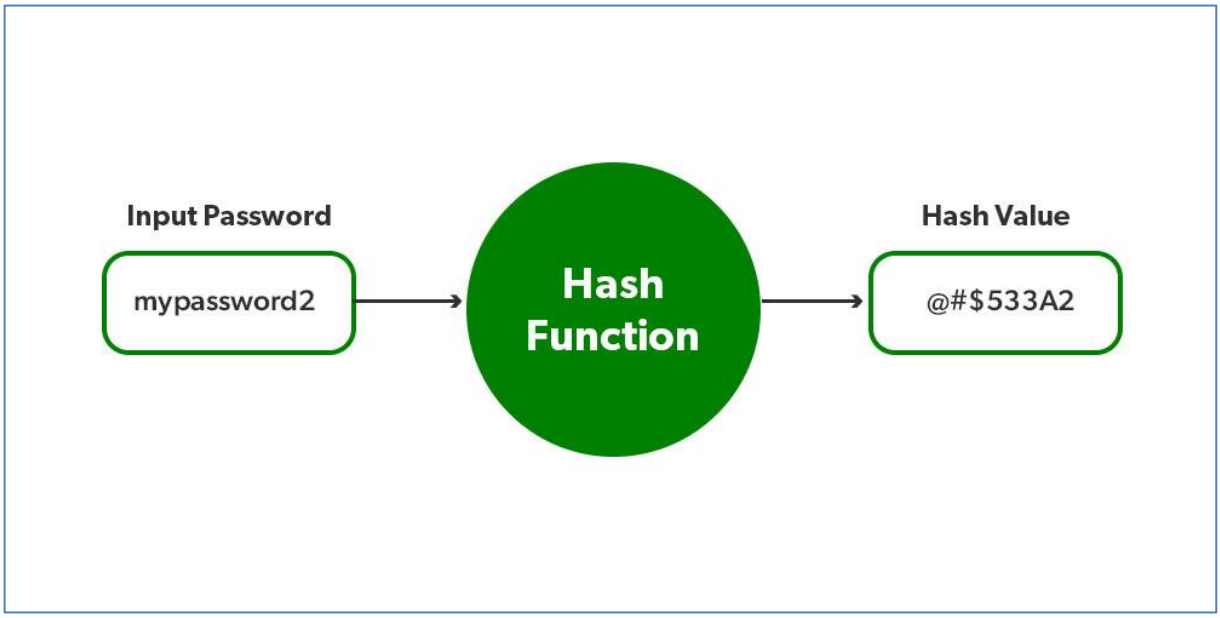
P_APP WEB_STORE



Gonzalo Javier Herrera Egoavil – CID2B
Vennes - Lausanne
24 périodes
Chef de projet : CSR

Table des matières

1 SPÉCIFICATIONS 3
1.1 TITRE 3
1.2 DESCRIPTION 3
1.3 MATÉRIEL ET LOGICIELS À DISPOSITION 3
1.4 PRÉREQUIS 3
1.5 CAHIER DES CHARGES 3
1.5.1 Dockerisation 3
1.5.2 Profil du client 3
1.5.3 HTTPS 3
1.5.4 Authentification par mot de passe 4
1.5.5 Vérification du token JWT 4
1.5.6 Administration 4
1.5.7 Protection contre les injections SQL 4
1.5.8 Utilisation de bcrypt 4
1.5.9 Versioning 4
1.5.10 Documentation 5
1.5.11 Utilisation de l'IA 5
1.6 LES POINTS SUIVANTS SERONT ÉVALUÉS 5
1.7 VALIDATION ET CONDITIONS DE RÉUSSITE 5
2 PLANIFICATION INITIALE 5
3 ANALYSE 6
3.1 CONCEPTUALISATION 6
3.2 EXPLICATION DU CODE 6
3.2.1 Contrôleurs 6
3.2.2 Database 7
3.2.3 Middleware 7
3.2.4 Views/Css 7
3.2.5 Server et routes 8
3.3 TRAVAIL FOURNI & ATTITUDE SUR LE PROJET 8
4 CONCLUSION 9
4.1 BILAN PERSONNEL 9
4.2 WEBOGRAPHIE 10
4.2.1 Utilisation de l'IA 10

1 SPÉCIFICATIONS

1.1 Titre

P_App-Webstore

1.2 Description

Développer un site e-commerce avec une authentification sécurisée (hashage personnalisé et JWT), incluant une page de login, une interface admin et une page utilisateur.

1.3 Matériel et logiciels à disposition

Un ordinateur standard de la section informatique avec Docker Desktop

1.4 Prérequis

I183-SécuritéApplication-CSR

1.5 Cahier des charges

1.5.1 Dockerisation

L'ensemble des services web sera conteneurisé.

1.5.2 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.

1.5.3 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.

1.5.4 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).

1.5.5 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.

1.5.6 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.

1.5.7 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.

1.5.8 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.

1.5.9 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.

1.5.10 Documentation

Journal de travail (1 ligne par quart d'heure de travail)

Rapport, contenant :
- Votre conceptualisation (schéma)
- Des explications sur votre code
- Une conclusion sur le travail fourni et sur l'attitude face au projet

1.5.11 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.

1.6 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

1.7 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

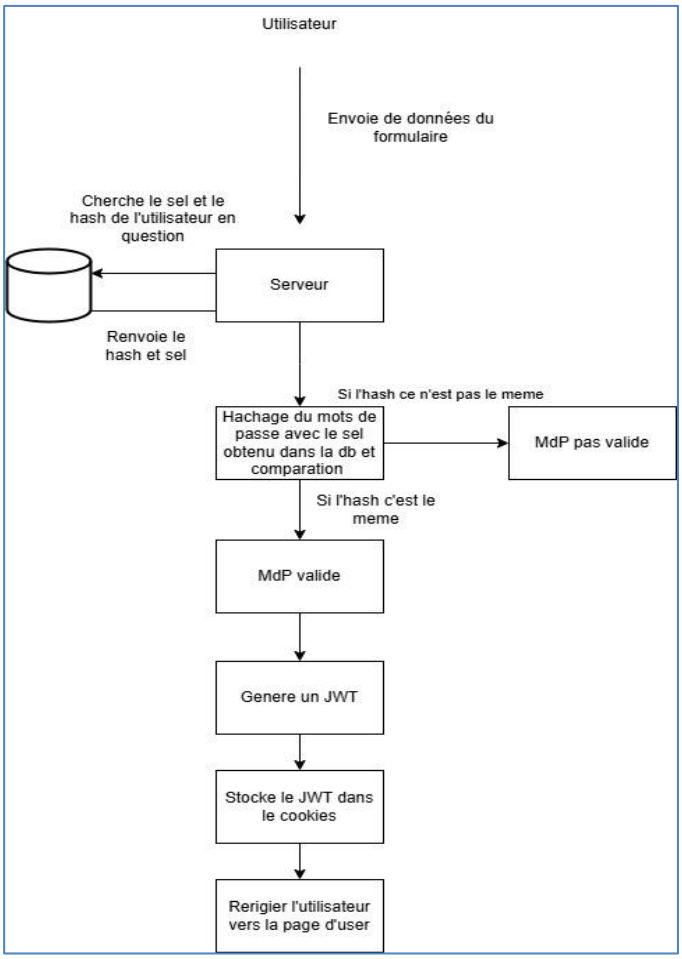
2 PLANIFICATION INITIALE

La planification à était réalisée de semaine en semaine, car au début on ne savait ce qu'il fallait faire, donc j'ai travaillé chaque semaine et à la fin je faisais la planification de la semaine suivante.

La planification se trouve dans le fichier **« JdT – Gonzalo Herrera.xlsm »** dans le dossier **« docs ».**

3 ANALYSE

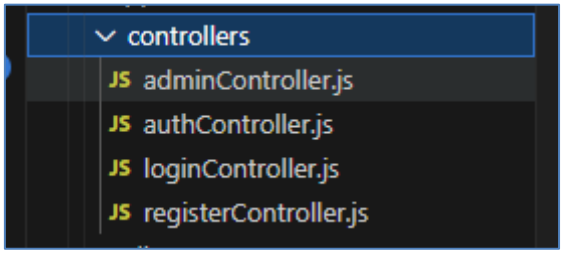
3.1 Conceptualisation



3.2 Explication du code

3.2.1 Contrôleurs

- adminController.js : Gère l'interface et les actions réservées à l'administrateur (listing des utilisateurs, recherche, etc.).
- authController.js : Fournit des fonctions utilitaires pour générer et vérifier les tokens JWT (utile pour l'authentification).
- loginController.js : Gère la connexion de l'utilisateur (vérification des identifiants, génération du token, etc.).
- registerController.js : Gère l'inscription d'un nouvel utilisateur (création du sel, hachage du mot de passe, insertion en base).



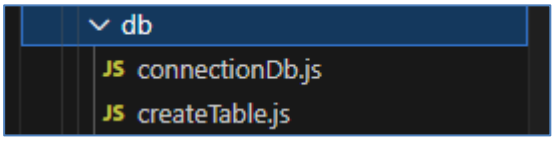
3.2.2 *Database*

connectionDb.js :

- Se connecte à la base de données MySQL "db_webshop" et exporte cette connexion.
- Fichier de création de la base et des utilisateurs

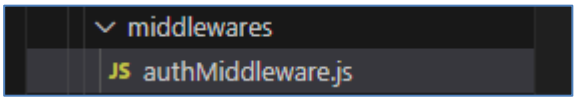
createTables.js

- Crée la base "db_webshop" si elle n'existe pas, puis crée la table "t_users".
- Insère les utilisateurs par défaut ("admin" et "user") si nécessaire.



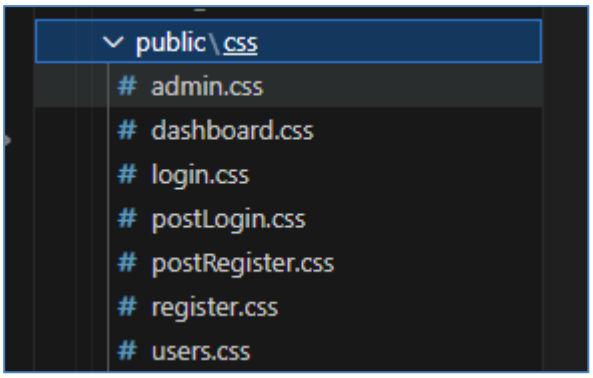
3.2.3 *Middleware*

Ce middleware vérifie la présence d'un token JWT dans les cookies. Il décode le token pour récupérer l'ID utilisateur et interroge la base de données pour obtenir ses informations.
Si l'utilisateur n'existe pas ou le token est invalide, il renvoie une erreur. Pour les routes "/admin", il s'assure que l'utilisateur a le rôle "admin". Sinon, il ajoute l'utilisateur à la requête et appelle next().



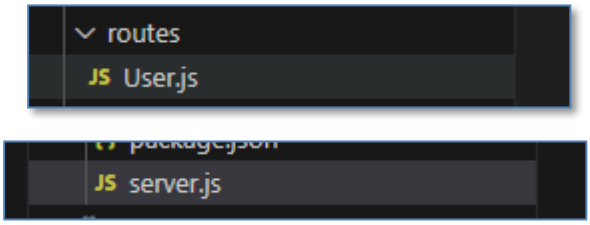
3.2.4 *Views/Css*

Le dossier public/css contient les fichiers de style CSS dédiés à chaque page ou section de l'application. Par exemple, admin.css pour la page d'administration, dashboard.css pour le tableau de bord, etc. Ainsi, chaque fichier gère le design et la mise en page spécifique à la vue correspondante.



3.2.5 *Server et routes*

Les routes définissent l'accès aux pages d'inscription, de connexion, de profil et d'administration.
Un middleware sécurise les zones protégées grâce à un système d'authentification par token.
Les contrôleurs gèrent la logique métier pour les différentes fonctionnalités de l'application.
Un script initialise la base de données et crée les tables nécessaires.
Enfin, le serveur configure les vues et les ressources statiques.

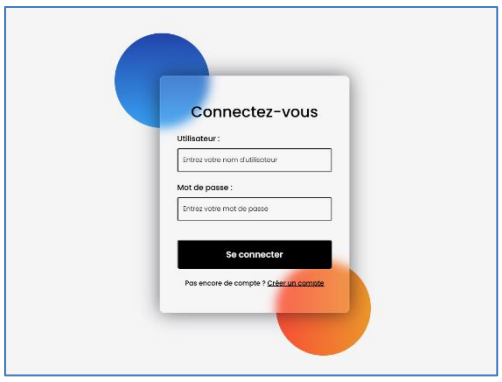


3.3 Travail fourni & attitude sur le projet

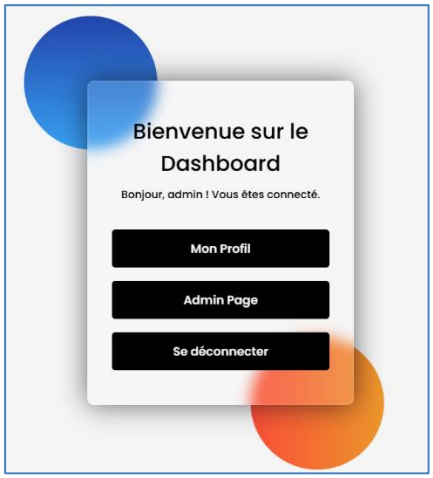
J'ai beaucoup aimé ce projet, surtout car ça m'a aidé à mieux comprendre comment ça fonctionne la protection des données et gestion de routes (URLs) protégées.

Voici des images de mon site web.

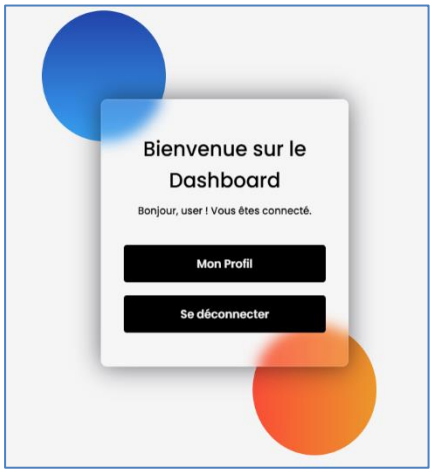
Login et page de registre

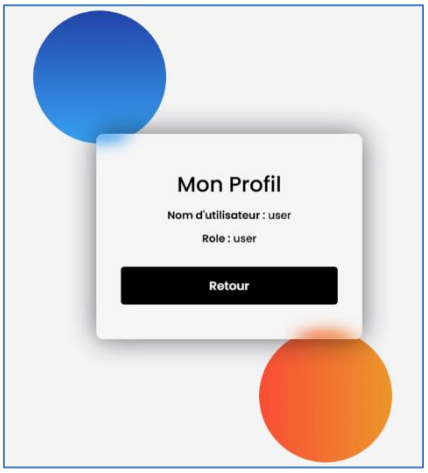


Page admin et panneau d'administration



Page user et profile





4 CONCLUSION

4.1 Bilan personnel

J'ai trouvé ce projet très stimulant, notamment grâce à la liberté offerte pour le design du site. Il m'a permis d'approfondir mes compétences en backend, bien plus que d'autres projets JavaScript. Avec le recul, j'aurais débuté en créant une vue d'ensemble visuelle des différents projets pour éviter une certaine confusion initiale. Cette expérience m'a vraiment permis de comprendre l'importance d'une bonne structuration dès le départ.

4.2 Webographie

4.2.1 Utilisation de l'IA

J'ai sollicité l'aide de ChatGPT pour mieux comprendre certains bugs et difficultés rencontrés pendant le développement, mais l'IA n'a jamais réalisé le projet à ma place.