Web Application

Guilherme Franco <xfranc01@stud.fit.vutbr.cz>
Jakub Wachowicz <262392@stud.fit.vutbr.cz>
Sandis Studers <xsandi01@stud.fit.vutbr.cz>
Louis Rives-Lehtinen <xrives00@stud.fit.vutbr.cz>

January 2024

1 Introduction

The aim of the project is to develop a web application that efficiently manages a complex course with diverse activities, tasks, and team projects. It includes features for task assignment, progress tracking, team project evaluation, assessment communication, export functionality, and load calculation, providing a comprehensive tool for course administration and evaluation.

2 An overview of the project

In this section, we will provide some background on the technologies used.

2.1 Spring, Maven

The project was implemented in Java using the Spring framework with Maven. Insights from a Spring lecture highlighted the appropriateness of this technology for our project, owing to its robust ecosystem and versatility for Web Applications. The Spring framework, renowned for its comprehensive set of tools and libraries, facilitated the development process by providing a well-organized and modular architecture.

Maven's dependency management system streamlined the integration of external libraries, making it easier to handle updates and maintain consistency across the project. The standardized project structure enforced by

Maven also enhanced collaboration among team members, as everyone could easily understand and navigate the codebase.

2.2 MySQL

Choosing MySQL as our database management system provides several advantages for our project. MySQL is a widely used relational database that offers robust features, strong performance, and scalability. Its compatibility with Spring's dependencies and Hibernate, ensures a smooth and efficient integration process. Hibernate simplifies the interaction with the database by providing a convenient abstraction layer over SQL, allowing us to work with objects rather than raw database tables.

The decision to maintain the database locally enhances ease of manipulation during development. This local setup allows developers to iterate rapidly, test different scenarios, and troubleshoot issues effectively. Moreover, working with a local database facilitates a more controlled environment for debugging and ensures that changes made during development don't impact the production database.

2.3 Java Persistence API

In our project, we leveraged the Java Persistence API (JPA) to interact with the MySQL database seamlessly. JPA is a Java specification that provides a standard way to manage relational data in Java applications. It simplifies database access by allowing developers to work with objects in their Java code, abstracting away the details of the underlying database operations.

- Entities (e.g., Course, Activity, Task, Criterion, ...):

  Each of these classes is annotated with JPA annotations such as @Entity, @Table, @Id, and @Column. These annotations define the mapping between the Java objects and the corresponding database tables and columns. JPA annotations help specify primary keys, column names, relationships between entities, and other metadata crucial for database operations.

- Repositories (e.g., CourseRepository, ActivityRepository, TaskRepository, CriterionRepository, ...):

These repositories extend the JpaRepository interface provided by Spring Data JPA. This interface exposes a set of generic methods for performing common database operations (e.g., save, findById, delete). By extending JpaRepository, our repositories inherit powerful querying methods, eliminating the need for boilerplate code for basic CRUD (Create, Read, Update, Delete) operations.

- Service Classes (e.g., CourseService, ActivityService, TaskService, CriterionService, ...):

  The service classes encapsulate business logic and orchestrate interactions with the database through JPA repositories. JPA's integration with Spring allows these services to be injected into other components, promoting modularity and separation of concerns.

- Custom Query Methods:

  The repositories feature custom query methods derived from method names. Spring Data JPA interprets these method names and generates the appropriate SQL queries, reducing the need for manual query creation. For example, methods like findByCourseId in ActivityRepository and findByActivityId in TaskRepository are custom query methods that fetch records based on specific criteria.

3 Results

Our project represents the successful implementation of a decent web application, showcasing fully functional pages and a seamless user experience. The challenges faced during team collaboration, particularly in merging our individual contributions, were met with effective problem-solving strategies, underscoring our team's resilience and adaptability.

Notably, the project's technological stack, centered around Spring, Maven, JPA, and Hibernate, provided a rich learning experience. The decision to employ Java and the Spring framework not only facilitated the development process but also ensured a modular and scalable architecture. Maven's role in dependency management streamlined integration and maintenance, contributing to a well-organized project structure

In conclusion, our project's success is attributed not only to the functional web application it delivers but also to the thoughtful selection and effective utilization of technologies. The collaborative effort, combined with the chosen stack and development practices, has resulted in a project that is not only functional but also well-architected, laying the foundation for future enhancements and expansions.

4 Conclusions and future work

4.1 Problems with the current solution

The main issue with our solution is that the database is locally situated, which does notmake it viable in a real-life situation. This localized database poses significant challenges in terms of scalability, accessibility, and data redundancy. In a real-world scenario where the application may need to cater to a larger user base or be deployed across multiple locations, relying on a local database becomes a bottlene The limited scalability of a locally situated database hinders the system's ability to handle increased data loads, concurrent user interactions, and growing storage requirements. This limitation could result in performance issues, slower response times, and an overall degraded user experience as the demand on the system intensifies.

Another area for enhancement in our current solution is the front-end design. While the existing interface is decent, there is room for improvement to make it even more user-friendly and visually appealing. User interfaces are integral to the overall user experience, and we aim to refine the front-end to ensure it not only meets but exceeds user expectations.

4.2 Future work

To overcome the limitations posed by the locally situated database, our immediate future work involves evaluating and implementing a more scalable and distributed database solution. This migration aims to improve scalability, enhance accessibility, and establish a robust data redundancy strategy. The adoption of cloud-based databases or distributed database systems will be explored to better support the growing needs of the application, ensuring seamless performance even in scenarios with increased data loads and concurrent user interactions.

Building upon the recognition of potential improvements in the front-end design, future work will focus on a comprehensive refinement of the user interface. Collaborating with UX designers and conducting user feedback sessions will be integral to this process. The goal is not only to address existing usability concerns but also to implement design enhancements that elevate the visual appeal of the interface. By incorporating modern design principles and responsive layouts, we aim to create a front-end that not only meets but exceeds user expectations, contributing to an overall positive and engaging user experience.

5 References

Guilherme Franco:
Jakub Wachowicz :
Sandis Studers :
Louis Rives-Lehtinen :

- Spring's official tutorial pages :

  https://spring.io/guides/gs/accessing-data-mysql/ https://spring.io/guides/gs/accessing-data-jpa/

- Baeldung.com tutorials about spring, jpa, hibernate:

  https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa https://www.baeldung jpa-hibernate

- MySQL official tutorial page:

  https://dev.mysql.com/doc/refman/8.0/en/tutorial.html

- Spring Boot Registration and Login with MySQL Database Tutorial:

  https://www.codejava.net/frameworks/spring-boot/user-registration-and-login-tutorial

- Spring Security Tutorial with Login Example:

  https://www.youtube.com/watch?v=tDZPdovCH4I

A Task and effort distribution

- Guilherme Franco was responsible for the implementation of the controllers and HTML pages for the Home page, Course Page, Criterion Page, Evaluation Page, Every Form Page (excluding Solver,workload entitites and their management) as well as minor Service and Repository class additons in most entities in order to fetch specific data (25% of effort).

- Jakub Wachowicz was responsible for Importing and Exporitng functionality. Part of the Entities and their's Controllers and Services - mostly Student and Team. Minor additions to database and HTML. (25% of effort).

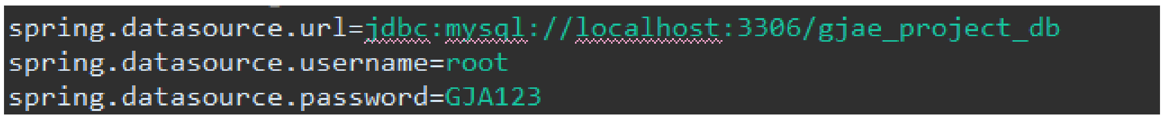
- Sandis Studers was responsible for setup of Java classes for most of the entities and their corresponding Controller-Service-Repository classes. Implementation of CRUD operations for all services. HTML pages and underlying logic for workload and solver entities and their management. Minor other functional, html, css additions. (25% of effort).

- Louis Rives-Lehtinen was responsible for database research, report writing, and programming the login, "users" related functions, pages' style sheet (css). Minor other functional methods, html additions. (25% of effort).

B Documentation

1. Download MySQL and import the database.sql file (you can copy past the line in the shell).

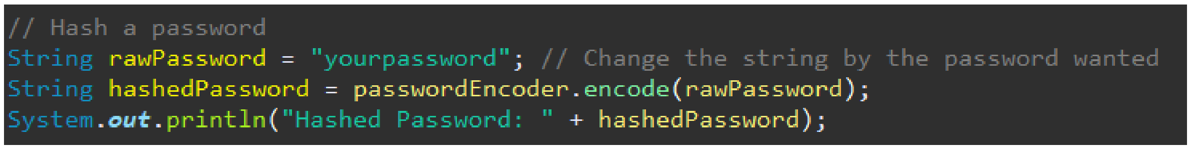
2. Access to the following file : src/main/resources/application.properties and modify the database configuration. Replace the username and password by the ones you configured during the MySQL installation. fig1



```
spring.datasource.url=jdbc:mysql://localhost:3306/gjae_project_db
spring.datasource.username=root
spring.datasource.password=GJA123
```

Figure 1

3. Create an USER in the MySQL shell, the command should be like this : INSERT INTO Users (username, email, password, name, role) VALUES ('xtest', 'test@gmail.com', 'hashedpassword', 'test', 'ADMIN''); To get the hashed version of your password, you need to modify the rawPassword in the MainProjectApplication.java class. The hashed version will be printed in the IDE's cmd when running the application.



```
// Hash a password
String rawPassword = "yourpassword"; // Change the string by the password wanted
String hashedPassword = passwordEncoder.encode(rawPassword);
System.out.println("Hashed Password: " + hashedPassword);
```

Figure 2

4. Run the application, get to the http://localhost:8080/login page and you will be able to login using the username and the password. Depending on your role 'ADMIN' or 'TEACHER', you will be redirected to either the admin or teacher page. Both of them will provide a link to the home page.