P_APP_183

Figure 1 - Image de titre - https://pxhere.com/fr/photo/1440395

Segalen Alban – mid2b
Vennes
24p
Cédric Schaffter

Table des matières

1 SPÉCIFICATIONS

1.1 Titre

P_APP_183 : Secured WebShop

1.2 Description

Création d'un site d'e-commerce sécurisé

1.3 Matériel et logiciels à disposition

Un ordinateur standard de la section informatique avec Docker Desktop

1.4 Prérequis

Suivre le module 183

1.5 Les points suivants seront évalués

- Le rapport
- Les planifications (initiale et détaillée)
- Le journal de travail
- Le code et les commentaires
- Les 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

2 ANALYSE

L'objectif de ce projet consiste à réaliser un mini site web sécurisé.

Il faudra stocker le mot de passe de manière sécurisée et empêcher les injections SQL.

Il faudra une page de connexion, une page qui affiche les détails de l'utilisateur connecté, et une page d'administration avec un champ de recherche.

3 CONCEPTION

La sécurité est centrale dans ce projet, c'est pourquoi une protection du mot de passe est mise en place. Le mot de passe est salé, poivré et haché avant d'être stocké dans la base de données.

Pour connecter les utilisateurs, l'application utilise des jetons JWT (Json Web Token). Les informations stockées dans le token sont : le nom de l'utilisateur, son identifiant et son droit administrateur. Le jeton est ensuite stocké dans les cookies.

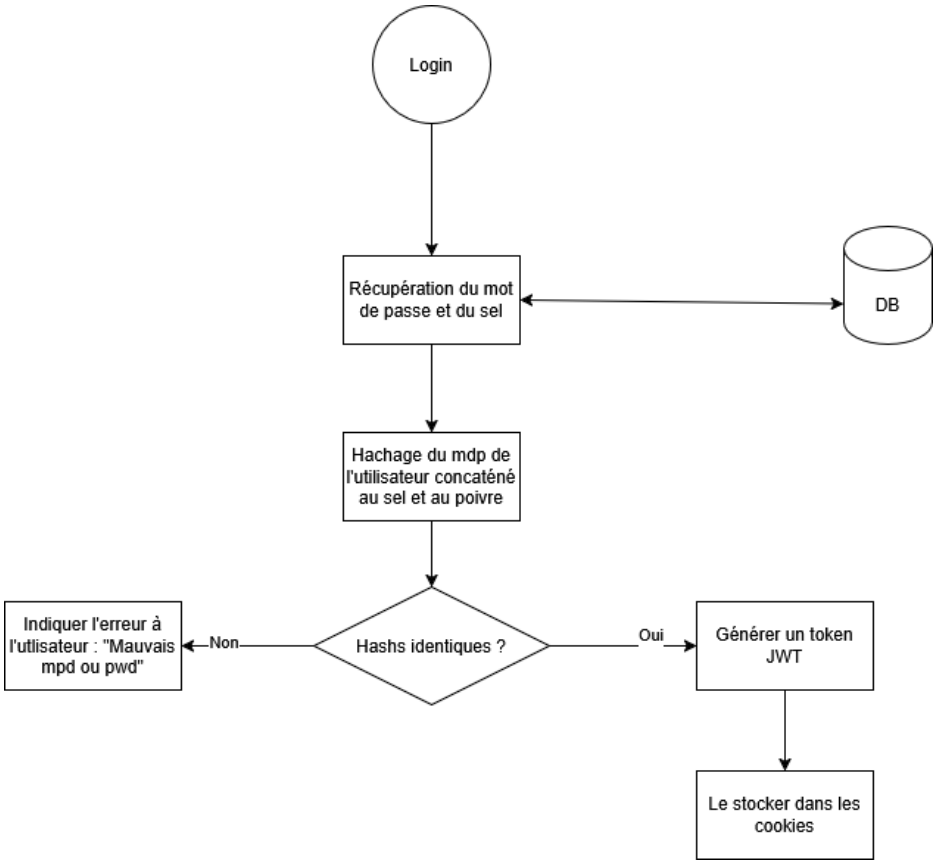

Figure 2 - Schéma de connexion à l'application

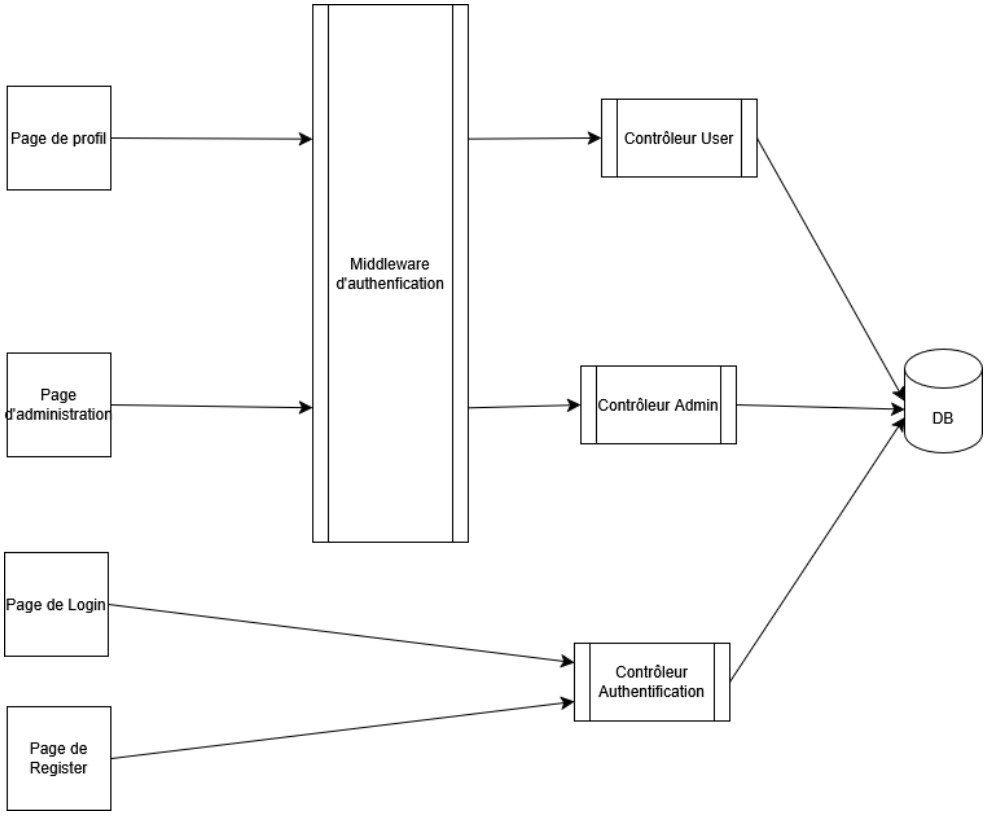

Figure 3 - Schéma de l'application

4 RÉALISATION

4.1 Dossier de Réalisation

4.1.1 Mise en place de docker

```
C:\Users\po37sqb\Documents\GitHub\P_APP_183_Secured_WebShop\Application>docker compose up -d
[+] Running 0/2
 - phpmyadmin Error                                                                                                  2.0s
 - db Error                                                                                                          2.0s
Error response from daemon: toomanyrequests: You have reached your pull rate limit. You may increase the limit by authen
ticating and upgrading: https://www.docker.com/increase-rate-limit
```

Figure 4 - Erreur lors de la composition du conteneur

Voici l'erreur qui est arrivée lors de la composition des conteneurs. Cette erreur est due au fait que trop de personnes essaie de télécharger des images en même temps depuis la même adresse IP. Pour résoudre cette erreur, il suffit de se connecter à docker.

4.1.2 Base de données

```sql
CREATE DATABASE db_webShop;
USE db_webShop;

CREATE TABLE t_users(
   users_id INT AUTO_INCREMENT,
   username VARCHAR(50) NOT NULL,
   password VARCHAR(256) NOT NULL,
   PRIMARY KEY(users_id),
   UNIQUE(username)
);
```

Figure 5 - Script de création de la base de données

4.1.3 Commit du .env

Après avoir ajouté le fichier .env qui contient les informations de connexion à la base de données. Les informations qui ont fuitées ne doivent plus jamais être utilisées. Le fichier à été remplacé par un fichier .env.example.

```
DBHOST=db_container
DBUSER=root
DBPASSWORD=root
DBPORT=3306
DBDATABASE=db_webShop
```

Figure 6 - Informations qui ont fuitées du .env

4.1.4 Procédure stockée d'insertion d'un utilisateur

Comme l'application permet la création d'un compte utilisateur et que le nom d'utilisateur doit être unique, il arrive qu'une erreur arrive quand un utilisateur essaie de s'inscrire avec un nom qui existe déjà. Pour éviter cela, une première solution consister à effectuer une requête pour vérifier que le

nom n'est pas déjà pris avant d'effectuer la requête d'insertion. Cependant, cette double requête n'est pas très écologique.

Pour éviter cela, une procédure stockée a été créées.

```
DELIMITER //
CREATE PROCEDURE INSERT_USER(IN useusername VARCHAR(50), IN
usepassword VARCHAR(64), IN usesalt VARCHAR(16), OUT sucess
BOOLEAN )
BEGIN
  DECLARE userCount DECIMAL(10,2) DEFAULT 0;

    SELECT COUNT(username)
    INTO userCount
    FROM t_users
    WHERE username = useusername;

    IF userCount > 0 THEN
        SET sucess = FALSE;
    ELSE
        SET sucess = TRUE;
        INSERT INTO t_users (username, password, salt)
VALUES(useusername, usepassword, usesalt);
    END IF;
END //
DELIMITER ;
```

Figure 7 - Procédure stockée d'insertion d'un utilisateur

Cette procédure prend le nom d'utilisateur, le mot de passe et le sel en entrée et retourne un booléen qui représente le succès de l'insertion. Cette

solution n'a pas été retenue car je n'ai pas réussi à récupérer le booléen de succès dans mon code.

La solution finalement retenue consiste à rediriger l'utilisateur à la page d'accueil si une erreur survient.

```
  //Inscription
  register: (req, res) => {
    //Récupération des credentials
    const password = escapeHTML(req.body.password);
    const confirmPassword =
escapeHTML(req.body.confirmPassword);
    const username = escapeHTML(req.body.username);

    if (password === confirmPassword) {
      //Génération du sel
      const salt = crypto.randomBytes(8).toString("hex");

      //Hachage du mot de passe
      const hash = crypto.hash("sha256", password + salt +
process.env.PEPPER);

      //Insertion de l'utilisateur dans la DB
      db.connect();
      db.query(
        "INSERT INTO t_users (username, password, salt, admin)
VALUES (?, ?, ?, ?)",
        [username, hash, salt, false],
        function (error) {
          return res.redirect("/login");
        }
      );
    }
  },
```

Figure 8 - Code de la fonction d'inscription

4.1.5 Protection contre l'injection SQL

L'utilisation de requête paramétrée avec la librairie mysql2 rend le site résistant aux injections SQL.

4.1.6 Protection contre les failles XSS

Pour se protéger des failles XSS, la librairie escape-html a été utilisée. Le moteur de template EJS échappe également automatiquement les caractères spéciaux

4.1.7 Routes

Voici les routes implémentées :

Get	/admin	Page d'administration
Post	/admin	Recherche – pas implémentée
Get	/login	Formulaire de connexion
Post	/login	Connexion
Get	/register	Formulaire d'inscription
Post	/register	Inscription
Get	/logout	Déconnexion
Get	/user/ :id	Page de profil

5 TESTS

L'application n'a pas beaucoup été testée.

Inscription	OK
Inscription avec un nom qui existe déjà	OK
Login avec un compte qui existe	OK
Login avec un compte qui n'existe pas	OK

6 CONCLUSION

6.1 Bilan des fonctionnalités demandées

Toutes les fonctionnalités ont été implémentées, à l'exception des suivantes :

- Implémentation de https
- Recherche sur les utilisateurs
- Approfondissement avec Sequelize et Bcrypt

6.2 Bilan personnel

J'ai bien aimé ce projet et ce module, car j'ai appris l'importance de la sécurité et les nombreuses possibilités de ce domaine.
Ce projet m'a permis d'approfondir mes connaissances en JavaScript ainsi que de découvrir EJS.

7 DIVERS

7.1 Journal de travail

Document ci-joint

8 WEBOGRAPHIE

9 ANNEXES

Le code de l'application est disponible à l'adresse https://github.com/ASETML/P_APP_183_Secured_WebShop/tree/main