DOKUMENTATION

Modul 295 «Backend für Applikationen realisieren"

Erstellung eines RESTful Backend für eine Kursverwaltung

Das Ziel dieses üK's ist ein funktionierendes PHP-Backend zu programmieren, welches in dem später folgenden üK Modul 294 «Frontend einer interaktiven Webapplikation» mittels eines auf React JS basierendes Frontend angesteuert werden kann.

Jannis Westphal
jannis.westphal@gmail.com

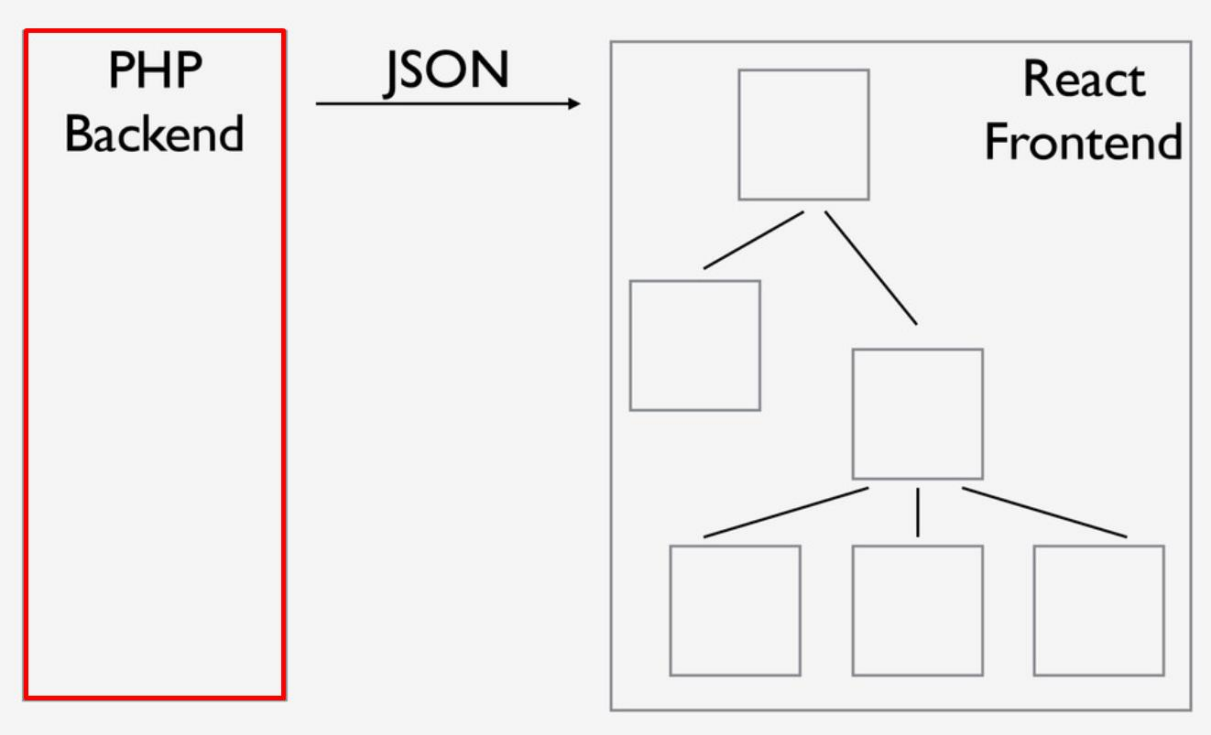
Inhalt

Auftrag

Das Thema lautet „Erstellung eines RESTful Backend für eine Kursverwaltung“.

Grundsätzlich geht es in diesem Modul darum ein funktionierendes PHP-Backend zu programmieren, welches dann in einem zukünftigen Kurs (üK Modul 294 „Frontend einer interaktiven Webapplikation“) mit einem auf React JS basierenden Frontend angesteuert werden kann.

Auf der folgenden Grafik wird dies nochmal verdeutlicht:



Anforderungen

Grundanforderungen

Programmiersprache:

- PHP (Version 8.1 oder neuer)

Erstellen einer RESTful API für eine Kursverwaltung:

- GET, um eine Ressource abzurufen
- PUT, um den Zustand einer Ressource zu ändern oder zu aktualisieren
- POST, um die Ressource zu erstellen
- DELETE, um die Ressource zu entfernen

Datenformat:

- Application/json

Implementierung aller CRUD-Operationen:

- Create, Datensatz anlegen
- Read oder Retrieve, Datensatz lesen
- Update, Datensatz aktualisieren
- Delete oder Destroy, Datensatz löschen

Sicherheit:

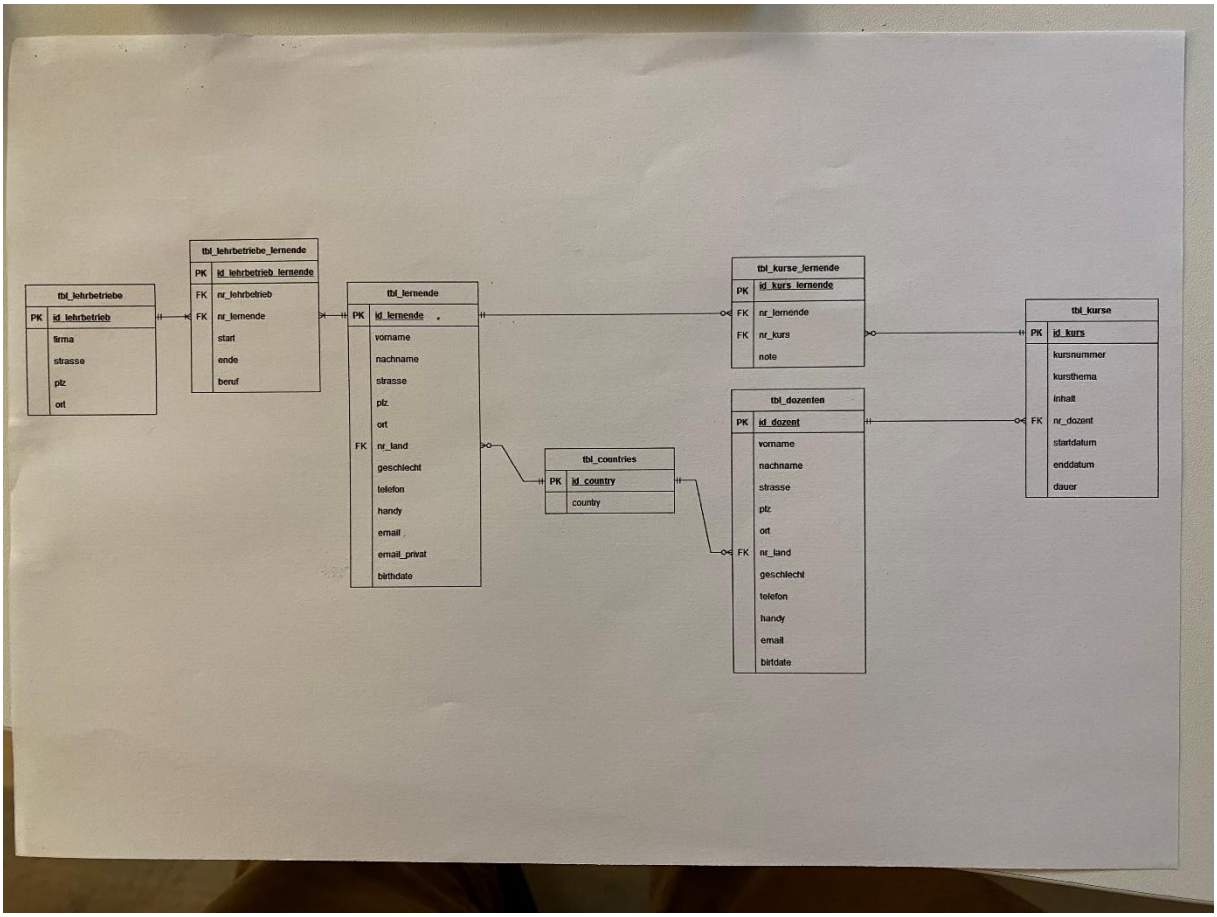
- POST-Parameter sind konsequent zu validieren
- Prepared SQL Statements sind konsequent zu verwenden

Optional: Authentifizierung

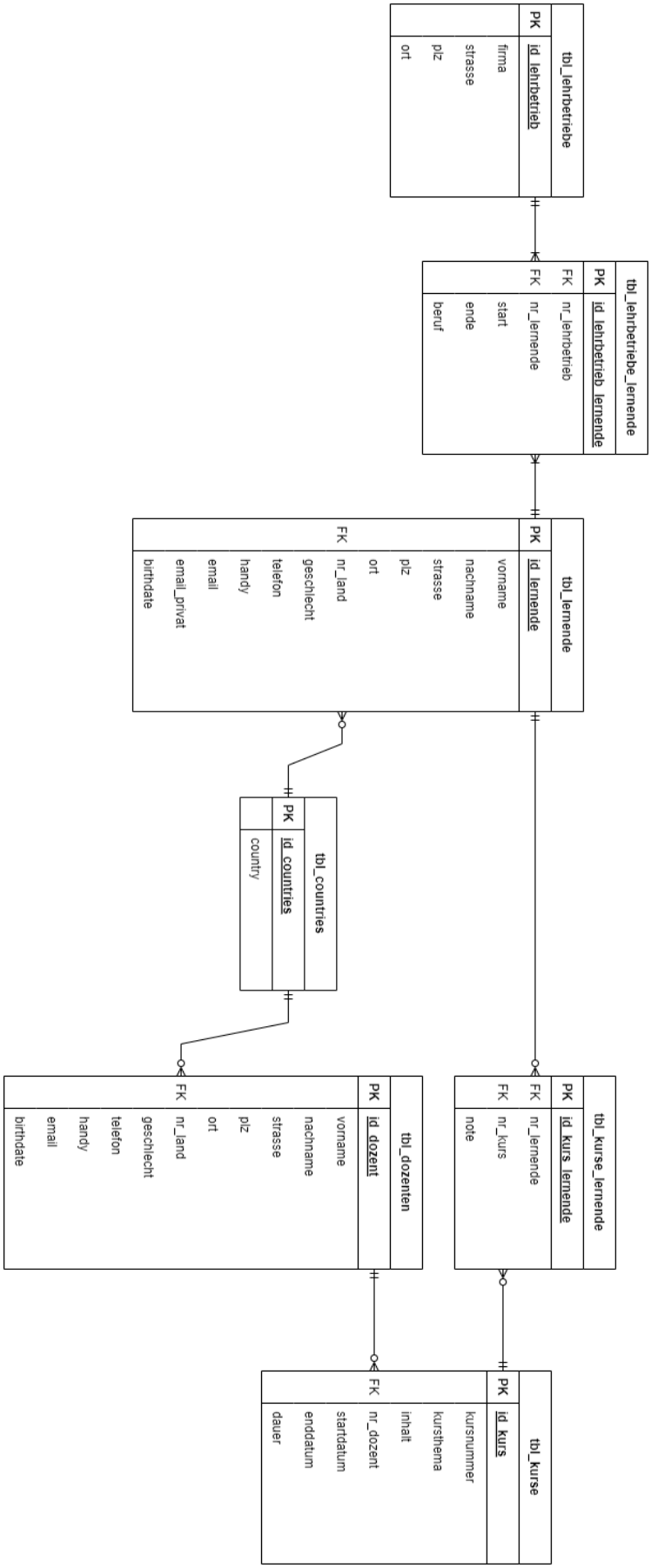
- Login und Authentifizierung mittels JWT Web Tokens

Funktionale Anforderungen

Es ist eine Datenbank mit MySQL oder MariaDB zu erstellen. Diese sollte eine Struktur wie diese haben:



Ich habe die gewünschte Struktur in diesem ERM festgehalten:



Die zuvor beschriebenen CRUD-Aktionen müssen für folgende API-Endpoints abgebildet sein:

- /lehrbetriebe/[id]
- /lernende/[id]
- /lehrbetriebe_lernende/[id]
- /laender/[id]
- /dozenten/[id]
- /kurse/[id]
- /kurse_lernende/[id]
- Optional: /benutzer/[id]

Endpoints für alle Einträge:

- z.B. /lehrbetriebe/all

Nicht funktionale Anforderungen

Kommentare im Quellcode

Der Quellcode muss gut mit Kommentaren erklärt sein.

Schnittstellen-Dokumentation

Es ist eine API-Dokumentation zu erstellen. Dafür ist ein geeignetes Tool zu evaluieren:

- swagger.io
- postman.com
- redocly.com
- stoplight.io
- eigene Tools

Softwareverwaltung

- Softwareablage auf Github oder anderer Versionsverwaltung

Testen

- Erstellung eines Testkonzepts
- Durchführen von Tests

Dokumentation

Es ist eine minimale Dokumentation von ca. 10 Seiten zu erstellen.

Diese Dokumentation enthält:

- Vergleichsmatrix der Evaluation des API-Dokumentation Tools
- Beschreibung Vorgehen für Commits ins Repository
- History der Commits

- Testkonzept
- Testberichte
- Link zur Schnittstellen-Dokumentation

API – Dokumentation

Evaluation Tool

Um evaluieren zu können, welches Tool das am meisten geeignete für die API – Dokumentation ist, habe ich mich entschieden eine Nutzwertanalyse zu erstellen.

Hierbei werden die folgenden Tools berücksichtigt:

- Swagger.io
- Postman.com
- Redocly.com

Ich habe diese Tools ausgewählt, weil sie insgesamt die gängigsten und bekanntesten sind.

Um die verschiedenen Tools miteinander vergleichen zu können, benötige ich zuerst Kriterien. Dabei habe ich mich für folgende entschieden:

- Kosten
- Benutzerfreundlichkeit
- Dokumentationsqualität
- Flexibilität
- Integration
- Community & Support

Natürlich sind nicht alle dieser Kriterien gleichwichtig für mich, weshalb ich eine prozentuale **Gewichtung** eingeführt habe. Das bedeutet, dass jedes Kriterium einen unterschiedlich hohen Prozentanteil bekommt. Je höher der Anteil, desto wichtiger das Kriterium und desto höher die potenziell zu erreichende Punktzahl.

Dazu kommt die **Bewertung** selbst, die in Form einer Note von 6 (sehr gut) bis 1 (sehr schlecht) vergeben wird. Diese Note wird dann mit der Gewichtung multipliziert und ergibt die insgesamten **Punkte** für dieses Kriterium.

Zuletzt werden alle Punkte addiert und ergeben den finalen Nutzwert des jeweiligen Tools. Das Tool mit dem höchsten Nutzwert ist somit für meine Situation das am besten geeignete Tool.

Nr	Kriterium	Gewichtung	swagger.io		postman.com		redocly.com	
			Bewertung	Punkte	Bewertung	Punkte	Bewertung	Punkte
1	Kosten	30	6	180	6	180	6	180
2	Benutzerfreundlichkeit	30	6	180	5	150	5	150
3	Dokumentationsqualität	20	5	100	5	100	5	100
4	Flexibilität	10	5	50	5	50	5	50
5	Integration	5	5	25	6	30	5	25
6	Community & Support	5	6	30	6	30	6	30
Nutzwert		100		565		540		535
Bewertung: 6 = sehr gut, 5 = gut, 4 = befriedigend, 3 = ausreichend, 2 = mangelhaft, 1 = ungenügend								

Wie aus der obigen Tabelle hervorgeht, ist „Swagger“ der Gewinner der Analyse. Da alle verglichene Tools sehr gut sind, ist es ein knappes Ergebnis. Letztendlich hat Swagger vor allem durch die sehr hohe Benutzerfreundlichkeit den Vorsprung gegenüber den anderen Tools erhalten.

Ich nutze für meine API-Dokumentation also **Swagger** als Tool.

Link

Ich habe für meine API-Dokumentation die Vorlage aus dem Swagger Editor genutzt und sie auf meine Daten angepasst. Durch diesen Editor ist es möglich die gesamte Dokumentation in einer YAML-Datei darzustellen. Dank der VS-Code Extension „OpenAPI (Swagger) Editor war es mir möglich direkt in VS-Code die YAML-Datei zu bearbeiten, unter anderem durch das praktische Preview-Fenster, das mir erlaubt zu sehen, wie meine API-Dokumentation im SwaggerUI aussieht. Diese Datei wird dann wiederum vom Swagger Editor in ein JSON umgewandelt, welches für das Swagger UI darstellbar ist.

Ich habe die YAML- und die JSON-Datei direkt in dieses Dokument eingebettet, sodass sie direkt geöffnet und gespeichert werden können:


swagger.yaml


swagger.json

Doppelklicken zum Öffnen der JSON-Datei

Ausserdem finden sich alle relevanten Dateien dazu im Ordner „Code/docs“.

Commits

Vorgehen

Bei den Commits orientiere ich mich weniger an vordefinierte Arbeitspakete, sondern an einen definierten Zeitrahmen. Das bedeutet, dass ich nicht etwa nach jedem fertigen Controller einen Commit mache, sondern eher ca. alle 2 Stunden meine neuen Änderungen commite und vor allem auch pushe. In den Commit-Messages ist deshalb auch nicht genau beschrieben, welche Dinge ich ganz genau verändert hab, da es mit

meiner Methode häufig gar nicht möglich ist all das in eine Message zu schreiben. Und letztendlich ist das auch nicht notwendig. Stattdessen finden sich in meinen Commit Messages nur sehr grobe und häufig vage Beschreibungen meiner Tätigkeiten. Jedoch kann ich diese genau wiedererkennen, wenn ich sie sehe und auf die Weise im Extremfall zurückverfolgen wann ich welche Änderung vorgenommen hab.

Wenn ich in einer Message eine Änderung beschreibe, nutze ich immer die Gegenwartsform, also z.B. «add new function» statt «added new function». Der Grund dafür ist, dass ich die neue Änderung selbst beschreibe um die das Programm erweitert wird und nicht einfach 'was **ich** gemacht habe'.

History

Dec 18, 2024

- `75c50e3` – add api-documentation & restructure folders
- `c5faa9d` – fill delete-files & remove 'benutzer' controller

Dec 11, 2024

- `f05bee2` – add delete-files & refactor

Nov 27, 2024

- `9a642d8` – add remaining controllers & validate foreign keys
- `277eb44` – add most controllers
- `109342a` – lots of stuff

Nov 20, 2024

- `d5f0681` – routes etc.
- `2706e0e` – restructure again
- `af2ea98` – add namespace
- `450a5c4` – restructure
- `22be9ea` – router + controllers
- `d25d955` – testing stuff out
- `8d725a9` – init

Tests

Konzept

Mein Testkonzept orientiert sich nach den Operationen (GET, POST, etc.) und nach den verschiedenen Endpoints. Das bedeutet es gibt einen Testfall für jede **Operation** auf jedem **Endpoint**. Je nach Operation erfasse ich noch einen **Input** und führe dann die Funktion aus. Zuletzt vergleiche ich das **erwartete** mit dem **tatsächlichen** Ergebnis und stell daraus fest, ob der Test erfolgreich war oder fehlgeschlagen ist.

Das setze ich in einer Tabelle mit den folgenden relevanten Angaben um:

- Operation
- Endpoint
- Input
- Expected
- Actual
- Result

Berichte

Umgesetzt habe ich die Testbericht in einer Excel-Tabelle mit den oben genannten Spalten. Diese Tabelle ist auch im Ordner „Documentation“ zu finden.

Testberichte

Operation	Endpoint	Input	Expected	Actual	Result
GET	api.test/lehrbetriebe		{"status":"success","message":"Lehrbetriebe retrieved successfu	{"status":"success","message":"Lehrbetriebe retrieved successfu	PASS
GET	api.test/lehrbetriebe/201		{"status":"success","message":"Lehrbetrieb found","data":{"id_	{"status":"success","message":"Lehrbetrieb found","data":{"id_	PASS
GET	api.test/lernende		{"status":"success","message":"Lernende retrieved successfully"	{"status":"success","message":"Lernende retrieved successfully"	PASS
GET	api.test/lernende/301		{"status":"success","message":"Lehrling found","data":[{"id_ler	{"status":"success","message":"Lehrling found","data":[{"id_ler	PASS
GET	api.test/lehrbetrieb_lernende		{"status":"success","message":"Entries retrieved successfully","	{"status":"success","message":"Entries retrieved successfully","	PASS
GET	api.test/lehrbetrieb_lernende/701		{"status":"success","message":"Lehrbetrieb_lernende found","d	{"status":"success","message":"Lehrbetrieb_lernende found","d	PASS
GET	api.test/laender		{"status":"success","message":"L\u00e4nder retrieved successfu	{"status":"success","message":"L\u00e4nder retrieved successfu	PASS
GET	api.test/laender/101		{"status":"success","message":"Land found","data":{"id_countr	{"status":"success","message":"Land found","data":{"id_countri	PASS
GET	api.test/dozenten		{"status":"success","message":"Dozenten retrieved successfully	{"status":"success","message":"Dozenten retrieved successfully"	PASS
GET	api.test/dozenten/401		{"status":"success","message":"Dozent found","data":[{"id_doz	{"status":"success","message":"Dozent found","data":[{"id_doze	PASS
GET	api.test/kurse		{"status":"success","message":"Kurse retrieved successfully","d	{"status":"success","message":"Kurse retrieved successfully","da	PASS
GET	api.test/kurse/501		{"status":"success","message":"Kurs found","data":[{"id_kurs":	{"status":"success","message":"Kurs found","data":[{"id_kurs":"	PASS
GET	api.test/kurse_lernende		{"status":"success","message":"Entries retrieved successfully","	{"status":"success","message":"Entries retrieved successfully","	PASS
GET	api.test/kurse_lernende_801		{"status":"success","message":"Kurs_lernende found","data":[{"	{"status":"success","message":"Kurs_lernende found","data":[{"	PASS
POST	api.test/lehrbetriebe	{"id_lehrbetrieb":201,"firma":"Hami	{"status":"success","message":"Lehrbetrieb created successfully	{"status":"success","message":"Lehrbetrieb created successfully	PASS
POST	api.test/lernende	{"id_lernende":"301","vorname":"M	{"status":"success","message":"Lehrling created successfully"}	{"status":"success","message":"Lehrling created successfully"}	PASS
POST	api.test/lehrbetrieb_lernende	{"id_lehrbetrieb_lernende":"701","fk	{"status":"success","message":"Lehrbetrieb_lernende created su	{"status":"success","message":"Lehrbetrieb_lernende created su	PASS
POST	api.test/laender	{"id_countries":"101","country":"Sw	{"status":"success","message":"Land created successfully"}	{"status":"success","message":"Land created successfully"}	PASS
POST	api.test/dozenten	{"id_dozent":"401","vorname":"Max	{"status":"success","message":"Dozent created successfully"}	{"status":"success","message":"Dozent created successfully"}	PASS
POST	api.test/kurse	{"id_kurs":"501","kursnummer":"12	{"status":"success","message":"Kurs created successfully"}	{"status":"success","message":"Kurs created successfully"}	PASS
POST	api.test/kurse_lernende	{"id_kurs_lernende":"801","fk_lerne	{"status":"success","message":"Kurs_lernende created successfu	{"status":"success","message":"Kurs_lernende created successfu	PASS
PUT	api.test/lehrbetriebe	{"firma":"Testfirma"}	{"status":"success","message":"Lehrbetrieb updated successfull	{"status":"success","message":"Lehrbetrieb updated successfull	PASS
PUT	api.test/lernende	{"vorname":"Marie"}	{"status":"success","message":"Lehrling updated successfully"}	{"status":"success","message":"Lehrling updated successfully"}	PASS
PUT	api.test/lehrbetrieb_lernende	{"start":"2011-08-01"}	{"status":"success","message":"Lehrbetrieb_lernende updated s	{"status":"success","message":"Lehrbetrieb_lernende updated s	PASS
PUT	api.test/laender	{"country":"Germany"}	{"status":"success","message":"Land updated successfully"}	{"status":"success","message":"Land updated successfully"}	PASS
PUT	api.test/dozenten	{"nachname":"Müller"}	{"status":"success","message":"Dozent updated successfully"}	{"status":"success","message":"Dozent updated successfully"}	PASS
PUT	api.test/kurse	{"kursnummer":"4321"}	{"status":"success","message":"Kurs updated successfully"}	{"status":"success","message":"Kurs updated successfully"}	PASS
PUT	api.test/kurse_lernende	{"fk_kurs":"502"}	{"status":"success","message":"Kurs_lernende updated success	{"status":"success","message":"Kurs_lernende updated successf	PASS
DELETE	api.test/lehrbetriebe/201		{"status":"success","message":"Lehrbetrieb deleted successfully	{"status":"success","message":"Lehrbetrieb deleted successfully	PASS
DELETE	api.test/lernende/301		{"status":"success","message":"Lehrling deleted successfully"}	{"status":"success","message":"Lehrling deleted successfully"}	PASS
DELETE	api.test/lehrbetrieb_lernende/701		{"status":"success","message":"Lehrbetrieb_lernende deleted su	{"status":"success","message":"Lehrbetrieb_lernende deleted su	PASS
DELETE	api.test/laender/101		{"status":"success","message":"Land deleted successfully"}	{"status":"success","message":"Land deleted successfully"}	PASS
DELETE	api.test/dozenten/401		{"status":"success","message":"Dozent deleted successfully"}	{"status":"success","message":"Dozent deleted successfully"}	PASS
DELETE	api.test/kurse/501		{"status":"success","message":"Kurs deleted successfully"}	{"status":"success","message":"Kurs deleted successfully"}	PASS
DELETE	api.test/kurse_lernende/801		{"status":"success","message":"Kurs_lernende deleted successfu	{"status":"success","message":"Kurs_lernende deleted successfu	PASS