REST-API-Spielaufgabe

Ziel der Aufgabe

- Erstelle eine REST-API mit Express.js, die ein spielerisches Erlebnis bietet.
- Verwende die API, um Spielgegenstände zu verwalten, aufzuleveln und in Kämpfen einzusetzen.

Wichtige Ressourcen

1. **Express.js Dokumentation**: expressjs.com
   https://github.com/expressjs/expressjs.com/tree/gh-pages/de
2. **MDN Web Docs (HTTP-Statuscodes)**: developer.mozilla.org
3. **CORS Dokumentation**: npmjs.com/package/cors

Grundidee des Spiels

- API verwaltet Spielgegenstände mit folgenden Eigenschaften:
  - `id`: Eindeutige Kennung
  - `name`: Name des Gegenstands (z.B. „Flammenschwert“)
  - `type`: Art des Gegenstands (z.B. „Waffe“, „Trank“, „Rüstung“)
  - `power`: Stärke des Gegenstands (numerisch)

Aufgabe 1: Projekt einrichten

1. **Ordner erstellen** und Konsole öffnen
2. **Projekt initialisieren**:

```
npm init -y
```

3. **Pakete installieren**:

```
npm install express cors
```

Aufgabe 1: Projekt einrichten (Fortsetzung)

4. **server.js erstellen** mit folgendem Grundgerüst:

```
const express = require('express');
const cors = require('cors');
const app = express();
const port = 4000;

app.use(cors());
app.use(express.json());

app.listen(port, () => {
  console.log(\`Server läuft auf Port ${port}\`);
});
```

5. **Server testen** mit:

```
node server.js
```

Aufgabe 2: Grundgerüst der API

- **GET-Route** `/items` erstellen:

```
let items = []; // Liste der Gegenstände

app.get('/items', (req, res) => {
  res.json(items); // Gibt die Liste zurück
});
```

Aufgabe 2: Grundgerüst der API (Fortsetzung)

- **POST-Route** `/items` erstellen, um neue Gegenstände hinzuzufügen:

```
app.post('/items', (req, res) => {
  const { name, type } = req.body;
  const newItem = {
    id: items.length + 1,
    name,
    type,
    power: Math.floor(Math.random() * 41) + 10,
  };
  items.push(newItem);
  res.status(201).json(newItem);
});
```

Aufgabe 3: Spielerische Routen

- **Kampfroute** `/battle` hinzufügen:
  - Spieler wählt einen Gegenstand per `id` aus
  - API erstellt einen zufälligen Gegner
  - Vergleicht die `power`-Werte und bestimmt den Gewinner

Hinweis: Verwende die Mechanik aus den Folien zu zufälligen Werten.

Aufgabe 3: Spielerische Routen

```
app.post('/battle', (req, res) => {
  const { id } = req.body;
  // gegenstand des Spielers finden
  // Zufälligen Gegner erstellen
  // power-Werte vergleichen
});
```

- **Spannung erhöhen**: Überlege dir eine spannende Spielbeschreibung!

Aufgabe 4: Schwierigkeitsgrad erhöhen

a) Validierung der Eingaben

- **Middleware** schreiben, die sicherstellt, dass `name` und `type` gültig sind.
- **Hinweis**: Sieh dir die Folien zu Middleware an.

beispiel:

```
function validateItem(req, res, next) {
  // Überprüfe `name` und `type`
  next();
}
```

b) PATCH-Route zum Aufleveln

- **Route** `/items/:id/levelup` erstellen
- Erhöhe den `power` -Wert eines Gegenstands um 1-10
- **Hinweis**: Verwende `Math.random()` und `Math.floor()`

c) Heiltrank-Route

- **Route** `/items/heal` erstellen, um einen Heiltrank zu verwenden
- Überprüfe, ob der Gegenstand ein „Trank“ ist und regeneriere `health`-Wert
- **Hinweis**: Verwende Bedingungen, um Heilung zu begrenzen

Bonusaufgabe: Turnier-Modus

- **POST-Route** `/tournament` erstellen
- Spieler wählt drei Gegenstände für ein Turnier
- API simuliert eine Serie von Kämpfen

Bonusaufgabe: Turnier-Modus (Hinweise)

- **Schleife verwenden**, um Kämpfe zu simulieren
- **Gegner zufällig** erstellen und `power`-Werte vergleichen
- Zähle die gewonnenen Kämpfe und bestimme das Ergebnis

Schnipsel:

```
let wins = 0;
// schleife für Kämpfe
// Ergebnis berechnen
```