SD Laboratory Assignment –

Stack Overflow

Laboratory teacher: Eng. Bindea Bogdan Nicusor

Student: Petruta George-Calin

Group: 30434

Contents

[1. Introduction 3](#_Toc98251790)

[2. Technical Stack 4](#_Toc98251791)

[2.1 Programming language 4](#_Toc98251792)

[2.2 Frontend framework 4](#_Toc98251793)

[2.3 Data storage and querying 4](#_Toc98251794)

[2.4 API testing and services 5](#_Toc98251795)

[3. Use cases 6](#_Toc98251796)

[3.1 User use case 6](#_Toc98251797)

[3.2 Admin use case 6](#_Toc98251798)

[3.3 Detailed use case: 7](#_Toc98251799)

[4. Architecture 7](#_Toc98251800)

[5. Testing 8](#_Toc98251801)

[5.1 Create new user: 9](#_Toc98251802)

[5.2 Add question: 9](#_Toc98251803)

[5.3 Add answer: 10](#_Toc98251804)

[5.4 Get user: 10](#_Toc98251805)

[5.5 Get question: 10](#_Toc98251806)

[5.6 Get answer: 10](#_Toc98251807)

[5.7 Get all users: 11](#_Toc98251808)

[5.8 Get all questions: 11](#_Toc98251809)

[5.9 Get all answers: 11](#_Toc98251810)

[5.10 Delete user: 11](#_Toc98251811)

[5.11 Delete question: 11](#_Toc98251812)

[5.12 Delete answer: 12](#_Toc98251813)

[6. Bibliography 12](#_Toc98251814)

# Introduction

The application to be designed is a very simple copy of Stack Overflow. It will permit the user to enter questions with tags, like comment and question, see and search questions based on their title and/or tags and comment on questions.

The user is not able to do anything before logging in beside, of course, sign-up.

The users can like each other’s contributions, but not their own. They can also edit or delete their own material on the application when they want.

Beside the feature of adding new content on the web application, there will be a system which will rank the users depending on their contributions to the site.

Their score will be showed on every comment, and it will be updated when they receive reactions from others and depending on that, the score will increase or decrease.

Also, beside the normal user, there will be an/some admins which can ban/unban users if there is a need for that and also delete/edit materials that are found to be against the rules of the application.

On short, the application will have 2 types of users: Admin and User and each of them will have the following available features:

* Admin:
  + Add users
  + Delete/Update comments/questions
  + Ban/Unban users
* User:
  + Add/edit/delete question
  + Add/edit/delete answer
  + Like/dislike content
  + Create account
  + Search questions

# Technical Stack

## Programming language

The project will be done using as a programming language **Java** and **Javascript (React library).**

**Java version used: 17**

The main reason for using Java is the versatility and the support it offers in everything the developer would get into.

**IDE** used: Intellij

## Frontend framework

The UI will be done using the **React** library.

**Visual studio code** is used as the text editor for the development of the UI.

## Data storage and querying

The application used for storing the database and executing queries is the Mysql Workbench.

## API testing and services

In order to test the correct functionality of the different endpoints of the website application, **Postman** will be used.

# Diagram Description automatically generatedUse cases

## User use case

## Diagram Description automatically generatedAdmin use case

## Detailed use case:

**User logs in, comments on a question and like another answer:**

The user accesses the application and logs in.

After he successfully logs in, he can then browse through the questions and select one of them by clicking on the button to the right.

A new page will be displayed, showing the information about the question: title, author, vote count, tags and content.

Below this information added by the author, the user can find a comment section.

There, the user can like other’s user answer. Below the comment section he will find a button to add a new comment. After he enters the comment and press the button, the answer will be shown on the question page.

# Architecture

The architecture used for the application is called Layered Architecture. It consists in 4 layers, each of them calling through request the layers (most of the time only the next layer is called. In case of a closed layered architecture only the one below is called) and then obtaining the information from bottom to up as shown in the picture below:

The 4 layers are called:

* + Presentation
  + Business
  + Persistence
  + Database

The presentation layer represent the view and what the user can see. Usually here we have the classes that are used in UI or in frontend.

The business layer consists of classes that do all the logic for obtaining the information for the presentation layer. Here you can find classes such as: mappers, services and/or configuration.

In the persistence layer most of the classes are used for obtaining and storing data from the database. In the application, the main classes that exists in this layer are marked as “DTO” (Data Transfer Object) and is used to obtain information from executing queries. Using the mappers described in the business layer, we obtain the model classes used by the application (which are not visible directly to the user yet).

The database layer contains the databases which is accessed through different properties files containing the necessary information for connecting to the database and retrieving the data.Diagram

Description automatically generated

Figure 4.1: An example of a closed layered architecture

## Database diagram:

Diagram

Description automatically generated

## Class diagDiagram Description automatically generated with medium confidenceram

# Testing

The main part of the application was tested manually through the UI, but some additional unit tests were added in the application.

Beside those, some tests were done using Postman. Below, some of the requests from Postman will be described.

**Disclaimer: the information from the body/url might need to be changed in order to work on your computer!**

## Create new user:

**Method: POST**

**URL:** <http://localhost:8080/users/create>

**BODY:**

{

    "idRole": 1,

    "username": "maria",

    "password": "pere",

    "email": "ana@gmail.com",

    "banned": **false**

}

## Add question:

**Method: POST**

**URL:** [http://localhost:8080 /question/create](http://localhost:8080/users/create)

**BODY:**

{

    "question":

    {

        "idUser": 3,

        "title": "Java",

        "text": "gere?",

        "creationDate": "2018-01-25T21:34:55"

    },

    "tags": [

        {

            "idTag" : "5",

            "name" : "java"

        }

    ]

}

## Add answer:

**Method: POST**

**URL:** [http://localhost:8080/answer/create](http://localhost:8080/answer/create%20)

**BODY:**

{

    "idUser": 2,

    "idQuestion": 4,

    "text": "perge",

    "creationDate": "2018-01-25T21:34:55"

}

## Get user:

**Method: GET**

**URL:** [http://localhost:8080 /users/2](http://localhost:8080/users/create)

**BODY: EMPTY**

## Get question:

**Method: GET**

**URL:** <http://localhost:8080/question/2>

**BODY: EMPTY**

## Get answer:

**Method: GET**

**URL:** <http://localhost:8080/answer/2>

**BODY: EMPTY**

## Get all users:

**Method: GET**

**URL:** [http://localhost:8080 /users/all](http://localhost:8080%20/users/all)

**BODY: EMPTY**

## Get all questions:

**Method: GET**

**URL:** <http://localhost:8080/question/all>

**BODY: EMPTY**

## Get all answers:

**Method: GET**

**URL:** <http://localhost:8080/answer/all>

**BODY: EMPTY**

## Delete user:

**Method: GET**

**URL:** [http://localhost:8080 /users/delete/2](http://localhost:8080%20/users/delete/2)

**BODY: EMPTY**

## Delete question:

**Method: GET**

**URL:** <http://localhost:8080/question/delete/2>

**BODY: EMPTY**

## Delete answer:

**Method: GET**

**URL:** <http://localhost:8080/answer/delete/2>

**BODY: EMPTY**

# Bibliography

1. <https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf>
2. <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>
3. <https://heap.io/topics/what-is-a-tech-stack>
4. <https://www.postman.com/>
5. <https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/use-case-diagram-example-website-extend-and-include-use-case/>
6. <https://www.baeldung.com/java-spring-mockito-mock-mockbean>