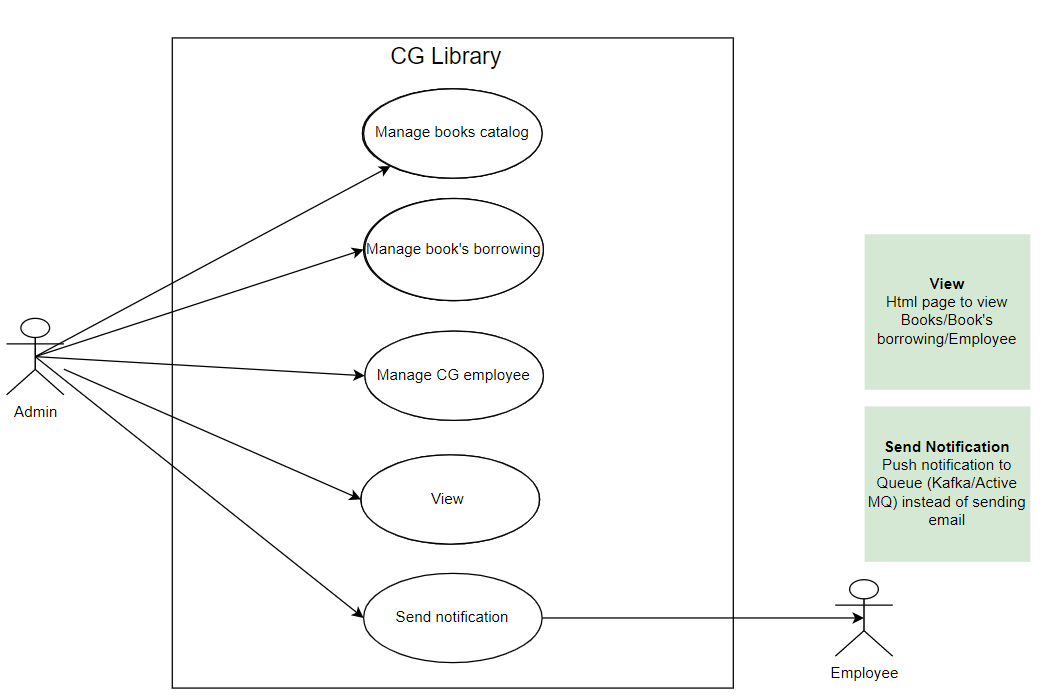
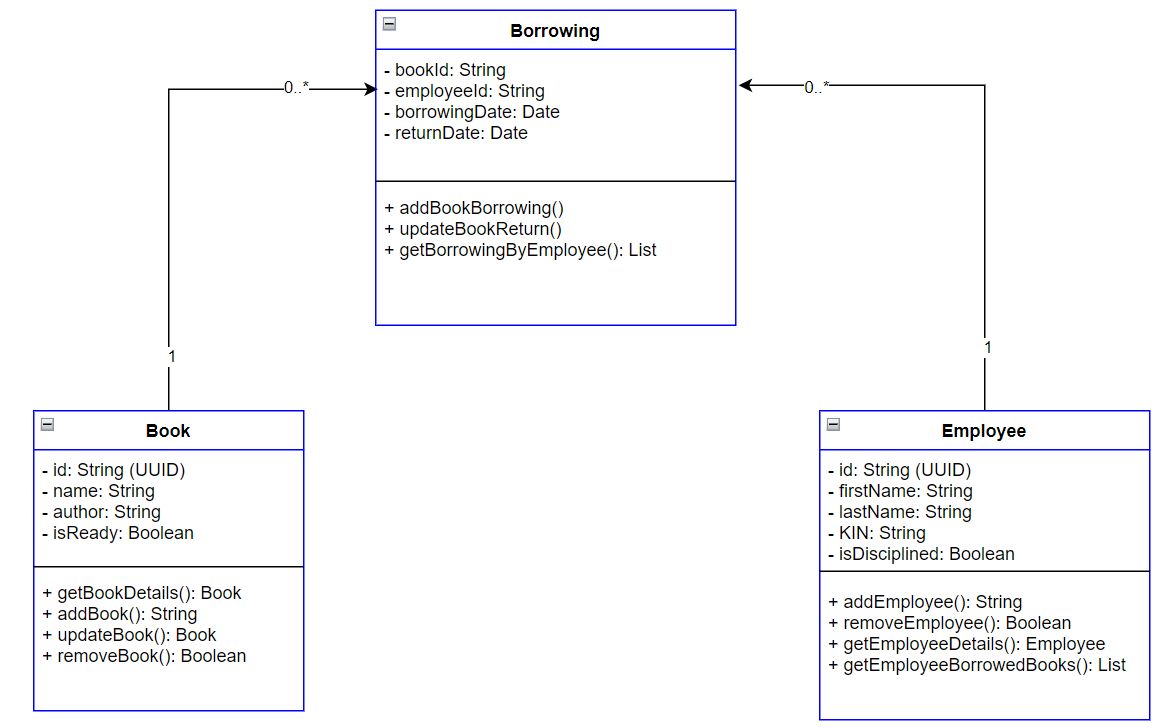
1. Exercise:
   1. What will you have learned from the exercise
      1. New stuff of Java 8 and many more
      2. Use Spring Boot to build Microservice
      3. Bring standalone services to Microservice architecture
      4. Understand about event driven architecture
      5. How to manage a transaction in a distributed environment.
      6. Moving services to containers world (Docker) – Advanced level (Optional)
      7. Orchestrate them using K8s – Advanced level (Optional)
   2. 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
      1. Manage books catalog.
      2. Manage book’s borrowing.
      3. Manage CG employees, who will borrow books.
      4. Send notification to employee when a book was returned, a book’s borrowing was added.
   3. Use Case diagram



* 1. Class Diagram



* 1. API Description
     1. 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} |

* + 1. 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} |

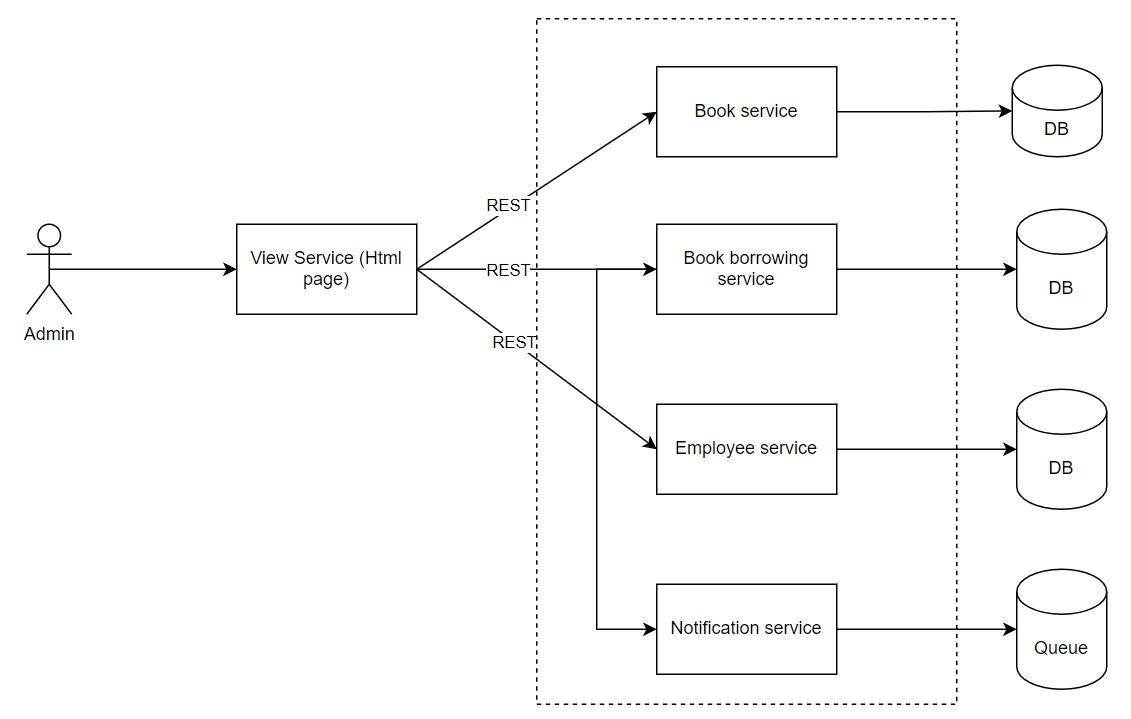
* + 1. Employee Service API

|  |  |  |
| --- | --- | --- |
| 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} |

* + 1. Notification Service

Used to send notification to employee once

* Book was returned
* New book’s borrowing was added
  1. Architecture



* 1. Where to apply the design principles?
     1. CQRS will be applied for Book service (first step), then other services later.
     2. SAGA should be applied for Borrow booking service. 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
  2. References
     1. 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//>
     2. 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/>
  3. Minimum Microservice architecture for reference

