TP 03
M° 206 Création d'une application cloud native
Année 2023-2024

Filière : DDOWFS
Niveau : Technicien spécialisé

Durée : 01 H 30

API d'authentification sécurisée avec Node.js, Express.js, Mongoose et JWT

Objectif: Développer une API RESTful pour gérer l'authentification des utilisateurs avec des mots de passe sécurisés en utilisant Node.js, Express.js, Mongoose et JSON Web Tokens (JWT).

Tâches:

1. **Créer un modèle Mongoose pour les utilisateurs:**
   - Définir un schéma Mongoose pour représenter un utilisateur avec les champs nom, email, motDePasse (haché) et role (administrateur ou utilisateur standard).
   - Exporter le modèle pour l'utiliser dans les routes Express.
2. **Implémenter l'inscription des utilisateurs:**
   - **Route POST /inscription:**
     - Recevoir les données d'inscription d'un utilisateur (nom, email, mot de passe) dans le corps de la requête.
     - Valider les données d'entrée (format d'email, longueur du mot de passe, etc.).
     - Hacher le mot de passe de l'utilisateur en utilisant bcrypt.
     - Créer un nouvel utilisateur dans la base de données MongoDB en utilisant le modèle Mongoose.
     - Envoyer une réponse appropriée indiquant le succès ou l'échec de l'inscription.
3. **Implémenter la connexion des utilisateurs:**
   - **Route POST /connexion:**
     - Recevoir les informations de connexion d'un utilisateur (email, mot de passe) dans le corps de la requête.
     - Rechercher un utilisateur dans la base de données par son adresse e-mail.
     - Si l'utilisateur est trouvé, comparer le mot de passe fourni avec le mot de passe haché stocké dans la base de données en utilisant bcrypt.
     - Si les mots de passe correspondent, générer un jeton JWT contenant l'identifiant de l'utilisateur et le rôle.
     - Envoyer le jeton JWT et les informations utilisateur pertinentes (nom, email, rôle) dans la réponse.

        - Si les mots de passe ne correspondent pas ou si l'utilisateur n'est pas trouvé, envoyer une réponse d'erreur appropriée.

4. **Protéger les routes avec JWT:**
    - Utiliser un middleware Express pour vérifier la présence et la validité du jeton JWT dans les requêtes entrantes.
    - Si le jeton est valide, extraire l'identifiant de l'utilisateur et le rôle du jeton et les attacher à l'objet de requête `req`.
    - Autoriser l'accès aux routes protégées en fonction du rôle de l'utilisateur (administrateur ou utilisateur standard).
    - Gérer les requêtes non authentifiées ou non autorisées en envoyant des réponses d'erreur appropriées.

Points clés:

- Hacher les mots de passe des utilisateurs avec bcrypt pour les stocker de manière sécurisée dans la base de données.
- Générer et utiliser des jetons JWT pour authentifier les utilisateurs et protéger les routes.
- Vérifier la validité des jetons JWT et extraire les informations utilisateur pertinentes.
- Autoriser l'accès aux routes en fonction du rôle de l'utilisateur.