QueueUnderflow

STUDENT NAME: Radu-Todor Alexandru

GROUP: 30433

# CONTENTS

[1. Introduction 3](#_Toc97840780)

[2. Diagrams 3](#_Toc97840781)

# 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.

# Diagrams

1. Use Case Diagram

A picture containing graphical user interface

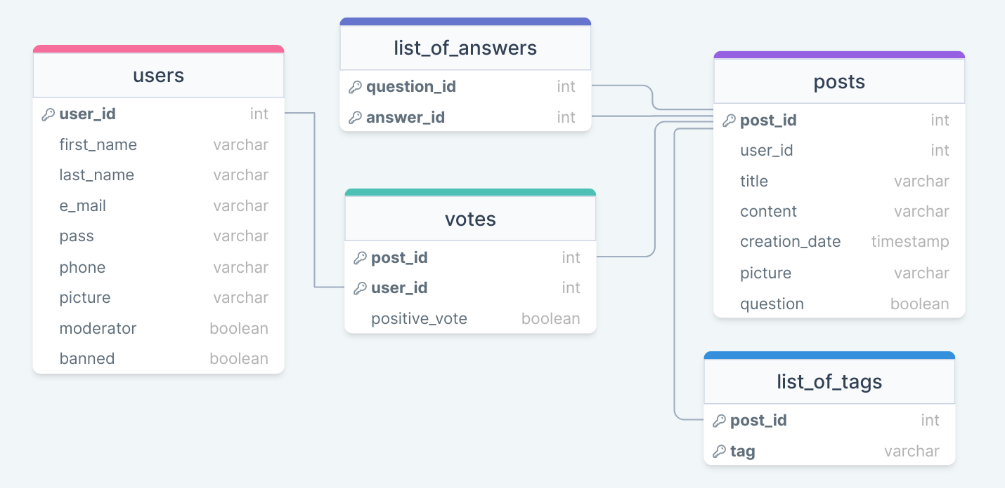
Description automatically generated

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.

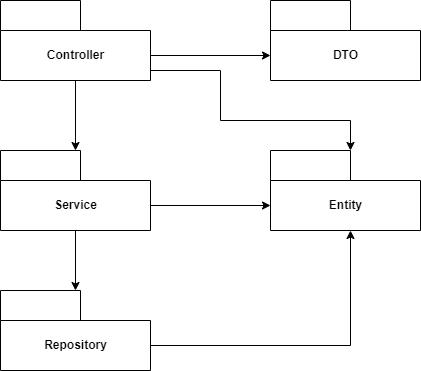
1. Database Diagram



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

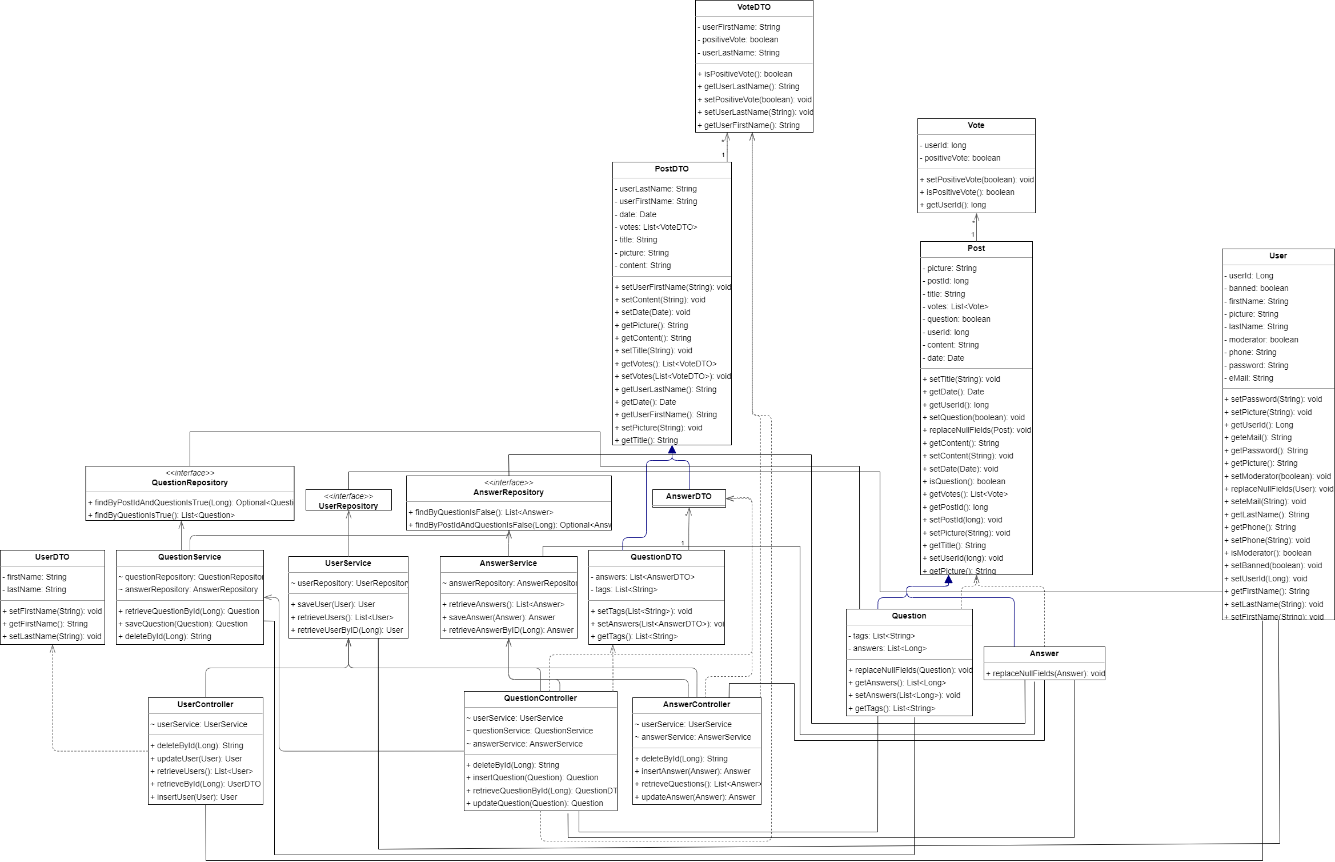


The database is build following the layered architecture pattern. In this architecture, we have the packages on the right with the following meanings:

* 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.