WEB application: the QUIZ MANAGER Project

# Team members:

Stefano Acosta, Alvaro Bilbao.

# Course:

AdvancedJava.

Subject description

The general objective of this project is to deliver an Quiz manager software, capable of providing an end user the possibility to manage quizzes creation, questions creations and simulate them in an information system (create, persist and manage them) through a friendly Graphical User Interface (GUI) which is based as a front-end client. The project is based on: Java as the back-end coding language and Typescript-HTML5-CSS for the front-end development, DerbyDB as the database, Tomcat as the server, Maven-Spring-JUnit-Hibernate-Jersey-Angular as frameworks, GIT as a version control tool, and Eclipse Java EE IDE as development tool.

The specific objectives are:

Access, create and modify quiz and questions information.

Persist quiz and questions data in a database.

Be robust, capable of good performance.

Propose a simple but efficient user interface.

# Subject analysis

## Major features

* User authentication
  + Provide means to create a user with at least a unique username and a password.
  + Allow any user who has a valid account to login into the Quiz management system.
* Quiz Management
  + Provide services to create a Quizzes and save them in the database
  + Provide the user with the option to manage existing Quizzes
    - Update
    - Delete
    - Create new
* Question Management
  + Provide services to create Questions and save them in the database
  + Provide the user with the option to manage existing Questions
    - Update
    - Delete
    - Create new
* GUI
  + Provide a friendly and intuitive graphical interface, which allows the user to authenticate and use the software in an easy way.

Application Feasibility

The application is well defined, we know beforehand the type of questions allowed for the quiz creation, and in this case the questions will only be MCQ (multiple choice). The database structure, and the interaction between objects is defined inside the application, thanks to the ORM framework.

The only unknown factor for the application is the actual data that will be entered by the users of the application.

Expected results

We expect that the user is able to perform the following operations over Quizzes and Questions:

1. Create
2. Read
3. Update
4. Delete

And he can also perform this operations on users:

1. Create
2. Login

And all these changes should be preserved in a Database.

Scope of the application (limits, evolutions)

The application is a web (java backend – front typescript/Angular) application, it is limited to perform only one action at any given moment per user, this means that a single user cannot perform more than one quiz or question creation at a given time. If the user wants to create multiple questions or quizzes, he must create them one by one, the same happens for the update and delete capabilities.

The user authentication is weak, it can be improved by JWT in angular as well as using encryption modules.

Conception

Data structures

For handling the users and identities, we have defined our own classes that represent them as shown in the data description schema.

To save the data, we are using DAO (Data access object), we have a DAO for: users, questions, quizzes, MCQuestions, userquiz.

A DAO is a data type that provides an interface to persist data into a database, without exposing the database itself, it provides methods for:

* Create (save something in the database)
* Read (search for the data that exists in the database)
* Update (modify something that already existed in the database)
* Delete (remove something from the database)

The Identities DAO provide functions to perform the 4 operations (or more in certain cases, which are combinations).

This DAO methods make use of other data structures like Lists and Sets.

To communicate with the database, we use a ORM framework, which relies on HQL, Tomcat and a derbyClient driver provided by oracle.

Global schema and major features schema

The application consists of 2 projects, web api exposure and core.

The schema for the project organization is the following.

