Enterprise Architecture Project

Implement a library application with the following requirements:

Functional requirements:

Book functionality:
We can add, remove, update and find books in the library application. You can search books on scancode, isbn, author name, available copies and title. A book can have multiple authors. We can have multiple copies of a book with the same isbn. Every book copy has an unique scancode.
The system keeps track of how many copies of a book are available in the library.

Customer functionality:
We can add, remove, update and find customers in the library application. You can search customers on customernumber, name and email address. Every customers has an unique customernumber.

Reservation functionality:
A customer can reserve multiple books.
When a customer returns a book that is reserved by another customer, then this customer should receive an email (let the EmailSender just do a System.out.println()).

Borrowing functionality.
A customer can checkout books from the library
A customer can return books to the library
A customer can have a maximum of 4 books at home at the same time (this number is configurable).
Books can be checked out for a maximum of 3 weeks (this number is configurable).
For every day that a book is returned too late, the customer needs to pay a fee of 50 cents per day (this amount is configurable).
The system keeps track of the outstanding (unpaid) fees for customers.
Customers can pay their fees to the librarian. The librarian will enter all payments into the system.

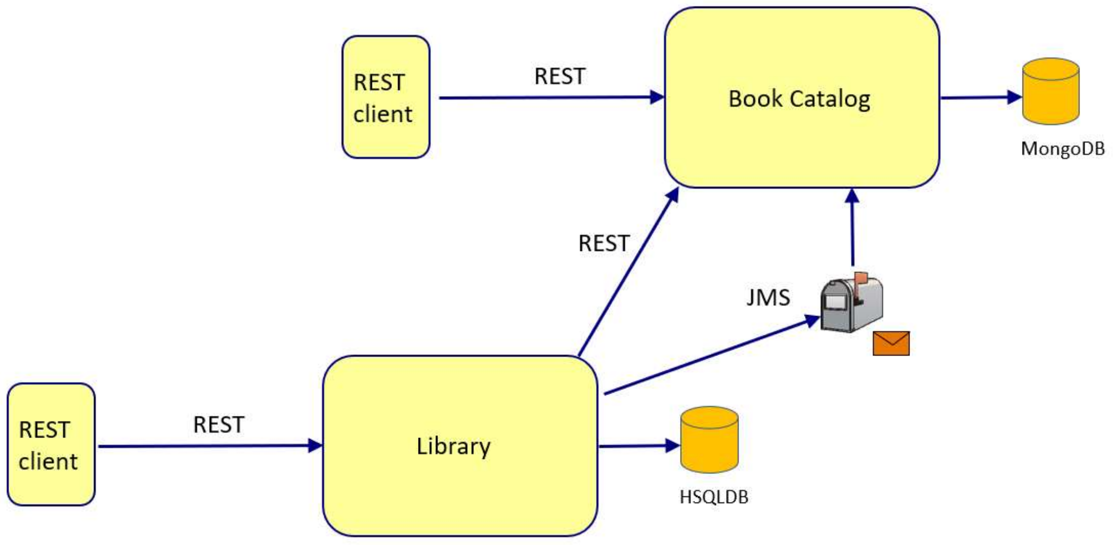
We can retrieve the following information:
- Get all data from a customer(including checked-out books, returned books, current fee balance, payments made)
- Get all books that are currently borrowed and are returned too late.

Architecture requirements

The library system consist of 2 separate applications:
Application 1: Book catalog application
- Contains the book collection with all book functionality described above. This functionality can be accessed through a REST interface. Write a REST client that calls the functionality of this application.
- Uses a Mongo database

Application 2: Library application
- Contains all the other remaining functionality. This functionality can be accessed through a REST interface. Write a REST client that calls the functionality of this application.
- Uses a HSQLDB database
- Use JPA for database access
- Use Spring events to notify a customer that a reserved book has become available.

- Add logging (console and file) to this application at different logging levels.
- This application should be monitored with Grafana.
- Use scheduling so that the application prints every 20 seconds a list of all customers that have an outstanding fee.
- The following data can be configured in *application.properties*:
    - Maximum number of books a customer can have at home at the same time.
    - Maximum number of weeks a book can be checked out
    - The URL to Application 1 (the Book catalog application)
    - The fee that needs to be paid for every day a book is returned to late.
- Use @ConfigurationProperties to read these properties
- All domain logic is tested in unit tests
- All queries in the repositories are tested.
- Borrowing functionality is tested with RestAssured
- Anytime a book is checked out, the library application sends a JMS message using ActiveMQ to the Book catalog application. The Book application keeps track of how many copies are available of a certain book.
- The Book Catalog application keeps the data of all books. It is important that the Library application always calls the Book Catalog application if it needs book data.



Deliverables:

- All code that you write. Your code should be in 4 separate Maven projects (Book catalog, Library and their corresponding REST clients) that can be opened with IntelliJ.
- One or more slides with diagrams (like UML class diagrams) that show the details of your design
- A video of a 20 minutes presentation of your application.

Presentation

- Minimal 20 minutes, maximal 30 minutes
- Your presentation should contain the following topics
  - Clearly explain your design using diagrams
  - Demo of all the important aspects of your code
    - Show running your rest client
    - Show the content of the database
    - Show that a message is sent
    - Show running you different tests

This project is done individually. You can discuss with your fellow students, but all your demo code and material needs to be created by yourself.

The due date for this project is **Thursday at 9:45 AM** (projects that are submitted later will lose points). On Thursday some students may present their project in class. During the project (Tuesday and Wednesday) you don't need to come to class.

On Thursday everyone needs to come to class.

How to submit your project?

The project needs to be submitted in the assignment section is sakai. You can submit the code and the slides, but you cannot upload your video (size is too large). Upload your video to google drive, make it **available to everyone**, and submit the link as part of your assignment. **It is important that the link is available to everyone.**

The grading of this project is based on:

- The quality of your design
- The quality of your code
- The quality of the presentation.
- Did you implement all requirements?