StackOverflow replica documentation

I. Backend

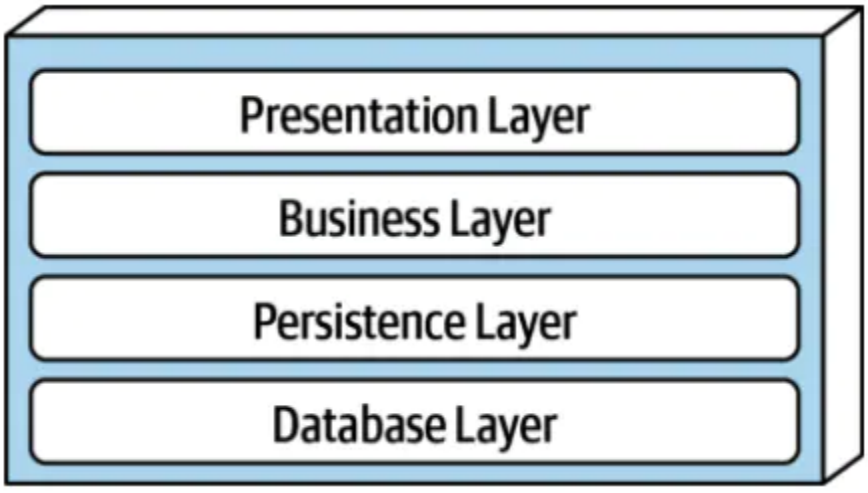
Short description

The website developed during this assignment will be a replica of the well known StackOverflow site used by many programmers when they encounter a bug or a problem and don't know how to solve it. There will be 3 types of users, unregistered who can only see the questions and the answers, the registered one can ask(add) questions and add answers, while also being able to delete their own question and answer. The last category is the moderator, which is a special type of user, him being able to delete answers and questions that he deems to be unappropriated, and can also decide to ban a specific user for an indefinitely period of time. There is also implemented a voting system, in which the users receive points for having a good answer, whist losing points for downvoting an answer or their own answer being downvoted.

Technologies

For this assignment I used Java Spring Boot for creating the back-end, PostgreSQL as a DBMS, this was because it offers the possibility of inheriting tables, and for the front-end I will be using Angular, alongside with Bootstrap, FontAwasome, and HTML.

Architecture



The architecture used will be the Layered Architecture. The Layered Architecture is the preferred architecture when building web applications that use CRUD operations on the data.

The presentation layer is mainly responsible fir the user interactions with the application, also known as the UI layer.

The business layer acts as a controller, handling the requests and responses.

The persistence layer is the one responsible for executing the actions and for processing the data before sending it to the database or to the controller.

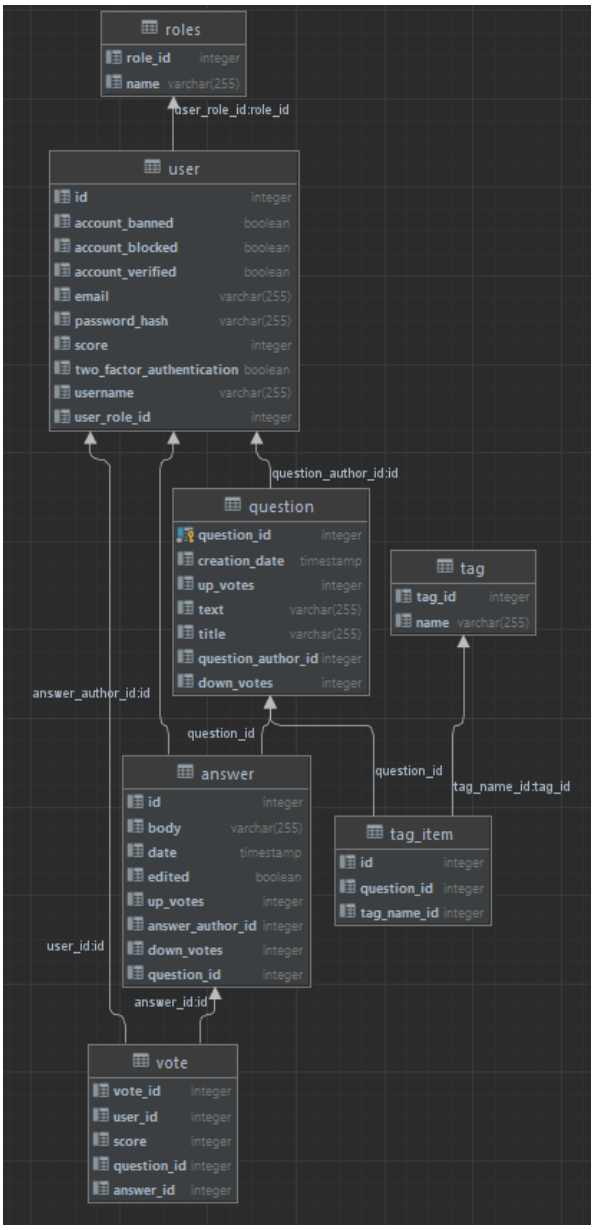
The database layer contains all the repository which interacts directly with the database. At this level I used Derived Queries from the JPA, in order to create custom queries easily and to avoid SQL injection attacks.

Database

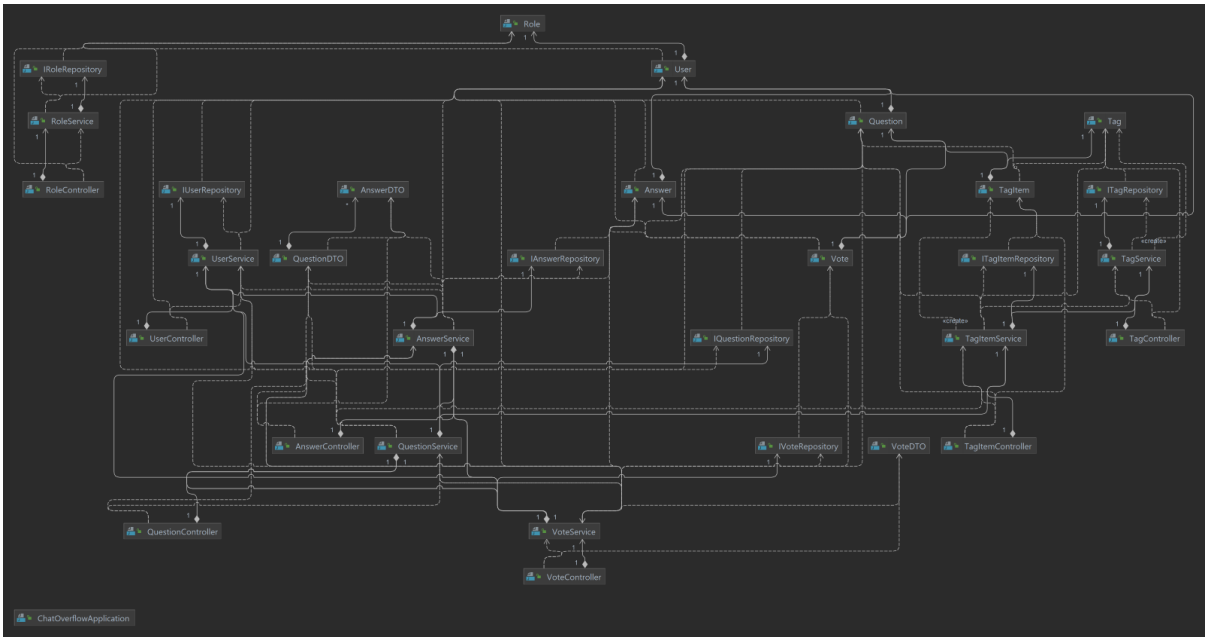
The database is composed of 7 tables , them being:

- Roles
- User
- Question
- TagItem
- Tag
- Answer
- Vote

Between Question and Tag being a many to many relationship, an extra table(tag item) had to be created.



Class diagram



Endpoints

There are several endpoints, as described below:

- Questions (/questions)
  - Get all questions (/all-questions)
  - get by name (/search) with the name parameter (acts like a normal search)
  - get question by id (/find-question) with the id parameter
  - delete question (/delete-question) with the id parameter
  - create question (/create-question) using a QuestionDTO in the body (is present in the exported postman file)
  - save question (/save-question) save the changes using the QuestionDTO
  - find questions by tag (/find-by-tag) having as parameter the tag
- Answers (/answers)
  - Create (/create) using the AnswerDTO in the body
  - Find by id (/find-answer) with the id as parameter
  - Edit (/edit) with the AnswerDTO in the body
  - Delete (/delete) with the id as parameter
- Vote (/vote)
  - Add vote (/add-vote) with the VoteDTO in the body
  - Change vote (/change-vote) with the VoteDTO in the body
  - Delete vote (/delete-vote) with the VoteDTO in the body
- Role (/roles)
  - View all roles (/all)
  - Add a role (/add) with the parameter name
- Users (/users)
  - Get all users (/all-users)
  - Find user by name (/search) with the parameter name

    - Find user by id (/find-user) with the parameter id
    - Delete a user (/delete-user) with the parameter id
    - Create user (/create-user) with the User Json in body
    - Edit user (/edit-user) with the User Json in body
- TagItem (/tags)
    - Get all tag items (/tagitems)
    - Get questions by tag (/find) with the parameter name
- Tag (/tag)
    - Get all tags (/all-tags)

Use case

The users logs in and searches for a question. He finds something which he thinks is useful and clicks on it. Because it wasn't useful, he presses the down vote button for the question. After that, he proceeds to create a new question. He creates it and after that he logs out.

II. Frontend

1. Technologies

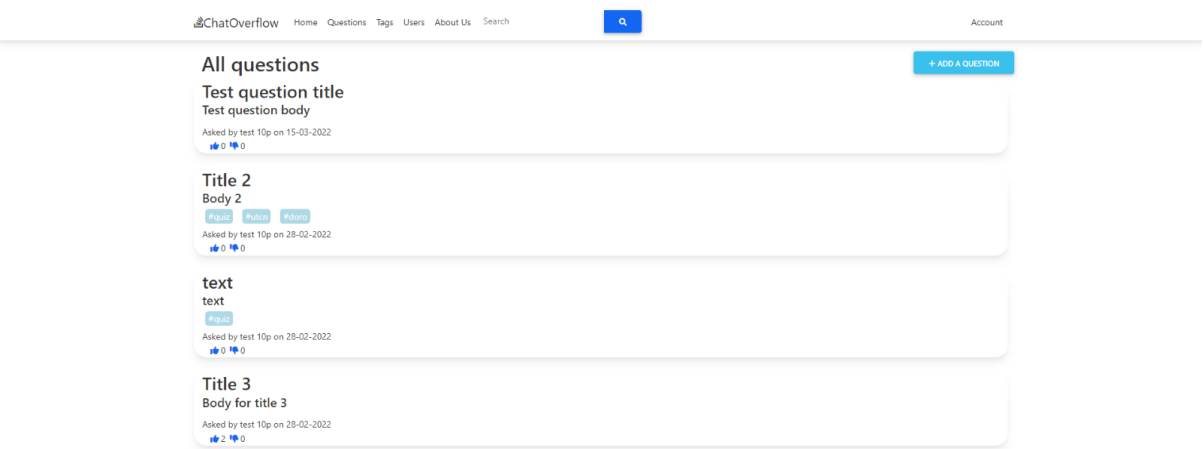
For the Front-End part of this assignment, I chose to use the Angular framework, a longside with Bootstrap, and Font Awesome, which provided the icons used in the buttons. A big advantage when using Angular is that we can reuse pieces of the UI, thus increasing the code reusability and decreasing the development time at the same time.

2. Architecture

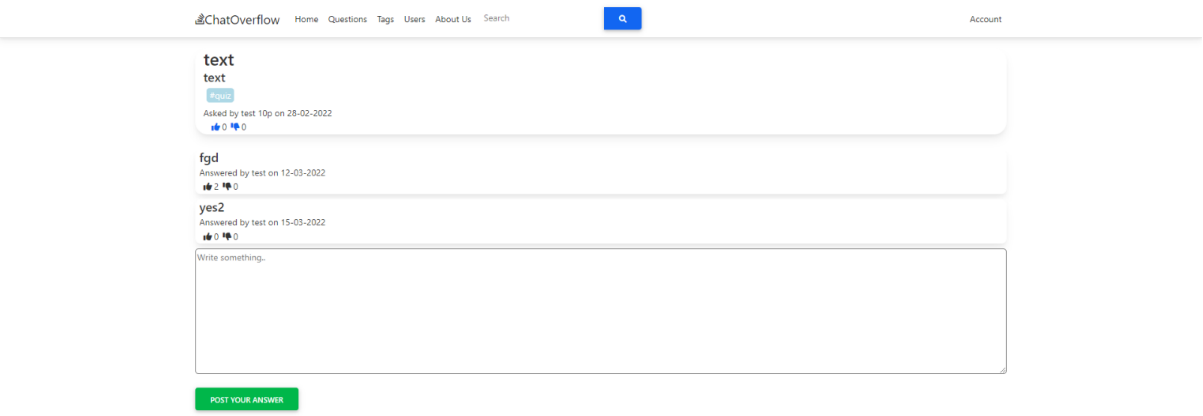
The architecture used in this Angular project can be compared to the Model-View-Controller architecture. The app routing module being similar to the controller layer, the services are just like those in the backed, responsible for sending the requests and parsing the JSONs, and alongside with the model classes creates the model layer. The last one is the view, which contains the components, with the HTML, SCSS and TypeScript parts which combined creates the components.

3. UI

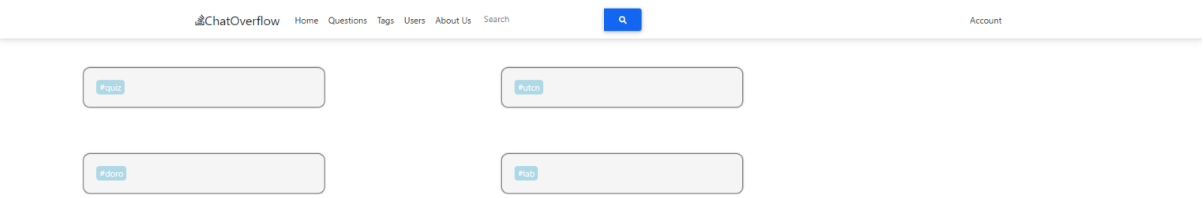
Questions page



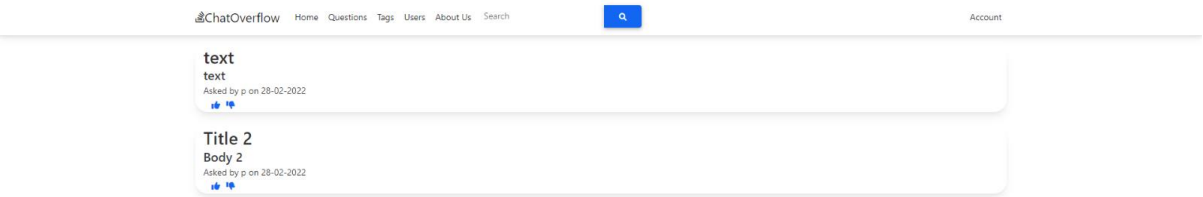
The Question Page



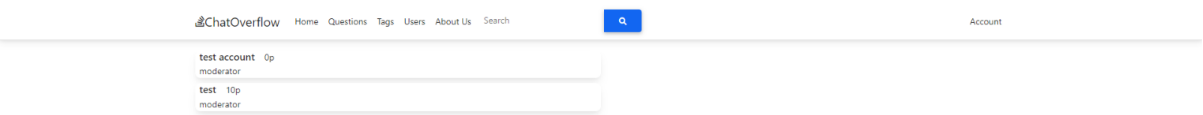
The Tags page



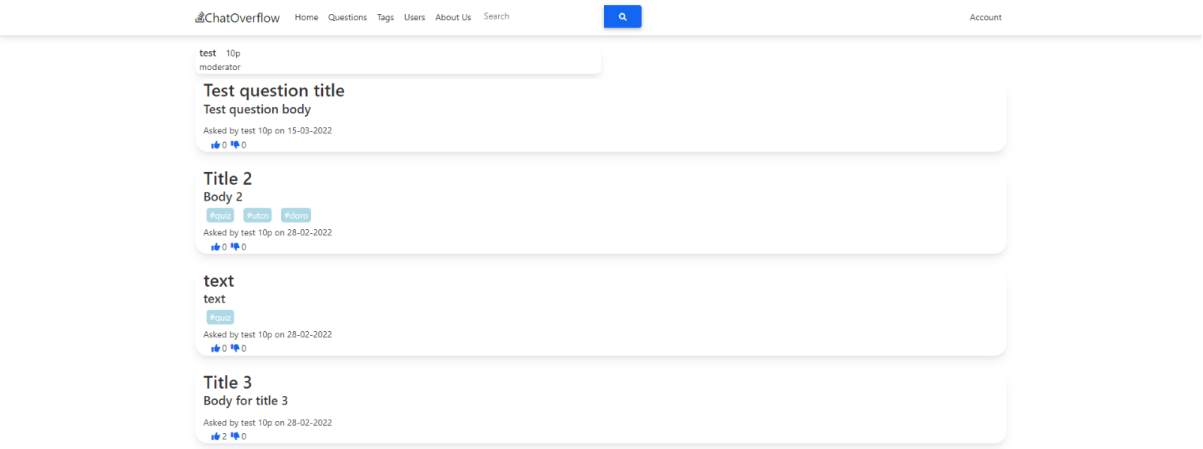
The Questions by tag



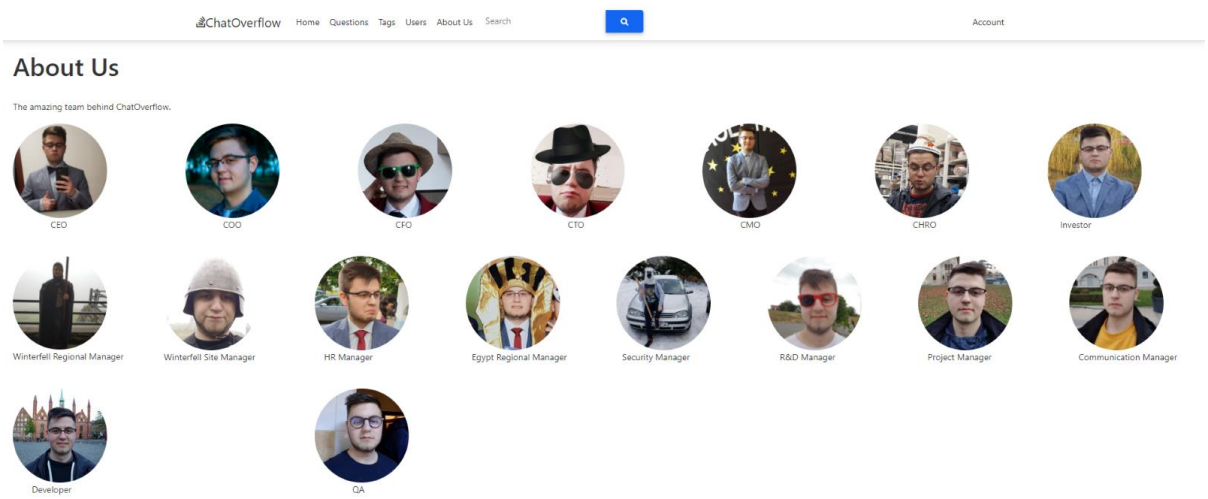
The Users page



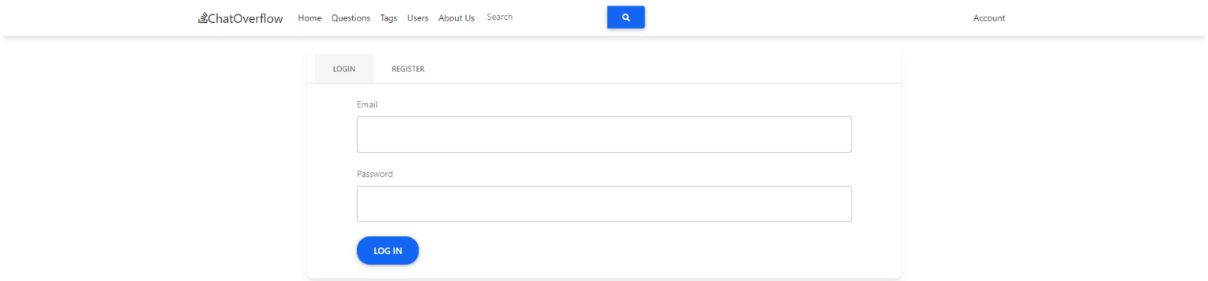
The User page



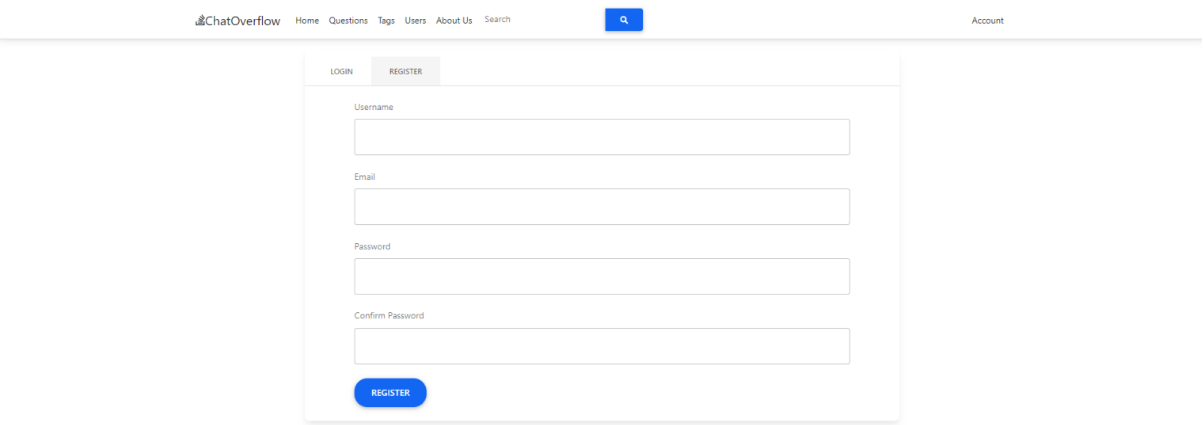
The well known about page



The login page



The register page



4. Testing

I created a testing module for each serviced linked to the backend: question service, user service, and tag service. In each testing module, I created 2 tests, one for verifying that the service can be instantiated and the other to verify that the service send the get response and receive the desired data.

5. Components

There are three types of components:

- Shared
    a. Navigation bar: which is present on every page
- Components
    b. Answer component
    c. Question component
    d. Tags component
    e. User component
- Pages
    f. About page component
    g. Add question page component
    h. Home page component
    i. Login page component
    j. Question page component
    k. Questions page component
    l. Questions by tag component

m. Register page component
n. Tags page components
o. User page component
p. Users page component

III. Bibliography

https://www.baeldung.com/spring-data-derived-queries

https://www.baeldung.com/configuration-properties-in-spring-boot

https://zetcode.com/springboot/postgresql/

https://www.codejava.net/frameworks/spring-boot/connect-to-postgresql-database-examples

https://www.baeldung.com/circular-dependencies-in-spring

https://stackoverflow.com/questions/3485347/circular-dependency-in-spring

set image path in angular 8 Code Example (codegrepper.com)

Angular - Testing

Testing Components – Testing Angular (testing-angular.com)

how to set default page in angular 8 Code Example (codegrepper.com)

Buttons · Bootstrap v5.0 (getbootstrap.com)

onchange Event (w3schools.com)