Apps.js

Die Datei app.js ist der **Haupteinstiegspunkt** für unsere Anwendung.  
Sie startet den Express-Server, verbindet sich mit der Datenbank und stellt die API-Routen bereit.

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

1️. **mport der benötigten Module**

import express from "express";

import dotenv from "dotenv";

import cors from "cors";

import connectDB from "./db.js";

import assetRoutes from "./routes/assetRoutes.js";

→ **express** ist das Webframework für den Server.  
→ **dotenv** lädt Umgebungsvariablen (z. B. für die Datenbankverbindung).  
→ **cors** erlaubt den Zugriff auf die API von anderen Domains.  
→ **connectDB** ist die Funktion zum Verbinden mit MongoDB.  
→ **assetRoutes** enthält die API-Routen für Assets.

2️. **Umgebungsvariablen laden**

dotenv.config();

→ Lädt die .env-Datei, um z. B. den **PORT** und die **Datenbank-URL** zu nutzen.

3️⃣ **Express-App initialisieren & Port definieren**

js

Code kopieren

const app = express();

const PORT = process.env.PORT || 3000;

→ Erstellt eine **Express-Instanz**.  
→ Setzt den **Port** aus den Umgebungsvariablen oder Standard **3000**.

4️. **Datenbankverbindung herstellen**

connectDB();

→ **Stellt die Verbindung** zur MongoDB-Datenbank her.

5️. **Middleware einbinden**

app.use(cors());

app.use(express.json({ type: "application/json", limit: "10mb" }));

app.use(express.urlencoded({ extended: true }));

→ **cors()** ermöglicht API-Zugriff von externen Clients.  
→ **express.json()** erlaubt das Verarbeiten von **JSON-Anfragen** (bis 10 MB Größe).  
→ **express.urlencoded({ extended: true })** verarbeitet **Formulardaten**.

6️. **Health-Check Route definieren**

app.get("/health", (req, res) => {

res.json({ status: "OK", timestamp: new Date().toISOString() });

});

→ Diese Route **prüft, ob der Server läuft**.

7️. **API-Routen für Assets einbinden**

app.use("/api/assets", assetRoutes);

→ Verknüpft **assetRoutes.js** mit der Route /api/assets.

8️. **Server starten**

app.listen(PORT, () => console.log(`✅ Server läuft auf http://localhost:${PORT}`));

→ Startet den **Express-Server** auf dem definierten Port.  
→ Gibt eine **Bestätigung in der Konsole aus**.

**📊 Übersicht der API-Routen in app.js**

|  |  |  |  |
| --- | --- | --- | --- |
| **Methode** | **Route** | **Funktion** | **Beschreibung** |
| **GET** | /health | Health-Check | Prüft, ob der Server läuft |
| **GET** | /api/assets | getAssets | Alle Assets abrufen |
| **POST** | /api/assets | createAsset | Neues Asset erstellen |
| **PUT** | /api/assets/:id | updateAsset | Asset mit ID aktualisieren |
| **DELETE** | /api/assets/:id | deleteAsset | Asset mit ID löschen |
| **GET** | /api/assets/status | getApiStatus | API-Status prüfen |

**🛠️ Warum ist dieser Code wichtig?**

✅ **Erstellt eine zentrale API-Schnittstelle** für den Zugriff auf die Daten.  
✅ **Saubere Trennung** von Routen, Controllern und Datenbanklogik.  
✅ **Einfache Wartung**, weil Middleware und Konfiguration gut strukturiert sind.  
✅ **Skalierbar**, da weitere Routen oder Services einfach hinzugefügt werden können.

**📌 Fazit:**

Die Datei app.js ist das **Herzstück** der API. Sie sorgt für die **Verarbeitung von Anfragen**, die **Datenbankverbindung** und den **Start des Servers**.