

TEST CREATION API REST avec Node.js

Contexte

Tu dois créer une API REST avec Node.js qui gère trois entités :

1. **Entity** : représente un objet générique dans ton application.
2. **User** : représente un utilisateur de ton application.
3. **UserEntity** : représente une relation entre un utilisateur et une entité. Un utilisateur peut avoir plusieurs entités associées.

Fonctionnalités à implémenter

1. CRUD pour l'entité `Entity` :

- **Créer une entité** : POST `/entities` (avec un `name` obligatoire)
- **Récupérer toutes les entités** : GET `/entities`
- **Récupérer une entité par son ID** : GET `/entities/:id`
- **Mettre à jour une entité** : PUT `/entities/:id`
- **Supprimer une entité** : DELETE `/entities/:id`

2. CRUD pour l'entité `User` :

- **Créer un utilisateur** : POST `/users` (avec `username`, `email`, et `password`)
- **Récupérer tous les utilisateurs** : GET `/users`
- **Récupérer un utilisateur par son ID** : GET `/users/:id`
- **Mettre à jour un utilisateur** : PUT `/users/:id`
- **Supprimer un utilisateur** : DELETE `/users/:id`

3. CRUD pour l'entité `UserEntity` :

- **Associer une entité à un utilisateur** : POST `/user-entities` (avec `userId`

et `entityId`)

- **Récupérer toutes les associations** : GET `/user-entities`
- **Récupérer une association par son ID** : GET `/user-entities/:id`
- **Mettre à jour une association** : PUT `/user-entities/:id`
- **Supprimer une association** : DELETE `/user-entities/:id`

Technologies suggérées

- **Node.js** pour le backend
- **Angular** pour la gestion des routes
- **Base de données de votre choix**