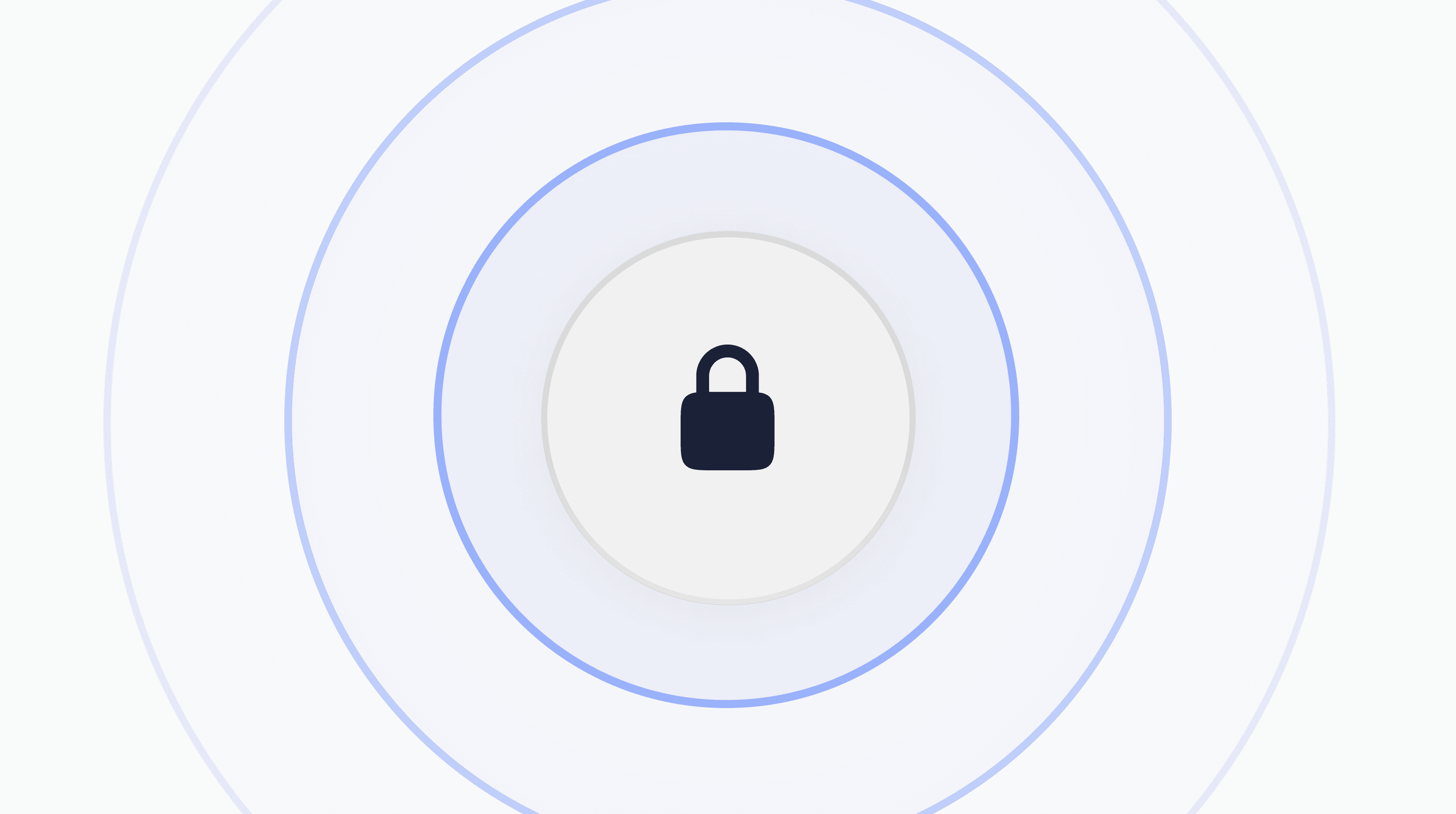
P\_APP\_183 : Secured WebShop



Yosef Nademo – CID2B

Vennes

24 périodes

Cédric SchaffterTable des matières

[1 Spécifications 3](#_Toc192831746)

[1.1 Titre 3](#_Toc192831747)

[1.2 Description 3](#_Toc192831748)

[1.3 Matériel et logiciels à disposition 3](#_Toc192831749)

[1.4 Prérequis 3](#_Toc192831750)

[1.5 Cahier des charges 3](#_Toc192831751)

[1.5.1 Caractéristiques des utilisateurs et impacts 3](#_Toc192831752)

[1.5.2 Fonctionnalités requises (du point de vue de l’utilisateur) 4](#_Toc192831753)

[1.5.3 Travail à réaliser par l'apprenti 4](#_Toc192831754)

[1.5.4 Si le temps le permet … 4](#_Toc192831755)

[1.5.5 Méthodes de validation des solutions 4](#_Toc192831756)

[2 Planification Initiale 5](#_Toc192831757)

[3 Analyse FONCTIONNELLE 5](#_Toc192831758)

[3.1 Conceptualisation 5](#_Toc192831759)

[3.1.1 Schéma de site web 5](#_Toc192831760)

[3.1.2 Schéma des Routes 8](#_Toc192831761)

[3.2 Explication de code 9](#_Toc192831762)

[3.2.1 Registration 9](#_Toc192831763)

[3.2.2 Login 10](#_Toc192831764)

[3.2.3 Details 11](#_Toc192831765)

[3.2.4 Recherche des-utilisateurs 12](#_Toc192831766)

[4 Conclusion 13](#_Toc192831767)

[4.1 Conclusion 13](#_Toc192831768)

[4.2 Bilan personnel 13](#_Toc192831769)

[5 Divers 14](#_Toc192831770)

[5.1 Journal de travail 14](#_Toc192831771)

[6 Annexes 14](#_Toc192831772)

# Spécifications

## Titre

Création d'un site e-commerce sécurisé avec gestion des utilisateurs et administration

## Description

Le projet consiste à créer une plateforme de commerce en ligne (webstore) sécurisée. Ce site offrira une authentification pour les utilisateurs (clients) et les administrateurs. La page d'authentification permettra aux utilisateurs de se connecter en toute sécurité via un mot de passe haché et salé. Un token JWT sera émis après une identification correcte de l'utilisateur, garantissant ainsi la sécurité des sessions. Le projet se concentrera sur l'authentification des utilisateurs, la gestion des rôles et l'administration du site, avec des protections contre les injections SQL et l'utilisation d'un certificat HTTPS pour assurer une connexion sécurisée. Ce projet vise à doter l'apprenant de compétences avancées en matière de sécurité web et de gestion des utilisateurs, tout en respectant les normes et les bonnes pratiques étudiées précédemment dans les modules.

## Matériel et logiciels à disposition

* **Matériel :** Un ordinateur standard de la section informatique avec accès à Docker Desktop pour la gestion des containers.
* **Logiciels :**
* Node.js pour le développement du backend.
* Docker pour la containerisation des services.
* IDE de développement.
* GitHub pour le versioning du code.

## Prérequis

* Connaissances de base en développement web, notamment en **HTML**, **CSS**, **JavaScript** et **Node.js**.
* Expérience avec les bases de données relationnelles (par exemple **MySQL**) et la gestion des utilisateurs.
* Connaissances en **sécurité web**, particulièrement concernant les mécanismes de hachage, de salage de mots de passe, et la gestion des tokens JWT.
* Compétences en **Docker** pour la mise en place d'un environnement de développement et de production containerisé.
* Compétence dans l'utilisation de **Git** et **GitHub** pour la gestion des versions du projet.

## Cahier des charges

### Caractéristiques des utilisateurs et impacts

es utilisateurs du site seront divisés en deux types principaux :

* **Clients** : Ils pourront se connecter via une page d’authentification et accéder à leur profil personnalisé.
* **Administrateurs** : Ils auront des privilèges supplémentaires pour rechercher des utilisateurs et accéder à des données sensibles.

**Impact sur la conception :**

* **Ergonomie** : La page de connexion et la page d'administration seront simplifiées et intuitives. Le design doit respecter les bonnes pratiques d'accessibilité.
* **Sécurité** : L'interface et les routes doivent être sécurisées par des mécanismes tels que l’authentification par mot de passe et le contrôle d'accès avec JWT. Les pages d'administration seront protégées contre les injections SQL.

### Fonctionnalités requises (du point de vue de l’utilisateur)

Le site comportera les fonctionnalités suivantes :

* **Page de connexion** : Permet à l’utilisateur de se connecter en utilisant son nom d’utilisateur et son mot de passe haché et salé.
* **Page d’administration** : Accessible uniquement aux administrateurs, elle permettra de rechercher des utilisateurs en fonction de leur nom et d’afficher les résultats.
* **Page de profil utilisateur(Page de détails )** : Affiche les informations de l'utilisateur connecté, récupérées via une route backend sécurisée.
* **Page d’acueill :** page general de conexion.

### Travail à réaliser par l'apprenti

L'apprenti devra :

* Concevoir l'architecture du site web et la mettre en œuvre en utilisant Node.js et MySQL.
* Implémenter l’authentification par mot de passe sécurisé avec hachage et salage, ainsi que l’émission et la vérification des tokens JWT.
* Développer la page d’administration avec une recherche par nom d'utilisateur.
* Assurer la protection contre les injections SQL.
* Utiliser Docker pour conteneuriser les services web et exécuter le projet de manière sécurisée en HTTPS.
* Rédiger un rapport détaillant la conceptualisation, les choix techniques, et les défis rencontrés.

### Si le temps le permet …

Si le temps permet, l’apprenti pourrait :

* Intégrer des fonctionnalités supplémentaires comme la gestion des rôles plus fine, avec des permissions pour chaque type d'utilisateur (client, administrateur, etc.).
* Ajouter une fonctionnalité de réinitialisation de mot de passe.

### Méthodes de validation des solutions

Les solutions seront validées de plusieurs façons :

* **Tests unitaires** : Les fonctions critiques, telles que l'authentification et la vérification des tokens JWT, seront couvertes par des tests unitaires.
* **Tests d'intégration** : La communication entre le frontend et le backend sera testée pour garantir la sécurité et l'intégrité des données.
* **Vérification de la sécurité** : Des tests seront effectués pour s'assurer que le site est protégé contre les injections SQL et d’autres attaques courantes.

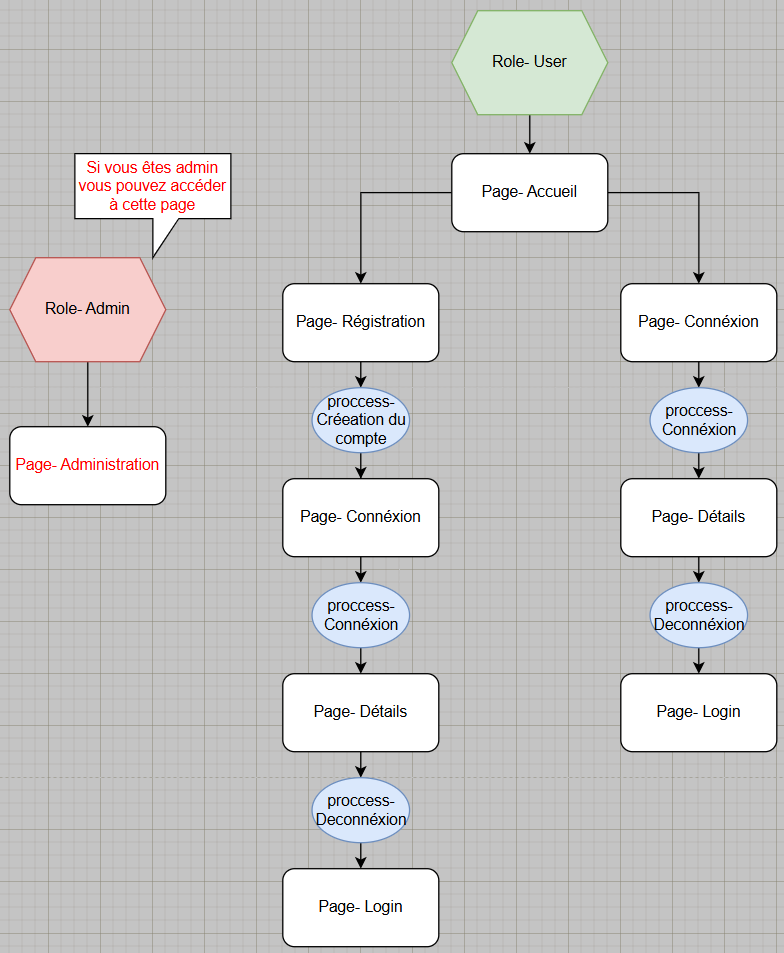
# Planification Initiale

La planification initiale est en forme du fichier Exe est mis au disposition dans dossier documentation, et est accessible par ce liens : [m-planification-jnltrav-Yosef-Nademo (1).xlsm](https://eduvaud-my.sharepoint.com/:x:/g/personal/pn25kdv_eduvaud_ch/ERkoOiUiEi5NozX_9pB_XyABW8Ymn-to2sQ5C27Pbr7dzA?email=yosef.nademo%40eduvaud.ch&e=kNUScd)

# Analyse FONCTIONNELLE

## Conceptualisation

### Schéma de site web



* **Rôle : Administrateur**

1. **Page - Administration :**
   * + **Description :** Cette page est accessible uniquement aux utilisateurs ayant le rôle d'administrateur. Il fournit des fonctionnalités administratives telles que la visualisation d'informations détaillées des utilisateurs.
     + **Accès :** les utilisateurs administrateurs peuvent accéder à cette page directement depuis la navigation principale ou via une route /admin.

* **Rôle : Utilisateur**
  1. **Page - Accueil (Accueil) :**
* **Description :** Ceci est la page d'accueil pour les utilisateurs réguliers. Il sert de page de destination où les utilisateurs peuvent naviguer vers d’autres parties du site.
* **Accès :** Tous les utilisateurs peuvent accéder à cette page.
  1. **Page - Inscription :**
* **Description :** Cette page permet aux nouveaux utilisateurs d'enregistrer un compte sur le site.
* **Processus - Création du compte (Account Creation) :**
  + **Description :** Les utilisateurs remplissent un formulaire d'inscription avec leurs coordonnées (nom d'utilisateur, mot de passe, etc.) et le soumettent pour créer un nouveau compte.
  + **Accès :** Accessible depuis la page d'accueil ou d'autres liens de navigation.
  1. **Page - Connexion (Login) :**
* **Description :** Cette page permet aux utilisateurs de se connecter à leurs comptes.
* **Processus - Connexion (Processus de connexion) :**
  + **Description :** Les utilisateurs saisissent leurs informations d'identification (nom d'utilisateur et mot de passe) et soumettent le formulaire pour se connecter.
* **Accès :** Accessible depuis la page d'accueil ou après inscription.
  1. **Page - Détails (Détails) :**
* **Description :** Cette page affiche des informations détaillées sur l'utilisateur, telles que ses informations de profil et les paramètres de son compte.
* **Processus - Connexion (Processus de connexion) :**
* **Description :** Après s'être connecté, les utilisateurs sont redirigés vers cette page pour afficher leurs détails.
* **Accès :** Accessible uniquement après une connexion réussie.
  1. **Processus - Déconnexion (Processus de déconnexion) :**
* **Description :** Ce processus permet aux utilisateurs de se déconnecter de leurs comptes.
* **Accès :** Accessible depuis la page de détails ou d'autres liens de navigation.

**Flux détaillé pour chaque rôle**

**Flux d'administration**

1. **Accéder à la page d'accueil :**

* Les utilisateurs administrateurs commencent à la page d'accueil.

1. **Accéder à la page d'administration :**

* Les utilisateurs administrateurs peuvent accéder à la page d'administration où ils peuvent gérer les utilisateurs et effectuer d'autres tâches administratives.

**Flux d'utilisateurs**

1. **Accéder à la page d'accueil :**

* Les utilisateurs réguliers commencent par la page d’accueil.

1. **Accéder à la page d'inscription :**

* Les nouveaux utilisateurs peuvent accéder à la page d'inscription pour créer un compte.

1. **Processus de création de compte :**

* Les utilisateurs remplissent le formulaire d'inscription et le soumettent pour créer un nouveau compte.

1. **Accéder à la page de connexion :**

* Après l'inscription, les utilisateurs sont dirigés vers la page de connexion.

1. **Processus de connexion :**

* Les utilisateurs saisissent leurs informations d’identification et se connectent.

1. **Accéder à la page Détails :**

* Une fois connectés, les utilisateurs sont redirigés vers la page de détails pour afficher les informations de leur profil.

1. **Processus de déconnexion :**

* Les utilisateurs peuvent se déconnecter à partir de la page de détails ou d’autres liens de navigation qui les redirigent vers la page de connexion.

**Résumé**

Le site est conçu pour fournir un flux clair et structuré tant pour les administrateurs que pour les utilisateurs réguliers. Les utilisateurs administrateurs ont accès à la page d'administration, tandis que les utilisateurs réguliers peuvent s'inscrire, se connecter, afficher leurs détails et se déconnecter. Chaque page et chaque processus sont conçus pour garantir une expérience utilisateur fluide et une gestion sécurisée des données utilisateur.

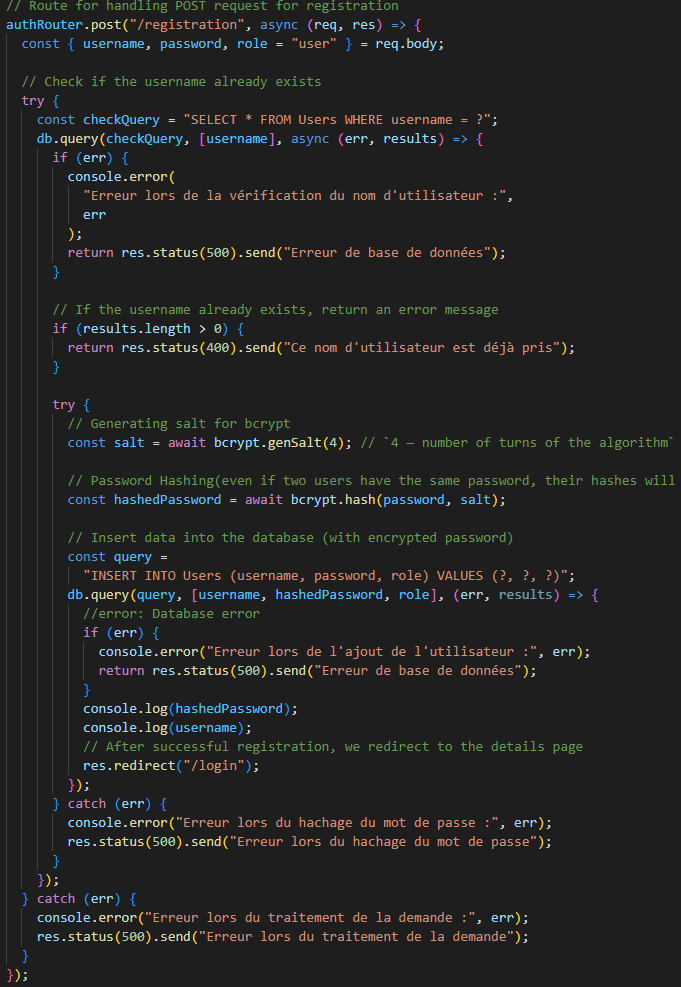
### Schéma des Routes

Cette section présente le tableau des itinéraires du site. Il contient des informations sur les méthodes HTTP (GET, POST, etc.), les URI (par exemple /home, /login), les données JSON transmises dans les requêtes et une description de chaque itinéraire. Ce tableau permet de structurer clairement l'interaction du client avec le serveur et simplifie le processus de développement et de support d'une application Web. Des informations détaillées sont fournies dans l'image ci-dessous.



## Explication de code

Dans cette section, je fournirai des extraits de code contenant les fonctions de base de ce projet informatique.



### Registration

Ce code définit une route **POST */registration*** qui permet aux utilisateurs de s'inscrire sur le site Voici explication:

1. **Récupération des données de la requête :** Le code extrait le nom d'utilisateur, le mot de passe et le rôle du corps de la requête.
2. **Vérification de l'existence du nom d'utilisateur :** Une requête SQL est exécutée pour vérifier si le nom d'utilisateur existe déjà dans la base de données. Si c'est le cas, une erreur est renvoyée.
3. **Hachage du mot de passe :** Si le nom d'utilisateur n'existe pas, un sel est généré pour le hachage du mot de passe à l'aide de **bcrypt.** Le mot de passe est ensuite haché.
4. **Insertion des données dans la base de données :** Les données de l'utilisateur (nom d'utilisateur, mot de passe haché, rôle) sont insérées dans la base de données.
5. **Redirection après inscription :** Après une inscription réussie, l'utilisateur est redirigé vers la page de connexion (*/login*).



### Login

Ce code définit une route **POST */login*** qui permet aux utilisateurs de se connecter à leur compte. Voici explication :

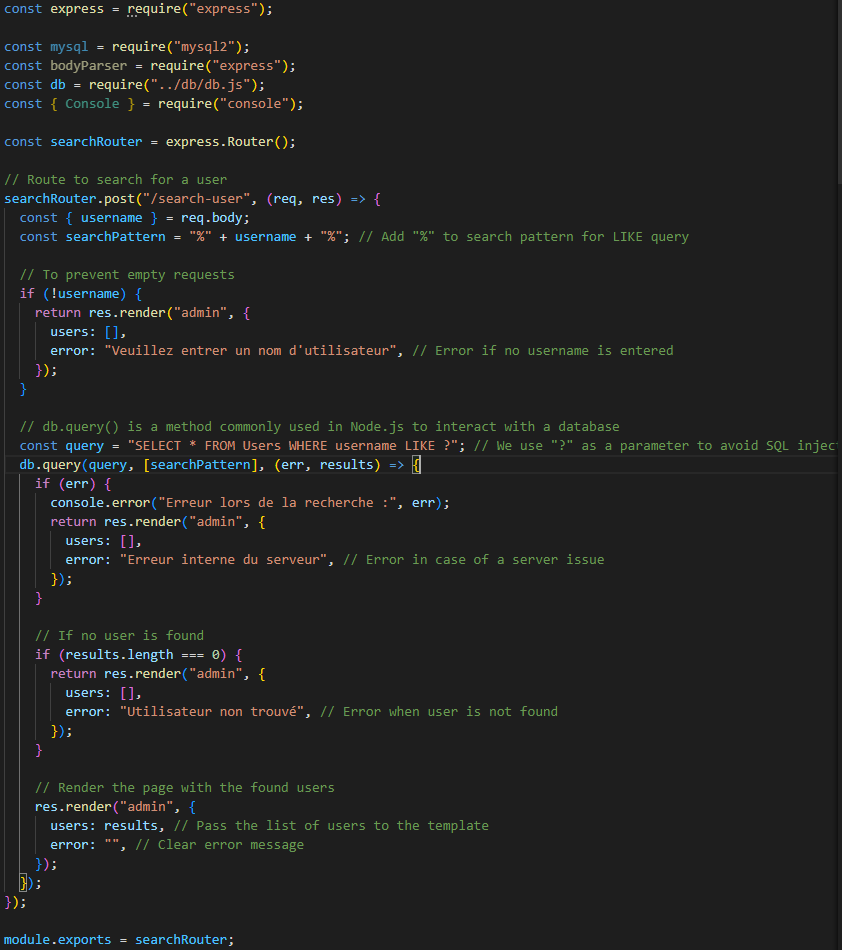
1. **Récupération des données de la requête :** Le code extrait le nom d'utilisateur et le mot de passe du corps de la requête.
2. **Vérification de l'existence de l'utilisateur :** Une requête SQL est exécutée pour vérifier si l'utilisateur existe dans la base de données. Si l'utilisateur n'existe pas, une erreur est renvoyée.
3. **Vérification du mot de passe :** Si l'utilisateur existe, le mot de passe fourni est comparé au mot de passe haché stocké dans la base de données à l'aide de bcrypt. Si les mots de passe ne correspondent pas, une erreur est renvoyée.
4. **Génération du token JWT :** Si les mots de passe correspondent, un token JWT est généré et renvoyé à l'utilisateur. Ce token est utilisé pour authentifier les requêtes futures.
5. **Redirection après connexion :** Après une connexion réussie, l'utilisateur est redirigé vers la page des détails (*/details*).



### Details

Ce code définit une route **GET */details*** qui permet d'afficher les détails de l'utilisateur connecté. Voici explication :

1. **Récupération du token JWT :** Le code extrait le token JWT de l'en-tête de la requête.
2. **Vérification de la présence du token** : Si le token n'est pas présent, l'utilisateur est redirigé vers la page de connexion *(/login).*
3. **Vérification du token JWT :** Le token est vérifié à l'aide de la clé secrète JWT. Si le token est invalide ou expiré, une erreur est renvoyée. Récupération des détails de l'utilisateur : Si le token est valide, une requête SQL est exécutée pour récupérer les détails de l'utilisateur à partir de la base de données en utilisant l'ID de l'utilisateur extrait du token.
4. **Affichage des détails de l'utilisateur :** Si l'utilisateur est trouvé, les détails de l'utilisateur sont rendus sur la page details. Si l'utilisateur n'est pas trouvé, une erreur est renvoyée.



### Recherche des-utilisateurs

Ce code définit une route **POST */search-user*** qui permet de rechercher des utilisateurs dans la base de données en fonction de leur nom d'utilisateur. Voici explication :

1. **Importation des modules nécessaires :** Le code commence par importer les modules express, mysql2, bodyParser, et db pour gérer les requêtes HTTP, interagir avec la base de données MySQL, et parser les données des requêtes.
2. **Création du routeur :** Un routeur Express est créé pour gérer les routes de recherche d'utilisateur.
3. **Définition de la route POST */search-user* :** Cette route prend le nom d'utilisateur envoyé dans le corps de la requête, construit un motif de recherche en ajoutant des caractères génériques (%) pour une requête SQL LIKE, et vérifie si le nom d'utilisateur est vide.
4. **Exécution de la requête SQL :** Si le nom d'utilisateur n'est pas vide, une requête SQL est exécutée pour rechercher les utilisateurs dont le nom correspond au motif de recherche. Les résultats sont ensuite rendus sur la page admin avec la liste des utilisateurs trouvés ou un message d'erreur approprié.

# Conclusion

## Conclusion

Le projet **P\_APP\_183 : Secured WebShop** a permis de concevoir et de développer une plateforme sécurisée intégrant des fonctionnalités essentielles comme l’authentification des utilisateurs, la gestion des rôles et l’administration du site. En mettant l’accent sur la sécurité, j’ai implémenté des techniques avancées telles que le hachage et le salage des mots de passe, l’émission de tokens JWT et la protection contre les injections SQL.

Ce projet m’a permis d’approfondir mes compétences en développement backend avec **Node.js**, en gestion de base de données avec **MySQL et PhPmyAdmin**, ainsi qu’en déploiement sécurisé grâce à **Docker** et **HTTPS**. La mise en place des tests unitaires et d’intégration a assuré la fiabilité du système, et la structuration du code a facilité son évolutivité et sa maintenance.

Malgré les défis rencontrés, notamment correction des erreurs de (Docker et Node), et l’intégration des mesures de sécurité, j’ai réussi à obtenir un résultat fonctionnel et conforme aux exigences initiales. Ce projet représente une expérience enrichissante qui me servira pour de futurs développements web sécurisés.

## Bilan personnel

**Qu’est que ce projet m’a appris ?**

* **Compétences techniques :**
  + L'implémentation de l'authentification sécurisée avec JWT et bcrypt.
  + La gestion des bases de données relationnelles avec MySQL et l'utilisation de requêtes SQL sécurisées.
  + La containerisation des applications avec Docker, facilitant le déploiement et la gestion des environnements de développement et de production.
* **Compétences en sécurité web :**
  + La protection contre les injections SQL et autres attaques courantes.
  + L'utilisation de certificats HTTPS pour sécuriser les communications entre le client et le serveur.
* **Compétences en gestion de projet :**
  + La planification et l'organisation du travail pour respecter les délais et les exigences du projet.
  + La rédaction de documentation technique et de rapports détaillés.

**Remerciements :**

* Je tiens à remercier mon tuteur, Cédric Schaffter, pour son soutien et ses conseils tout au long de ce projet.
* Merci également à mes collègues et amis pour leurs retours et leur aide précieuse.

# Divers

## Journal de travail

Le journal de travail vous pouvez trouve dans ficher «GitHub\Secured-WebShop\Documentation\m-planification-jnltrav-Yosef-Nademo.xlsm »

Ou il est accessible par ce liens : [m-planification-jnltrav-Yosef-Nademo (1).xlsm](https://eduvaud-my.sharepoint.com/:x:/g/personal/pn25kdv_eduvaud_ch/ERkoOiUiEi5NozX_9pB_XyABW8Ymn-to2sQ5C27Pbr7dzA?email=yosef.nademo%40eduvaud.ch&e=kNUScd)

# Annexes

* <https://www.youtube.com/watch?v=AzA_LTDoFqY&t=30s&ab_channel=SamMeech-Ward>
* <https://app.diagrams.net/>
* <https://stackoverflow.com/questions/63007066/error-err-module-not-found-cannot-find-package-src>
* <https://stackoverflow.com/questions/15809611/bcrypt-invalid-elf-header-when-running-node-app>
* <https://stackoverflow.com/questions/46398592/node-js-error-return-process-dlopenmodule-path-makelongfilename>
* <https://stackoverflow.com/questions/46384591/node-was-compiled-against-a-different-node-js-version-using-node-module-versio>