JAVA 2 - Project documentation from
**Valerii Drobiazgho**

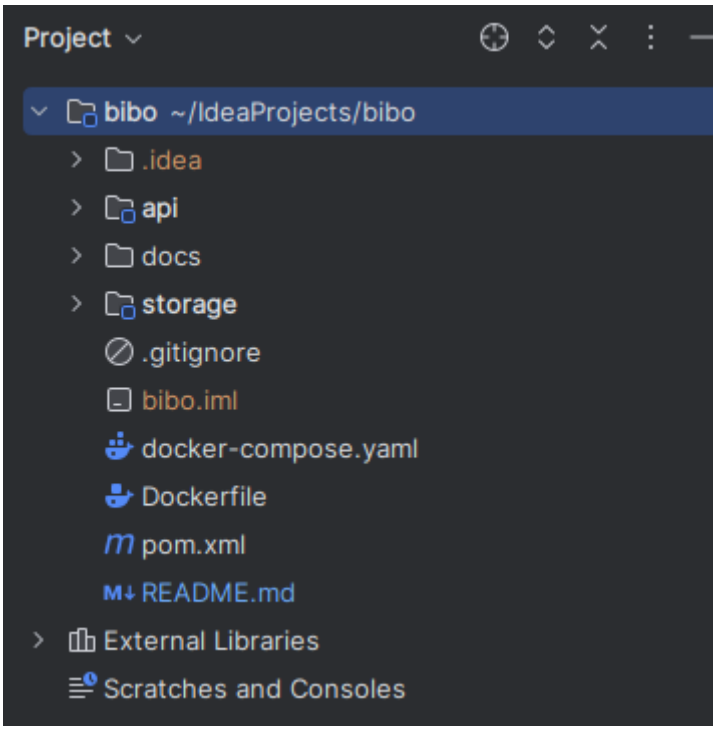
Projectidea

The idea of the project is to create a library management system - a software application designed to manage the work of a online library. The system allows users (hereinafter referred to as "Persons") to search for books, borrow them and return them, and in the future, it is possible to implement such functions as the ability for librarians to manage the inventory of books, track borrowed books and process user accounts.

The main idea, why a library

- I wanted to learn how a micro service approach would be based on the library
- Learning how the interaction works in a lending context
- What happens approximately when a book is borrowed from a library

The project tree:



There are two modules. The first module is called api and the second is called storage.

API:
All interactions with the outside world are managed in these modules. This includes the remaining endpoints and, in the MVC context, the controllers.

STORAGE:
All entities that are managed and used by Spring Boot and Co are managed in these modules. This also includes interfaces for the repository and its data access.

There are three entities. The first is the book, which contains all the necessary information about a book that can be borrowed. The second is the person. This holds further data about a person. The third entity is intended as an intermediate entity, which stores the borrowing in a separate table. This entity contains the ID of both previously considered entities. This is used internally as the FK (Foreign Key).

The “docs” folder is used to manage documents related to this project.

What do what, a short overview:

- **API Module**:
  - Rest Client
  - Main entrypoint for the java application
  - FunctionHandler (MVC)
  - *UnitTest about controller (broken)*
- **Storage Module**:
  - Data Entity Storage for Spring Boot
  - Low level logic
  - UnitTest about entity

Used tools during the project:

- IntelliJ (IDE)
- Debian VM (Linux)
- Spring Boot
- Draw.io
- Swagger Ui (localhost:8080/(swagger-ui/index.html)
- Maven (BuildTool)
- Curl and Postmann as Request/Rest UnitTesting
- Containerized (Docker and DockerCompose)
- Git ( git lab fh-erfurt)

Design pattern decision:

- security (auth tokens, bearer tokens etc not spectated regarding the bad time management )
- h2 memory datenbank for fast deploy without volumes

To be noted regarding to run the application over the IDE (IntelliJ) :

Problem

- UnitTests runnable for every UnitTest class, not the hole project map
- RestUnitTest only manually posible (PostMann or CURL or **localhost:8080/swagger-ui/index.html (recommended)** )

Lessons Learned

- docker multi module java build
    - multi module can get a hole of dependency problems
    - together with spring boot ( application scanning and this )
- different solutions in the internet (not matter what the problem fix, but it fix and for specific java version)
- different solutions about java in the ground
- BuildTool (Maven) and this procedure
- Spring Boot ( Rest Service) learned in a small case
- Time management is very important, to many working hours outside the the context backfire bade result for me
- time pressure (extern work)
- unit test with java and spring boot regarding multi maven project not clear for me, cant fixed them in this available time
- error messages not even helpful (example, plugin error)
- some times, i have the impression, that what i do the problem fixed but i dont know why

**HOW TO  Run**

Simple start over docker-compose

- navigate to root project (strcuture ./bibo)
- make clear, you installed the dependencies (docker and docker-compose and use it as root or fake root)
- native with docker run the command:
    - create the docker image
        - run “sudo docker build –tag bibo . “
    - run “sudo docker run -p 8080:8080 -t bibo-api –network=host”
- with docker compose
    - run “docker-compose up”