RESTful Bookstore API Project Report

Utkarsh Dubey

July 18, 2025

1 Introduction

This report details the design, development, and implementation of a RESTful Bookstore API as part of the Elevate Labs Java Developer Internship. The primary objective was to build a robust, scalable, and well-documented backend service for managing a bookstore's inventory. The application provides comprehensive CRUD (Create, Read, Update, Delete) functionalities for books and authors, demonstrating modern Java backend development principles and the Spring Boot framework. The project follows a layered architecture (Controller-Service-Repository) to ensure separation of concerns and maintainability. It also incorporates advanced features such as server-side pagination, sorting, and dynamic filtering to handle large datasets efficiently.

2 Abstract

The RESTful Bookstore API is a full-featured backend system developed using Java 17 and Spring Boot 3.3.1. It exposes REST endpoints to manage book and author entities, with data persistence handled by Spring Data JPA and an H2 in-memory database. The API supports complex queries through URL parameters for pagination, sorting, and criteria-based filtering. Interactive API documentation is provided via Swagger UI. A simple static frontend (books.html) demonstrates API consumption, including real-time data fetching and pagination controls. The final application is a production-ready service showcasing core backend development skills.

3 Tools and Technologies Used

4 Steps Involved in Building the Project

The project was developed incrementally, following a logical progression from data modeling to advanced feature implementation:

1. Project Initialization: Bootstrapped using Spring Initializr with dependencies: Spring Web, Spring Data JPA, and H2 Database.

2. Data Modeling: Created Book and Author JPA entities with attributes and a many-to-one relationship.

Category	Tool / Technology	Purpose
Backend Framework	Spring Boot 3.3.1	Core framework for building web applicatio
Language	Java 17	Primary programming language
Data Persistence	Spring Data JPA, Hibernate	Object-relational mapping and database int
Database	H2 In-Memory Database	Fast, zero-setup development and testing
API Documentation	SpringDoc OpenAPI (Swagger 3)	Generate interactive API documentation
Build Tool	Apache Maven	Dependency management and build automa
Web Server	Embedded Tomcat	Default server provided by Spring Boot
Development IDE	Visual Studio Code	Integrated development environment
API Testing	Postman, Swagger UI	Testing and validating REST endpoints

Table 1: Tools and Technologies

3. Repository Layer: Implemented BookRepository and AuthorRepository by extending JpaRepository for CRUD operations.

4. Service Layer: Developed BookServiceImpl and AuthorServiceImpl to handle business logic, data validation, and repository coordination.

5. Controller Layer: Exposed REST endpoints using @RestController in BookController and AuthorController to handle HTTP requests and responses.

6. Frontend Integration: Created a books.html page with HTML, CSS, and JavaScript to consume the API, demonstrating client interaction.

7. Debugging & Troubleshooting:

    - Fixed LazyInitializationException using @Transactional and @JsonManagedReference/@JsonBackReference for JSON serialization.
    - Resolved NoSuchMethodError by correcting Spring Boot parent version in pom.xml.

8. Advanced Querying: Extended repositories with JpaSpecificationExecutor, refactored service and controller layers for dynamic pagination, sorting, and filtering.

9. Database Migration: Migrated from MySQL to H2 in-memory database by updating pom.xml and application.properties.

10. API Documentation: Added springdoc-openapi for Swagger UI, providing comprehensive endpoint documentation.

5 Conclusion

The RESTful Bookstore API was successfully completed, meeting all objectives. The result is a well-structured, documented backend service demonstrating proficiency in Spring Boot and REST API design. The project provided practical experience in resolving backend challenges, such as database relationship management, lazy loading, and dependency conflicts. Advanced features like pagination and dynamic filtering offered insights into building efficient applications. The final API reflects the skills and knowledge gained during the Elevate Labs internship.