**Projektdokumentation: Tutorial Management API**

**1. Projektübersicht**

Das Projekt „Tutorial Management API“ ist eine RESTful Web-API zur Verwaltung von Tutorials. Die API ermöglicht das Erstellen, Lesen, Aktualisieren und Löschen (CRUD) von Tutorial-Datensätzen, sowie die Suche nach Tutorials anhand von Titel oder Beschreibung.

Das Projekt basiert auf Java mit dem Spring Boot Framework und verwendet eine relationale In-Memory-Datenbank (H2) über JPA zur Persistenz.

**2. Hauptfunktionen**

* **Tutorials abrufen:** Alle Tutorials oder gefiltert nach Titel oder Beschreibung.
* **Tutorials erstellen:** Neue Tutorials anlegen mit Titel und optionaler Beschreibung.
* **Tutorials aktualisieren:** Vorhandene Tutorials per ID ändern.
* **Tutorials löschen:** Einzelne Tutorials oder alle Tutorials entfernen.
* **Fehlerbehandlung:** Service-Ausnahmen werden abgefangen und mit sinnvollen HTTP-Statuscodes kommuniziert.

**3. Architektur und Technologien**

* **Programmiersprache:** Java 17 (oder höher)
* **Framework:** Spring Boot (Web, Data JPA)
* **Datenbank:** JPA-kompatible relationale DB (z.B. H2 für Entwicklung, MySQL für Produktion)
* **Build-Tool:** Maven oder Gradle
* **API-Dokumentation:** OpenAPI / Swagger
* **Mapping:** MapStruct für DTO <-> Entity Transformation

**4. Paketstruktur**

* **controller:** REST-Controller, welche die HTTP-Endpunkte bereitstellen.
* **service:** Geschäftslogik und Validierung.
* **repository:** Schnittstelle zur Datenbank, nutzt Spring Data JPA.
* **model:** JPA-Entities und Domain-Modelle.
* **dto:** Data Transfer Objects für API-Kommunikation.
* **mapper:** MapStruct-Interfaces zur Umwandlung zwischen Entities und DTOs.
* **config:** Konfiguration, z.B. für OpenAPI/Swagger.
* **resources:** Applikationsressourcen (application.properties etc.)

**5. API-Endpunkte (Auszug)**

| **HTTP-Methode** | **Pfad** | **Beschreibung** | **Response-Code** |
| --- | --- | --- | --- |
| GET | /api/tutorials | Alle Tutorials abrufen | 200, 204 |
| GET | /api/tutorials/{id} | Tutorial nach ID abrufen | 200, 404 |
| POST | /api/tutorials | Neues Tutorial anlegen | 201, 500 |
| PUT | /api/tutorials/{id} | Tutorial aktualisieren | 200, 404, 500 |
| DELETE | /api/tutorials/{id} | Tutorial löschen | 204, 500 |
| DELETE | /api/tutorials | Alle Tutorials löschen | 204, 500 |

**6. Fehlerbehandlung**

Alle Service-Methoden können eine***ServiceException*** werfen, die internen Fehler kapselt. Der Controller fängt diese ab und gibt entsprechende HTTP-Statuscodes zurück, meist ***500 Internal Server Error.***

**7. Konfiguration**

Die API-URLs für verschiedene Umgebungen (Development, Test) sind in der ***application.properties*** konfiguriert. Swagger/OpenAPI wird über ***OpenAPIConfig*** konfiguriert und stellt eine interaktive API-Dokumentation bereit.

**8. Ausführen und Testen**

* Projekt mit Maven bauen:

**mvn clean install**

* Spring Boot Anwendung starten:

**mvn spring-boot:run**

* API-Dokumentation im Browser öffnen:

**http://localhost:8081/swagger-ui.html**

* Tests können über Postman, Curl oder andere REST-Clients durchgeführt werden.

**9. Weiterentwicklungsmöglichkeiten**

* Authentifizierung und Autorisierung (z.B. OAuth2, JWT)
* Paging und Sorting der Tutorials
* Erweiterte Suchfilter
* Erweiterung der Tutorial-Entity (z.B. Autor, Veröffentlichkeitsdatum)
* Verwaltung von Tutorial-Benutzern
* Integration mit Frontend-Anwendung