**Documentația aplicației de Book Management**

**1. Prezentare generală**

Book Management este o aplicație construită cu folosind Java, Spring Boot și MySQL. Aplicația oferă funcționalități pentru gestionarea cărților, autorilor, genurilor literare, oferind endpoint-uri pentru operații CRUD (Create, Read, Update, Delete). Backend-ul a fost implementat în Java, iar pentru documentarea endpoint-urilor API a fost folosit Swagger.

**Caracteristici principale:**

1. Crearea conturilor și autentificarea.

2. Vizualizarea detaliilor și informațiilor cărților.

3. Vizualizarea informațiilor despre autori.

4. Adăugarea cărților în diferite liste personale (to-read, reading, completed).

5. Crearea recenziilor pentru cărți.

6. Evaluarea cărțiilor (de la 1 la 5).

7. Adăugarea și ștergerea cărților.

8. Adăugarea și ștergerea autorilor.

**2. Arhitectura aplicației**

**Componentele principale ale aplicației:**

**- Controllers:** Crează cererile HTTP (GET, POST, PUT, DELETE) și mapează răspunsurile.

**- Services:** Conțin logica de business a aplicației.

**- Repositories:** Interacționează cu baza de date cu ajutorul JPA.

**- Models:** Definesc entitățile.

**3. Instrucțiuni de configurare**

Este nevoie de:

- Java 21 (sau mai recent).

- Maven (pentru gestionarea dependențelor).

- Baza de date MySQL.

**Pașii pentru rularea aplicației:**

1. Descarcă codul sursă sau clonează repository-ul.

2. Configurează conexiunea la baza de date în fișierul application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/book\_management

spring.datasource.username=root

spring.datasource.password=root

3. Construiește proiectul folosind Maven:

mvn clean install

4. Rulează aplicația:

mvn spring-boot:run

5. Accesează Swagger UI la <http://localhost:8080/swagger-ui.html>.

**4. Documentația API**

URL de bază: <http://localhost:8080/api>.

**Endpoint-uri:**

4.1 Obține toate cărțile

* URL: /books
* Metodă: GET
* Descriere: Returnează o listă cu toate cărțile din baza de date.
* Exemplu de răspuns:

[

{

"id": 1,

"title": "Carte A",

"author": "Autor A"

},

{

"id": 2,

"title": "Carte B",

"author": "Autor B"

}

]

4.2 Obține o carte după ID

* URL: /books/{id}
* Metodă: GET
* Descriere: Returnează detalii despre o carte după ID-ul său.
* Parametri: id (Path Variable): ID-ul cărții pe care dorești să o obții.
* Exemplu de răspuns:

{

"id": 1,

"title": "Carte A",

"author": "Autor A"

}

4.3 Adaugă o carte nouă

* URL: /books
* Metodă: POST
* Descriere: Adaugă o carte nouă în baza de date.
* Exemplu de corp al cererii:

{

"title": "Carte C",

"author": "Autor C"

}

* Exemplu de răspuns:

{

"id": 3,

"title": "Carte C",

"author": "Autor C"

}

3.4 Actualizează o carte

* URL: /books/{id}
* Metodă: PUT
* Descriere: Actualizează detaliile unei cărți existente.
* Parametri: id (Path Variable): ID-ul cărții pe care dorești să o actualizezi.
* Exemplu de corp al cererii:

{

"title": "Carte A Actualizată",

"author": "Autor A Actualizat"

}

* Exemplu de răspuns:

{

"id": 1,

"title": "Carte A Actualizată",

"author": "Autor A Actualizat"

}

3.5 Șterge o carte

* URL: /books/{id}
* Metodă: DELETE
* Descriere: Șterge o carte din baza de date după ID-ul său.
* Exemplu de răspuns:

{

"message": "Cartea cu ID-ul 1 a fost ștearsă."

}

**5. Documentația Swagger**

API-ul este documentat folosind Swagger. Accesează Swagger UI pentru a vizualiza și testa API-ul: <http://localhost:8080/swagger-ui.html>.

**Exportarea documentației Swagger:**

- Accesează specificațiile API în format JSON la http://localhost:8080/v2/api-docs.

- Salvează fișierul JSON și integrează-l în instrumente precum Postman sau ReDoc pentru o vizualizare suplimentară.