COMPTE RENDU TP2 DEVELOPPEMENT WEB

KITSOUKOU & DJEUNGA

3D_TD2_TP4

Conclusion

Dans ce projet, nous avons franchi une étape supplémentaire dans le développement d'API en intégrant une base de données SQLite3 pour gérer la persistance des données des utilisateurs. Cela a permis d'approfondir nos compétences dans la gestion des données stockées et d'enrichir les fonctionnalités de l'API en ajoutant une dimension de stockage durable, essentielle dans un environnement de production.

Principales étapes et fonctionnalités :

1. Configuration de SQLite3 : Nous avons initialisé une base de données SQLite, idéalement adaptée aux projets légers ou à l'étape de prototypage d'une application. La base de données a été configurée pour stocker les informations des utilisateurs, avec des champs comme `id`, `firstName`, et `lastName`, et permettre un accès rapide et performant aux données.

2. Création des routes CRUD avec SQLite3 : En utilisant SQLite3, chaque route de l'API REST a été reliée aux données stockées :

- POST (Créer) : Nous avons créé la route pour ajouter de nouveaux utilisateurs à la base de données, avec des vérifications pour garantir l'intégrité des données (validation des champs requis).

- GET (Lire) : Les routes GET permettent d'obtenir soit la liste complète des utilisateurs, soit un utilisateur spécifique en fonction de son `id`. Cette structure assure un accès rapide aux informations stockées.

- PUT (Mettre à jour) : La route PUT permet de modifier les informations des utilisateurs existants en mettant à jour les enregistrements dans la base de données. Cette étape a introduit l'importance de vérifier l'existence des utilisateurs avant de procéder à toute mise à jour.

- DELETE (Supprimer) : Enfin, la route DELETE supprime un utilisateur en fonction de son `id`, démontrant la gestion des suppressions dans une base de données relationnelle.

3. Gestion des erreurs et retours d'informations : Chaque route inclut des vérifications d'erreurs pour garantir une communication claire avec le client de l'API, informant si l'utilisateur n'est pas trouvé, si une requête est incorrecte, ou si une opération réussit. Ces messages sont essentiels pour une expérience utilisateur fluide et pour faciliter le débogage.

4. Interaction avec SQLite3 : En manipulant SQLite3 à travers Node.js, nous avons exploré des requêtes SQL simples pour interagir avec les données, démontrant l'importance des bases de données relationnelles dans la gestion des applications modernes. Ce projet nous a également permis de voir comment structurer les données en tables pour une meilleure organisation et intégrité des données.

Ce projet nous a permis d'approfondir notre compréhension des API REST et d'apprendre à intégrer une base de données dans une application Node.js. Grâce à cette intégration, notre API est maintenant capable de gérer des données de manière persistante, offrant ainsi une base solide pour évoluer vers une application de production. Nous avons également appris les bonnes pratiques en matière de validation et de gestion des erreurs, assurant ainsi la robustesse et la fiabilité de notre API.