# Report on the Forum Project: Development of a Forum System Using Java and Spring Framework

Author: Dinmukhamed Osmanov;  
Group: Computing technology and software – 35  
Project name: Openforum  
Data: 11/17/2024

## Introduction

The goal of this project was to develop a fully functional forum system using Java and the Spring Framework. The project aimed to provide a user-friendly platform for discussions, including features for creating categories, topics, messages, and comments. The system supports user roles and permissions, allowing for different levels of access and control.

### Project Overview

The development of the forum was carried out in several phases:

1. **Requirement Analysis**: Understanding the functional and non-functional requirements of the forum.

2. **System Design**: Designing the architecture and database structure for the application.

3. **Implementation**: Writing the core logic for the forum’s functionality, including user management, topic management, and messaging.

4. **Testing**: Ensuring the system works as expected by writing and running comprehensive tests.

5. **Deployment**: Configuring the environment for deploying the application to production.

6. **Documentation**: Preparing this report and other relevant documentation for the project.

This report provides an in-depth explanation of the system’s development, architecture, functionality, and testing.

## Requirements

### Functional Requirements

1. **User Authentication and Authorization**: Users must be able to register, log in, and manage their profiles.
2. **Category and Topic Management**: Users with appropriate roles should be able to create, edit, and delete categories and topics.
3. **Message and Commenting System**: Users can post messages in topics, as well as comment on messages.
4. **Role-based Access Control**: The system must support different user roles, such as admin, moderator, and user, each with distinct permissions.

### Non-Functional Requirements

1. **Performance**: The system should handle multiple concurrent users and a growing amount of data efficiently.
2. **Scalability**: The architecture should support future scaling, allowing additional features or users to be added without significant rewrites.
3. **Reliability**: The system must be robust, with minimal downtime and error handling.
4. **User Interface**: The system should be intuitive and easy to navigate.

## System Architecture

### Overview of Architecture

The system is designed using a **multi-layered architecture**, consisting of several distinct layers that work together to provide the required functionality.

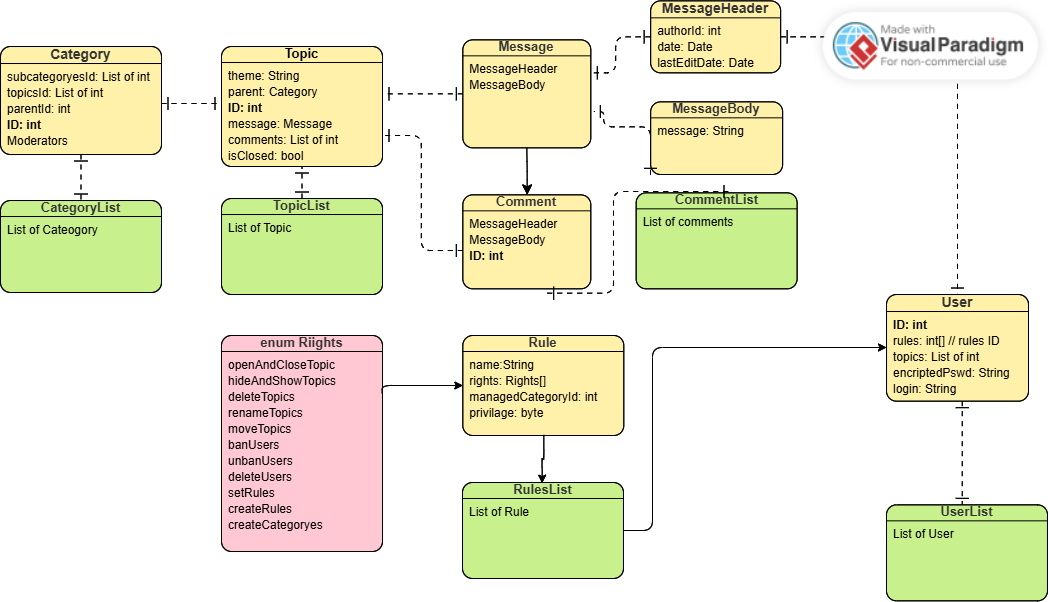
1. **Presentation Layer**: This layer is responsible for interacting with the user and displaying the results. It is built using **Thymeleaf** templates.
2. **Service Layer**: Contains the core business logic and processes requests from the controller layer. It interacts with repositories to fetch and manipulate data.
3. **Data Access Layer**: This layer consists of the repositories that handle database interactions. It is built using **Spring Data JPA** and communicates with the **PostgreSQL** database.

### Database Schema

The database schema is designed to store information about users, roles, topics, messages, comments, and categories. The following tables are included:

* **Users**: Stores user information such as username, email, password, and role.
* **Roles**: Defines different user roles, such as admin, moderator, and user, along with their respective permissions.
* **Categories**: Represents the different categories within the forum.
* **Topics**: Represents topics within each category.
* **Messages**: Stores the messages posted within each topic.
* **Comments**: Stores comments made on messages by other users.

### Architecture Diagram



### 

## Technologies Used

### Programming Language

* **Java**: The core language for implementing the forum system. Java is a widely-used, robust language suitable for enterprise-level applications.

### Frameworks

* **Spring Boot**: Used to simplify the configuration and deployment of the application. It provides built-in features like dependency injection, security, and data access.
* **Spring Data JPA**: Simplifies database interactions by using repositories to handle common CRUD operations.
* **Spring Security**: Implements authentication and authorization features, enabling role-based access control.

### Database

* **PostgreSQL**: A powerful, open-source relational database management system (RDBMS) used for storing data related to the forum.

### Frontend

* This project is called "openforum". Because it is openly stored in github and does not have a frontend. Anyone who wants to create their own forum can take the project and create their own design for their needs. Or just practice with creating a frontend for beginner developers.

### Tools

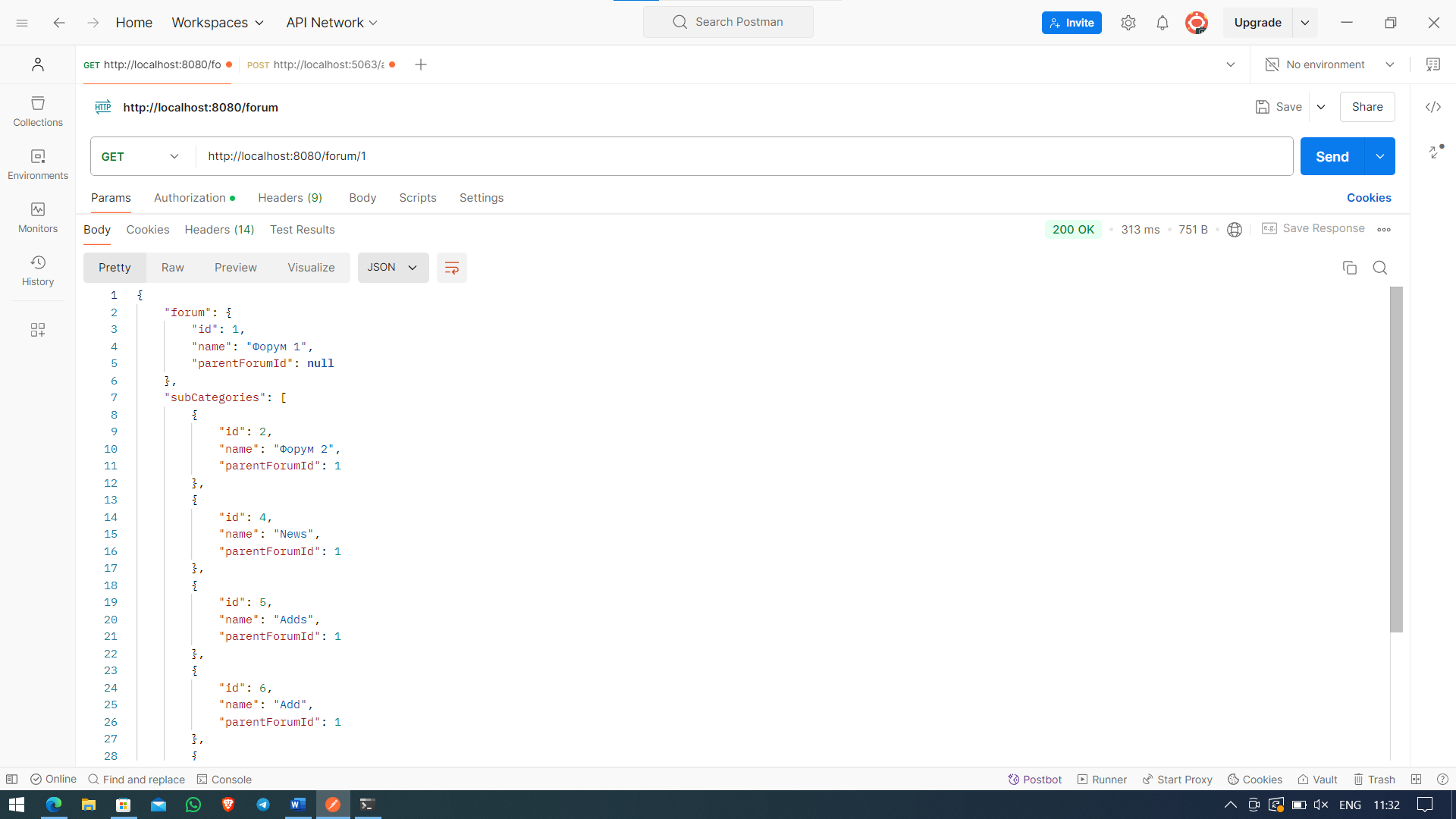
* **JUnit**: Used for unit and integration testing of the service and repository layers.

## System Features and Implementation

### Category and Topic Management

Administrators and moderators can create and manage categories and topics. Categories are hierarchical, meaning a category can contain subcategories. Topics are created within these categories and represent discussion threads.

**Watch category:**For watch some category, I just need send HTTP GET request to the server with category ID.

[http://localhost:8080/forum/1**](http://localhost:8080/forum/1)

**Full response:**{

"forum": {

"id": 1,

"name": "Форум 1",

"parentForumId": null

},

"subCategories": [

{

"id": 2,

"name": "Форум 2",

"parentForumId": 1

},

{

"id": 4,

"name": "News",

"parentForumId": 1

},

{

"id": 5,

"name": "Adds",

"parentForumId": 1

},

{

"id": 6,

"name": "Add",

"parentForumId": 1

},

],

"topics": [],

"path": [

"Форум 1"

]

}

This endpoint return object **ForumViewDTO** in own response body.

**ForumViewDTO:**

public class ForumViewData {

    private ForumDTO **forum**;

    private List<ForumDTO> **subCategories**;

    private List<TopicDTO> **topics**;

    private List<String> **path**;

}

ForumViewDTO dependence of **ForumDTO:**

public class ForumDTO {

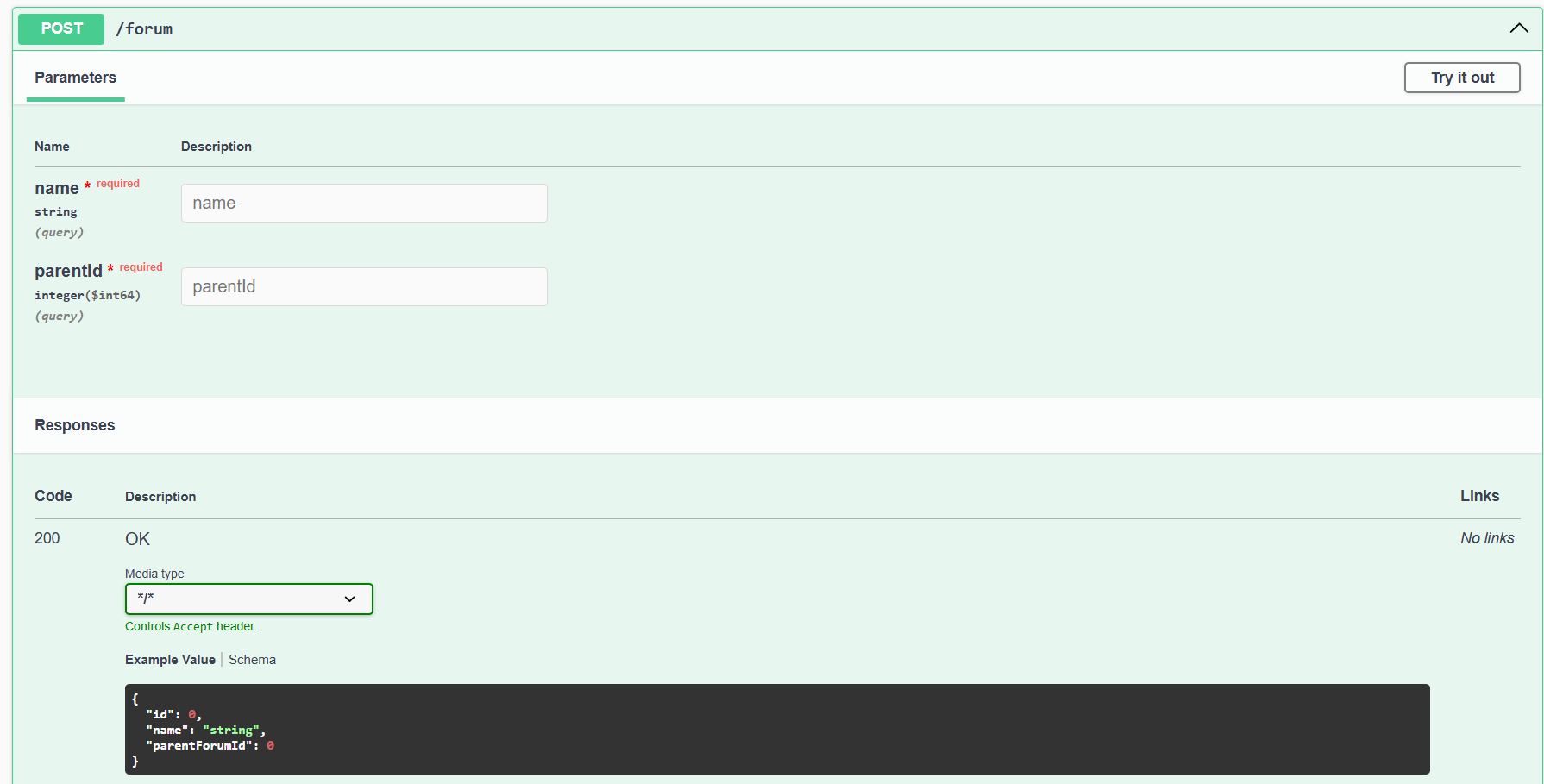
    private Long id;

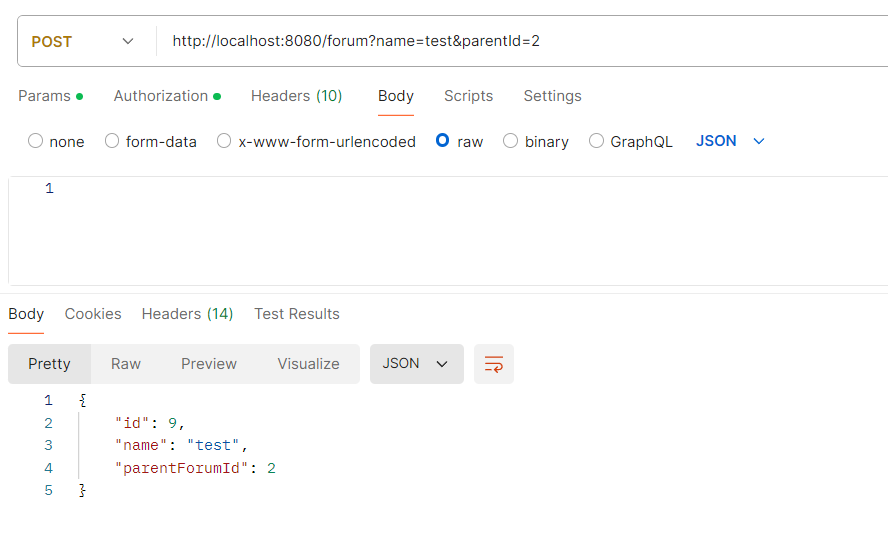
    private String name;

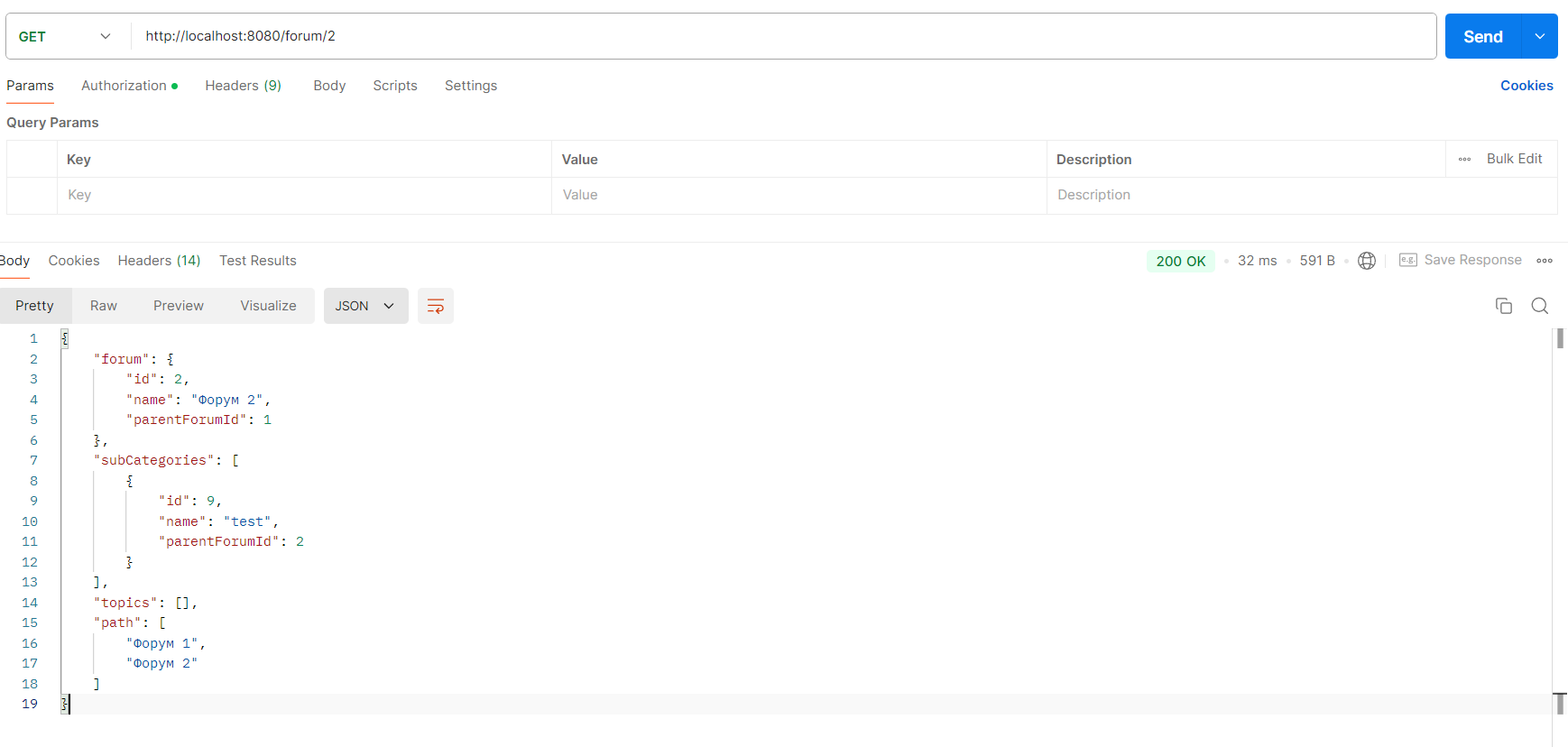
    private Long parentForumId;

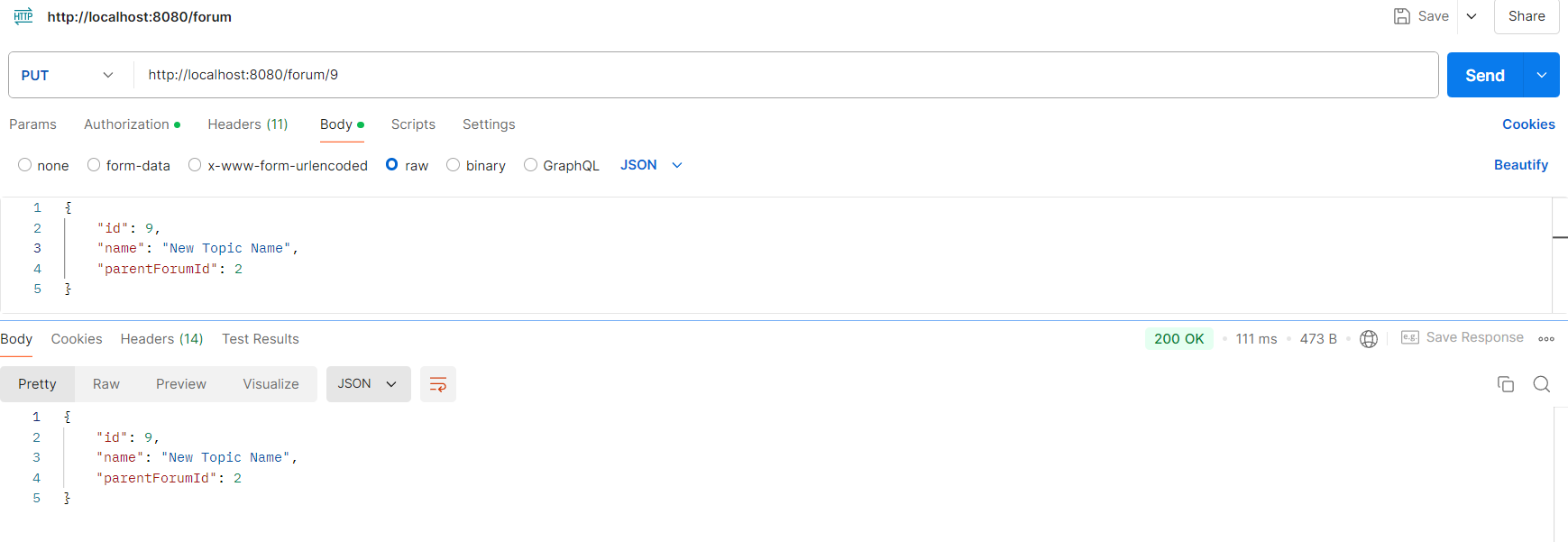
}

**Creating new forum:**

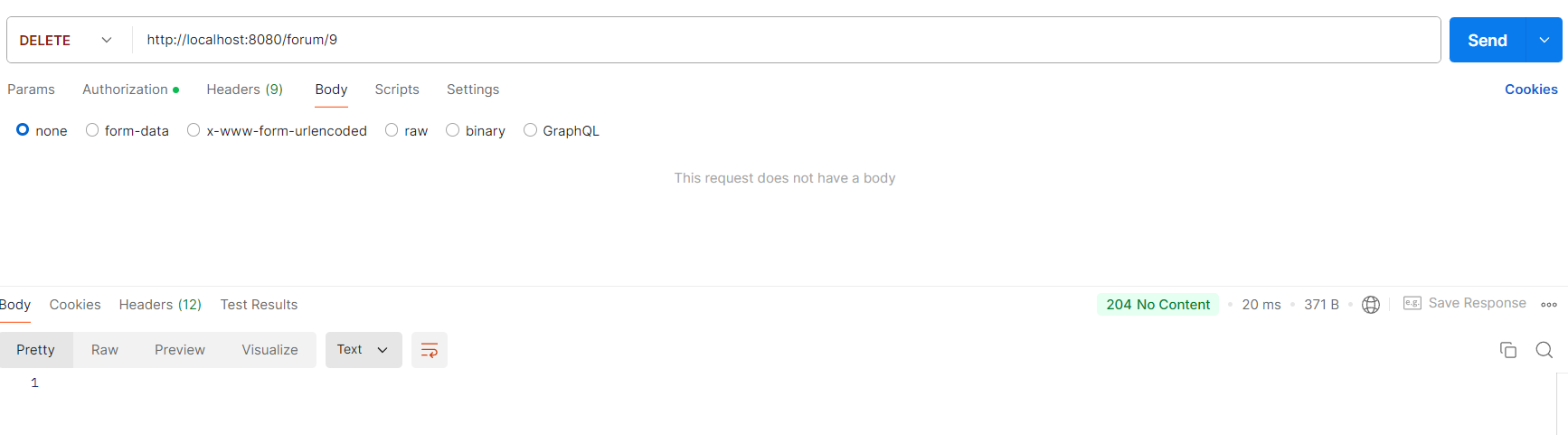
****

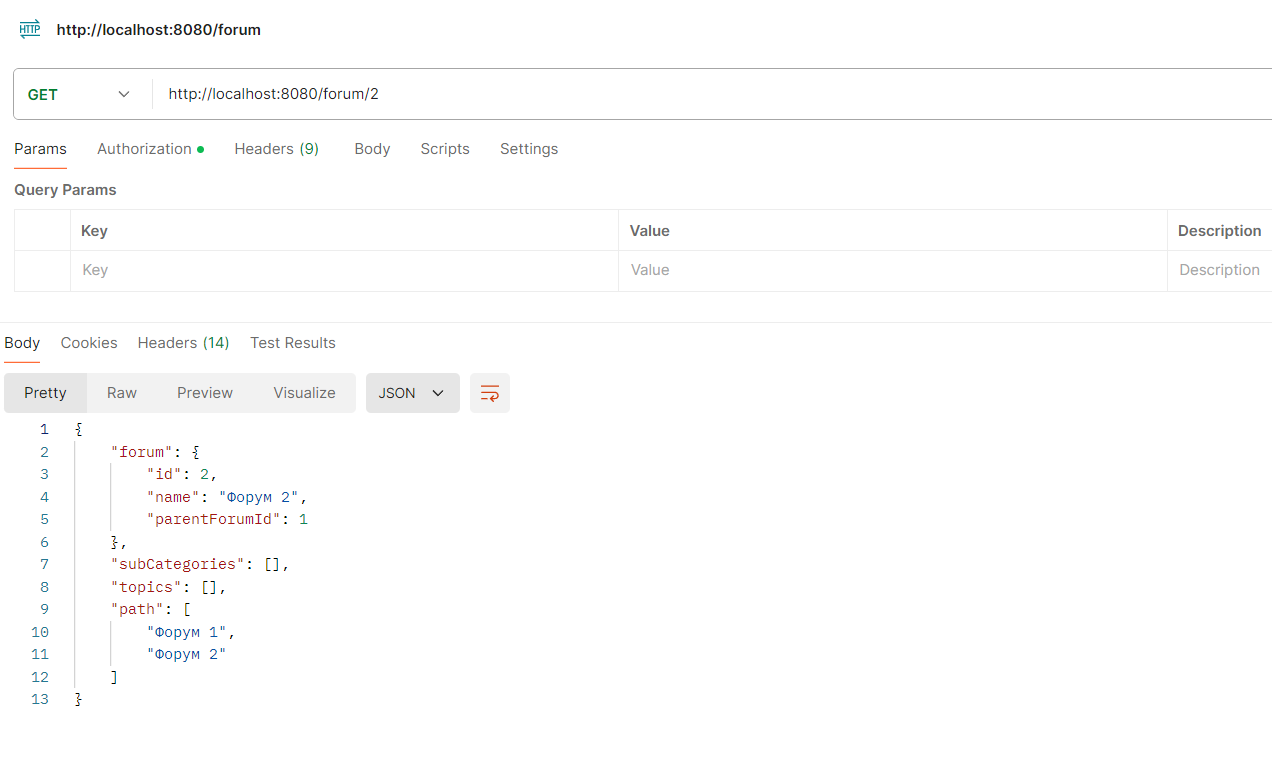
****

****

****

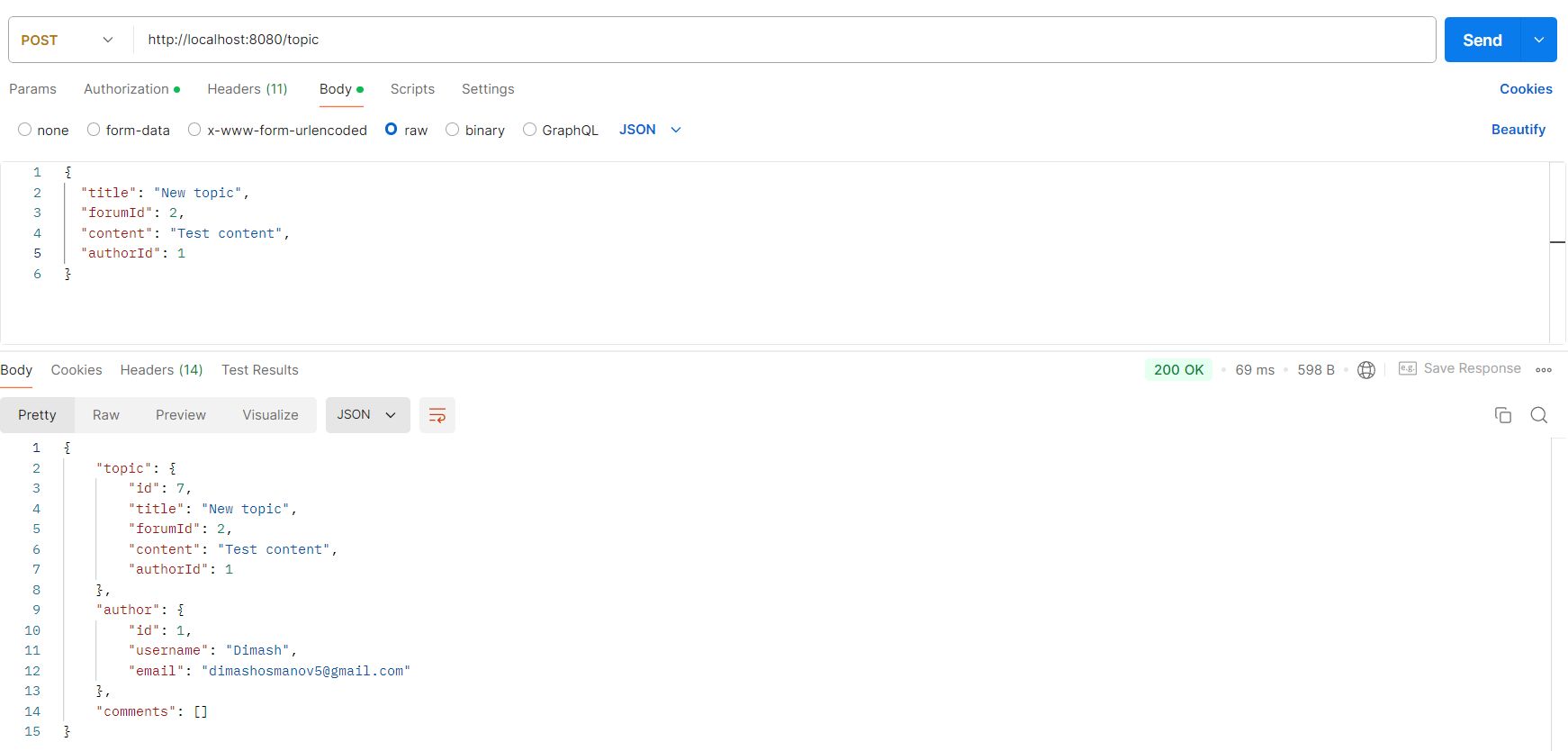
**Deleting forum:**

****

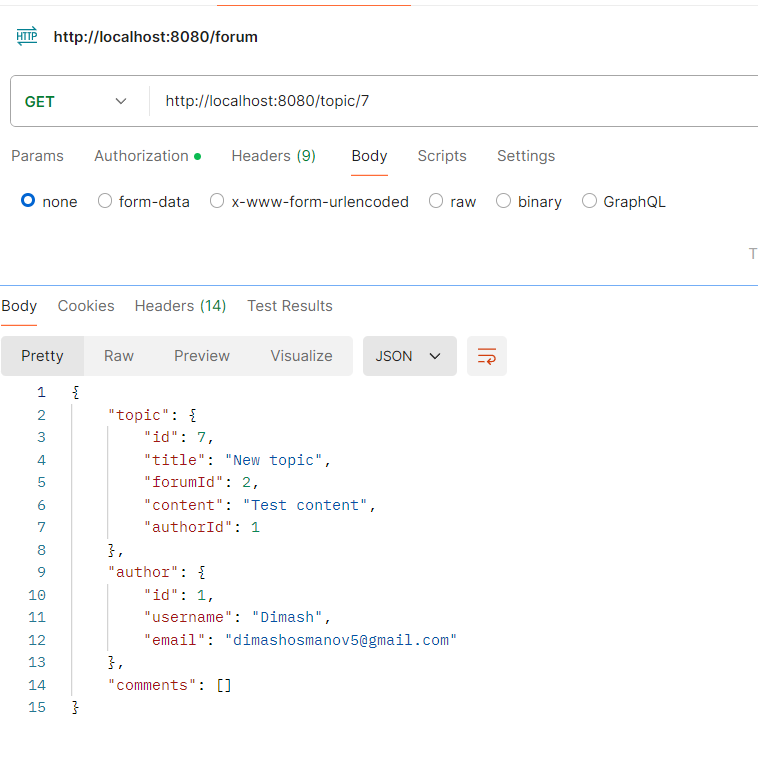
****

**Topic**

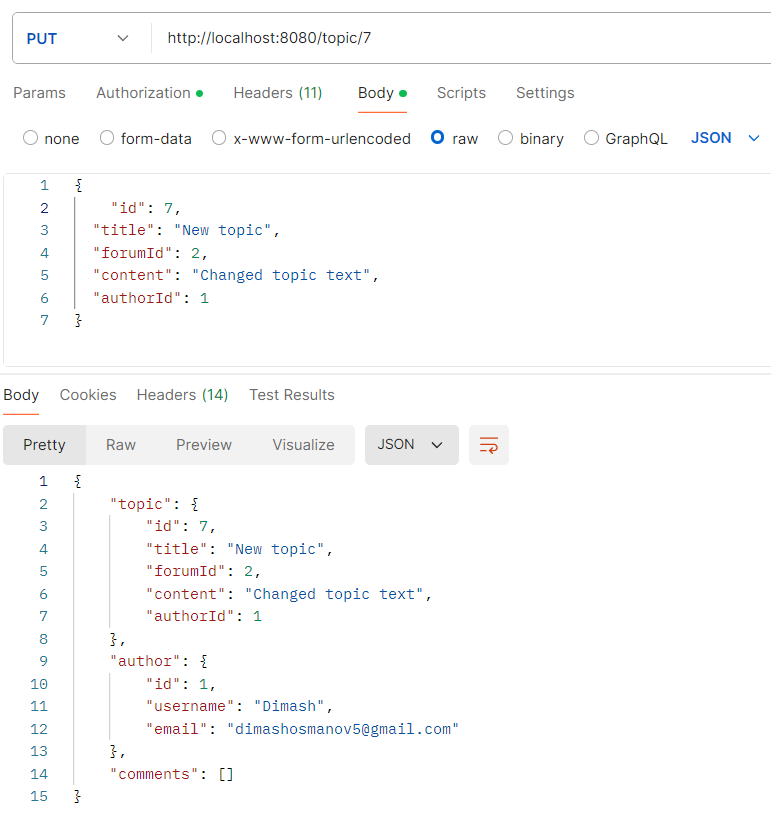
Create topic:



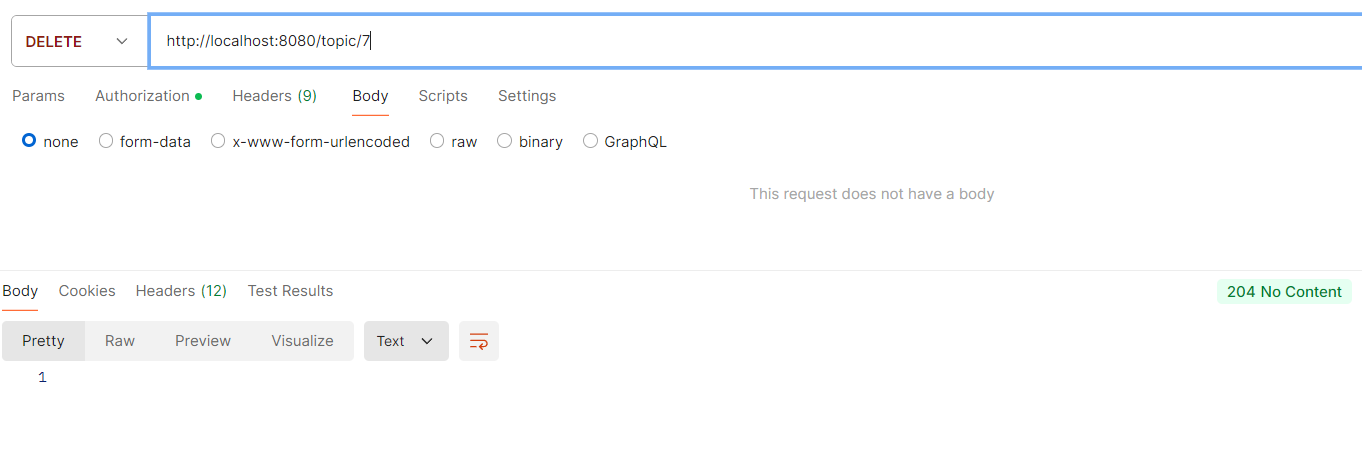
Read topic:



Update topic:



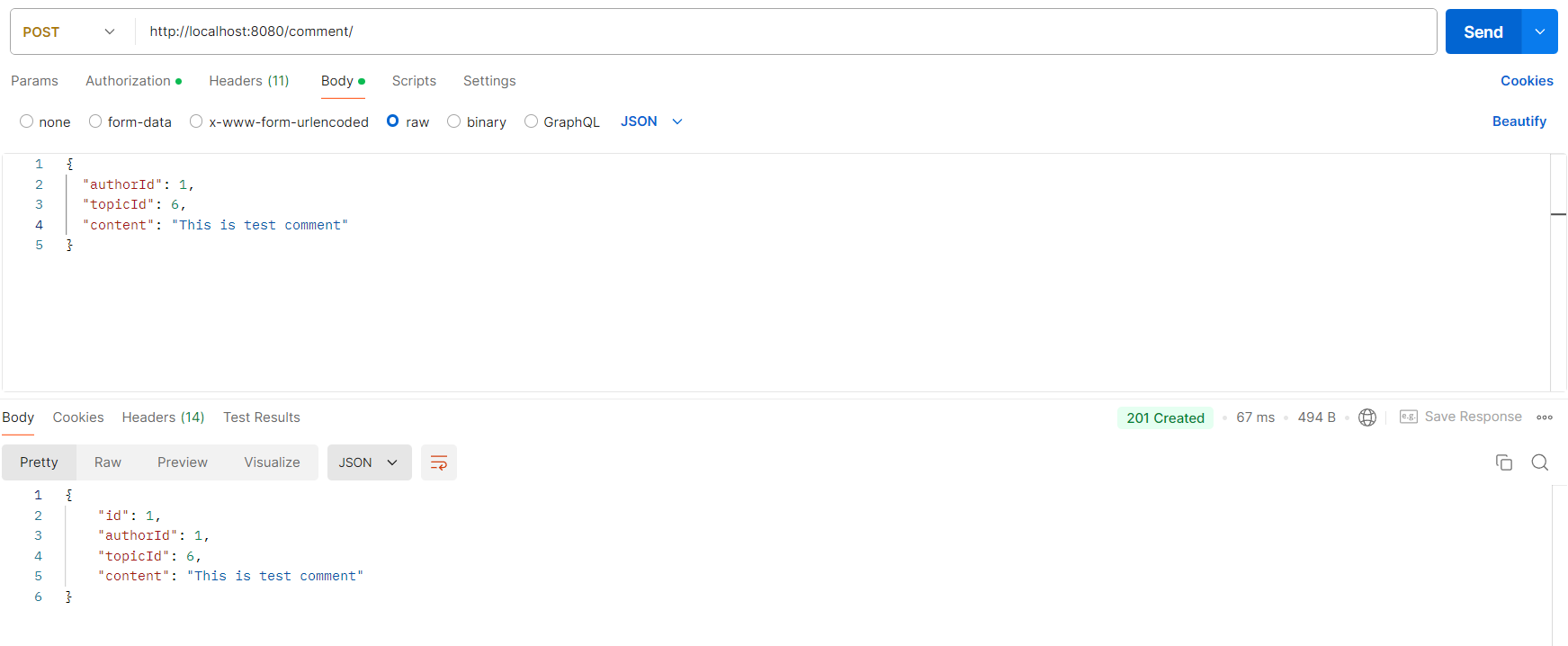
Delete topic:



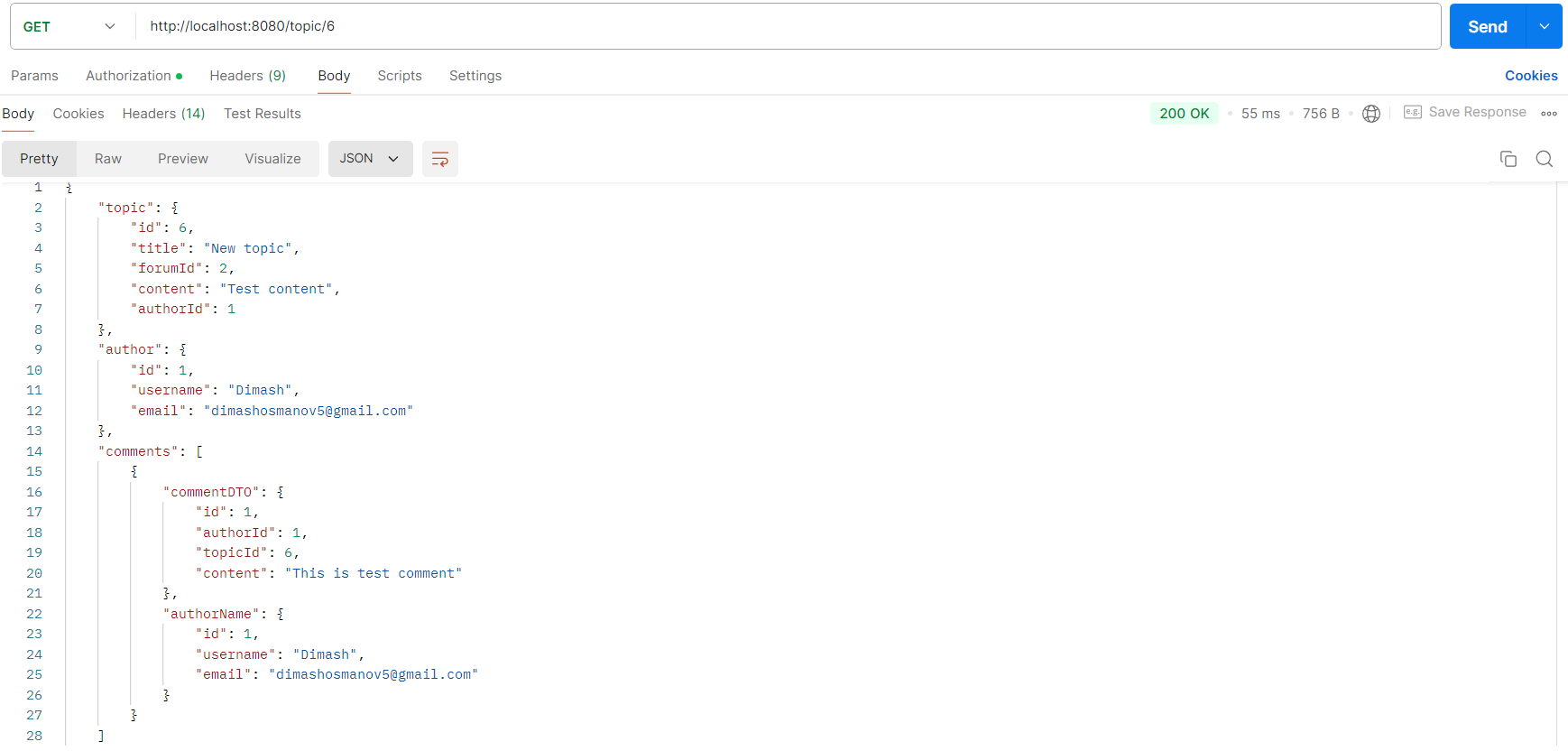
### Comments

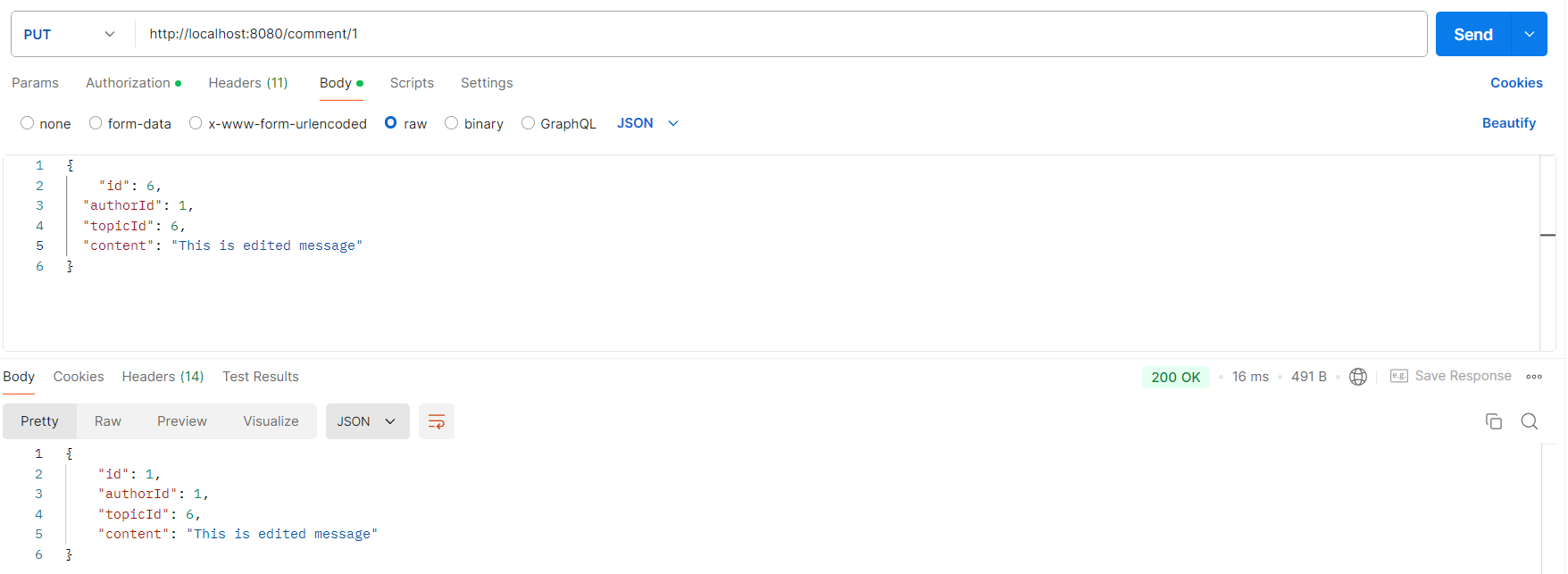
Users can post messages within topics and comment on existing messages. This feature allows for threaded discussions, where each message can have multiple comments. The system supports nested commenting, making it easy to follow conversations.

**Create message:**

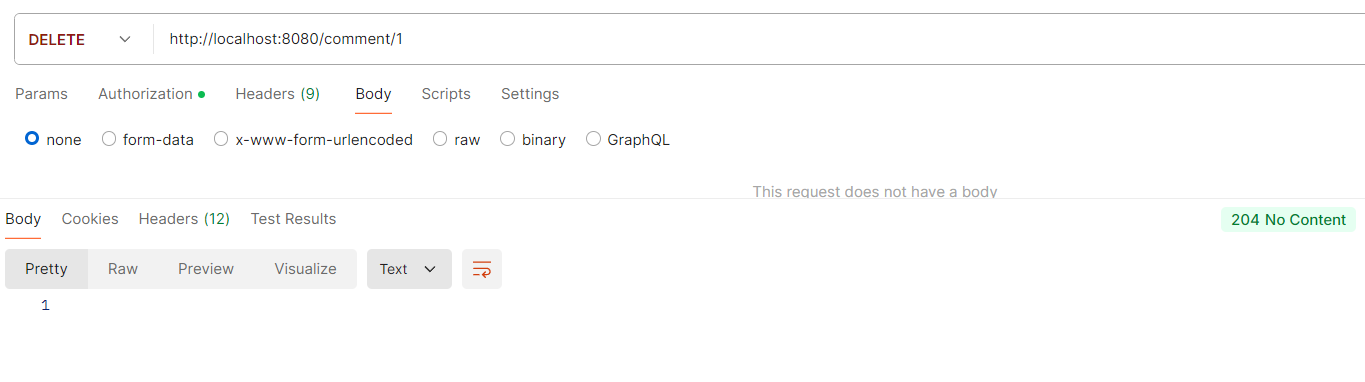
****

**Read messages:**

****

**Update message:  
**

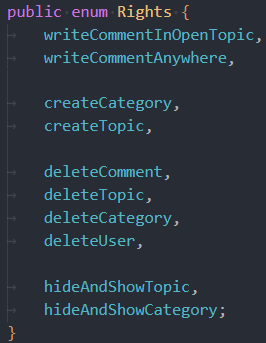
**Delete message:**

****

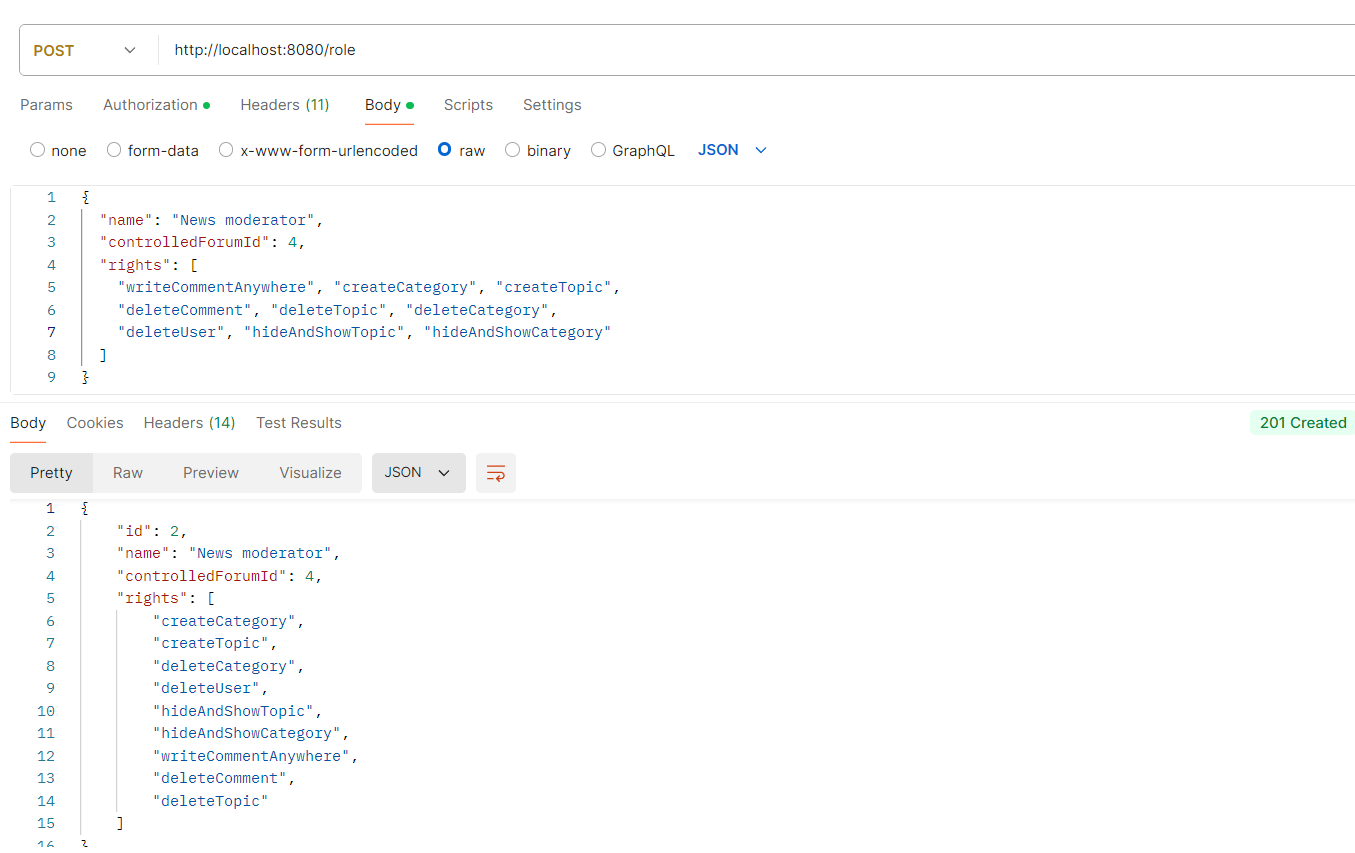
### Roles and Permissions

The system supports a **role-based access control (RBAC)** model. Users are assigned specific roles that determine their access level. In this project will be dynamic roles. Admin can create own roles with some permissions.

Admin can select what permissions set to role one of these rights:



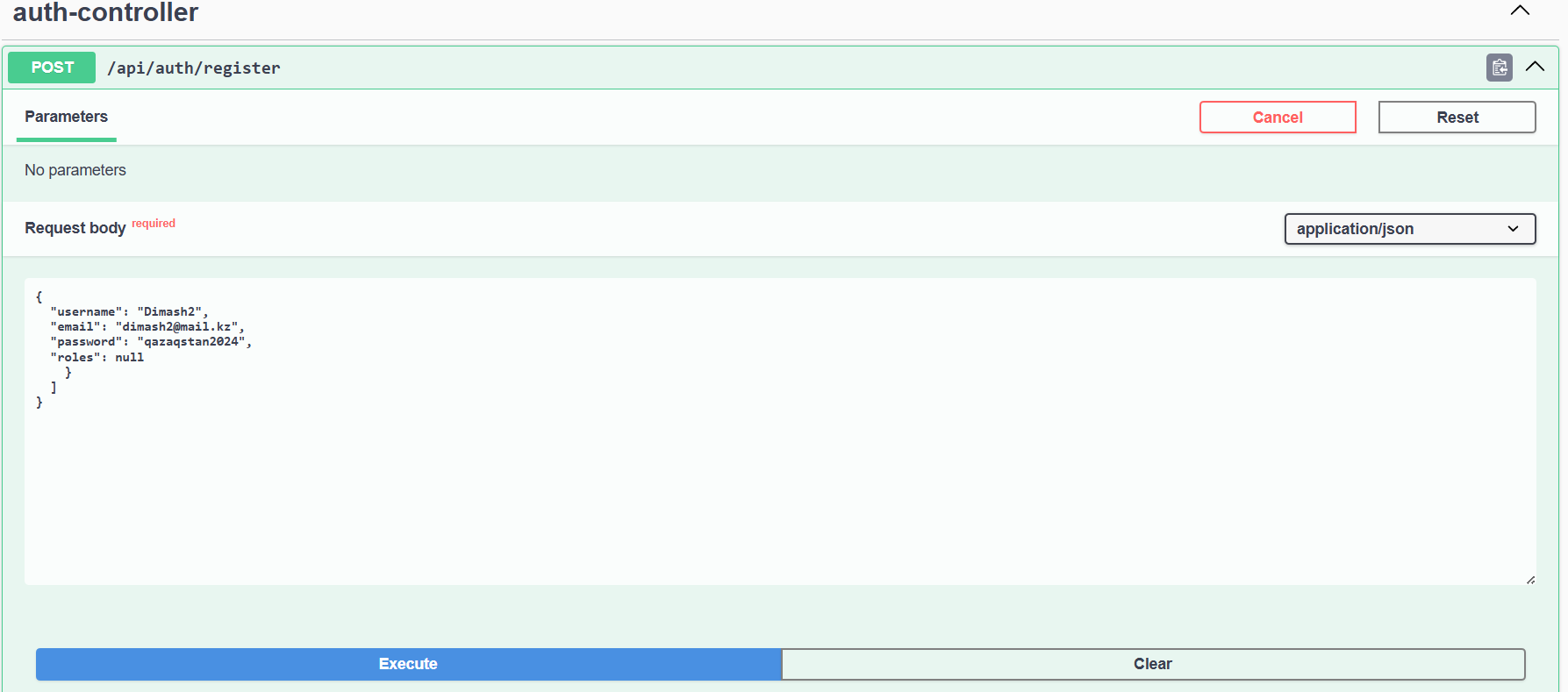
**Create role:**

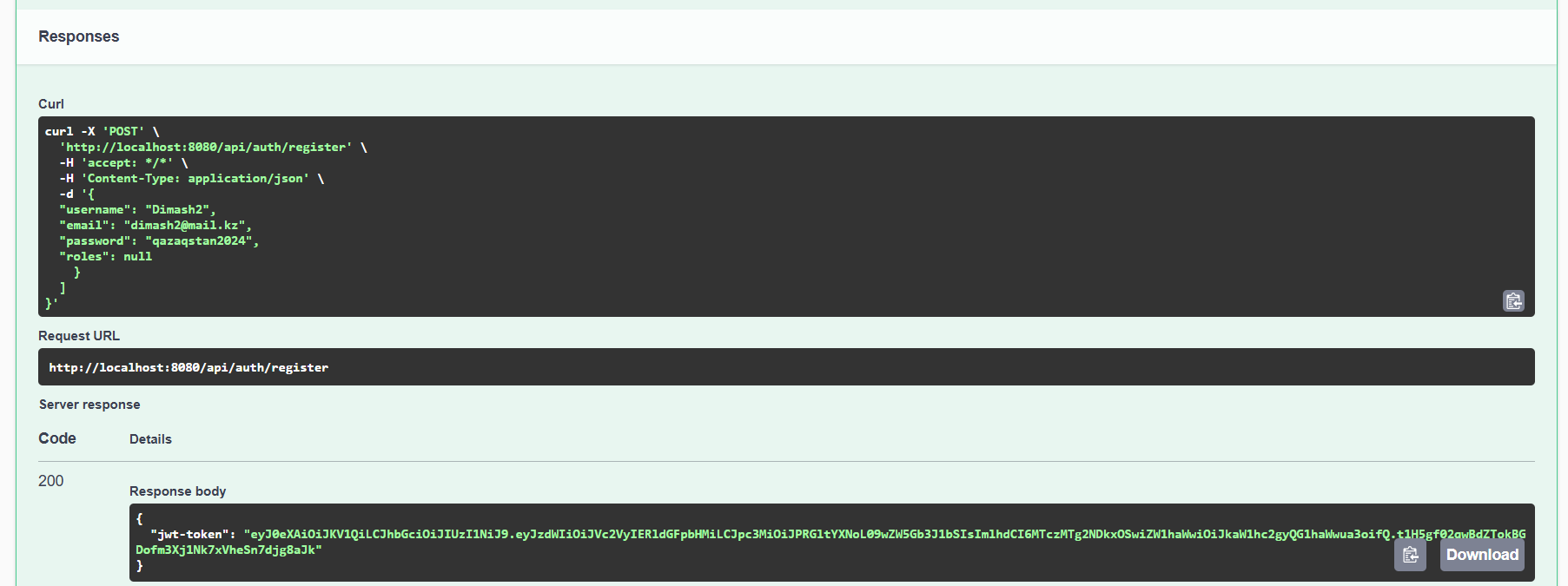
****

### Authentication

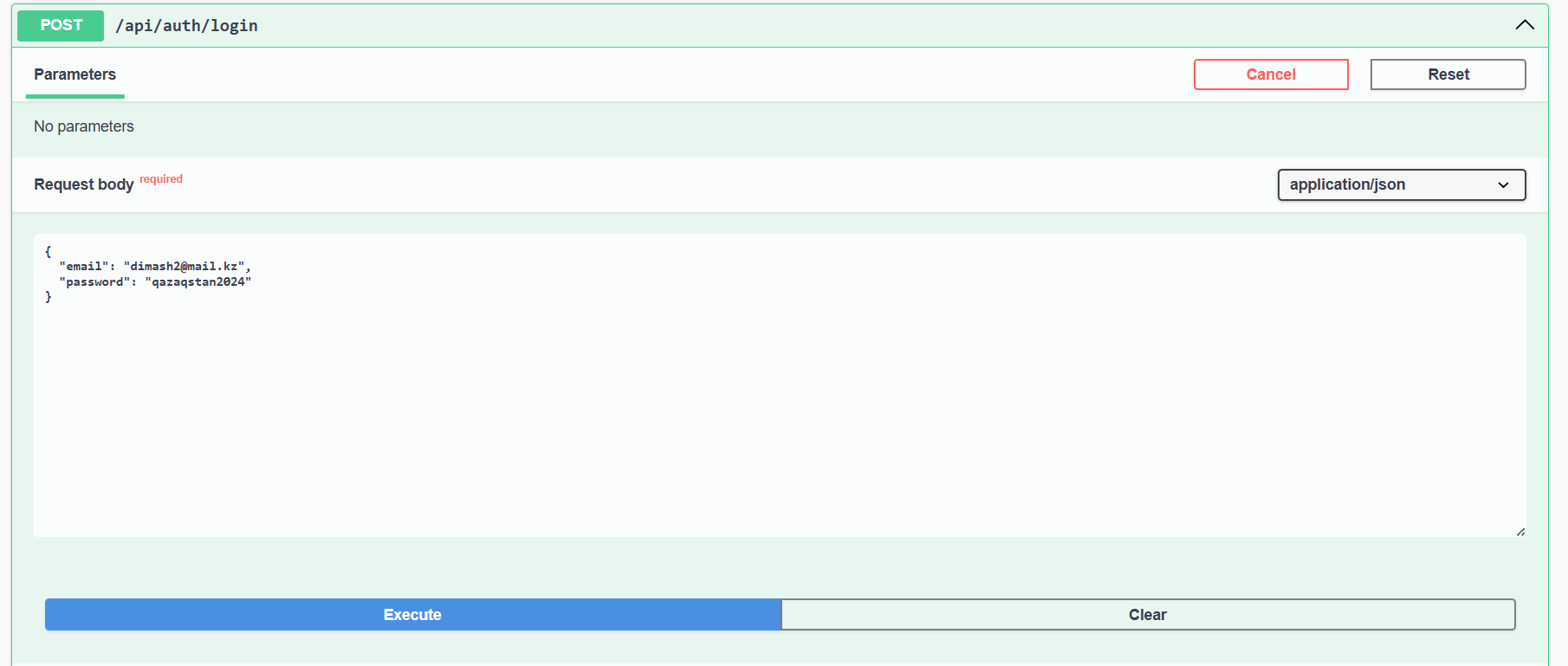
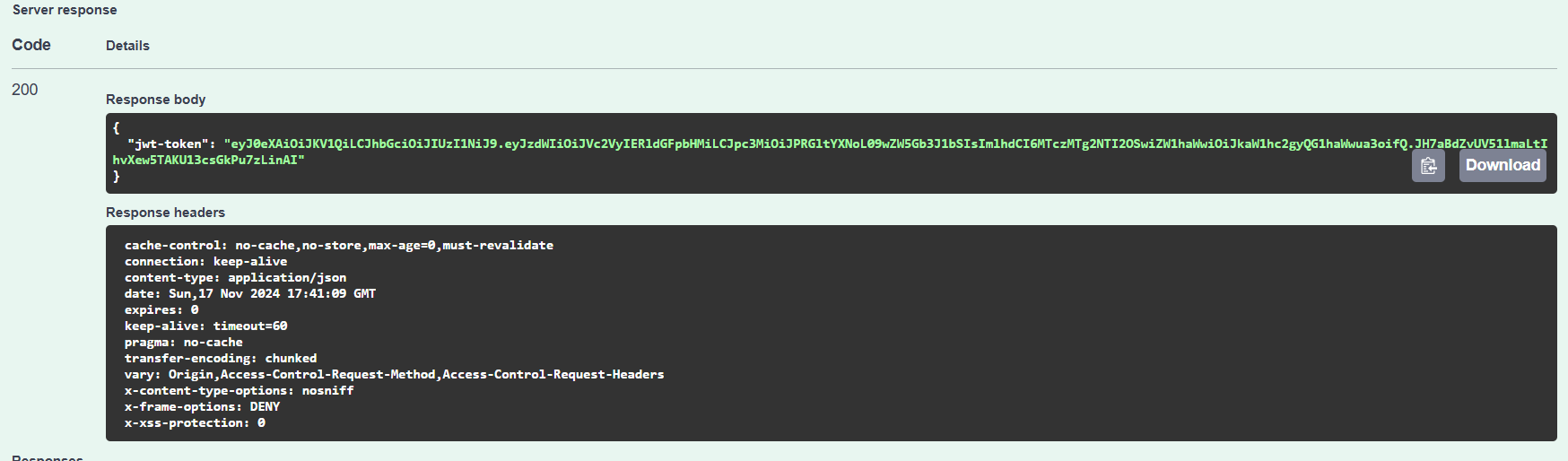
In this project for authentication used JWT tokens. For create token user need to complete registration or login in system. For authentication has a AuthController with 2 endpoints.

**Register:**

****

****

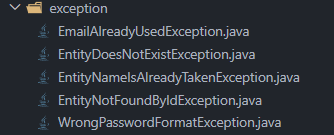
**Login:**

**** ****

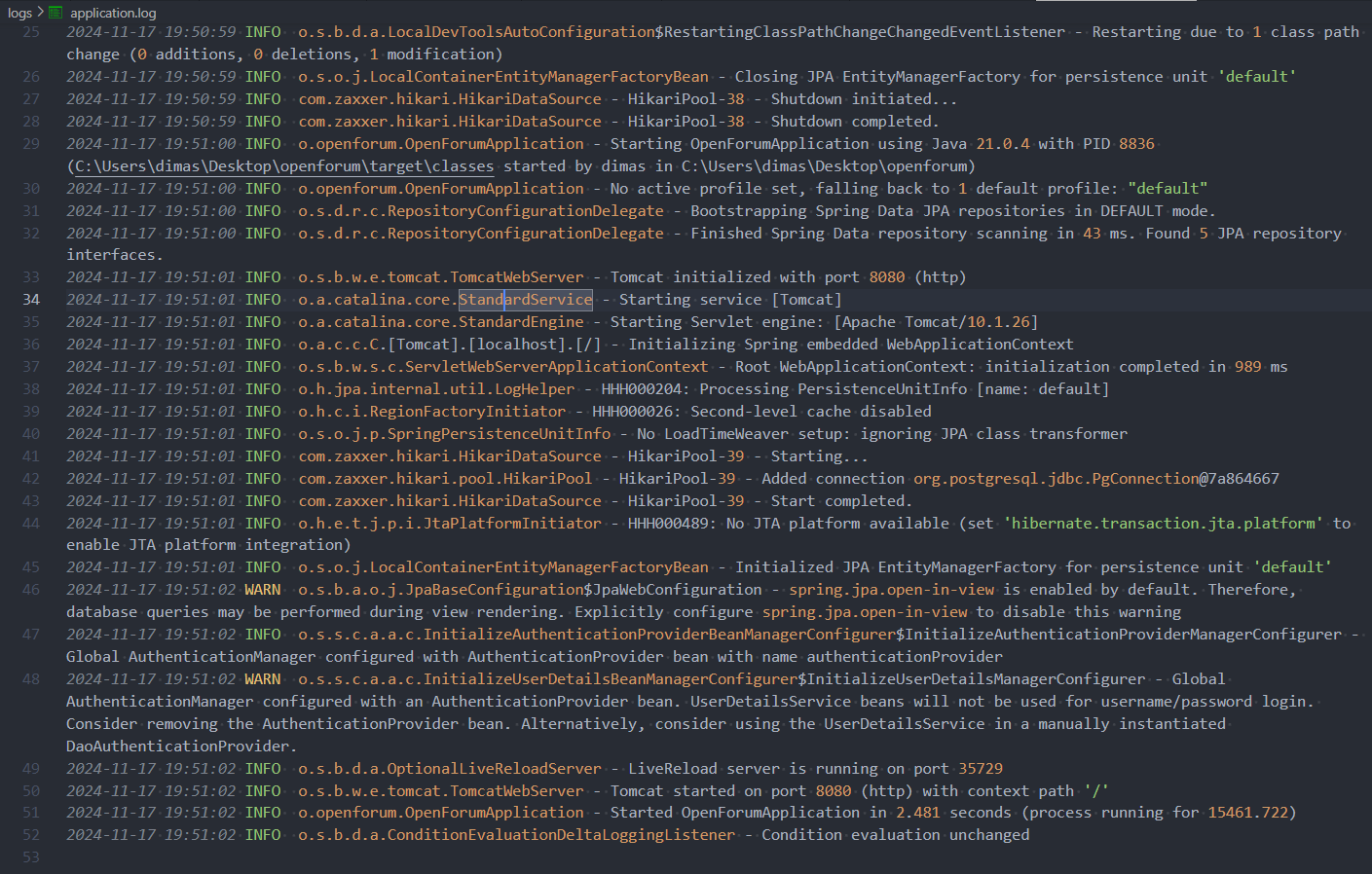
### System Logs and Error Handling

The application includes logging for all major actions, such as user logins, category/topic creation, and errors. This helps in debugging and monitoring system activity.

Custom exceptions:



Project logs will saved in file with path “/logs/application.log”

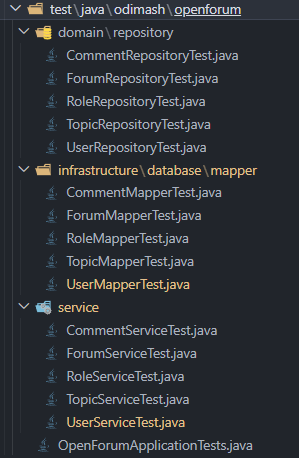


## Testing

### Testing Methodology

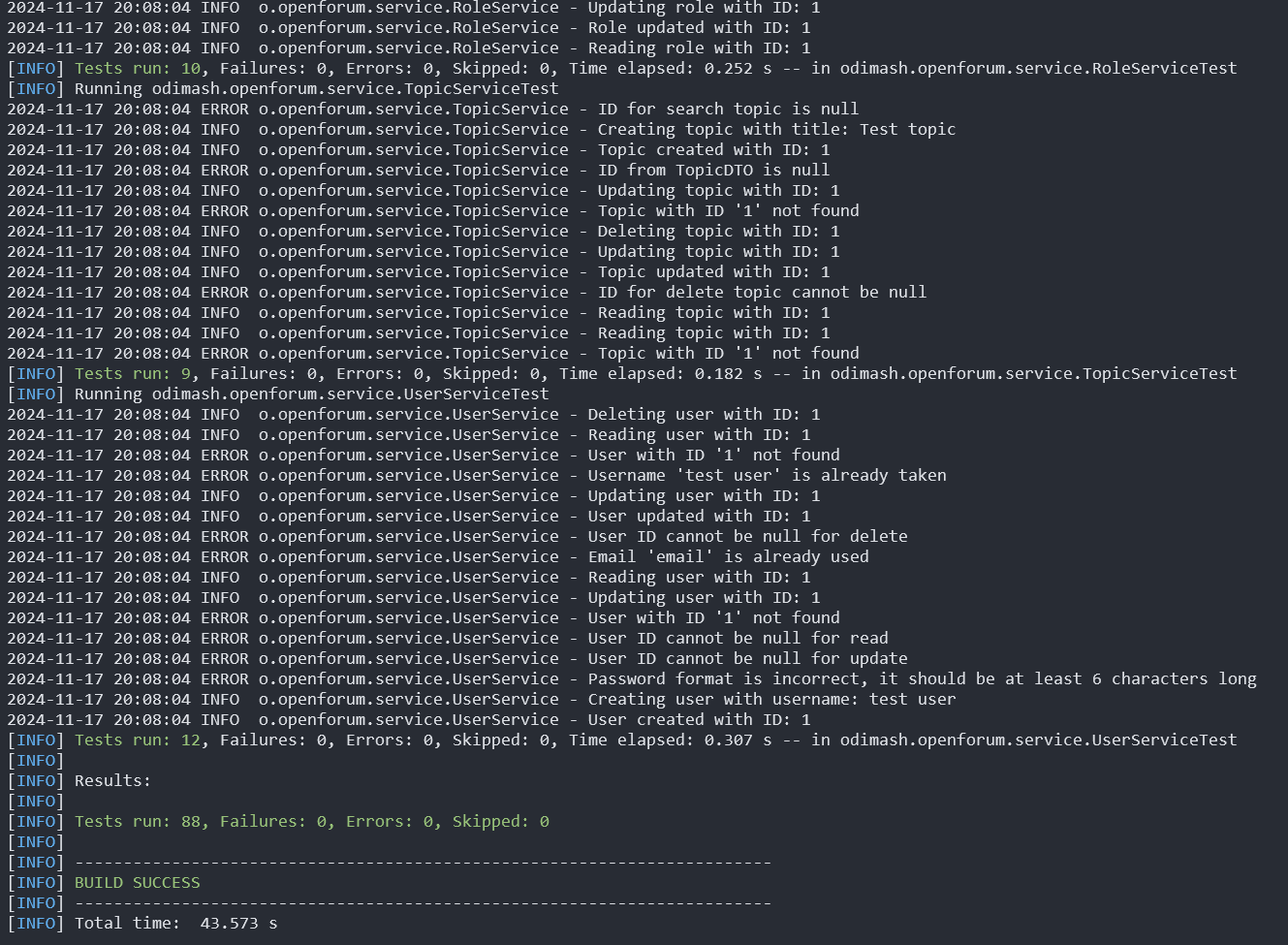
The project followed the **Test-Driven Development (TDD)** methodology, where tests were written before implementing the functionality. This ensured that all components of the system were thoroughly tested and met the requirements.

Tests were written for: - **Unit tests**: Testing individual components, such as the service and repository layers.



### Test Coverage

Test coverage was measured to ensure that critical paths of the application were tested. The code coverage report showed that all key features, including were thoroughly tested.



### Performance Testing

Performance testing was conducted to evaluate how the system handles multiple concurrent users. The system showed promising results under load, with no significant performance degradation observed.

*Insert screenshot of performance test results*

## Appendices

### Appendix 1: Project Structure

### src

### ├───main

### │ ├───java

### │ │ └───odimash

### │ │ └───openforum

### │ │ ├───config

### │ │ ├───domain

### │ │ │ ├───entity

### │ │ │ └───repository

### │ │ ├───exception

### │ │ ├───infrastructure

### │ │ │ ├───controller

### │ │ │ ├───database

### │ │ │ │ ├───dto

### │ │ │ │ └───mapper

### │ │ │ └───viewdata

### │ │ └───service

### │ └───resources

### │ ├───static

### │ └───templates

### └───test

### └───java

### └───odimash

### └───openforum

### ├───domain

### │ └───repository

### ├───infrastructure

### │ └───database

### │ └───mapper

### └───service

### Conclusion

The development of the forum project has been an exciting and educational journey, involving the implementation of key features such as user management, authentication, role-based access control, and database management. The project was structured using a layered architecture, which allowed for clear separation of concerns and facilitated maintainability. By leveraging Java with the Spring Boot framework, we were able to create a robust and scalable solution for a forum application.

**Current Achievements:**

1. **User Authentication**: The implementation of JWT-based authentication has ensured secure access to the platform. This allows users to log in and access protected resources with a token, enhancing the overall security of the application.
2. **Role-Based Access Control (RBAC)**: We successfully integrated role-based access control, allowing for the customization of user permissions within the application. This ensures that only authorized users can access specific areas of the forum, enhancing both security and user experience.
3. **DTO and Entity Management**: The use of Data Transfer Objects (DTOs) and entities has simplified the communication between the layers of the application, allowing for a clean and efficient flow of data.
4. **Database Design and Integration**: The PostgreSQL database was successfully integrated into the application. The database design ensures scalability, flexibility, and consistency of the data, making it easy to handle a large number of users and forum posts.
5. **Logging and Exception Handling**: A comprehensive logging system was implemented to monitor the application’s behavior, while custom exception handling ensures that users receive informative error messages without exposing sensitive information.

**Current Challenges:**

While the forum project has achieved its core functionality, there are a few areas that still require attention:

1. **Scalability**: As the forum grows in terms of users and posts, the application may face performance issues, particularly in handling large datasets. Optimizing database queries and introducing caching mechanisms will be necessary to handle a growing user base efficiently.
2. **Testing**: While many core functionalities have been tested, there is still room to improve test coverage. Specifically, integration tests and end-to-end tests could be added to ensure that the entire system works seamlessly.
3. **Front-End Implementation**: The project currently lacks a front-end interface, which limits the overall user experience. The next step should be to implement a user-friendly and responsive front-end using technologies like React or Angular.
4. **Search and Sorting**: Currently, the search and sorting functionalities are basic and could be enhanced. Adding more advanced search capabilities (e.g., by date, user, or topic) and improving the performance of these features would provide a better user experience.
5. **Security Enhancements**: Although JWT authentication is implemented, additional security measures, such as two-factor authentication (2FA), could be added to further enhance the platform’s security.
6. **User Notifications**: Implementing a notification system for users, such as email alerts for new posts or replies, would improve engagement and user experience.

**Future Improvements:**

Looking ahead, the following features and improvements are planned for future development:

1. **Microservices Architecture**: To improve scalability and maintainability, we may refactor the application into microservices. This will allow each component (e.g., user management, posts, comments) to be handled independently, enabling more efficient development and scaling.
2. **Real-Time Features**: Adding real-time features such as live chat or notifications using WebSockets would enhance user interaction and engagement within the forum.
3. **Mobile Application**: A mobile version of the forum could be developed, allowing users to access the forum from their smartphones. This would make the platform more accessible and expand its user base.
4. **Internationalization (i18n)**: To support users from different regions, the forum could be localized in multiple languages, broadening its appeal and accessibility.
5. **Advanced Search Features**: Improving the search functionality with more advanced filters and capabilities will help users find relevant content more quickly and efficiently.

In conclusion, while the forum project has achieved a solid foundation with essential features and functionality, there is still significant room for growth and improvement. Future enhancements will focus on performance optimization, user experience, and additional features that will make the forum more engaging, secure, and scalable.