

Énoncé d'exercice : Création d'un CRUD pour un User

 Propriétaire	 Marine
 Étiquettes	

Objectif

Réalisez une application Node.js avec Express et Mongoose permettant de créer, lire, mettre à jour et supprimer des utilisateurs (CRUD). L'application doit respecter une structure simple, avec des fichiers séparés pour les fonctionnalités principales.

Contraintes

- **Technologies utilisées :**
 - Express
 - Mongoose
 - dotenv pour gérer les variables d'environnement
- **Structure minimale :**
 - `app.js` : Point d'entrée de l'application
 - `routes/userRouter.js` : Gestion des routes pour le CRUD
 - `models/User.js` : Modèle Mongoose pour les utilisateurs
 - `config/db.js` : Fichier de connexion à MongoDB
 - `.env` : Fichier pour stocker les variables d'environnement (comme la connexion à MongoDB)

Fonctionnalités à implémenter

1. Création d'un utilisateur (POST /users)

Un utilisateur doit contenir les champs suivants :

- `name` (string, obligatoire)
- `email` (string, unique, obligatoire)
- `password` (string, obligatoire)
- `age` (number, optionnel)

2. Lecture des utilisateurs

- **(GET /users)** : Récupère tous les utilisateurs.
- **(GET /users/:id)** : Récupère un utilisateur spécifique grâce à son ID.

3. Mise à jour d'un utilisateur (PUT /users/:id)

Permet de modifier les informations d'un utilisateur.

4. Suppression d'un utilisateur (DELETE /users/:id)

Supprime un utilisateur par son ID.

Instructions

1. Créer le fichier `app.js` :

- Configurez Express avec le middleware nécessaire (ex. : `express.json()`).
- Chargez les variables d'environnement avec `dotenv` .
- Connectez MongoDB via un fichier `config/db.js` .
- Montez le routeur pour les utilisateurs.

2. Créer le fichier `models/User.js` :

- Définissez un schéma Mongoose pour le modèle `User` avec les champs spécifiés.

3. Créer le fichier `routes/userRouter.js` :

- Implémentez les routes du CRUD directement dans ce fichier en utilisant les méthodes du modèle Mongoose.

4. Créer le fichier `config/db.js` :

- Configurez la connexion à MongoDB avec Mongoose. Utilisez une variable d'environnement pour la chaîne de connexion.

5. Configurer un fichier `.env` :

- Ajoutez une variable `MONGO_URI` contenant l'URL de connexion à MongoDB.

Bonus

1. Séparation en controllers, services et repositories :

Refactorisez votre code pour déplacer la logique métier des routes dans des **controllers**.

- Les **controllers** contiendront la logique liée aux requêtes (ex. : `req`, `res`).
- Les **services** contiendront les règles métier.
- Les **repositories** contiendront les interactions avec la base de données.

2. Gestion des erreurs avec des exceptions personnalisées :

- Implémentez une gestion centralisée des erreurs en utilisant un middleware dédié.
- Créez une classe personnalisée pour les erreurs (ex. : `AppError`) afin de gérer différents types d'exceptions (ex. : 404, 500).
- Gérez les erreurs MongoDB (comme les erreurs de validation ou d'unicité).

3. Suppression "douce" (Soft Delete)

- Implémentez une suppression logique des utilisateurs au lieu d'une suppression physique :
 - Ajoutez un champ `isDeleted` au modèle.
 - Filtrez les utilisateurs pour exclure ceux qui ont `isDeleted: true`.