Arbeitsblatt: Projekt mit FastAPI, PostgreSQL, Docker und Weboberfläche

1 Hinweis zur Arbeitsweise

Dieses Projekt ist ausdrücklich mit Unterstützung durch eine geeignete KI zu erarbeiten. Die KI soll für Planung, Entwurf, Codegerüste, Tests und Dokumentation genutzt werden. Alle Ergebnisse sind fachlich zu prüfen, anzupassen und nachvollziehbar zu begründen. Wichtige Prompts und Antworten werden dokumentiert.

2 Zielsetzung

Es ist eine kleine Anwendungssoftware umzusetzen, bestehend aus einem FastAPI Backend, einer PostgreSQL Datenbank und einer einfachen Weboberfläche in HTML und JavaScript. Bei komplexeren Interaktionen darf eine kleine Anwendung mit Vue.js eingesetzt werden. Das Projekt wird in einem GitHub Repository versioniert und reproduzierbar in Containern betrieben.

3 Vorgegebene Eckdaten

3.1 Architektur

- Zwei Container: ein Container für das FastAPI Backend, ein Container für PostgreSQL.
- Standardfall für das Frontend: statische HTML und JavaScript Dateien oder ein gebautes SPA werden vom Backend Container ausgeliefert. Optional ist ein separates Frontend zulässig, wenn dies begründet wird.
- Gemeinsames Netzwerk der Container. Die Anwendung ist lokal erreichbar.

3.2 Datenbank und Ports

- Es ist PostgreSQL zu verwenden.
- Interner Datenbankport 5432. Externer Hostport 5433.
- Das Backend ist unter Port 8000 erreichbar.

3.3 Containerisierung

- Docker Images für Backend und Datenbank werden verwendet. Für das Backend soll eine schlanke Python Runtime eingesetzt werden, zum Beispiel python:3.14-slim oder neuer.
- Persistente Daten der Datenbank werden über ein Volume gespeichert.
- Ein docker compose File ist zu erstellen. Der konkrete Inhalt wird nicht vorgegeben. Die Konfiguration muss die oben genannten Eckdaten erfüllen.

3.4 Repository und Versionierung

- Das Projekt ist in einem GitHub Repository versioniert.
- Eine aussagekräftige README beschreibt Zweck, Architektur, Startanleitung und Nutzung.

4 Projektvorschläge

Die folgenden Vorschläge dienen als Ausgangspunkt. Es ist ein Vorschlag auszuwählen oder in Absprache ein gleichwertiges Thema zu formulieren.

4.1 Schülerverwaltung

4.1.1 Kurzbeschreibung

Verwaltung einer Liste von Schülerinnen und Schülern mit Klasse und Punktestand.

4.1.2 Datenmodell

Entität Student mit den Attributen id, name, class_name, points.

4.1.3 ER-Diagramm in Worten

Student (id, name, class_name, points). id ist Primärschlüssel.

4.1.4 Endpunkte

GET /students, POST /students, GET /students/id, PUT /students/id, DELETE /students/id.

4.1.5 Benutzeroberfläche

Tabelle mit Filter nach Klasse und Formular zum Anlegen und Bearbeiten.

4.2 Bücherverwaltung

4.2.1 Kurzbeschreibung

Verwaltung einer kleinen Bibliothek mit Verfügbarkeitsstatus.

4.2.2 Datenmodell

Entität Book mit id, title, author, year, available.

4.2.3 ER-Diagramm in Worten

Book (id, title, author, year, available). id ist Primärschlüssel.

4.2.4 Endpunkte

GET /books, POST /books, GET /books/id, PUT /books/id, DELETE /books/id.

4.2.5 Benutzeroberfläche

Liste mit Suchfeld und Filter nach Autor sowie Checkbox für Verfügbarkeit.

4.3 Mini Kantine

4.3.1 Kurzbeschreibung

Anzeige eines Menüs und Anlegen einfacher Bestellungen.

4.3.2 Datenmodell

MenuItem mit id, name, price. Order mit id, customer, menu_item_id, quantity. Beziehung Order viele zu eins MenuItem.

4.3.3 ER-Diagramm in Worten

MenuItem (id, name, price), Order (id, customer, quantity, menu_item_id). menu_item_id ist Fremdschlüssel auf MenuItem.id.

4.3.4 Endpunkte

GET /menu, POST /orders, GET /orders.

4.3.5 Benutzeroberfläche

Auswahl eines Menüeintrags, Eingabe der Menge, Anzeige der Bestellungen.

4.4 Film oder Serien Favoriten

4.4.1 Kurzbeschreibung

Verwalten einer Liste persönlicher Favoriten mit Bewertung und Status.

4.4.2 Datenmodell

Movie mit id, title, year, rating, watched.

4.4.3 ER-Diagramm in Worten

Movie (id, title, year, rating, watched). id ist Primärschlüssel.

4.4.4 Endpunkte

GET /movies, POST /movies, GET /movies/id, PUT /movies/id, DELETE /movies/id.

4.4.5 Benutzeroberfläche

Liste mit Filter nach gesehen und ungesehen sowie Eingabe einer Bewertung.

4.5 Notizen mit Kategorien

4.5.1 Kurzbeschreibung

Notizen mit Titel, Inhalt und optionaler Kategorie.

4.5.2 Datenmodell

Note mit id, title, content, category, created_at.

4.5.3 ER-Diagramm in Worten

Note (id, title, content, category, created_at). id ist Primärschlüssel.

4.5.4 Endpunkte

GET /notes, POST /notes, GET /notes/id, PUT /notes/id, DELETE /notes/id.

4.5.5 Benutzeroberfläche

Liste mit Suchfeld und Filter nach Kategorie. Editorfeld für den Inhalt.

4.6 Aufgabenplaner für Lehrkräfte

4.6.1 Kurzbeschreibung

Verwaltung von Unterrichtsaufgaben, Abgabeterminen und Klassen.

4.6.2 Datenmodell

Task mit id, title, description, due_date, class_name, completed.

4.6.3 ER-Diagramm in Worten

Task (id, title, description, due_date, class_name, completed). id ist Primärschlüssel.

4.6.4 Endpunkte

GET /tasks, POST /tasks, GET /tasks/id, PUT /tasks/id, DELETE /tasks/id.

4.6.5 Benutzeroberfläche

Liste aller Aufgaben mit Filter nach Klasse und Status sowie Möglichkeit, Aufgaben als erledigt zu markieren.

4.7 Kursverwaltung

4.7.1 Kurzbeschreibung

Verwaltung von Kursen mit Teilnehmern.

4.7.2 Datenmodell

Course mit id, name, teacher. Student mit id, name. Enrollment mit course_id, student_id.

4.7.3 ER-Diagramm in Worten

Course (id, name, teacher), Student (id, name), Enrollment (course_id, student_id). Enrollment.course_id ist Fremdschlüssel auf Course.id, Enrollment.student_id auf Student.id.

4.7.4 Endpunkte

GET /courses, POST /courses, GET /courses/id, GET /courses/id/students, POST /enrollments, DELETE /enrollments.

4.7.5 Benutzeroberfläche

Liste der Kurse mit Ansicht der zugeordneten Studierenden und Formular zum Hinzufügen und Entfernen von Teilnehmern.

4.8 Inventarverwaltung

4.8.1 Kurzbeschreibung

Verwaltung von Geräten oder Gegenständen, etwa Schul-IT oder Leihgeräte.

4.8.2 Datenmodell

Item mit id, name, category, status, location, assigned_to.

4.8.3 ER-Diagramm in Worten

Item (id, name, category, status, location, assigned_to). id ist Primärschlüssel.

4.8.4 Endpunkte

GET /items, POST /items, GET /items/id, PUT /items/id, DELETE /items/id.

4.8.5 Benutzeroberfläche

Tabelle mit Filter nach Kategorie oder Status sowie Formular für neue Geräte.

4.9 Terminplaner

4.9.1 Kurzbeschreibung

Erfassung und Anzeige von Terminen mit Beschreibung und Datum.

4.9.2 Datenmodell

Event mit id, title, date, description, location.

4.9.3 ER-Diagramm in Worten

Event (id, title, date, description, location). id ist Primärschlüssel.

4.9.4 Endpunkte

GET /events, POST /events, GET /events/id, PUT /events/id, DELETE /events/id.

4.9.5 Benutzeroberfläche

Liste oder einfache Kalenderansicht der Termine mit Eingabeformular.

4.10 Feedback-System

4.10.1 Kurzbeschreibung

Sammeln und Auswerten von Rückmeldungen zu Projekten oder Unterricht.

4.10.2 Datenmodell

Feedback mit id, name, message, rating, created_at.

4.10.3 ER-Diagramm in Worten

Feedback (id, name, message, rating, created_at). id ist Primärschlüssel.

4.10.4 Endpunkte

GET /feedback, POST /feedback, DELETE /feedback/id.

4.10.5 Benutzeroberfläche

Formular für neue Rückmeldungen und Liste aller Einträge, sortiert nach Datum.

4.11 Stundenplan-Generator

4.11.1 Kurzbeschreibung

Verwaltung von Fächern und Zuordnung zu Tagen und Zeiten.

4.11.2 Datenmodell

Subject mit id, name, teacher. ScheduleEntry mit id, subject_id, day, start_time, end_time.

4.11.3 ER-Diagramm in Worten

Subject (id, name, teacher), ScheduleEntry (id, subject_id, day, start_time, end_time). subject_id ist Fremdschlüssel auf Subject.id.

4.11.4 Endpunkte

GET /schedule, POST /schedule, GET /subjects, POST /subjects.

4.11.5 Benutzeroberfläche

Anzeige des Stundenplans pro Tag und Eingabemaske zur Planung.

4.12 Rezeptverwaltung

4.12.1 Kurzbeschreibung

Sammlung von Kochrezepten mit Zutatenlisten.

4.12.2 Datenmodell

Recipe mit id, title, description. Ingredient mit id, name, recipe_id, amount, unit.

4.12.3 ER-Diagramm in Worten

Recipe (id, title, description), Ingredient (id, name, amount, unit, recipe_id). recipe_id ist Fremdschlüssel auf Recipe.id.

4.12.4 Endpunkte

GET /recipes, POST /recipes, GET /recipes/id, PUT /recipes/id, DELETE /recipes/id.

4.12.5 Benutzeroberfläche

Liste der Rezepte mit Detailansicht und Eingabeformular für Zutaten.

5 Anforderungen an die API Dokumentation

Die Schnittstellen sind ausführlich zu dokumentieren. Die Dokumentation enthält mindestens:

1. Übersicht der Endpunkte mit Pfad und HTTP Methode.
2. Beschreibung des Zwecks jedes Endpunkts.
3. Request Angaben je Endpunkt:
   - Pfadparameter mit Namen, Typ und Bedeutung.
   - Query Parameter mit Namen, Typ, Bedeutung und Standardwerten.
   - Request Body Struktur mit Attributen, Typen, Pflichtfeldern und zulässigen Werten.
4. Response Angaben je Endpunkt:
   - Statuscodes mit Bedeutung.
   - Response Body Struktur mit Attributen, Typen und Beispielen.
   - Fehlerfälle mit Beispielantworten und Hinweisen zur Behebung.
5. Hinweise zu Paginierung, Sortierung und Filtern, falls verwendet.
6. Hinweise zu Authentisierung oder Autorisierung, falls verwendet.
7. Beispielaufrufe mit curl oder einem vergleichbaren Werkzeug.

6 Datenbankentwurf und ER-Diagramm

1. Das ER-Diagramm enthält Entitäten, Attribute, Kardinalitäten sowie Primär- und Fremdschlüssel.
2. Die Abbildung auf das relationale Schema für PostgreSQL ist zu beschreiben. Datentypen, Indizes und Constraints werden begründet.
3. Optional können Migrationsskripte verwendet werden. Ein Initialisierungsskript ist zulässig.

7 Backend

1. Das Backend wird mit FastAPI umgesetzt.
2. Eingaben sind sinnvoll zu validieren.
3. Die automatisch generierte API Dokumentation unter dem Pfad slash docs ist funktionsfähig und entspricht der Implementierung.

8 Benutzeroberfläche

1. Eine einfache Oberfläche in HTML und JavaScript ruft die Endpunkte auf und stellt die Daten dar.

2. Bei komplexeren Interaktionen darf eine kleine Anwendung mit Vue.js erstellt und gebaut werden. Für die Abgabe werden die gebauten Dateien im Backend Container ausgeliefert oder es wird eine begründete Alternative verwendet.

9 Readme und Projektdokumentation

1. Die Readme beschreibt kurz und klar Zweck und Funktionsumfang des Projekts. Sie enthält eine Architekturübersicht, Startanleitung, Konfigurationshinweise, bekannte Einschränkungen und Beispielaufrufe.
2. Eine Projektdokumentation beschreibt die wichtigsten Designentscheidungen, das ER-Diagramm, die API Spezifikation und die Nutzung der KI. Wichtige Prompts und Antworten werden mit Datum festgehalten.

10 Abgabe und Nachweis

1. GitHub Repository mit vollständiger Projektstruktur.
2. Reproduzierbarer Start der beiden Container. Die Anwendung ist unter Port 8000 erreichbar. Die Datenbank ist unter Port 5433 erreichbar.
3. Die API Dokumentation ist im Repository enthalten und in der laufenden Anwendung überprüfbar.