsi que webshop\_183-nodejs-1.

## 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.

## 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.

Changer le port :

// Démarrage du serveur

app.listen(443, () => {

    console.log('Server running on port 443');

});

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 la demande pour produire un certificat auto-signé :

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

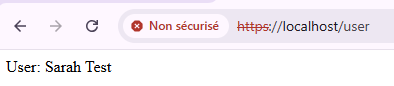
// Charger les clés SSL

const privateKey = fs.readFileSync("keys/privkey.key", "utf8");

const certificate = fs.readFileSync("keys/certificate.crt", "utf8");

const credentials = { key: privateKey, cert: certificate };

Le certificat est auto-signé alors pour chrome il n’est pas reconnu c’est pourquoi il ne fonctionne pas.



## 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).

## 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.

## 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.

## 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.

## 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.

## 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.

# Conclusion

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.