TODO List - Katell Maro

Prioritization of the features

First, I write my TODO List for the project and I prioritized the tasks:

1. Set-up of the environment

Before beginning all the other tasks, I had to install the tools required for the project (TOMCAT, a JDK, an IDE, a database, JSTL, Postman). The requirements of the project is to use TOMCAT ⅞ with Java ⅞, I chose to use version 8 of both. It took one day to install and set-up all these tools.

2. Persistence of data

The first priority is the persistence of data. A todo list is useless if users cannot retrieve their tasks. I begin with this point because the implementation of the application is different if there is or not a database.

3. View a task list
4. Add, remove task
5. Check, uncheck task

Then, I choose to develop the main features of a TODO list : get all the tasks, add a task, remove a task and check/uncheck a task. I planned to develop only the back-end for these features and to test them with Postman.

6. Simple connection

After the development of the main features, I planned to create a simple connection service. Users can open a session with only a login (without a password). Users can only see their tasks. For that feature, I developed only the back-end and tested it with Postman. At this point, the back-end of the application is functional.

7. Interface

The interface will be simple, with two screens, one for the connection and one for the TODO List.

8. Tests for previous features

I planned to write tests for the previous features. Unit tests are not really relevant for the creation, reading, delete and check of tasks, because there is no processing in these methods, there is just access to the database. Instead, I would have developed integration tests. These tests would have allowed me to check if the methods retrieve or update the data in the database correctly.

9. Secured connection with a password

Then, I would have developed a secured connection with an encrypted password. The back-end and the front-end would have been developed at the same time.

10. Errors management

A user-friendly app has to manage the common errors (bad password, not unique login for creation of account, technical error ...).

11. Tests for previous features

I would have written integration tests for the securised connection and unit tests for errors management.

12. Add last updated time and a description for task

This feature is not really high-priority because it is not a key feature of a TODO list. To have the last updated time and the description is a bonus. I would have developed the back-end, front-end and the tests for this feature.

13. Improvement of the interface

The last task is the improvement of the interface, more user-friendly and pretty.

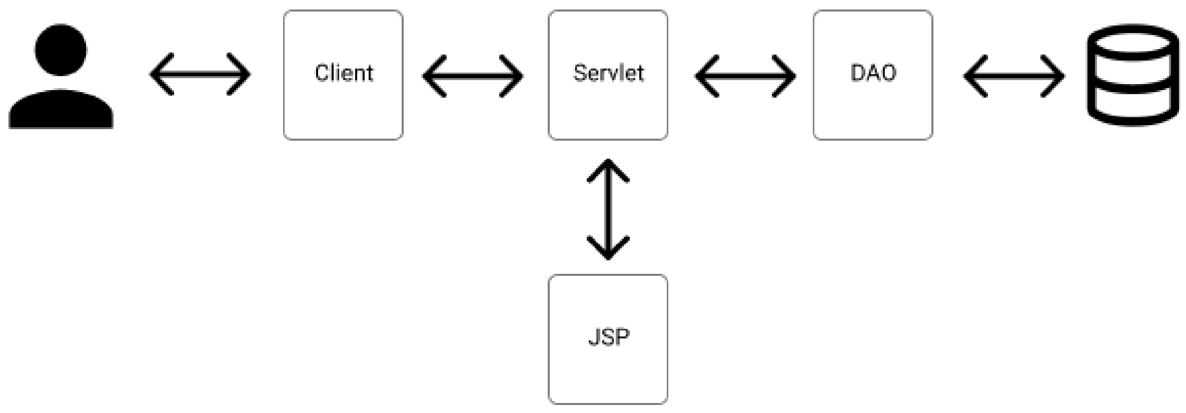
I had the time to develop the 7 first tasks.

Technical choices

Tools

I chose to use the ORM JPA to simplify the interaction with the database. With the annotations in JPA it is easier to create database tables and more maintainable.

Organisation



- I used a DAO to communicate with the database because I prefered to separate the responsibility between servlets and the DAO. The DAO has the responsibility to access the database and the servlets have the responsibility of the requests from the client.
- I used JSP for the front-end because it is easier for the first version of the application. But I planned to use Angular to improve the interface because Angular allows me to create a dynamic interface.