**Stackoverflow Application**

1. **Introduction**

The project represents a full stack application that simulates the functionalities of Stackoverflow. It can be a simpler version (only with users and without scores), or a more complex one that is closer to reality. Basically the site will have a login form for the users/moderators to log in and then ask/answer questions that are posted by others. Each user must be able to address questions and maybe like/dislike the content posted by others.

The moderators are “super-users” that can edit/remove the content on the site and ban/unban users. The application needs to have a messaging system that alerts the users that are being banned.

**2. Technology**

For the development of the full-stack application that represents the StackOverflow app, I have used a combination of Spring Boot and Angular as the primary technologies.

Spring Boot, which is an open-source Java-based framework, is used to develop the backend of the application. It provides a robust set of features and functionalities that enable developers to create scalable, flexible, and high-performance web applications quickly. Additionally, it comes with pre-built modules for database integration, security, and web services that significantly reduce development time and effort.

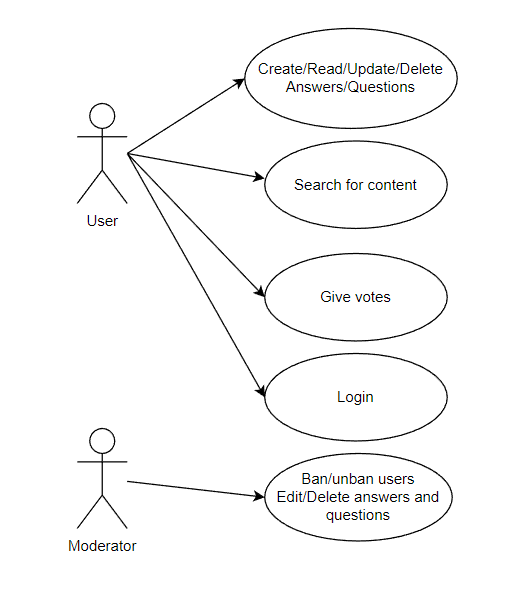
For the frontend, I have used Angular, which is a TypeScript-based open-source framework. Angular is known for its robustness, scalability, and performance. It offers a set of tools and libraries for building dynamic, responsive, and modern web applications.

To develop the application, I have used IntelliJ IDEA as the IDE for Spring Boot and Visual Studio Code for Angular. IntelliJ IDEA is a powerful and feature-rich IDE that provides excellent support for developing Java-based applications. Visual Studio Code is a lightweight, cross-platform code editor that offers extensive support for TypeScript and Angular development.

* Java 17
* Spring Boot 3.0
* MySql Database

**3. Use case diagram**

A use case diagram is a type of UML (Unified Modeling Language) diagram that is used to describe the functionality of a system from a user's perspective. It shows the different actors (users or other systems) that interact with the system and the various use cases (functions or features) that the system provides to each of these actors. Use case diagrams are useful for understanding the requirements of a system and for communicating those requirements to stakeholders. They are often used in the early stages of software development to help identify the different types of users that will interact with the system and the specific tasks they will perform.

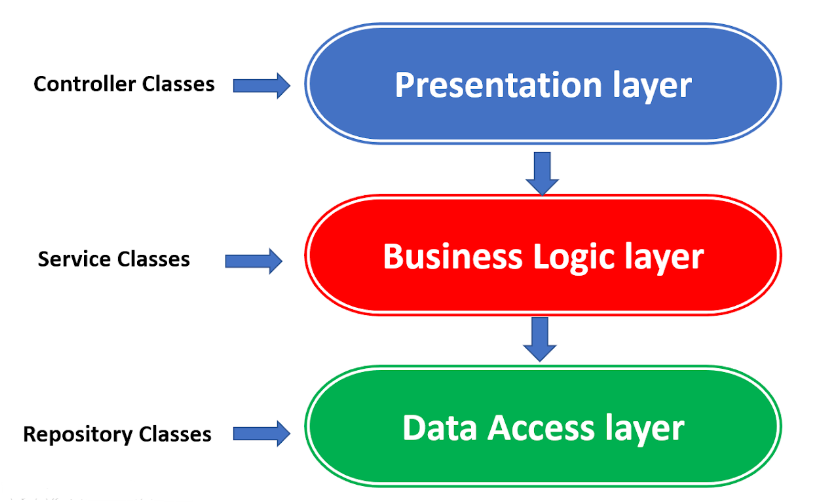


**4. Architecture**

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers.

In Spring Boot we have what we call an MVC Architecture (model, view, controller) and its visual representation looks like this :



The controller classes contains all of the endpoints that are used for establishing the communication between client and server. Usually, it does not contain logic, it is like a gateway for the data between the frontend and the backend.

The service classes are used for the business logic of the application. The controllers are using service methods and pass the data that is coming from the frontend. After the operation is completed, the service is usually sending the resulted data back to the controller.

The repository classes are used by the service ones and their role is to manage the database CRUD operations and to retrieve data.

**5. Package diagram**

The Spring Boot application is well organised in packages, having related classes that deserve a purpose. We will have the four main packages like in the figure below. In a large application we will have a lot more packages(for configuration, errors, mappers, DTO’s), but this ones are mandatory. I have explained above all the functionalities of the classes that compose this packages.

Diagram

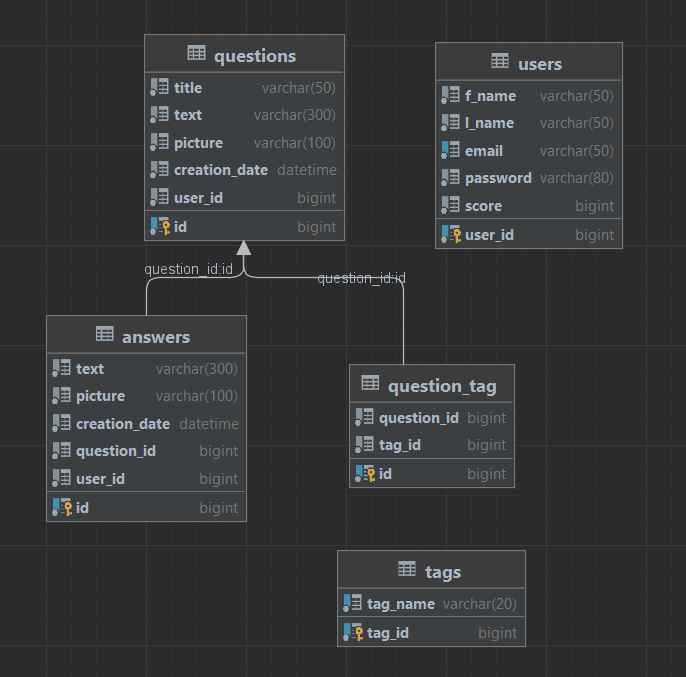
Description automatically generated

**6. Class diagram**



**7. Database diagram**

Database diagrams graphically show the structure of the database and relations between database objects.

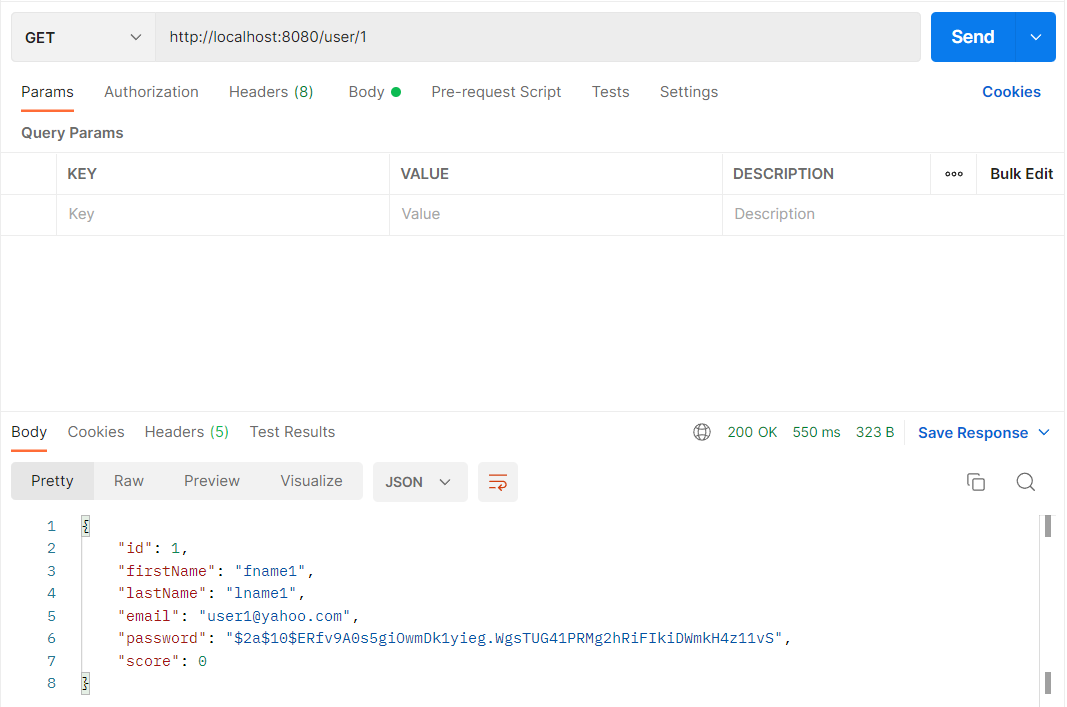


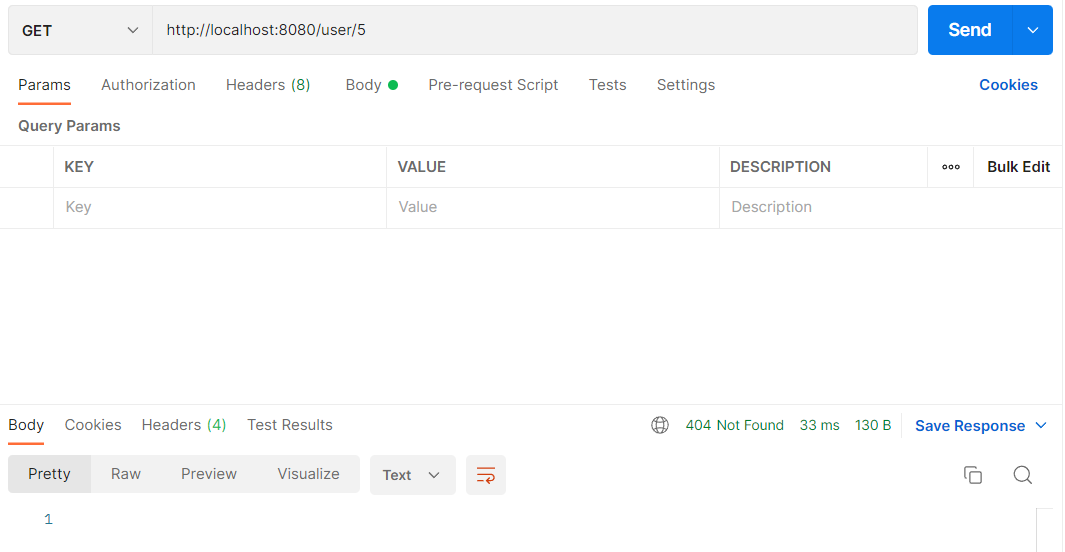
**8. Endpoints requests**

* for User -> “/user”
* for Answer -> “/answer”
* for Question -> “/question”

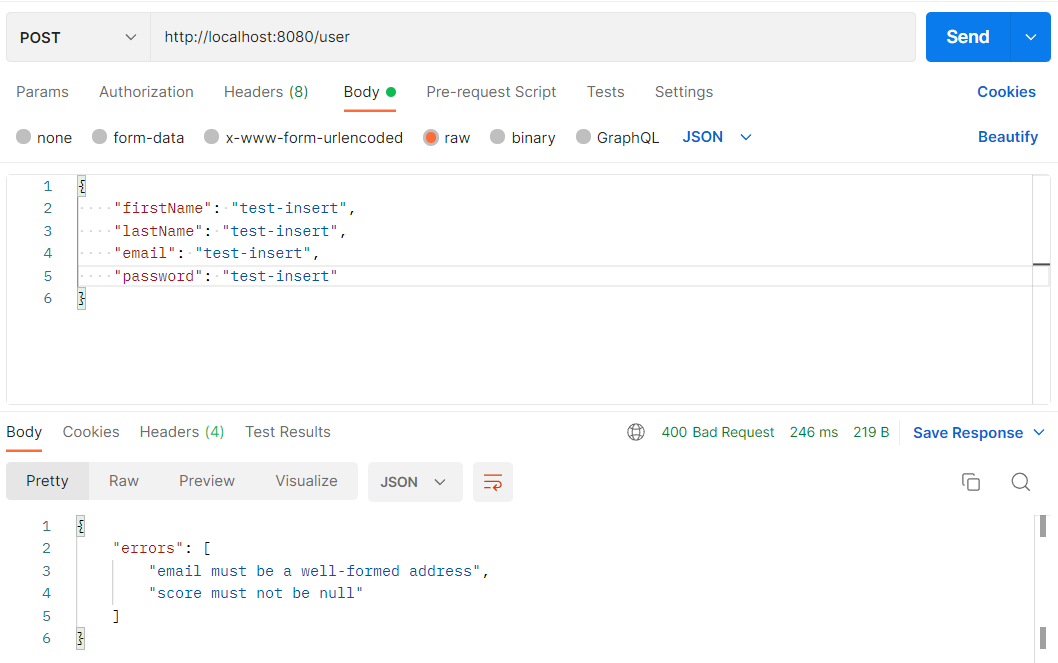
Below I have attached some photos that represent how the basic CRUD requests work in my application. I have used the class ResponseEntity to provide the result for the request and I have implemented a small validation and HttpStatuses as well for a better understanding.

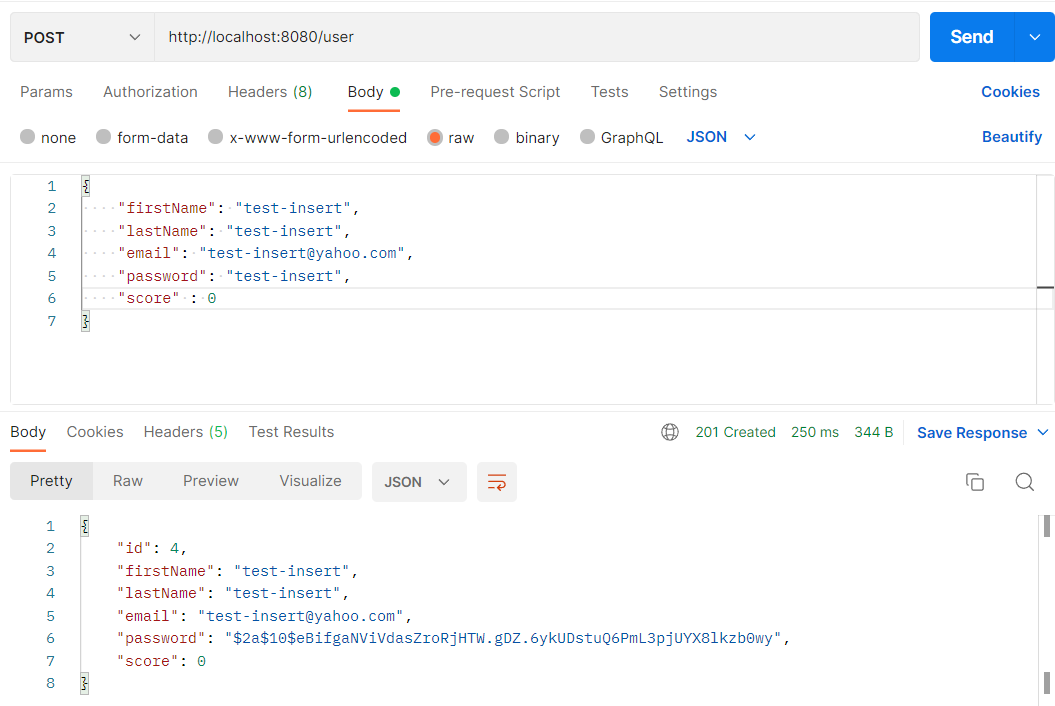
**GET**



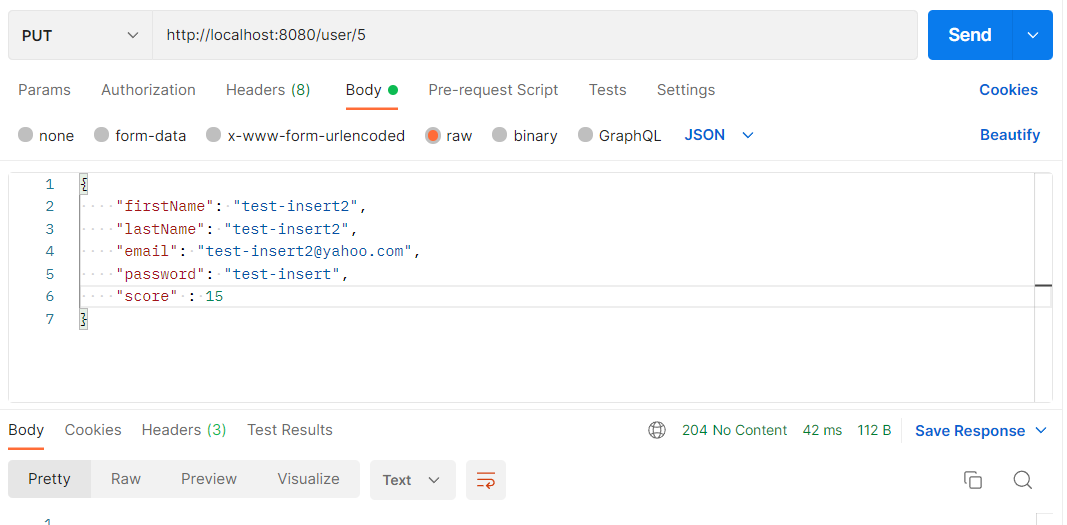


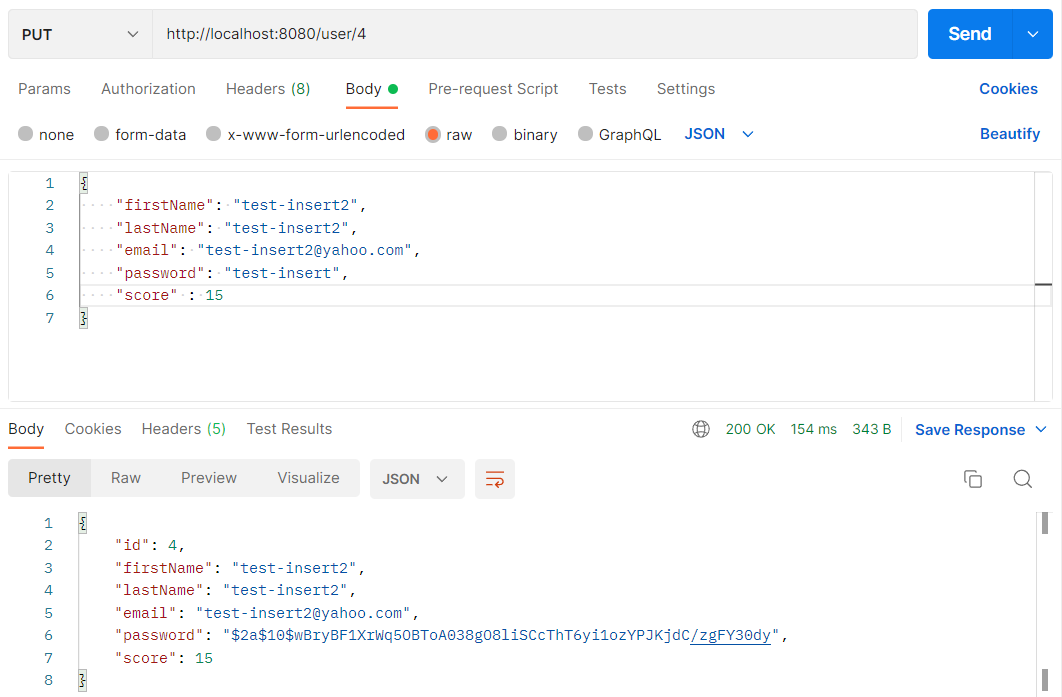
**POST**





**PUT**





**DELETE**

