QueueUnderflow

STUDENT NAME: Radu-Todor Alexandru

GROUP: 30433

# CONTENTS

[1. Introduction 3](#_Toc97840780)

[2. Diagrams 3](#_Toc97840781)

[3. Routing 5](#_Toc97840780)

4. Features…………………………………………………………………………………………………………………………………………..6

# Introduction

This is a project aiming to design a copy of the StackOverflow platform, developed for the laboratory of the Software Design course of my university. On this platform, the user (which can be a normal user or a moderator) will be able to ask questions or answer on questions from other users, edit his posts or upvote/downvote posts from other users. This program is a web application, so it has a frontend, a backend, and a database.

The database is written in MySQL and is developed using MySQL Workbench 8.0 CE. The project includes a file “QueueUnderflowDatabase.sql” which contains a basic database for testing purposes.

This backend is written in Java Spring and is developed using JetBrains IntelliJ IDEA. The backend is designed using layered architecture, and the layers are explained in the Package Diagram part of the documentation.

The frontend is written using React JavaScript and is developed using JetBrains WebStorm IDEA.

# Diagrams

1. Use Case Diagram

A picture containing graphical user interface

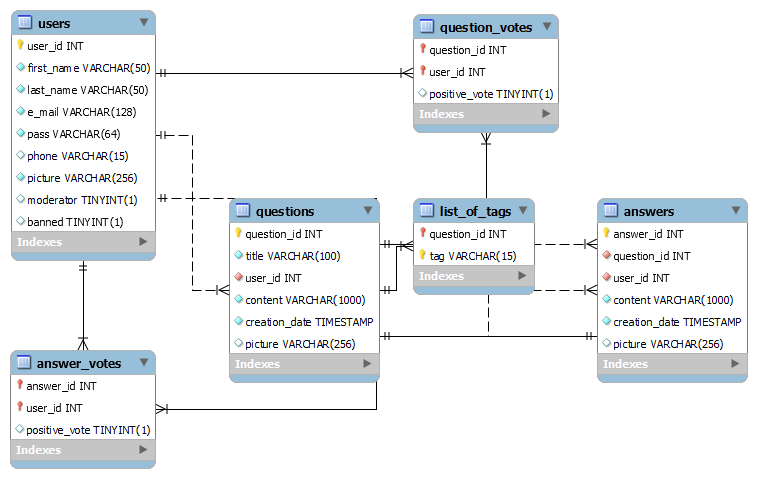
Description automatically generated

The Use Case diagram presents the actions that can be performed by each type of user.

We have two types of users: a normal user, and a moderator. The normal user, after logging in, can post a new question, answer a question from another user, upvote or downvote a post, and modify (or even delete) one of his posts.

The moderator is a special type of user. It can do the same things a normal user can, and along with that, he can also modify (or even delete) any post that is inappropriate and ban users from the site in case of bad behavior.

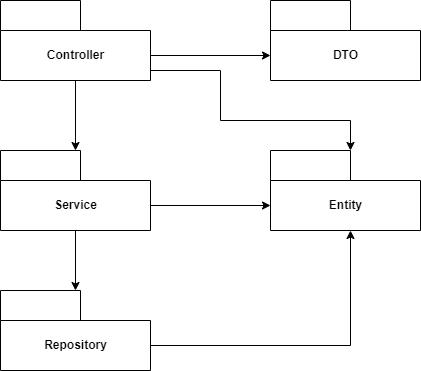
1. Database Diagram



The database has a total of 5 tables. The first table is “users” and is used to save data about the users, and the table “votes” saves information about the votes on each question/answer, to be precise, who voted and the type of vote.

The next table I want to talk about is “posts”. It contains common information about the questions and answers and is in a one-to-many relationship with “votes”. “Posts” with “votes” together can be considered a table for answers. The table for questions would be formed out of “posts” and “votes”, combined with “list\_of\_tags” (one-to-many relationship) and “list\_of\_answers” (one-to-many relationship). These two tables are the only difference in the database between answers and questions, along with the field “question” from “post” which tells if a question or not (is a answer).

1. Package Diagram

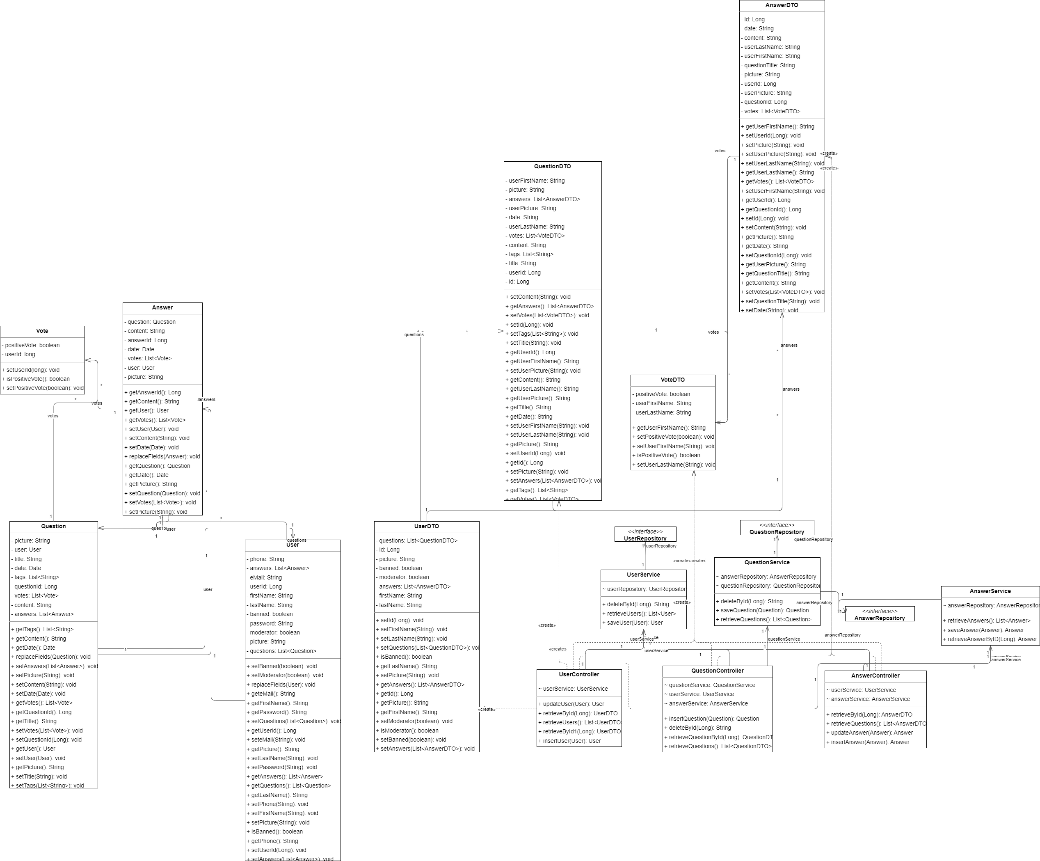


The database is build following the layered architecture pattern. In this architecture, we have the packages on the right with the following meanings:

* Repository: Used to make the connection between the database and the backend,
* Entity: Used to transfer lines from tables from the database into Objects,
* Service: Used to compute the logic of the backend,
* Controller: Used to make the connection between the database and the frontend,
* DTO: Used to make the Objects that are sent to the frontend.

Another important thing to say about this architecture is how the packages Controller, Service and Repository are layered: Controller can only use Service and Service can only use Repository, any other combination between them is forbidden.

1. Class Diagram

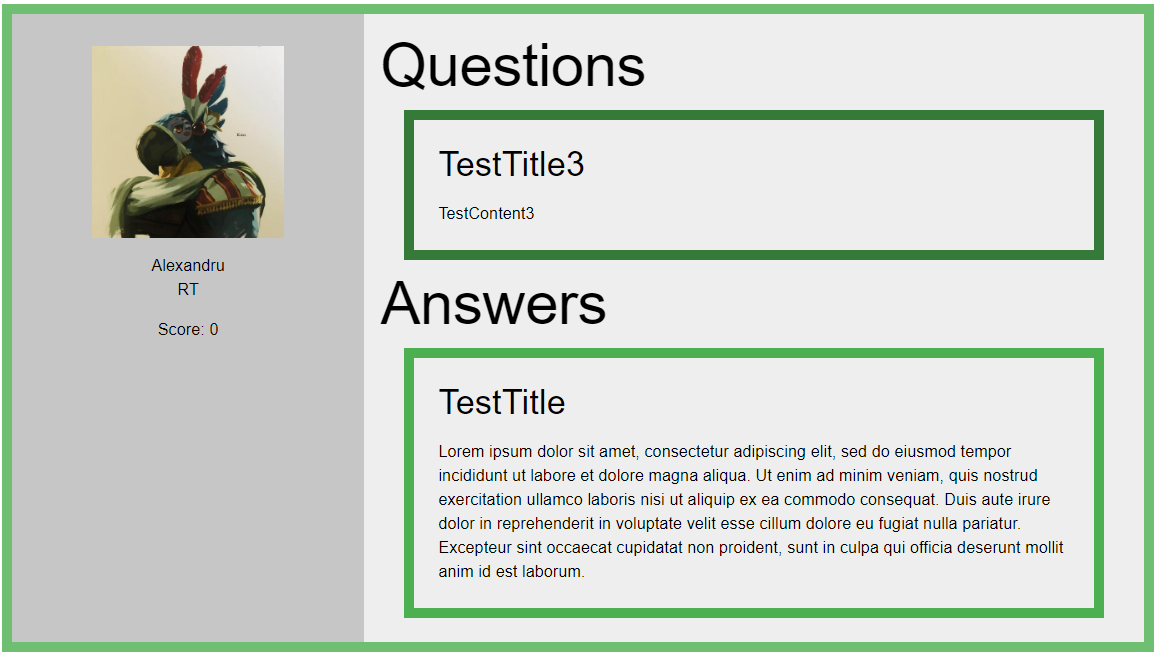


The classes follow the pattern described in the Package Diagram section, and each entity has a class in each package. We have three main entities, Answer, made with the help of Post and Votes, which maps the instances of a answer, Question, made with the help of Post and Votes, which maps the instances of a question, and User, which maps the instances of a user. Each of them is an Entity and has a Repository, a Service, a Controller, and a DTO.

# Routing

* 1. User

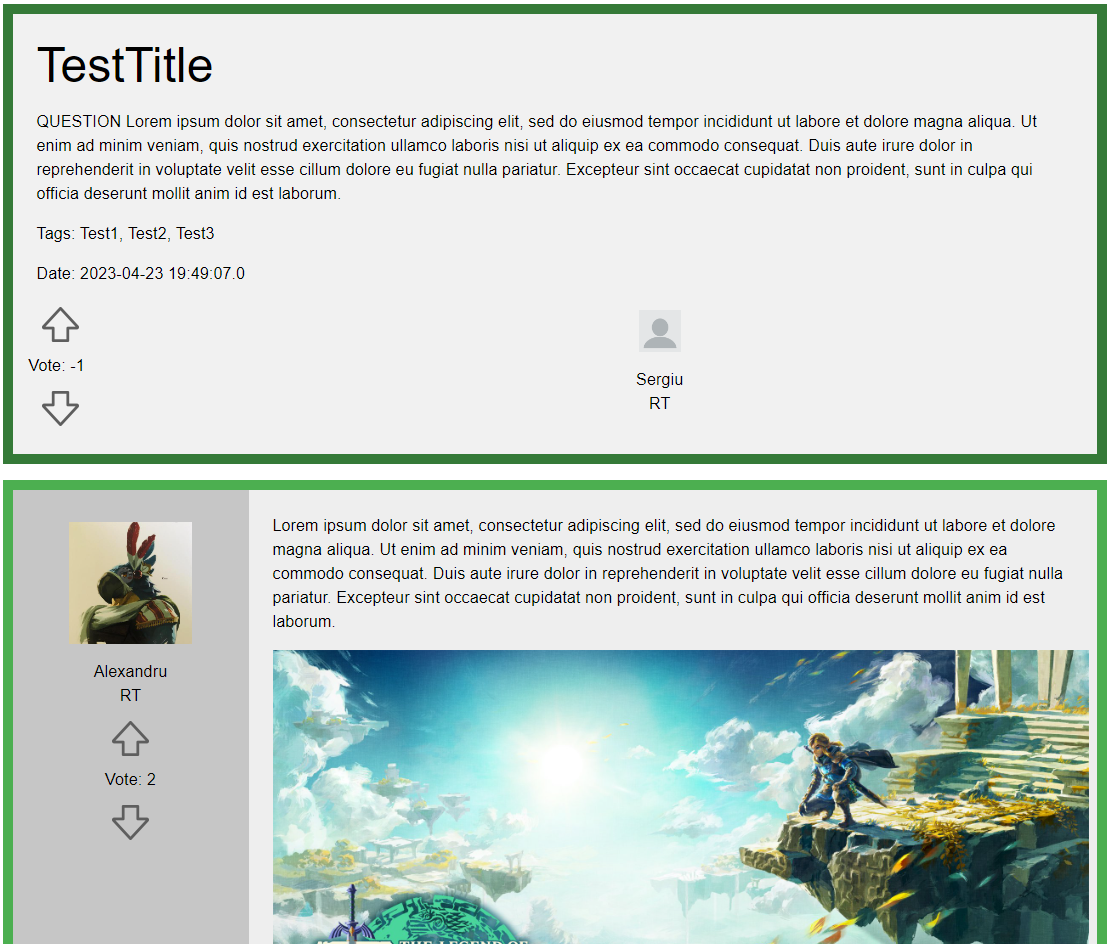
The path to get to a user is “./users/{id}”, where “id” represents the id of the user. The page contains information about the user, such as its profile picture, first name, last name, user score (which is computed by the votes on the user’s posts), and if the user’s a moderator or banned. This page also contains a list of questions posted, with the title and the question’s text, as well as a list of answers, with the title of the question and the text of the answer. Pressing a question or an answer, it redirects you to the page of the question.



* 1. Question

The path to get to a question is “./questions/{id}”, where “id” represents the id of the question. The page contains the question with the title, content, a descriptive photo if it exists, a list of tags (pressing a tag redirects you to all questions with the same tag), the date when it was posted, the user who posted it (pressing the user redirects you to his user page), as well as a vote counter.

The page also contains a list with all the answers. An answer is composed of its text, a picture of the answer if it exists, the date when it was posted, the user who posted it (which can be pressed), and a vote counter.



1. **Features**
   1. Questions

On the home page, the list of questions from the database is displayed, sorted by date, with the most recent question being displayed first.

If a user is logged in, he can ask a new question. On the home page, we have a “NEW QUESTION” button, which will send the user to the following page where he can write his question. Some fields are required, while others aren’t. After the question is made, the user is redirected to the question page.

A screenshot of a questionnaire

Description automatically generated with low confidence

After the question is made, the user is redirected to the question page.

A picture containing text, screenshot, software

Description automatically generated

On this page, the user has two buttons, the first one being the edit button. Pressing it opens a new form where the user can modify the question.

A screenshot of a computer

Description automatically generated

The other button is the delete button, which deletes the question and its answers and redirects the user to the home page.

Another feature is the search. The user can press a tag from a question to see all the questions with the same tag, and he can also use the search field to search for questions with a certain title. The user can also see other user’s questions (or his) on their profile page.



* 1. Answers

If a user is logged in, under every question there is a field where he can give an answer. After submitting the answer, he can still give other answers to the same question. Some fields are required while others aren’t.

A screenshot of a computer

Description automatically generated with medium confidence

The same edit and delete operations that were done on an own question can be done on an own answer. For a question, all its answers are displayed below.

A screenshot of a video game

Description automatically generated

* 1. Votes

The user can upvote or downvote a question or answer that is not his. He can only have one vote for an answer or question, but he can change the vote, and the total vote count is displayed. The vote count can also be negative. For a question, the list of answers is displayed decreasingly based on the vote count.

A picture containing font, symbol, white, design

Description automatically generated A picture containing symbol, font, logo, graphics

Description automatically generated A picture containing font, symbol, logo, white

Description automatically generated

* 1. Bonus Feature 1

Each user has a user score computed in the following way, starting from 0:

* +2.5 when his question is upvoted
* -1.5 when his question is downvoted
* +5 when his answer is upvoted
* -2.5 when his answer is downvoted
* -1.5 when he downvotes an answer
  1. Bonus Feature 2