Übung 11+12: REST

Beachtet bitte, dass nach der Umsetzung des RESTful Service euer Blog vorerst nicht über HTML (views) angezeigt werden kann!

Aufgabe: Controller ändern (getAllPosts, createPost und readPost)

Bevor wir mit einer RESTful API anfangen können, müssen wir einige Vorbereitungen treffen. Unser „blogController“ ist aktuell so konfiguriert, dass dieser eine Response in Form einer View an den Client sendet. Bei einem RESTful Service muss der Controller allerdings immer mit einem JSON antworten.

Passen Sie Ihren blogController bzw. die darin enthaltenen Funktionen so an, dass diese immer mit einem Status und JSON antworten.

Beispiel für „getAllPosts“

```
const getAllPosts = (req, res) => {
    res.status(200).json({ "success": true, data: posts })
}
```

Aufgabe: PUT Post im Controller (updatePost)

In Übung 5 haben Sie eine Funktion geschrieben, welche einen neuen Post in Ihrem Blog erzeugt. Erstellen Sie nun eine entsprechende PUT-Route, welche einen vorhandenen Post aktualisiert. Welche schon vorhandenen Funktionen können Sie in diesem Zusammenhang nutzen? Denken Sie auch daran einen entsprechenden HTTP Status Code an den Client zurückzusenden.

Hinweis: Sehen Sie sich die Methode „findIndex“ einmal genauer an.

Aufgabe: DELETE Post im Controller (deletePost)

Vervollständigen Sie Ihren RESTful Service, indem Sie eine Route für das Löschen eines Posts implementieren. Denken Sie auch daran einen entsprechenden HTTP Status Code an den Client zurückzusenden.

Aufgabe: REST API für Blog

Erstellen Sie eine neue Routen-Datei „api.js“. Binden Sie diese in der app.js-Datei unter der Route „/api“ ein.

Erstellen Sie innerhalb dieser API die Routen für einen RESTful Service des Blogs. Diese sollten auf die in der Tabelle angegebenen Routen reagieren und die entsprechenden Funktionen des Controllers (getAllPosts, createPost, updatePost, deletePost und readPost) aufrufen.

http-Methode	Route	Funktion
GET	http://localhost/api/blog/	Gibt eine Liste (JSON) aller Posts zurück.
POST	http://localhost/api/blog/	Erstellt einen neuen Post in der JSON-Datei und gibt diesen auch im JSON Format zurück
GET	http://localhost/api/blog/id	Gibt einen einzelnen Post in einem JSON-Format zurück
PUT	http://localhost/api/blog/id	Aktualisiert einen schon vorhandenen Post und gibt diesen im JSON-Format zurück
DELETE	http://localhost/api/blog/id	Löscht einen Post

Überprüfen Sie mittels Postman oder ThunderClient alle Schnittstellen.

Aufgabe: Benutzer (*)

Zur Vorbereitung der nächsten Übungen können Sie das im Blog erworbene Wissen (und Code) nutzen, um einen RESTful Service „users“ anzulegen. Die entsprechenden Routen können Sie der „api.js“-Datei hinzufügen. Erstellen Sie auch eine entsprechende JSON-Datei (Model) welche ID, Benutzername, Passwort (noch im Klartext) und E-Mail-Adresse speichern kann. Der User-Service sollte, analog zum Blog-Service, die GET, POST, PUT und DELETE Methoden implementieren. Testen Sie auch diesen Service mit Postman oder ThunderClient.