Examen Pratique : Création d'une API REST en Node.js pour la Gestion des Œuvres d'Art

Contexte

Vous allez créer une API REST pour gérer des œuvres d'art. La validation des données sera réalisée à l'aide des objets Proxy en JavaScript. L'API doit permettre de créer, lire, mettre à jour, et supprimer (CRUD) les œuvres d'art. Prisma sera utilisé pour gérer la base de données.

Exigences

- **Technologies** : Utilisez Node.js avec Express.js, Prisma pour la base de données et PostgreSql comme moteur de base de données. La validation sera effectuée via un Proxy en JavaScript.
- **Durée** : 3 heures

1 Configuration du Projet

1. Initialisez un projet Node.js et installez les dépendances suivantes :
   - `express` : Pour créer le serveur.
   - `body-parser` : Pour gérer le parsing des requêtes JSON.
   - `prisma` et `@prisma/client` : Pour interagir avec la base de données.
   - `nodemon` : Pour le rechargement automatique en développement.
   - `dotenv` : Pour manipuler le **.env**.
2. Configurez Prisma :
   - Initialisez Prisma avec `npx prisma init` pour générer les fichiers de configuration.
   - Dans `schema.prisma`, définissez le modèle `Artwork` avec les champs suivants :
     - `id` : entier, auto-incrémenté, clé primaire
     - `title` : chaîne de caractères, requis
     - `artist` : chaîne de caractères, requis
     - `year` : entier, optionnel
     - `description` : chaîne de caractères, optionnel
     - `type` : chaîne de caractères, requis, doit correspondre à une liste de valeurs autorisées (ex. : "peinture", "sculpture", "dessin", "acii art")
3. Exécutez la migration `npx prisma migrate dev --name init`

4. Dans le fichier **seed/seed.js** créer un méthode **seedRun** qui permet d'inserer dans votre base de donnéé postgres les objects je trouvant dans le fichier **seed/db_data_seed.json**

5. Dans votre package.json créer un script db:seed qui exécute le fichier **seed/seed.js**

2 Controller pour les Œuvres d'Art

1. Dans votre package.json créer un script dev qui démarre le serveur express se trouvant dans **src/index.js**

2. Implementer toute les méthodes du services **src/ArtWorkService.js**.

3. Créez une function `artworkDataValidator` dans le fichier utils. Cette fonction retourne un objet Proxy (avec la trap **set**) pour valider les données. Mettez en place les règles de validation suivantes :

   - `title`, `artist`, et `type` doivent être des chaînes non vides.
   - `type` doit être parmi les valeurs autorisées.
   - `year`, s'il est défini, doit être un entier positif.

4. Implement toute les méthodes du controller **src/ArtWorkController.js**.
   `NB` :

   - N'oublier pas d'utiliser la fonction **artworkDataValidator** pour valider les données dans (create & update).
   - Aussi envoyer les bon status de reponse.

3 Route pour les Œuvres d'Art

1. Tester via votre client http préférer l'endpoit /test.

2. Implémentez les endpoints suivants dans `src/router.js` et connectez-les au contrôleur :

   - **POST /artworks** : Crée une nouvelle œuvre d'art en validant les données via le Proxy.
   - **GET /artworks** : Récupère toutes les œuvres d'art.
   - **PUT /artworks/:id** : Met à jour une œuvre d'art en utilisant le Proxy pour la validation.
   - **DELETE /artworks/:id** : Supprime une œuvre d'art par son ID. Si ce dernier existe.

4 Recherche et Filtrage

1. **Filtrage par Type,Année et par artist** : Créez un endpoint **GET /artworks/filter** qui prend en paramètres query (ou GET) `type`, `year` et **artist** pour retourner uniquement les œuvres correspondant aux critères.