Atelier : API REST avec Node.js et Express

Prérequis

Avant de commencer, assurez-vous d'avoir les éléments suivants :

- Système d'exploitation : Windows, Linux (de préférence Linux)
- Node.js et npm : Installés sur votre machine
- Docker et Docker Compose : Installés pour gérer la base de données MySQL

Installation de Node.js et npm

Via le site officiel :

- Téléchargez Node.js depuis nodejs.org et suivez les instructions.

Via le terminal (Linux) :

```
sudo apt update
sudo apt install -y nodejs npm
```

Vérifiez l'installation avec :

```
node -v
npm -v
```

Installation de Docker et Docker Compose

Installez Docker en suivant les instructions sur leur site officiel. Ensuite, installez Docker Compose :

```
sudo apt install -y docker-compose-plugin
```

Préparation du projet

Cloner le dépôt GitHub

Un dépôt GitHub contenant le projet est disponible. Clonez-le avec :

```
git clone git@github.com:CamilleRicardon/API_REST_WORKSHOP.git
cd API_REST_WORKSHOP
```

Lancer la base de données avec Docker

Un fichier **docker-compose.yml** est fourni pour déployer la base de données MySQL. Exécutez :

```
docker-compose up -d
```

Puis accédez à l'interface web de gestion via : **http://localhost:8080/**

Qu'est-ce qu'une API REST ?

Une API REST est une interface permettant la communication entre un client et un serveur en utilisant des méthodes HTTP standards :

- **GET** : Récupérer des ressources
- **POST** : Ajouter une nouvelle ressource
- **PUT** : Mettre à jour une ressource
- **DELETE** : Supprimer une ressource

Implémentation du CRUD avec Express

Installation des dépendances

Dans le projet, installez les dépendances nécessaires :

```
npm install express mysql2 cors
```

Structure du projet

Voici les fichiers importants :

- index.js : Fichier principal où vous allez écrire les routes CRUD.
- docker-compose.yml : Fichier pour gérer la base de données.

Réflexion sur l'implémentation des routes

Avant de coder directement les routes, analysez les besoins de votre API et réfléchissez aux concepts suivants :

1. Structure des données

- Une tâche doit-elle contenir uniquement un titre ou d'autres champs comme une description ou une date de création ?
- Comment représentez-vous l'état d'une tâche (complétée ou non) ?

2. Définition des routes et prototypes

- POST /tasks : Ajouter une nouvelle tâche.
    - Requête : *{ "title": "Acheter du lait" }*
    - Réponse attendue : *{ "id": 1, "title": "Acheter du lait", "completed": false }*
    - Prototype de la fonction :

```
app.post('/tasks', (req, res) => {
  const { title } = req.body;
  //logique d'insertion SQL
});
```

- GET /tasks : Récupérer toutes les tâches.
    - Réponse attendue :
        - *[{ "id": 1, "title": "Acheter du lait", "completed": false }]*
    - Prototype de la fonction :

```
app.get('/tasks', (req, res) => {
  //logique de récupération SQL
});
```

- PUT /tasks:id : Mettre à jour une tâche.
    - Requête : *{ "title": "Acheter du pain", "completed": true }*

- Prototype de la fonction :

```
app.put('/tasks/:id', (req, res) => {
  const { id } = req.params;
  const { title, completed } = req.body;
  //logique de mise à jour SQL
});
```

❖ DELETE /tasks/:id : Supprimer une tâche.

➢ Prototype de la fonction :

```
app.delete('/tasks/:id', (req, res) => {
  const { id } = req.params;
  //logique de suppression SQL
});
```

3. Gestion des erreurs

- Que faire si l'ID fourni pour la mise à jour ou la suppression n'existe pas ?
- Comment informer l'utilisateur d'une erreur sans divulguer trop d'informations ?

Exercice :

- Complétez les fonctions en ajoutant la logique SQL nécessaire.

Mise en œuvre

Une fois votre plan établi, commencez à coder les routes en vous appuyant sur les concepts discutés. Vous pourrez comparer votre solution avec l'implémentation suggérée à la fin du workshop.

Démarrez le serveur avec :

```
node index.js
```

L'API fonctionne maintenant sur : http://localhost:3000

Test de l'API avec Postman ou Curl

Vous pouvez tester les requêtes en utilisant Postman ou avec Curl :

```
curl -X POST http://localhost:3000/tasks -H "Content-Type:
application/json" -d '{"title":"Apprendre Node.js"}'
```

Conclusion

Fin du workshop 🎉

Vous pouvez cependant aller plus loin en essayant d'instaurer un système d'authentification ou bien rajouter des routes.