Book Library Application

Cognifide is creating a book library management platform. The goal of the project is to enable libraries to share information about their book collections with customers. We want the system to be backed by a WEB API that will provide several endpoints defined below. Your task is to create a Java application that will provide those endpoints and serve the relevant information based on data read from a JSON file.

List of tasks below will help you plan your work and verify the completeness of your project.

Each task consists of parts. Parts marked with **the tile icon are optional** so you can skip part of a task in case of problems. Please let us know if you have any additional questions.

Optional tasks will give you some bonus points and are divided into 2 levels of advancement:

◩ - ambitious but less challenging to implement

◈ - very ambitious and more challenging to implement

And… if you would like to impress us even more - our next 2 superheroes will share with you some details regarding our **bonus tasks**!

In the provided .zip file you will be able to find additional information regarding application requirements and architecture as well as other files that will help you in creating the application.

Tasks

☐ **Build tools, environment and code structure**

Setup code structure. Prepare tools for build and deployment of your application.

Choose and set up the code structure and build tools that will allow to build the application, deploy it as well as run tests using one command. The data being processed by the application should be stored in memory and can be lost after stopping it. You can choose either, **Maven 3.3 or newer, Gradle 4.5 or newer** as a build tool. Choose **JDK 8 or newer**.

☐ **Code sharing**

You need a way to share your code with us. There are a couple ways to do that.

You can share your code using a zip file (and sending it to us via email) or you can use a web-based hosting service such as www.github.com. In this case setup **private repository** and share link to it with us. Remember that for a private repository you need to specify Github users that will have access to it (please share it with our recruitment account - username: **cognifiderecruitment**).

You can choose one or implement both optional tasks below:

During your work with code use Git - a Version Control System. Store all your code on the master branch. All commits are made to this one branch.

Store only stable code on a master branch. Store the code you are working on, on feature branches. After the work on a feature branch is finished, you can merge it to the master branch.
If you are interested in how it works in real life, you may read more about Git workflows.

☐ **Frameworks and libraries**

Sometimes choosing the right tool to do the job can save you a lot of time and effort. Below are examples of libraries you may want to check out before creating the first line of code.

☐ **Application frameworks**

You can choose **one** of the following options:

Most popular and easy to learn is Spring or Java EE.

A more ambitious approach is to use reactive programming framework, e.g Vert.x or WebFlux.

☐ **Testing libraries**

Choose just one, two or even all of them:

- Use JUnit 4 or JUnit 5.
- Use Mockito as a mocking library.
- Use RestAssured to test your endpoints.

Bonus tech stack task

You may also use frameworks and libraries of your own choice and justify (in a few sentences) your decision and benefits of using those solutions. Find more details about where to include your justification in the “Documentation” section.

☐ **Data**

In a .zip file, you can find “books.json” which represents the data you can work with. Read the data from a JSON file.

Choose just one, two or even all of the following optional tasks:

- Allow switching the dataset from the command line.
- Fetch the data using Google Books APIs.
- To serialize/deserialize JSON you can use e.g. GSON or Jackson.

☐ **Book details endpoint**

Expose the endpoint which allows reading details about a book. It will return a book identified by the given ISBN number in the form of a JSON document or return a 404 if the book does not exist in the data set.

Choose just one, two or even all of the following optional tasks:

- Make the endpoint to respond with data from the provided dataset.
- Cover the code with tests.
- Design RESTFul convention for endpoint address.

☐ **Books category endpoint**

Similarly to the last point create an endpoint that will list all books that are assigned to the requested category (empty list if no books belong to the category).

Choose just one, two or even all of the following tasks:

- Make the endpoint to respond with data from the provided dataset.
- Cover the code with tests.
- Design RESTFul convention for endpoint address.

Bonus full-stack task

Use the Books by category endpoint you provided and create a simple dashboard that will enable its user to choose a category (e.g. using a simple select) to display all books from the chosen category. You can find an example mockup of such dashboard in the attached requirements file.

☐ **Rating endpoint**

Lists all authors and their rating in descending order of the average rating of their books. If a book is not rated, it should not be taken into account in the calculation of its author rating.

Implement just one, two, three or even all of the following tasks:

- Make the endpoint to respond with data from the provided dataset.
- Implement an algorithm that will calculate the author's rating. Remember that books without rating should NOT be taken into account when calculating rating. The cleaner and simpler solution, the better.
- Cover the code with tests.
- Design RESTFul convention for endpoint address.

☐ **Code quality**

We value clean code. It should be pleasant to read and have structure easy to understand. If you are interested in object oriented design, take a look at SOLID principles.

Choose one or all of the following options.

Use static analysis tool or plugin (e.g. sonarlint) to diagnose the code. Fix reported issues.

Use at least one Design Pattern and explain why you have used it.

☐ **Documentation**

To make it easier for you to describe how to run the program and for us to understand how to run it we prepared a template for the setup instructions we expect you to deliver.

The Homework app can't require any other 3rd party software to be installed manually (e.g. manual setup of the Tomcat Server or MySQL database). However, you may use any libraries or software you need to do this homework. Just remember about the rule above.

Provide README.md file which contains:

- Short description of the application in general.
- Description of the application framework you have used. If you have used different tech stack than proposed, justify your choices here.
- Description of the testing framework used. If you used a different framework than proposed - include your justification here.
- Design Patterns used and why they were used.
- Instructions on how to run your application.

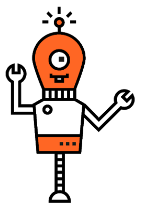
Additional option: provide documentation of the created APIs. It should be possible to generate documentation using one of the documentation generators. For example Javadoc.

What we will look at

We want to get to know better your coding skills and approach to solving algorithmic problems.

We will pay attention to and rate:

- code structure and naming conventions,
- design patterns and algorithmic solutions,
- code readability,
- presence of unit tests,
- libraries and tools used to solve the problems (right tools for the right tasks),
- project completeness (if all requirements are fulfilled),
- solution complexity (the simpler the better).



And what's most important… Good luck and have fun!

If in doubt, make sure you read the requirements carefully and be creative. Explain your decisions. In case you become stuck at any point of solving this homework - please feel free to let our team know via email. We will be more than happy to help you. :)

Java Recruitment Team