Kravspecifikation – Bibliotekssystem

Som en inlämning till kursen Java Spring ska man bygga ett bibliotekssystem med ramverket Spring i Java. Man ska följa trelager arkitektur, Repository, Service och Controller, för tydlighet och skalbarhet.

Entiteter

Se den medföljande SQLite-databasen (`MySimpleLibrary.db`) för exakt struktur av:

- books: (book_id, title, publication_year, available_copies, total_copies, author_id)
- authors: (author_id, first_name, last_name, birth_year, nationality)
- users: (user_id, first_name, last_name, email, password, registration_date)
- loans: (loan_id, user_id, book_id, borrowed_date, due_date, returned_date)

G-krav (Grundläggande nivå)

1. Book Management

1. GET /books** - Lista alla böcker
2. GET /books/search** - Sök böcker på title eller author (query parameters)
3. POST /books** - Skapa ny bok

2. Author Management

1. GET /authors** - Lista alla författare
2. GET /authors/name/{lastName}** - Hämta författare via efternamn
3. POST /authors** - Skapa ny författare

4. User Management

1. GET /users/email/{email}** - Hämta användare via email
2. POST /users** - Skapa ny användare

5. Loan Management

1. GET /users/{userId}/loans** - Hämta användarens lån
2. POST /loans** - Låna bok (kräver userId och bookId)
3. PUT /loans/{id}/return** - Returnera bok
4. PUT /loans/{id}/extend** - Förläng lån

6. Service Logic (G)

1. Kontrollera boktillgänglighet vid låning
2. Minska/öka availableCopies vid lån/retur
3. Sätt dueDate till +14 dagar vid låning

7. DTOs (G)

1. Grundläggande DTOs för alla entiteter
2. BookWithDetailsDTO (med Author-info)
3. UserDTO (utan password)

8. Testing (G)

1. Unittest för LoanService.createLoan()
2. Integrationstest för POST /loans

VG-krav (Avancerad nivå)

9. Transactional Management

1. @Transactional** för LoanService.createLoan()
2. @Transactional(readOnly = true)** för read operations

10. Pagination

1. Implementera **Pageable** för GET /books
2. Support för sorting och filtering

11. ResponseEntity

1) Använd **ResponseEntity** för alla endpoints
2) Korrekt HTTP status codes (200, 201, 404, 400)

12. DTO Mapping Patterns

Visa minst 3 olika mappningstekniker:

1) Manuell mappning
2) Builder pattern
3) Stream API för listor

13. Custom Queries

1) **@Query** med JPQL och native SQL
2) Optional return types

14. Exception Handling

1) **@ControllerAdvice** för global exception handling
2) Custom exceptions (BookNotFoundException, etc.)

15. Extended Testing (VG)

1) Mock testing med **@MockBean**
2) Repository tests med **@DataJpaTest**
3) Tester för felscenarier

Teknisk Stack

- Spring Boot 3.x
- Spring Web + Spring Data JPA
- SQLite databas
- Maven