Brief 3 : Gestionnaire de Tâches Simple

Objectif :

Créer une application backend de gestion des tâches en Node.js avec Express.js et MongoDB. L'application permettra aux utilisateurs de gérer leurs tâches en mettant en œuvre les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur une base de données MongoDB.

Instructions :

1. Initialisation du Projet

1. Configurer un projet Node.js avec `npm init`.
2. Installer les dépendances nécessaires (`express`, `mongoose`, `body-parser`).
3. Créer un fichier principal `index.js`.

2. Fonctionnalités :

1. **Ajouter une tâche** :
   Les utilisateurs peuvent entrer un titre et une description pour créer une tâche.
2. **Afficher toutes les tâches** :
   Listez les tâches avec leurs propriétés (titre, description, statut "complété").
3. **Mettre à jour une tâche** :
   Permet de modifier le statut d'une tâche pour indiquer si elle est complétée ou non.
4. **Supprimer une tâche** :
   Supprimez une tâche en utilisant son identifiant unique.
5. **Persistance des données** :
   Les données doivent être stockées dans une base MongoDB, locale ou sur MongoDB Atlas.

---

3. Gestion des Erreurs :

1. Validez les entrées utilisateur :
   - Les champs requis (`title`, `description`) ne doivent pas être vides.
   - Gérez les erreurs de connexion à MongoDB.
2. Gérez les cas où une tâche demandée pour la mise à jour ou la suppression n'existe pas.

---

4. Organisation du Code :

1. Créer un modèle Mongoose pour les tâches (`Task`) dans un dossier `models/`.
2. Créer des routes spécifiques pour chaque opération CRUD dans un dossier `routes/`.
3. Implémentez les principes SOLID :
    - **Single Responsibility** : Les modèles, les routes et la configuration sont dans des fichiers distincts.
    - **Open/Closed** : Facilitez l'ajout de nouvelles fonctionnalités (ex. tri des tâches par statut).

Structure du Programme :

Routes principales :

1. **POST** `/api/tasks` : Ajouter une tâche.
2. **GET** `/api/tasks` : Afficher toutes les tâches.
3. **PUT** `/api/tasks/:id` : Mettre à jour le statut d'une tâche.
4. **DELETE** `/api/tasks/:id` : Supprimer une tâche.

Méthodes spécifiques :

- `createTask(title, description)` : Pour ajouter une tâche.
- `getAllTasks()` : Pour afficher toutes les tâches.
- `updateTask(id, completed)` : Pour mettre à jour une tâche existante.
- `deleteTask(id)` : Pour supprimer une tâche.

Livrables :

1. **Code Source :**
    - Hébergé sur un dépôt GitHub, avec des commits clairs pour chaque fonctionnalité.
2. **Documentation :**
    - Explication des fonctionnalités principales.
    - Instructions pour exécuter l'application.
3. **Gestion de Projet avec Trello :**
    - Créez un tableau Trello pour organiser les tâches avec les colonnes suivantes :
        - **À faire** : Liste des tâches à accomplir.
        - **En cours** : Tâches en cours de réalisation.
        - **En revue** : Tâches terminées mais en attente de validation.
        - **Terminé** : Tâches finalisées.