1 - Set up and make sure that the communications work correctly
2 - Front end
3 - Back end

Final Exam - Mock

Software Development and Integration

Instructions

Your reasoning and problem solving skills will be evaluated as much as the final solution. Read the entire exam carefully before beginning so you can choose the right order in which to solve the different exercises.

You have 110 minutes to solve this exam.

General guidelines

1. Access the following Github classroom url: https://classroom.github.com/a/Erl61-cz
2. Follow the required steps to create the repository and clone it.
3. You should work in the local repository. **You must perform a commit at least every 15 minutes.**
4. Once you have finished, you must:
    1. **Make a final commit and tag it with a release v1.0**
    2. Compress the entire project in a zip file and upload it onto Canvas
    3. **You must include the generated .git folder in order for your exam to be evaluated**
    4. Make sure that you delete the */target* folder, to make sure that the solution has an acceptable size
5. You will also include a *references.txt* folder, in which you will include any reference used throughout. This includes but is not limited to: previous labs from class, Stackoverflow, ChatGPT, etc. You must include the entire url for the reference to be evaluated and accepted.

Instructions

Develop a web application that allows a teacher to manage all their students in a class. This involves adding new students to the roster, as well as displaying previously existing students. The teacher will be able to export the entire student roster as a CSV file at any point.

Using the Vaadin framework, you will develop an application containing:

- A form which allows the teacher to add a new student, containing:
  - First name
  - Last name
  - Date of birth
  - Gender (Male or Female)
  - UUID (auto-generated, you may use external libraries for this)

- A table or grid that lets the teacher visualize all information pertaining the student roster. When a student is added, it will be refreshed automatically with the latest information.

Using the *opencsv* library (specifically, the CSVWriter class), you will allow the teacher to export all the information to as a CSV file: https://www.geeksforgeeks.org/writing-a-csv-file-in-java-using-opencsv/

When adding a new student to the roster, you must display a notification on the screen with a successful message.

Communication with the backend will take place using the *HttpRequest* library. The backend will provide a RESTful API which will receive the requests with the student data and be in charge of creating the CSV file export.

All students registered will be saved in a JSON file which will be persisted to disk. It will be used as a database whenever the application is launched.

Front end requirements

1. The Java version shall be Amazon Corretto 17.
2. You will use the **Vaadin** framework version 24.
3. All dependencies will be managed using **Maven**.
4. To create the Vaadin application, use the initializer located at: https://vaadin.com/hello-world-starters (choose Spring boot).
5. All code used to manage json files will be through the *gson* library seen in class.
   1. GroupID: com.google.code.gson
   2. Artifact ID: gson
   3. Version: 2.6.2
6. **Define the GroupID of the project as “es.ufv.dis.mock” and the Artifact ID with your initials.**
7. The UI will contain the form allowing for the input of the fields required to create a student. Extra marks for a nice design.

Back end requirements

The RESTful API will receive the requests from the front end application. It will respond to 3 requests: a **POST** request to create a new student, and a **GET** request which will return all existing students in the JSON file. Finally, a **POST** request which will generate the CSV export, save it in the backend app under the **exports** folder. Extra credits if you are able to make the front end download the generated CSV file from the backend.

You must create the backend API using the Spring Initializer: https://start.spring.io/

Unit Tests

You must write unit tests using the JUnit v4 library. You must at least cover tests for creating a new student in the backend.

Deliverables

1. The folder with both projects and the *.git* folder in a zip file.
2. CSV exports generated in the **exports** folder of the backend project.
3. The *references.txt* document.

Evaluation Criteria

Front end: 40%, back end: 40%, tests and git: 20%.

Specifically:

- Are you able to correctly implement the API methods?
- Do you use git properly?
- Do you write the required unit tests?
- Do you develop the front end app using Vaadin as requested?
- Do you use the Java HttpRequests library as required?
- Do you reason through your solution properly?
- Do you generate the required JSON and CSV files?