Pair Programming – API

Antworten von René Wollny | 28.06.2023

Wir erstellen eine beispielhafte API, die CRUD-Operationen (Create, Read, Update, Delete) für Restaurants implementiert. Es sollen Informationen über Restaurants in JSON-Objekten gespeichert werden. Die JSON-Objekte werden zunächst als Liste im Arbeitsspeicher (in-memory) gespeichert. Ein Restaurant-Objekt besteht aus mindestens dem Namen, der Adresse und einer Kategorie (italienisches Restaurant, griechisches Restaurant, etc.).

1. Implementiere einen GET-Endpunkt (/restaurants), der die Liste der JSON-Objekte an den Client zurückgibt. Du kannst damit die gespeicherten Restaurants anzeigen.

2. Implementiere einen POST-Endpunkt (/restaurant) zum Speichern neuer Restaurants. Analysiere zuerst, ob das ankommende JSON-Objekt Name, Adresse und Kategorie enthält. Wenn ja, überprüfe, ob das Restaurant bereits in der Liste aller Restaurants vorhanden ist. Falls nicht, füge es hinzu. Als Identifikator kannst du vorerst den Namen verwenden.

3. Implementiere einen GET-Endpunkt (/restaurant/:name), der den Namen des Restaurants als Parameter empfängt und ein einzelnes Restaurant-Objekt zurückliefert, sofern es vorhanden ist.

4. Implementiere einen PUT-Endpunkt (/restaurant/:name), der den Namen des zu aktualisierenden Restaurants als Parameter empfängt. Der ankommende JSON-Payload soll das aktualisierte Restaurant-Objekt enthalten. Überprüfe, ob ein Restaurant mit dem angegebenen Namen existiert. Wenn ja, ersetze das vorhandene Restaurant-Objekt durch das neue und sende das aktualisierte Restaurant-Objekt an den Client zurück.

5. Implementiere einen DELETE-Endpunkt (/restaurant/:name), der den Namen des zu löschenden Restaurants als Parameter empfängt. Überprüfe, ob ein Restaurant mit dem angegebenen Namen existiert. Wenn ja, lösche es und sende das gelöschte Restaurant-Objekt an den Client zurück.

Hinweise:

* Verwende Express.js, um die API zu erstellen und die Endpunkte zu implementieren.
* Achte darauf, dass du die erforderliche Middleware und Routen einrichtest.
* Um die Restaurants im Arbeitsspeicher zu speichern, kannst du eine einfache Liste verwenden.
* Beim Implementieren des POST-Endpunkts solltest du den eingehenden JSON-Payload analysieren und sicherstellen, dass die erforderlichen Felder vorhanden sind, bevor du das Restaurant hinzufügst.
* Der GET-Endpunkt für ein einzelnes Restaurant sollte überprüfen, ob das Restaurant mit dem angegebenen Namen existiert. Wenn ja, sende das Restaurant-Objekt an den Client zurück.
* Beim Implementieren des PUT-Endpunkts solltest du überprüfen, ob das zu aktualisierende Restaurant mit dem angegebenen Namen vorhanden ist. Wenn ja, ersetze das vorhandene Restaurant-Objekt durch das neue und sende das aktualisierte Restaurant-Objekt an den Client zurück.
* Für den DELETE-Endpunkt überprüfe, ob das zu löschende Restaurant vorhanden ist, bevor du es löschst.
* Stelle sicher, dass die API-Endpunkte korrekte HTTP-Statuscodes und Rückgabewerte senden, um den Client über den Erfolg oder Fehler der Anfrage zu informieren.

Viel Erfolg bei der Implementierung der beispielhaften CRUD-API für Restaurants!