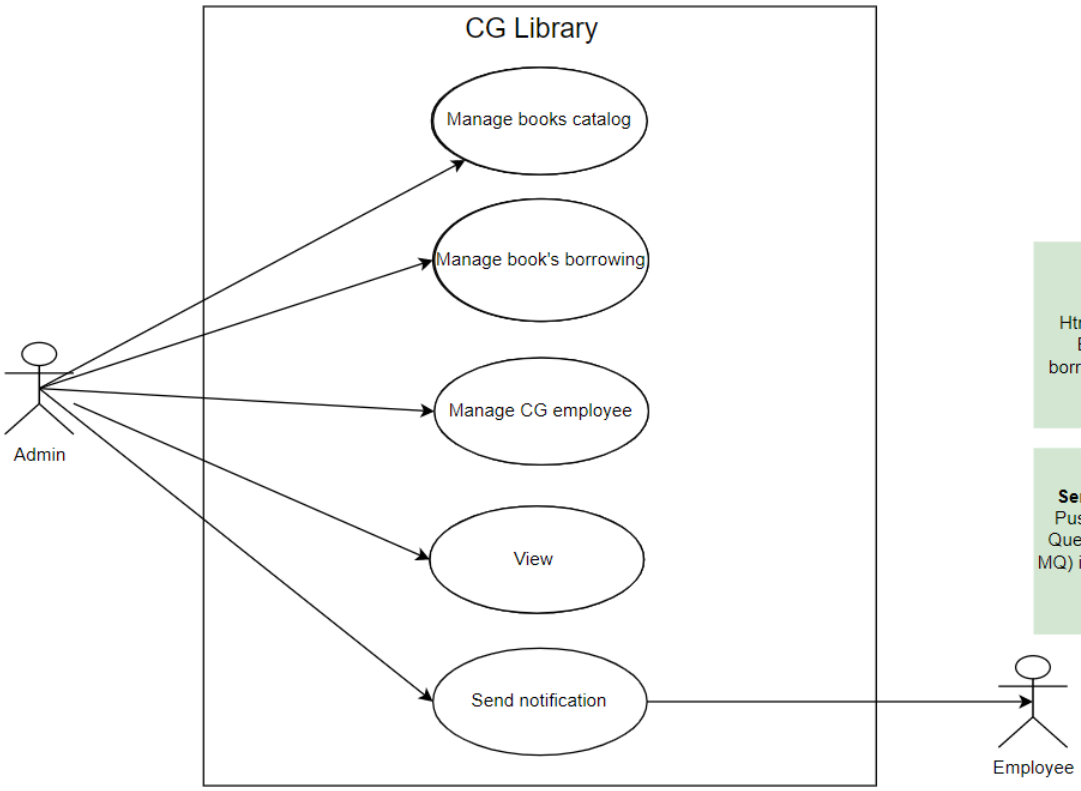
1. Exercise:
    a. What will you have learned from the exercise
        i. New stuff of Java 8 and many more
        ii. Use Spring Boot to build Microservice
        iii. Bring standalone services to Microservice architecture
        iv. Understand how event driven architecture works
        v. Understand how event sourcing works
        vi. How to manage a transaction in a distributed environment.
        vii. Moving services to containers world (Docker) – Advanced level (Optional)
        viii. Orchestrate them using K8s – Advanced level (Optional)
    b. About CGLibrary: It's a library management system. It covers every aspect and every task that librarians, as well as users, might use. Use this application you can do the following things
        i. Manage books catalog.
        ii. Manage book's borrowing.
        iii. Manage CG employees, who will borrow books.
        iv. Send notification to employee when a book was returned, a book's borrowing was added.
    c. Use Case diagram



    d. Class Diagram

e. API Description

i. Book Service API

Functionality	Method	Path
Get book details	GET	/api /v1/books/{bookId}
Add book	POST	/api/v1/books
Update book	PUT	/api/v1/books/{bookId}
Delete book	DELETE	/api/v1/books/{bookId}

ii. Book Borrowing Service API

Functionality	Method	Path
Get book borrowing by employee	GET	/api /v1/borrowing/{employeeId}
Add a new borrowing	POST	/api/v1/borrowing
Update a book return	PUT	/api/v1/borrowing/{employeeId}/{bookId}

iii. Employee Service API

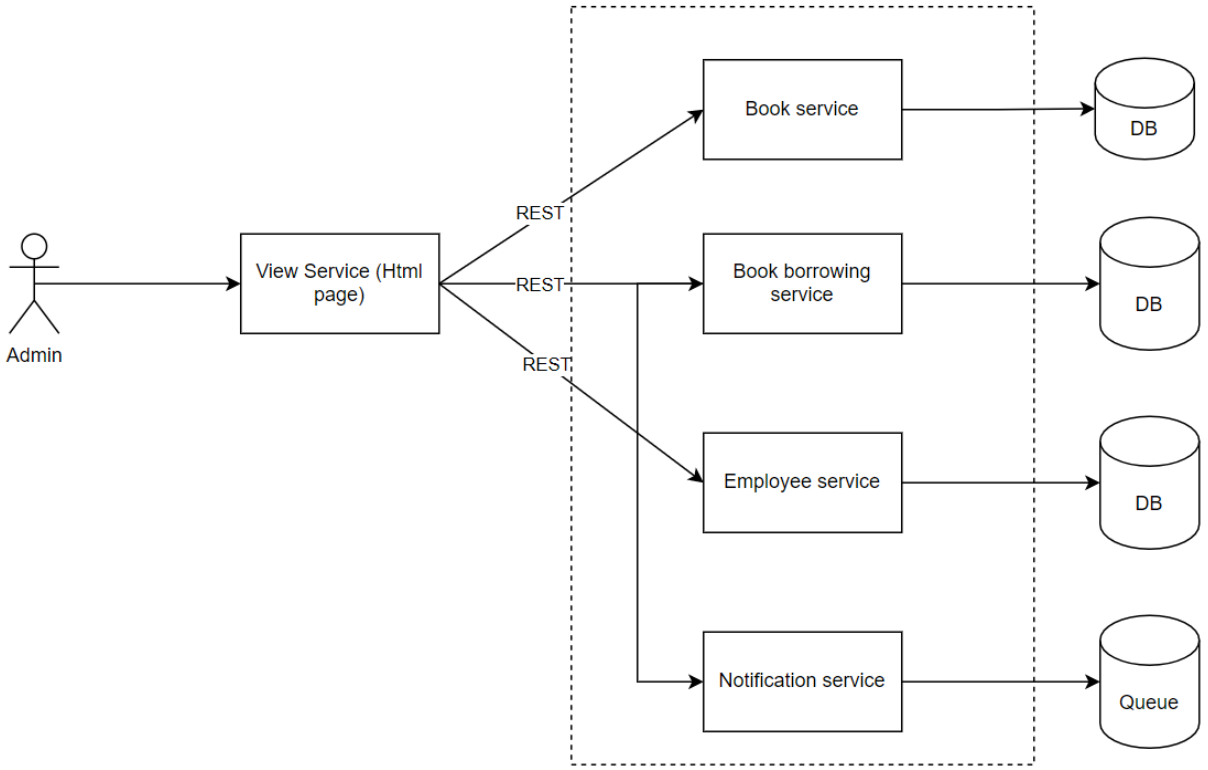
Functionality	Method	Path
Get employee details	GET	/api /v1/employees/{employeeId}
Get borrowed books for employee	GET	/api /v1/employees/{employeeId}/books
Add new employee	POST	/api/v1/employees
Remove employee	DELETE	/api/v1/employees/{employeeId}

iv. Notification Service

Used to send notification to employee once

- Book was returned
- New book's borrowing was added

f. Architecture

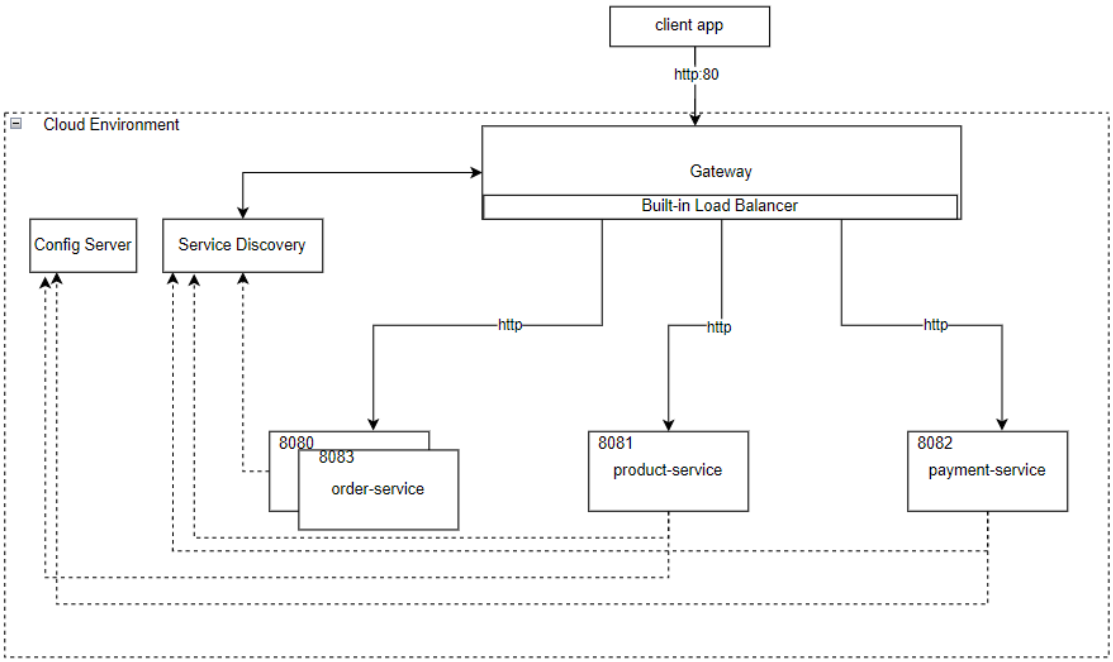


g. Where to apply the design principles?

i. CQRS will be applied for all services except the Notification service.

ii. SAGA should be applied at Borrow booking service (using orchestration-based SAGA). Based on the rules of borrowing a book to determine when a transaction is rolled back (for instance: employee gets disciplined)
    1. Create a record in the Borrowing table.
    2. Update status of book to false
    3. Employee who is not disciplined is valid for borrowing a book

h. References
    i. How to implement CQRS
        1. https://progressivecoder.com/event-sourcing-and-cqrs-with-axon-and-spring-boot-part-1/
        2. https://progressivecoder.com/implementing-event-sourcing-using-axon-and-spring-boot-part-1/
        3. https://progressivecoder.com/implementing-event-sourcing-with-axon-and-spring-boot-part-2//
        4. https://progressivecoder.com/implementing-event-sourcing-with-axon-and-spring-boot-part-3//
    ii. How to implement SAGA (orchestration-based Saga)
        1. https://progressivecoder.com/wp-content/cache/all/saga-pattern-implementation-with-axon-and-spring-boot-part-1/index.html
        2. https://progressivecoder.com/saga-pattern-implementation-axon-spring-boot-part-2/
        3. https://progressivecoder.com/saga-pattern-implementation-axon-spring-boot-part-3/
        4. https://progressivecoder.com/saga-pattern-implementation-with-axon-and-spring-boot-part-4/
i. Minimum Microservice architecture for reference



Additional design principles

1. Distributed tracing
2. Circuit breaker
3. Access token

Design Principles

1. **Service Discovery**
2. **API Gateway**
3. **Centralized configuration**
4. **Database per service**
5. **CQRS**
6. **Event Sourcing**
7. **SAGA**

Technologies stack

Service Discover	Gateway
Netfix Eureka	Spring Cloud API Gateway Netfix Ribbon (built-in Load Balancer)