AssetController.js

Die Datei assetController.js enthält die **Logik für die Verwaltung von Assets** in der Datenbank. Sie stellt verschiedene Funktionen bereit, um Assets **zu erstellen, abzurufen, zu aktualisieren und zu löschen**.  
Zusätzlich gibt es eine **Status-Endpoint**, um zu prüfen, ob die API läuft.

**Ablauf des Codes (assetController.js)**

1️ **mport des Asset-Modells**

import Asset from "../models/assetModel.js";

→ Damit wird das **MongoDB-Model Asset** importiert, um Datenbankoperationen auszuführen.

2️. **Abrufen aller Assets (GET /api/assets)**

export const getAssets = async (req, res) => { ... };

→ Ruft **alle Assets aus der Datenbank ab** und gibt sie als JSON zurück.  
**Fehlerhandling:** Falls ein Problem auftritt, wird eine Fehlermeldung gesendet.

3️. **Erstellen eines neuen Assets (POST /api/assets)**

export const createAsset = async (req, res) => { ... };

→ Erstellt **ein neues Asset** mit den Daten aus der Anfrage und speichert es in der Datenbank.  
**Fehlerhandling:** Falls Daten fehlen oder falsch sind, gibt die API einen Fehler zurück.

4️. **Aktualisieren eines Assets (PUT /api/assets/:id)**

export const updateAsset = async (req, res) => { ... };

→ Aktualisiert **ein bestehendes Asset** anhand der ID und gibt das **aktualisierte Objekt** zurück.  
**Fehlerhandling:** Falls das Asset nicht existiert, gibt die API eine **404-Fehlermeldung** zurück.

5️. **Löschen eines Assets (DELETE /api/assets/:id)**

export const deleteAsset = async (req, res) => { ... };

→ **Löscht ein Asset aus der Datenbank** anhand der ID und gibt eine Bestätigung zurück.  
**Fehlerhandling:** Falls das Asset nicht gefunden wurde, gibt die API eine Fehlermeldung aus.

6️. **Statusabfrage der API (GET /api/assets/status)**

export const getApiStatus = (req, res) => { ... };

→ Gibt einfach { "message": "Asset API ist aktiv" } zurück, um zu testen, ob die API läuft.

**📊 Zusammenfassung**

|  |  |  |
| --- | --- | --- |
| **Methode** | **Route** | **Beschreibung** |
| GET | /api/assets | Alle Assets abrufen |
| POST | /api/assets | Neues Asset erstellen |
| PUT | /api/assets/:id | Asset aktualisieren |
| DELETE | /api/assets/:id | Asset löschen |
| GET | /api/assets/status | API-Status prüfen |

**🛠️ Warum ist dieser Code wichtig?**

✅ Er stellt **alle wichtigen CRUD-Funktionen** für die Assets bereit.  
✅ Er enthält **sauberes Fehlerhandling**, um Probleme zu erkennen.  
✅ Er bietet eine **einfache Statusprüfung**, um die API schnell zu testen.

**📌 Fazit:**  
Der assetController.js ist das **Herzstück der Asset-Verwaltung**. Er stellt die wichtigsten Funktionen bereit, damit die API sinnvoll genutzt werden kann. 🚀