Bookstore Microservice Application

Objective

In this programming assignment, you will implement a microservice architecture for a simple bookstore application using Spring Boot. You will create two microservices: one for managing books and another for managing orders. The microservices will communicate with each other via Open Feign and will be registered on Eureka for service discovery and load balancing.

Requirements

Book Management Service

Create a RESTful service with Spring Boot named book-management-service. The service should expose the following RESTful APIs:

- GET /books: Fetches a list of all books
- GET /books/{id}: Fetches details of a book by its ID
- POST /books: Adds a new book
- PUT /books/{id}: Updates details of a book
- DELETE /books/{id}: Deletes a book by its ID

Use any SQL Database to save book information. Each book will have:

- ID
- Title
- Author
- Price

Order Management Service

Create another RESTful service with Spring Boot named order-management-service. The service should expose the following RESTful APIs:

- GET /orders: Fetches a list of all orders
- GET /orders/{id}: Fetches details of an order by its ID
- POST /orders: Creates a new order
- DELETE /orders/{id}: Cancels an order

Orders will have the following fields:

- ID
- Customer Name
- Book ID
- Order Status (New, Processing, Completed, Cancelled)

Use Open Feign to call the book-management-service to fetch book details when creating a new order.

Eureka Service Registry

Create a Eureka Server and register both book-management-service and order-management-service on it for service discovery and load balancing.

Steps to Complete the Assignment

- Create three separate Spring Boot projects: one for Eureka Server, one for book-management-service, and one for order-management-service.
- Implement the functionalities for book-management-service and order-management-service as described in the Requirements section.
- Create a Eureka Service Registry and configure both services to register with Eureka.
- Use Open Feign in order-management-service to call the APIs of book-management-service.

Bonus Points

- Implement API Rate Limiting in either of the services
- Use Spring Boot Actuator to expose operational endpoints (like /health, /info, etc.)

Submission

- Push the code of each Service/ Eureka Server to Github.
- Submit the submission form at https://interviewbit.typeform.com/to/nEApWzsV with the links to the Github repositories.

Happy Coding!