Advance Java Project : MyQuiz

Technical Specification Document

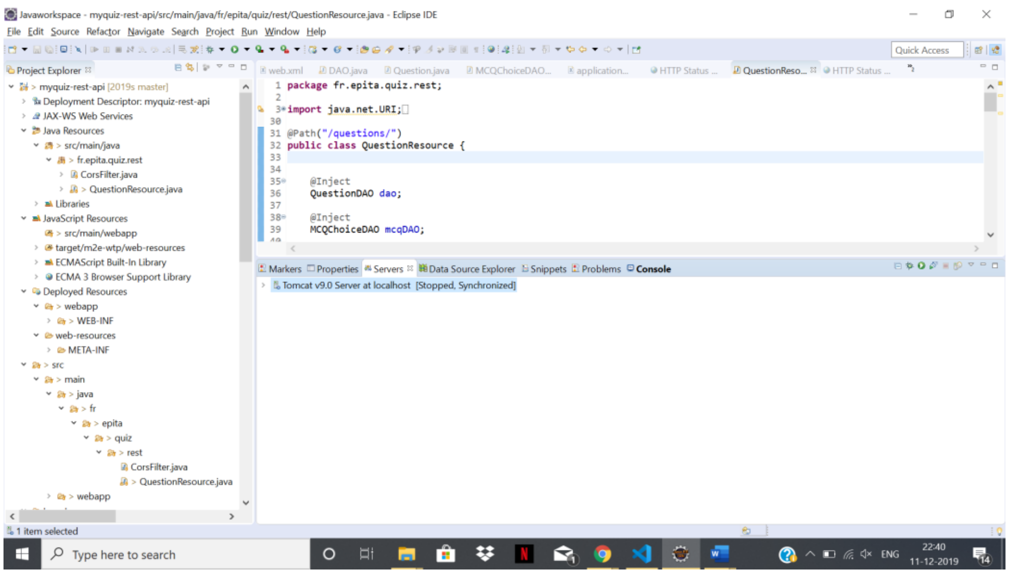
-Indhu ARIVALAGAN

Technical Description:

The purpose of this project is to create an application for the management of quiz preparation and execution using Angular 8 (API, web-based, oriented).

Introduction:

This project is a web-based application that helps perform CRUD operations on open questions and MCQ – preparation and execution using Angular 8.This document is prepared for the Advanced Java Project by Indhu ARIVALAGAN for her computer science master’s program at L’École Pour l'Informatique et les Techniques Avancées (EPITA).



Javaworkspace - myquiz-rest-api/src/main/java/fr/epita/quiz/rest/QuestionResource.java - Eclipse IDE

```
package fr.epita.quiz.rest;

import java.net.URI;

@Path("/questions/")
public class QuestionResource {


    @Inject
    QuestionDAO dao;

    @Inject
    MCQChoiceDAO mcqDAO;
```

Tomcat v9.0 Server at localhost [Stopped, Synchronized]

Project Overview:

There are two modules in this project: the admin module and the student module. In the student module, the user can only attend the quiz, whereas in the admin module, the administrator can add questions, delete questions, update questions, search questions, search / display list of questions, attend a quiz, evaluate the quiz.

Project Scope:

- Automatically assemble quizzes using open questions, Boolean and multiple choices.
- Auto-grading for multiple-choice questions.
- Creation of data access objects using CRUD Methods.
- Create a configuration file for the application's properties.
- The API is made with an Angular 8.

Software Requirements:

Technologies Used:

BACKEND/WEBSERVER	Java 8, Spring-Hibernate, Apache Tomcat 9, JPA
FRONTEND	Angular 8
DATABASE	H2
VERSION CONTROL	GIT

Project Dependencies:

- Install Java 8
- Install H2 JDBC
- Install Spring-Hibernate
- Angular 8
- Create table in database

Database:

This project uses the H2 database And the values that are sent through the angular are stored in the database.

Acronyms and Abbreviations:

ACRONYMS	ABBREVIATIONS
CRUD	(Create, Read, Update, Delete) → Relational database application functions.
DAO	(Data Access Object) → Service class to communicate with the H2 database.
JPA	(Java Persistence API) → is a Java application programming interface specification that describes the management of relational data in applications using Java Platform.
MCQ	(Multiple Choice Question) → Questions with possible responses listed in our implementation using radial buttons.
JDBC	(Java Database Connectivity) → Application Programming Interface(API) for Java Programming Language.

Project Requirements:

1. Create a quiz based on the requirements of the user.
2. Use questions stored in database (using h2 database).
3. Allows the user to take the quiz.
4. At the end of the assessment, correct open & MCQ questions and display results.
5. Export the quiz in a plain text format.
6. Use CRUD operations on a question.
7. Display the output using Angular 8.

System Specification:

API's:

Generated APIs to connect from front end to backend are listed as follows,

Add Question→http://localhost:8080/myquiz-rest-api/rest/questions/addQuestion

Add Options→http://localhost:8080/myquiz-rest-api/rest/questions/options

Delete Question→http://localhost:8080/myquiz-rest-api/rest/questions/delete/

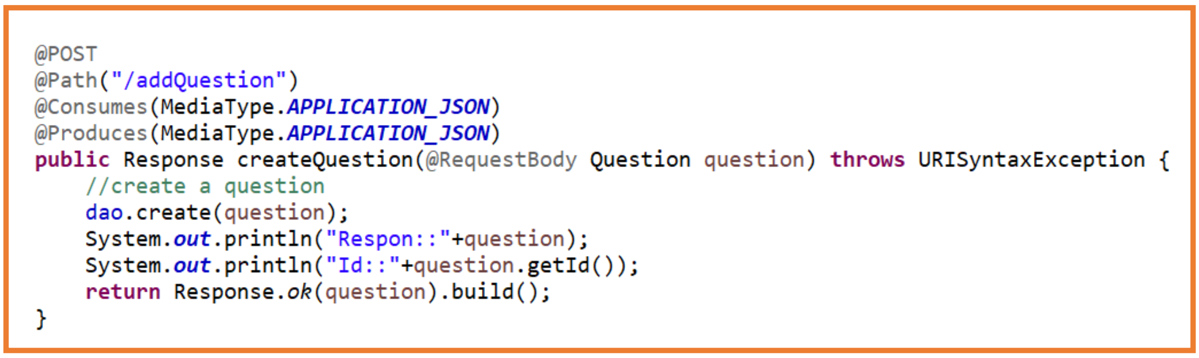
Update Question→http://localhost:8080/myquiz-rest-api/rest/questions/update/

Search Question→http://localhost:8080/myquiz-rest-api/rest/questions/search?qContent=

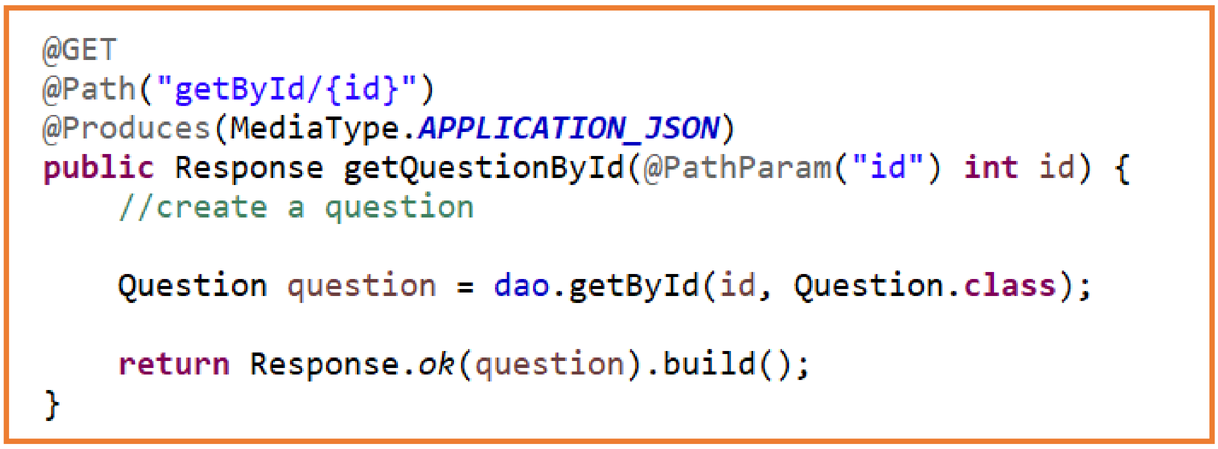
List of Questions→http://localhost:8080/myquiz-rest-api/rest/questions/allQuestions

Functional Requirements:

- Can be able to Add Questions connecting to the database(h2), i.e., The questions already used and stored in the database can be used by users.



```
@POST
@Path("/addQuestion")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response createQuestion(@RequestBody Question question) throws URISyntaxException {
    //create a question
    dao.create(question);
    System.out.println("Respon::"+question);
    System.out.println("Id::"+question.getId());
    return Response.ok(question).build();
}
```



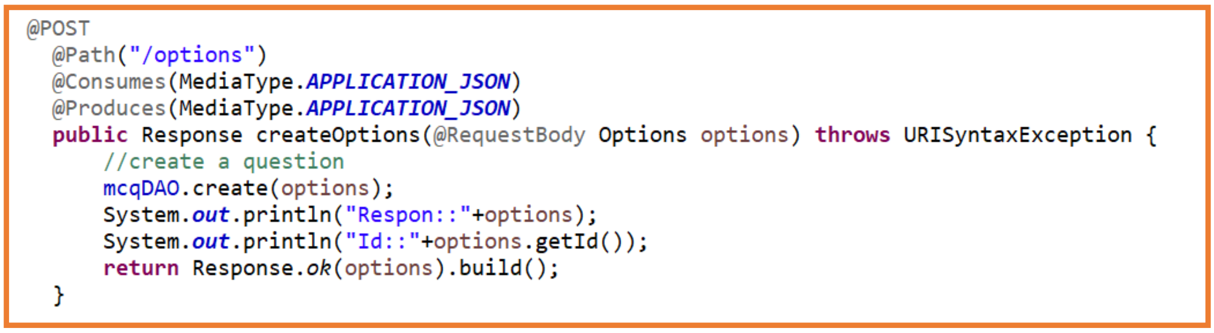
```
@GET
@Path("getById/{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response getQuestionById(@PathParam("id") int id) {
    //create a question

    Question question = dao.getById(id, Question.class);

    return Response.ok(question).build();
}
```

API For ADD QUESTION → http://localhost:8080/myquiz-rest-api/rest/questions/addQuestion

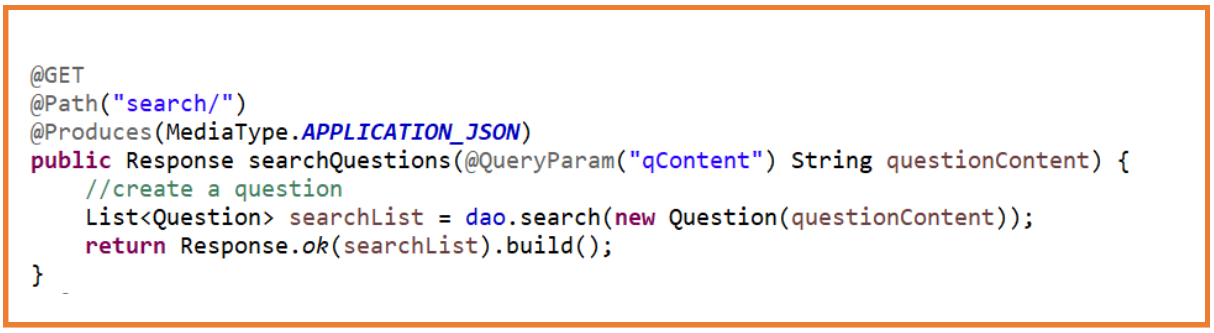
- Can be able to Options connecting to the database(h2)



```
@POST
  @Path("/options")
  @Consumes(MediaType.APPLICATION_JSON)
  @Produces(MediaType.APPLICATION_JSON)
  public Response createOptions(@RequestBody Options options) throws URISyntaxException {
      //create a question
      mcqDAO.create(options);
      System.out.println("Respon::"+options);
      System.out.println("Id::"+options.getId());
      return Response.ok(options).build();
  }
```

API For ADD OPTIONS → http://localhost:8080/myquiz-rest-api/rest/questions/options

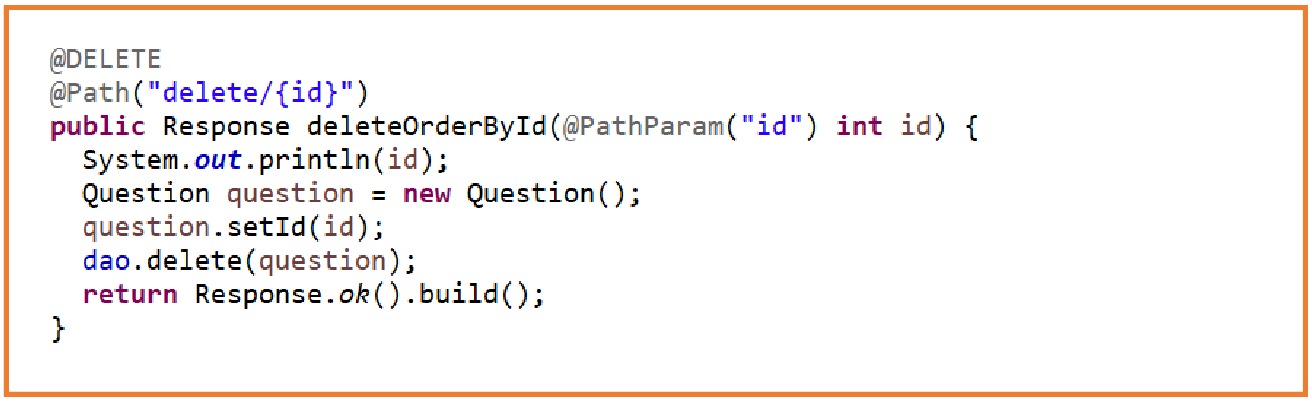
- Can be able to Search Questions connecting to the database(h2)



```java
@GET
@Path("search/")
@Produces(MediaType.APPLICATION_JSON)
public Response searchQuestions(@QueryParam("qContent") String questionContent) {
    //create a question
    List<Question> searchList = dao.search(new Question(questionContent));
    return Response.ok(searchList).build();
}
```

API For SEARCH QUESTION →http://localhost:8080/myquiz-rest- api/rest/questions/search?qContent=

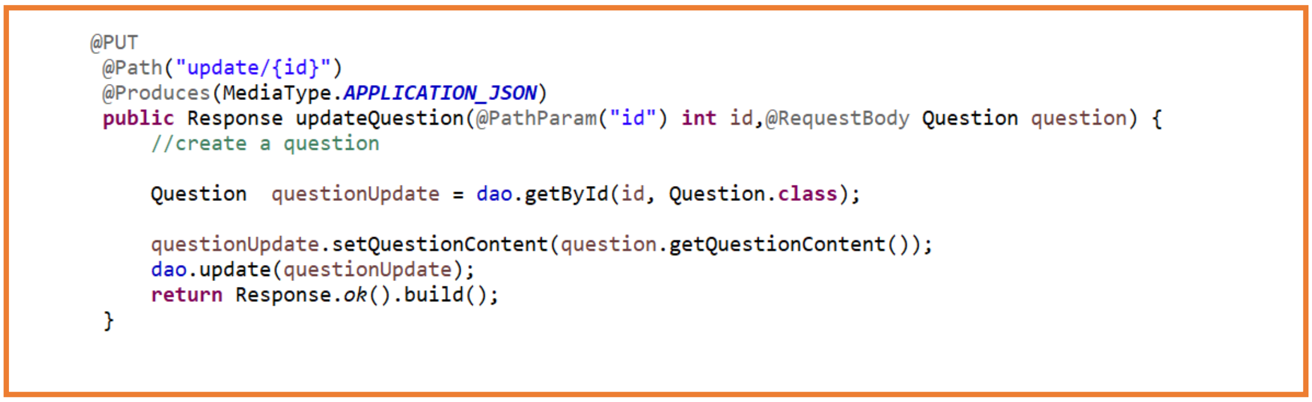
- Can be able to Delete Questions connecting to the database(h2)



```java
@DELETE
@Path("delete/{id}")
public Response deleteOrderById(@PathParam("id") int id) {
  System.out.println(id);
  Question question = new Question();
  question.setId(id);
  dao.delete(question);
  return Response.ok().build();
}
```

API For DELETE QUESTION → http://localhost:8080/myquiz-rest-api/rest/questions/delete

- Can be able to Update Questions connecting to the database(h2)



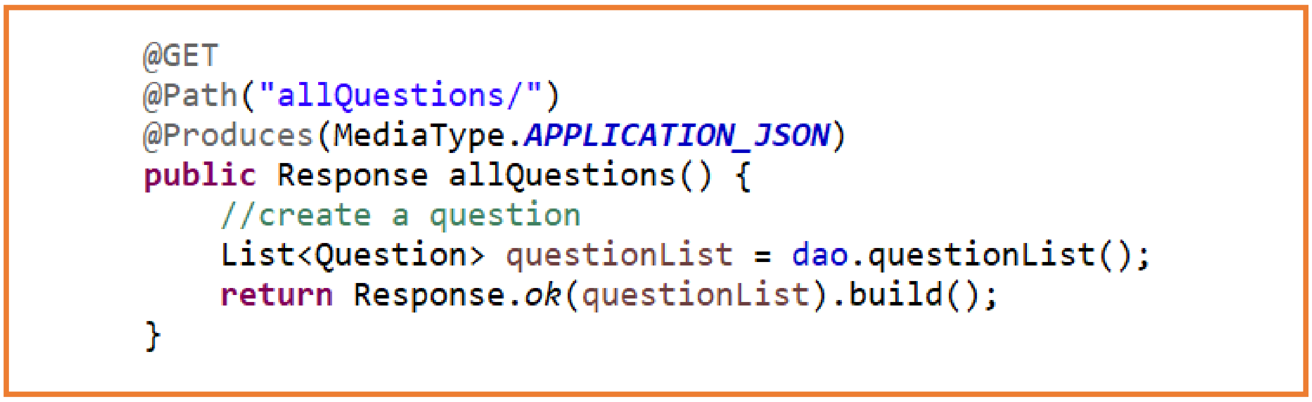
```
@PUT
@Path("update/{id}")
@Produces(MediaType.APPLICATION_JSON)
public Response updateQuestion(@PathParam("id") int id,@RequestBody Question question) {
    //create a question

    Question  questionUpdate = dao.getById(id, Question.class);

    questionUpdate.setQuestionContent(question.getQuestionContent());
    dao.update(questionUpdate);
    return Response.ok().build();
}
```

API For UPDATE QUESTION→ http://localhost:8080/myquiz-rest-api/rest/questions/update

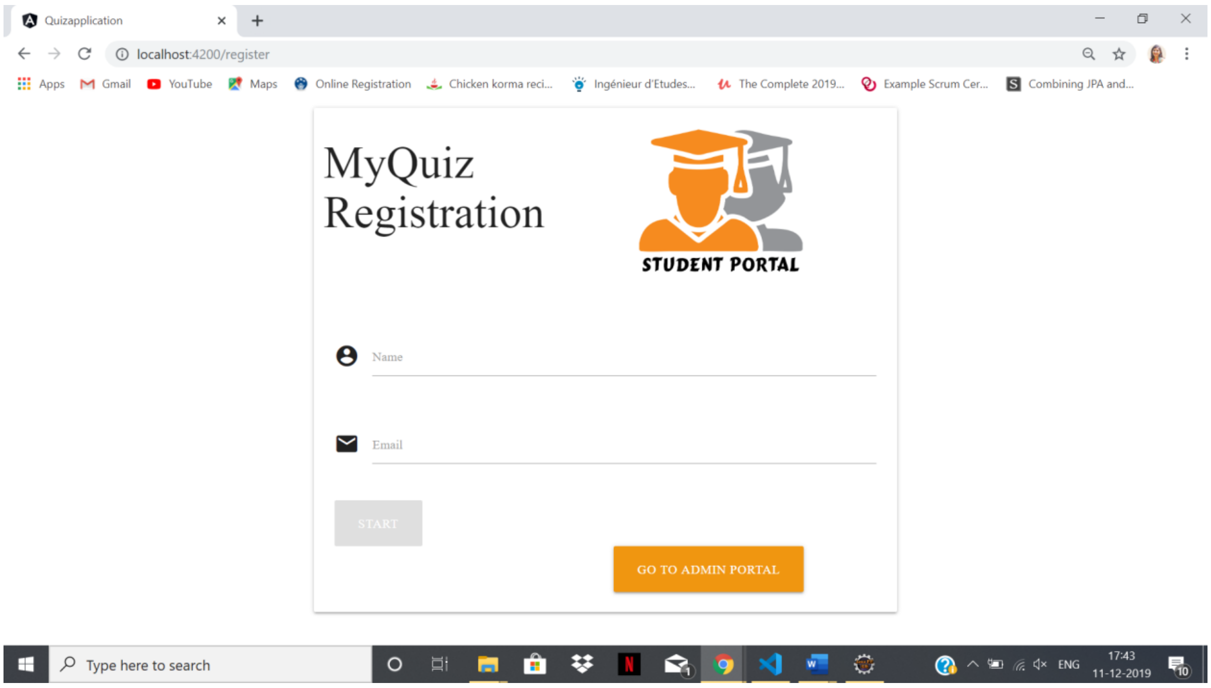
- Can be able to List Questions in the database(h2)



```
@GET
@Path("allQuestions/")
@Produces(MediaType.APPLICATION_JSON)
public Response allQuestions() {
    //create a question
    List<Question> questionList = dao.questionList();
    return Response.ok(questionList).build();
}
```

API For LIST QUESTION→ http://localhost:8080/myquiz-rest-api/rest/questions/allQuestions

Welcome Page Looks Like this!



User Interface Design:

Refer to the user guide.

Hardware Interfaces

- ✓ Operating System : Windows xp or more, MAC or UNIX
- ✓ Processor : Pentium 3.0 GHz or higher
- ✓ RAM : 256 MB or more
- ✓ Hard Disk : 10 GB or more

Bibliography:

https://thomas-broussard.fr/work/java/courses/project/advanced.xhtml