March 1, 2025



MF Library

Project Description

ETOUNDI II Eugene

etoundisebastien@gmail.com

Lyon, France

# Context

The **MF Library** project was born from the need to explore and apply Spring Cloud in a real-world microservices architecture. The objective is to gain hands-on experience in Spring Boot, Spring Cloud, and microservices deployment, while designing a scalable and maintainable application. By implementing industry’s best practices, this project will serve as a practical learning experience in distributed systems, API management, and cloud infrastructure.

# Objective

The goal of MF Library is to create an online platform where users can access a vast collection of free public domain books. The platform will allow users to:

* Search, browse, and read books available in the database.
* Create an account to personalize their experience.
* Rate and review books to provide community-driven recommendations.

This system should be scalable, resilient, and easy to extend, ensuring a seamless experience for users.

# Technical Description

## System Architecture

MF Library follows a microservices architecture using Spring Cloud, allowing independent services to communicate efficiently while ensuring modularity.

[*Deployment diagram*]

* User Service
  + Handles authentication and user profile management.
  + Manages roles and permissions for accessing different features.
* Book Service
  + Stores and retrieves book information (title, author, genre, etc.).
  + Supports book categorization and metadata management.
* Review Service
  + Allows users to leave reviews and ratings for books.
  + Connects user feedback to books for community engagement.
* API Gateway
  + A central entry point for routing requests to appropriate microservices.
  + Manages authentication, rate-limiting, and request forwarding.
* Service Discovery & Configuration (Spring Cloud Eureka & Config Server)
  + Helps in dynamic service registration and automatic discovery.
  + Manages centralized configuration for microservices.

## Technologies & Tools

* Database
  + PostgreSQL is the primary database engine chosen for its robustness and compatibility with Spring Boot.
  + Each microservice will have its own database instance.
* Docker & Kubernetes
  + Cloud Hosting (AWS or Local Kubernetes Cluster)
  + Spring Cloud Config & Eureka Server
  + CI/CD Pipeline (GitHub Actions)
* Security & API Management
  + Spring Security + JWT/OAuth2
  + Rate-limiting in API Gateway
  + Role-Based Access Control (RBAC)

# UML Diagrams

## Deployment diagram

A diagram of a computer

AI-generated content may be incorrect.

Figure 1 Deployment diagram

## Components diagram

A diagram of a service

AI-generated content may be incorrect.

Figure 2 Component diagram

## Flow of a user request

A diagram of a server

AI-generated content may be incorrect.

Figure 3 Flow of a user request

## Domain class diagram

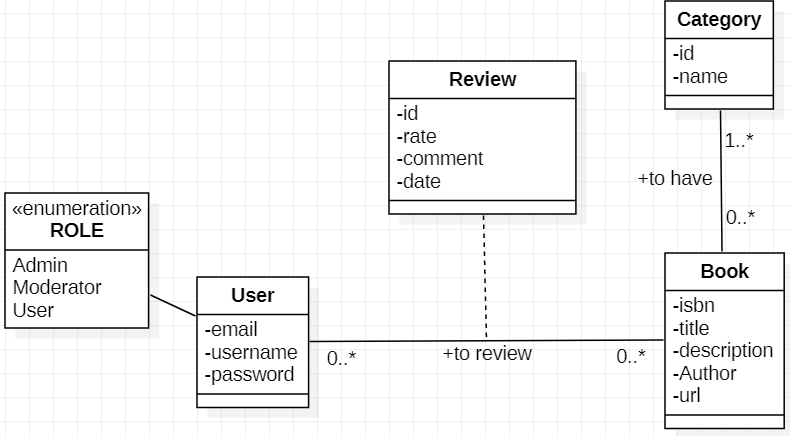


Figure 4 Domain class diagram

# References

* [Best Practices in Spring Boot Project Structure | by Nadeem Khan(NK) | LearnWithNK | Medium](https://medium.com/learnwithnk/best-practices-in-spring-boot-project-structure-layers-of-microservice-versioning-in-api-cadf62bd3459)