**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with ‘A’ grade,

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-111



**Seminar Report**

**on**

**“IMPLEMENTATION OF REST API USING SPRING BOOT”**

Submitted by

**Jigyasa Gupta (1DS20CS090)**

**Inchara B K (1DS20CS087)**

**M Nikitha (1DS20CS110)**

**Sixth Semester B.E (CSE)**

**2023-2024**

Under the guidance of

**Prof. Nidhishree**

**Asst. Professor**

**Dept. of CSE**

**DSCE, Bangalore**

**Department of Computer Science and Engineering**

**Dayananda Sagar College of Engineering**

**Bangalore-78**

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

DAYANANDA SAGAR COLLEGE OF ENGINEERING

Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore - 560078

**Department of Computer Science & Engineering**

****

**CERTIFICATE**

This is to certify that the project entitled **Implementation of REST API using Spring Boot** is a bonafide work carried out by **INCHARA BK [1DS20CS087], JIGYASA GUPTA [1DS20CS090], M NIKITHA [1DS20CS110]** in partial fulfilment of 6th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2022-23.

**Prof. Nidhishree Dr.Ramesh Babu Dr.B G Prasad**

Asst. Professor, Vice principal & Head Principal,

Department of CSE, Department of CSE, DSCE

DSCE DSCE

Signature.................... Signature.................... Signature.................

Name of the Examiners: Signature with date:

1. .................................... ………………………………
2. .................................... ……………………………….

**ACKNOWLEDGEMENT**

We are pleased to have successfully completed the Mini project **“Implementation of REST API using Spring Boot”**. We thoroughly enjoyed the process of working on this project and gained a lot of knowledge doing so.

We would like to take this opportunity to express our gratitude to **Dr. B G Prasad**, Principal of DSCE, for permitting us to utilize all the necessary facilities of the institution.

We also thank our respected Vice Principal, HOD of Computer Science & Engineering, DSCE, Bangalore, **Dr. Ramesh Babu D R**, for his support and encouragement throughout the process.

We are immensely grateful to our respected and learned guide, **Prof. Nidhishree**, Asst. Professor, CSE, DSCE for his/her valuable help and guidance. We are indebted to them for their invaluable guidance throughout the process and their useful inputs at all stages of the process.

We also thank all the faculty and support staff of Department of Computer Science, DSCE. Without their support over the years, this work would not have been possible.

Lastly, we would like to express our deep appreciation towards our classmates and our family for providing us with constant moral support and encouragement. They have stood by us in the most difficult of times.

Inchara BK , 1DS20CS087

Jigyasa Gupta , 1DS20CS090

M Nikitha , 1DS20CS110

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL. NO** | **CONTENT** | **PG. NO** |
|  | Abstract | 5 |
|  | Introduction | 6-7 |
|  | Literature Survey | 8-9 |
|  | System design & Methodology | 10-12 |
|  | Snapshots and Results | 13-15 |
|  | Conclusion & Future Enhancements | 16 |
|  | References | 17 |
|  | Appendix | 18-19 |

**ABSTRACT**

The project aims to implement a RESTful API using Spring Boot to perform CRUD (Create, Read, Update, Delete) operations. The API provides a standardized and efficient way to interact with a database or data source, allowing clients to manipulate resources using HTTP methods such as GET, POST, PUT, and DELETE. The implementation utilizes the Spring Boot framework, which simplifies the development process by providing out-of-the-box features and configurations. The project follows the principles of REST (Representational State Transfer), allowing clients to access and modify resources through well-defined endpoints. The core components of the implementation include entity classes, a repository interface, and a controller class. Entity classes represent the data models and are annotated with appropriate annotations for mapping to a database. The repository interface extends Spring Data JPA's `JpaRepository` to provide ready-to-use CRUD operations for the entities. The controller class handles incoming HTTP requests and interacts with the repository to perform CRUD operations on the resources. Endpoints are defined using appropriate annotations, such as `@GetMapping`, `@PostMapping`, `@PutMapping`, and `@DeleteMapping`, and request parameters are mapped to method arguments. The controller returns the desired HTTP response, such as the requested resource, a created resource, or a success status.

The project demonstrates the power and simplicity of Spring Boot for building RESTful APIs. It showcases how to create, retrieve, update, and delete resources using a combination of entity classes, repositories, and controllers. The API provides a flexible and scalable solution for managing data and allows clients to integrate and interact with the system easily. By leveraging the capabilities of Spring Boot and adhering to RESTful principles, this project enables developers to quickly develop and deploy robust APIs with CRUD functionality. It serves as a foundation for building more complex systems that require data manipulation through standardized HTTP operations.

**Chapter 1**

**INTRODUCTION**

In today's interconnected world, where applications and services need to seamlessly communicate with each other, the need for a standardized approach to enable such interactions has become increasingly vital. This is where the concept of REST API comes into play. REST, which stands for Representational State Transfer, has emerged as a dominant architectural style for designing networked systems. It offers a simple yet powerful set of principles that allow developers to build scalable and interoperable web services. REST API revolves around the idea of resources, which can be any entity or object that we want to expose to the clients for interaction. These resources are represented by unique identifiers called Uniform Resource Identifiers (URIs), typically expressed as URLs. Through a RESTful API, clients can perform various operations on these resources using standard HTTP methods such as GET, POST, PUT, and DELETE. One of the key principles of REST is statelessness. In a stateless architecture, the server does not store any information about the client's state between requests. Each request from the client contains all the necessary data for the server to process it. This design principle allows for better scalability and reliability, as the server can easily handle requests from multiple clients without the burden of maintaining session state. Another fundamental aspect of REST API is its emphasis on uniform interfaces. RESTful services adhere to a set of well-defined conventions for interacting with resources. The most common convention is the use of HTTP verbs to indicate the desired operation on a resource. For example, a GET request retrieves the representation of a resource, while a POST request creates a new resource. This uniformity simplifies the design and consumption of APIs, as clients can leverage their existing knowledge of HTTP semantics. REST API provides numerous benefits to developers and organizations. Firstly, it promotes loose coupling between the client and server, allowing them to evolve independently. This means that the server's internal implementation can change without affecting the clients, as long as the contract defined by the API remains consistent. Secondly, RESTful services are highly scalable, as they can leverage the statelessness and caching capabilities of HTTP to handle a large number of concurrent requests. Additionally, REST API encourages the use of hypermedia, which allows the server to provide links and metadata within the response to guide clients in their interactions. In terms of usage scenarios, REST API finds extensive application in web development, mobile app development, and integration of different systems. It is particularly well-suited for building distributed systems, microservices architectures, and APIs for external consumption. RESTful services have gained widespread adoption due to their simplicity, scalability, and compatibility with existing web infrastructure. In conclusion, REST API offers a standardized approach for building web services that can seamlessly communicate with each other. Its simplicity, scalability, and emphasis on loose coupling make it a popular choice for designing distributed systems. By understanding the principles and characteristics of RESTful architecture, developers and organizations can leverage its power to create robust and interoperable applications in today's interconnected world. Hibernate is an open-source object-relational mapping (ORM) framework that provides a convenient way to map Java objects to relational database tables. It simplifies the process of storing, retrieving, and manipulating data in a database, allowing developers to focus more on the business logic of their applications.

In the context of the project implementing a REST API using Spring Boot for CRUD operations, Hibernate can be used as the underlying ORM framework to handle the database interactions. Here's an introduction to Hibernate and its key features:

Object-Relational Mapping (ORM):

Hibernate enables developers to map Java classes to database tables and provides mechanisms for automatically generating SQL queries and managing the mapping between objects and database records. This abstraction layer eliminates the need to write low-level SQL queries, allowing developers to work with objects and classes instead.

Configuration:

Hibernate provides a flexible configuration approach, allowing developers to define the database connection details, dialect, caching, and other settings. Configuration can be done using XML files, Java annotations, or a combination of both. In a Spring Boot project, Hibernate can be configured using `application.properties` or `application.yml` files.

Entity Mapping:

With Hibernate, developers can define entity classes that represent the data models of the application. These classes are annotated with Hibernate-specific annotations, such as `@Entity`, `@Table`, and `@Column`, to define the mapping between the Java objects and the database tables, columns, and relationships.

By integrating Hibernate into the project, developers can leverage its powerful features to simplify database operations, enhance performance, and improve maintainability. Hibernate abstracts away the complexities of SQL and provides a high-level, object-oriented approach to data persistence, enabling developers to focus on the core functionality of the application.

**Chapter 2**

**LITERATURE SURVEY**

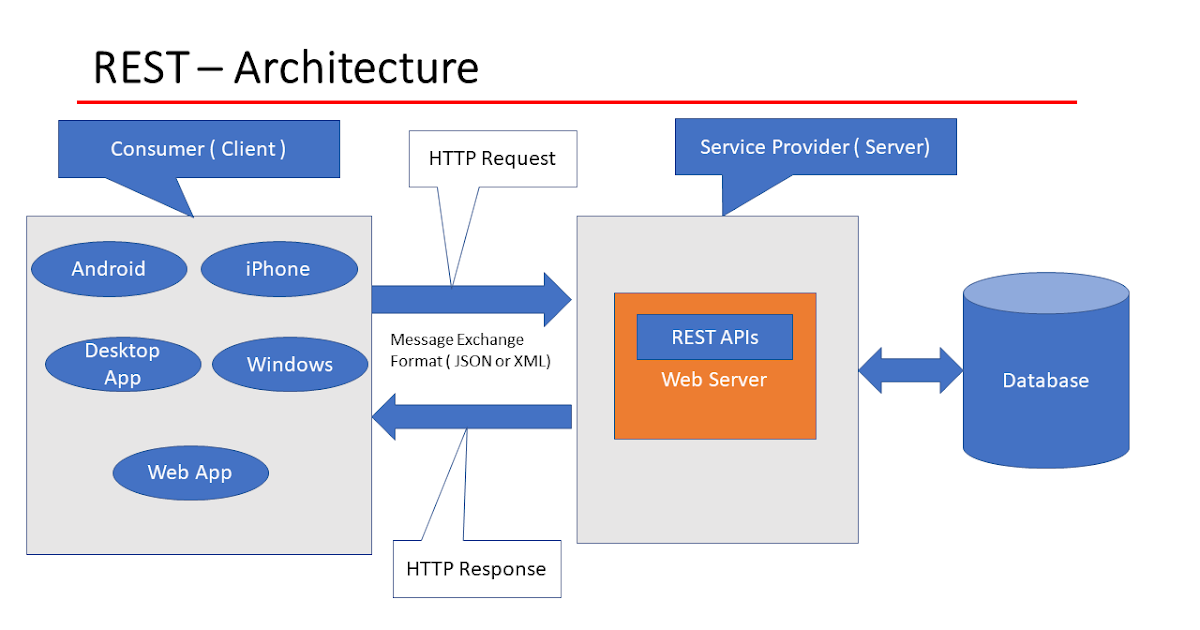
* For the project, Implementation of REST API using spring boot, the following 7 research/survey papers were referred before the implementation process. These papers are studied and tabulated as follows:

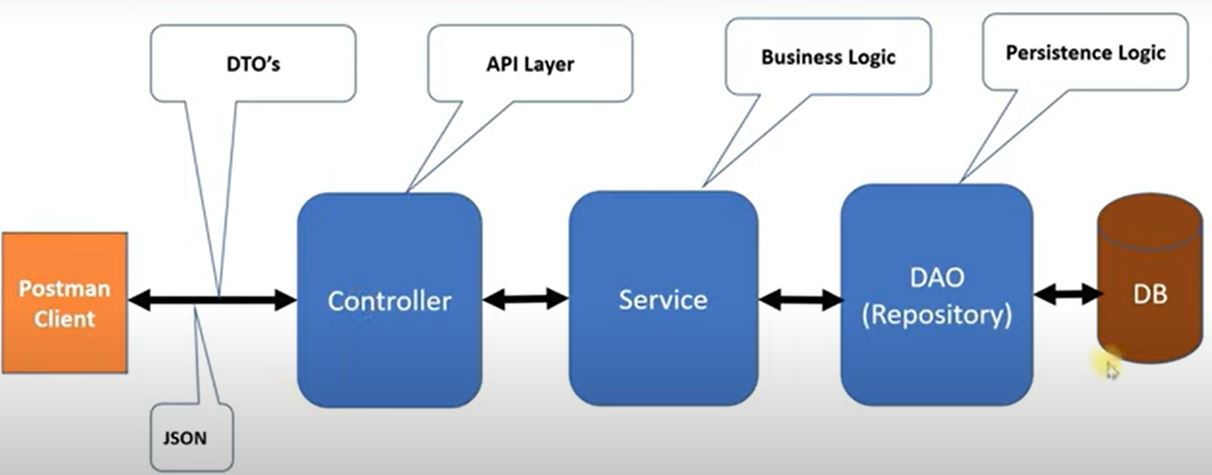
|  |  |  |  |
| --- | --- | --- | --- |
| **Title** | **Author** | **Inference** | **Datasets/Database** |
| 1. An Analysis of Public REST Web Service APIs | Andy Neumann, Nuno Laranjeiro, Jorge Bernardino | core features of the API, such as the number of operations, URI design schema, output format selection  HTTP verbs, implementation of HTTP method overrides, specification of content types in messages, appropriate HTTP status codes in responses, support for input and output formats. | Client-Server principle, Stateless principle |
| 2. A Study of the Effectiveness of Usage Examples in  REST API Documentation | S M Sohan, Frank Maurer, Craig Anslow, Martin P. Robillard | Request method, Request URL, Request headers, Request body | Use of Data Type & Data Format |
| 3. Framework for REST API | Sujan Y M, Dr. Shashidhara H R | RESTFUL Key Elements, Client-Server Model, restful Conversations, HTTP methods for RESTFUL Services Authorization of REST API | Client-Server principle, Stateless principle |
| 4.Client-Server Model | Haroon Shakirat Oluwatosin  School of Computing Universiti Utara Malaysia Kedah, Malaysia | It splits the processing of application across multiple machines  It allows easier sharing of resources from client to servers.  It reduces data replication by storing data on each server instead of client. | Databases as servers |
| 5. JAVA SPRING BOOT REST WEB SERVICE INTEGRATION WITH JAVA  ARTIFICIAL INTELLIGENCE WEKA FRAMEWORK | Željko Jovanović, Dijana Jagodić, Dejan Vujičić, Siniša Ranđić | This paper presents the integration of Java Spring Boot framework with WEKA – Java artificial intelligence  framework. For a presentation, developed REST Web service is demonstrated. As a result, it returns result of prediction  for four prediction algorithms that are used over some sample weather data. | WEKA (Waikato Environment for  Knowledge Analysis)  supports the methods for data  processing and evaluation of learning results |
| 1. HTTP request response architecture | Christopher Liang | SSL  HTTP request  HTTP response | Databases |
| 7. A Comparative study of SOAP vs REST web services provisioning techniques for mobile host | Dr. K S Wagh, Ravindra Thool | SOAP architecture  REST architecture  Selecting the right architecture based on the needs of the project | Databases as servers |

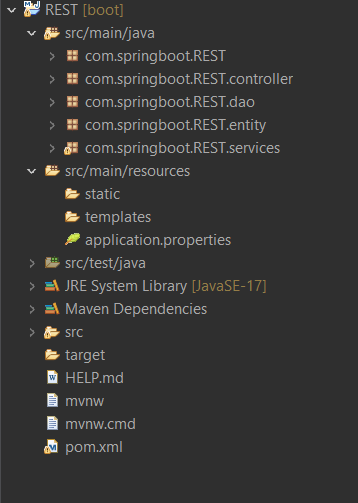
**Chapter 3**

**SYSTEM DESIGN & METHODOLOGY**





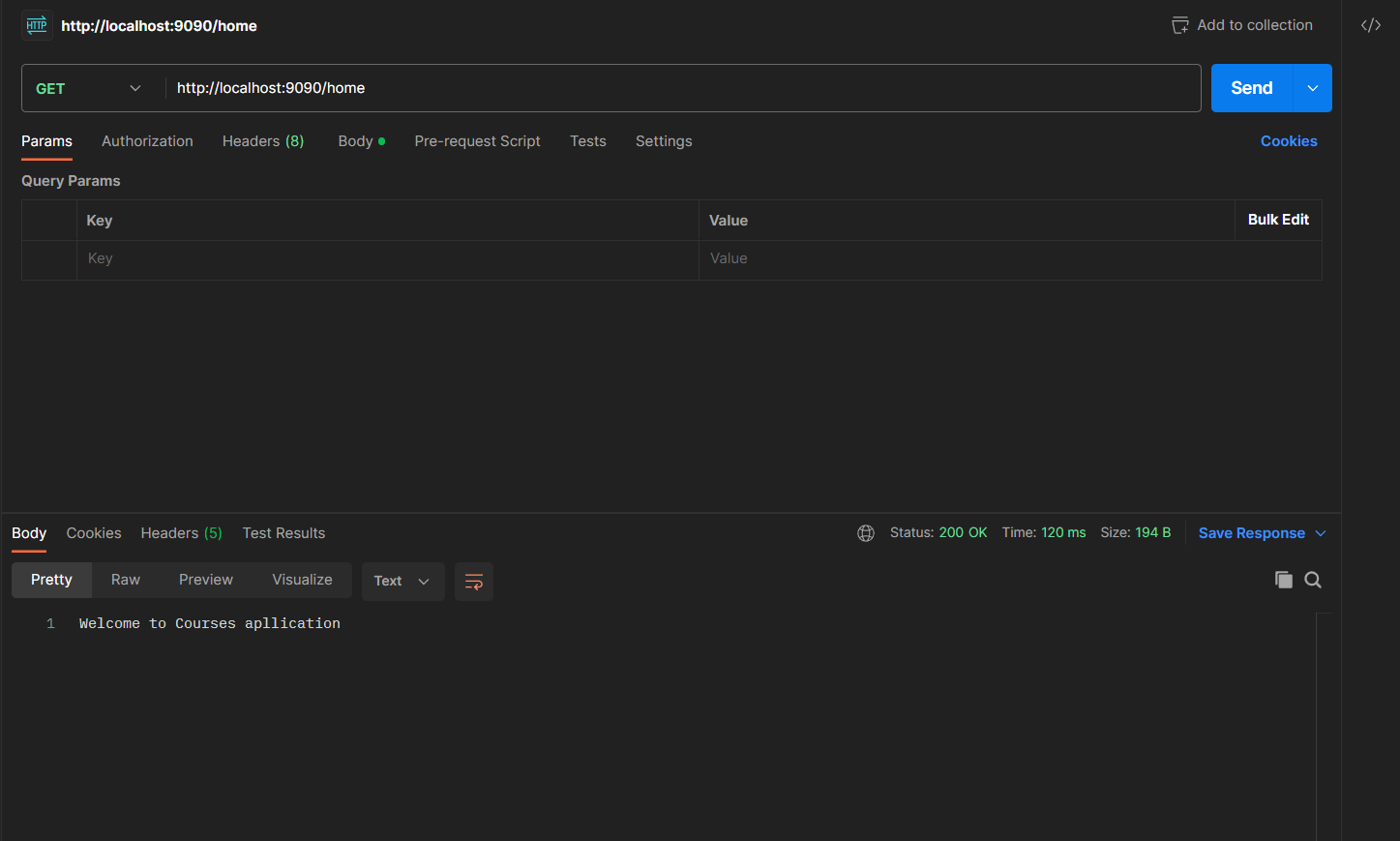
****

****

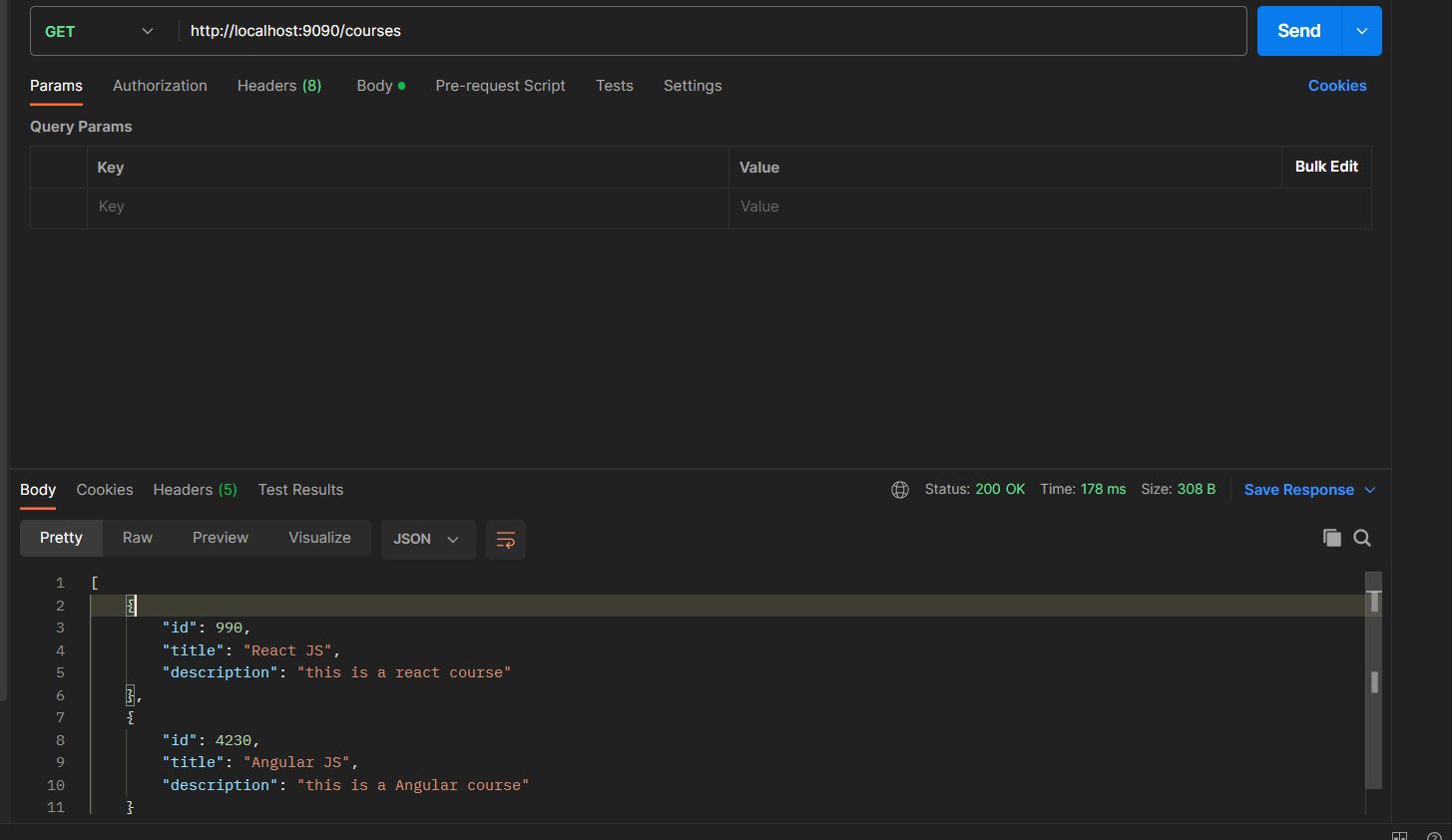
**Chapter 4**

**SNAPSHOTS AND RESULTS**

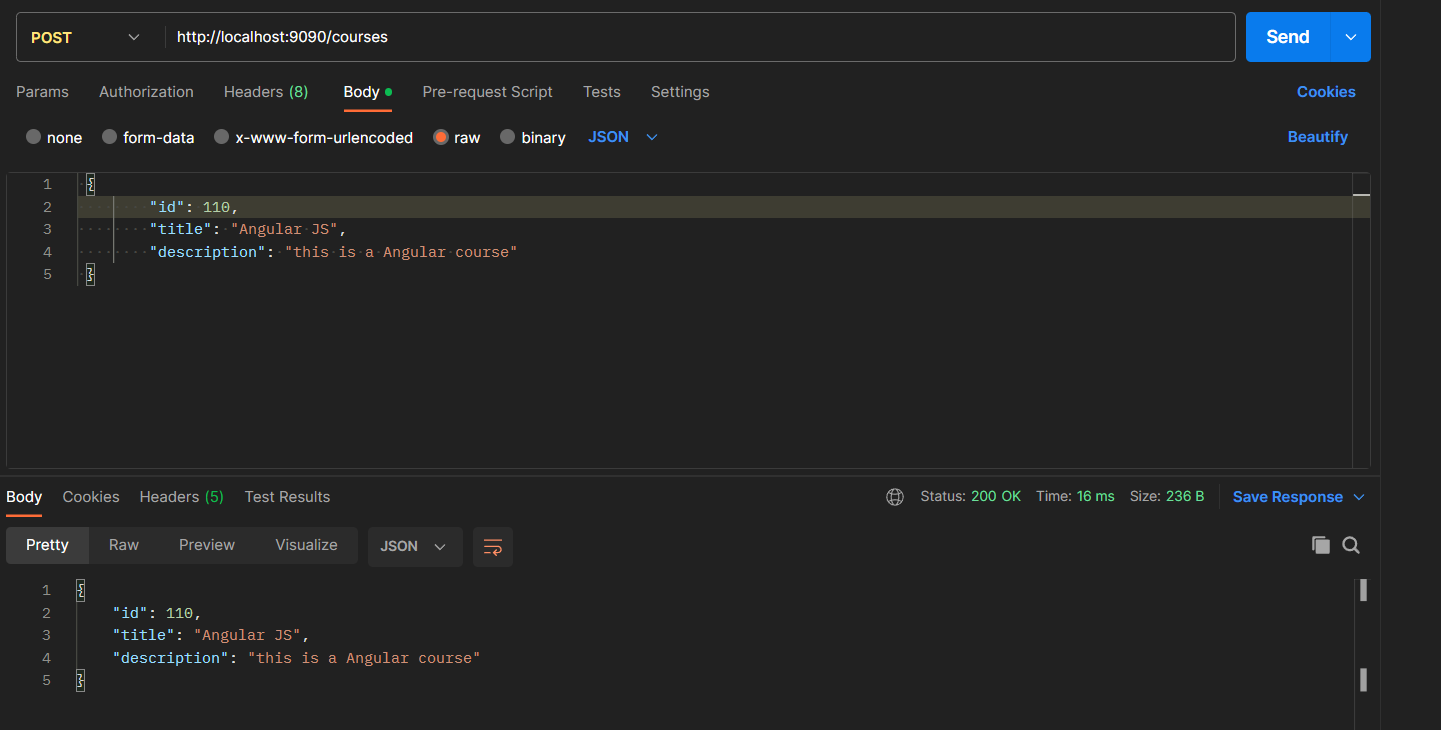
1. **/home**



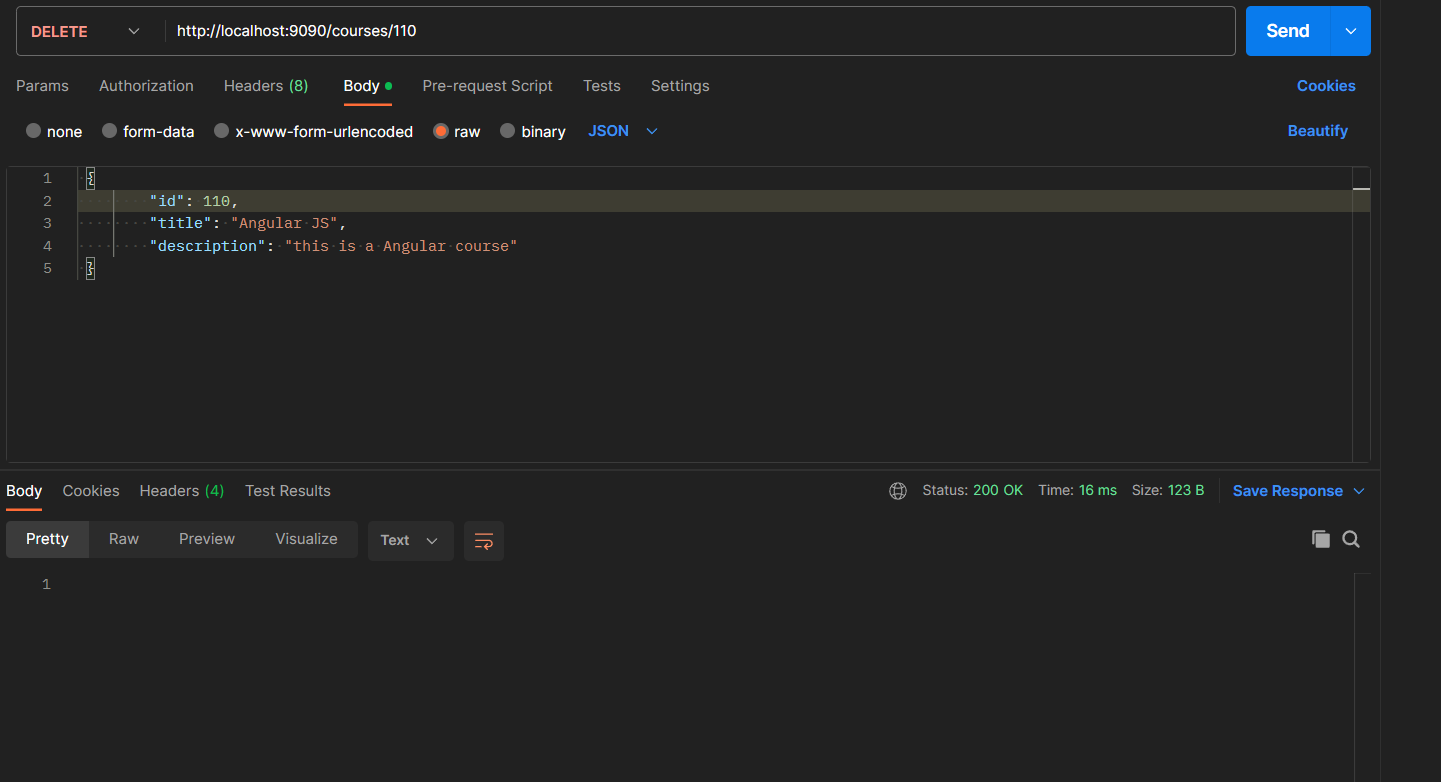
1. **/courses (GetMapping)**

****

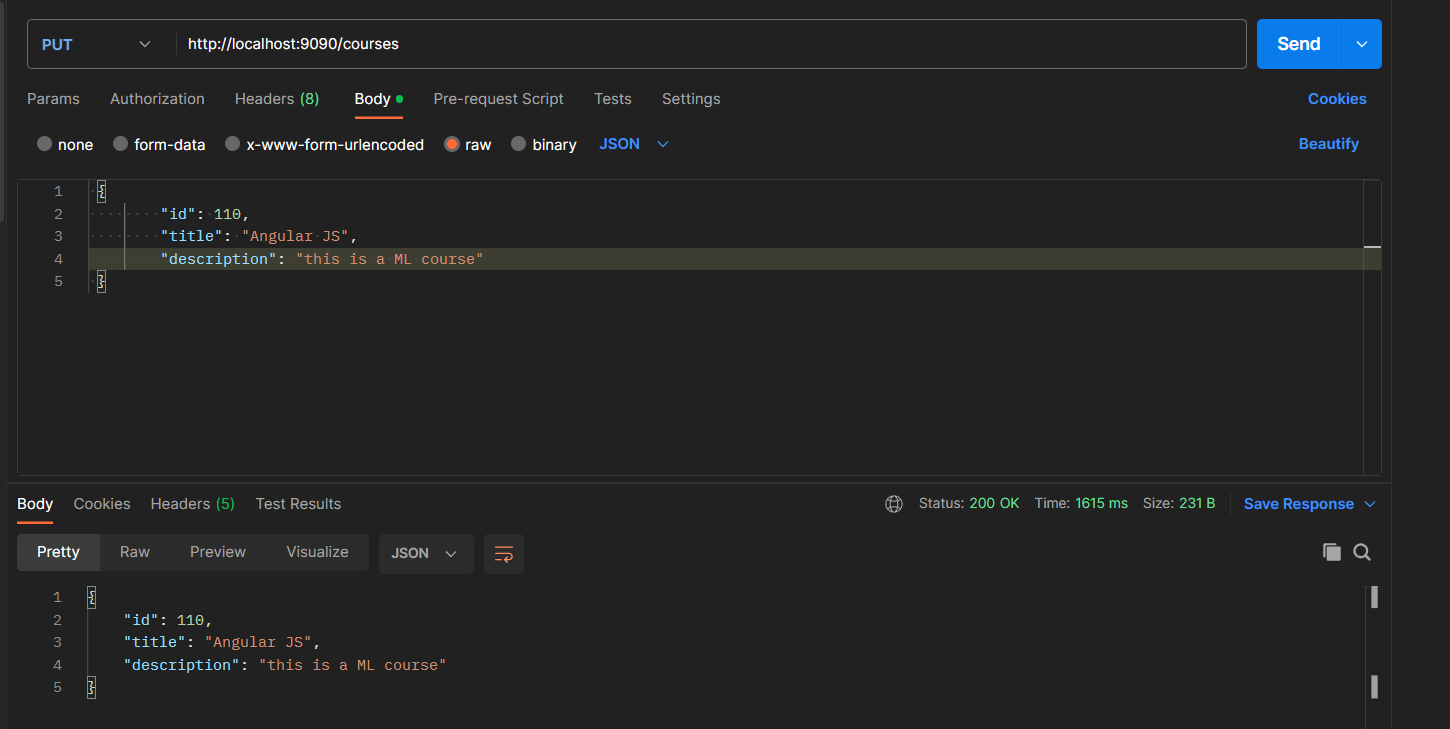
1. **/courses (PostMapping)**

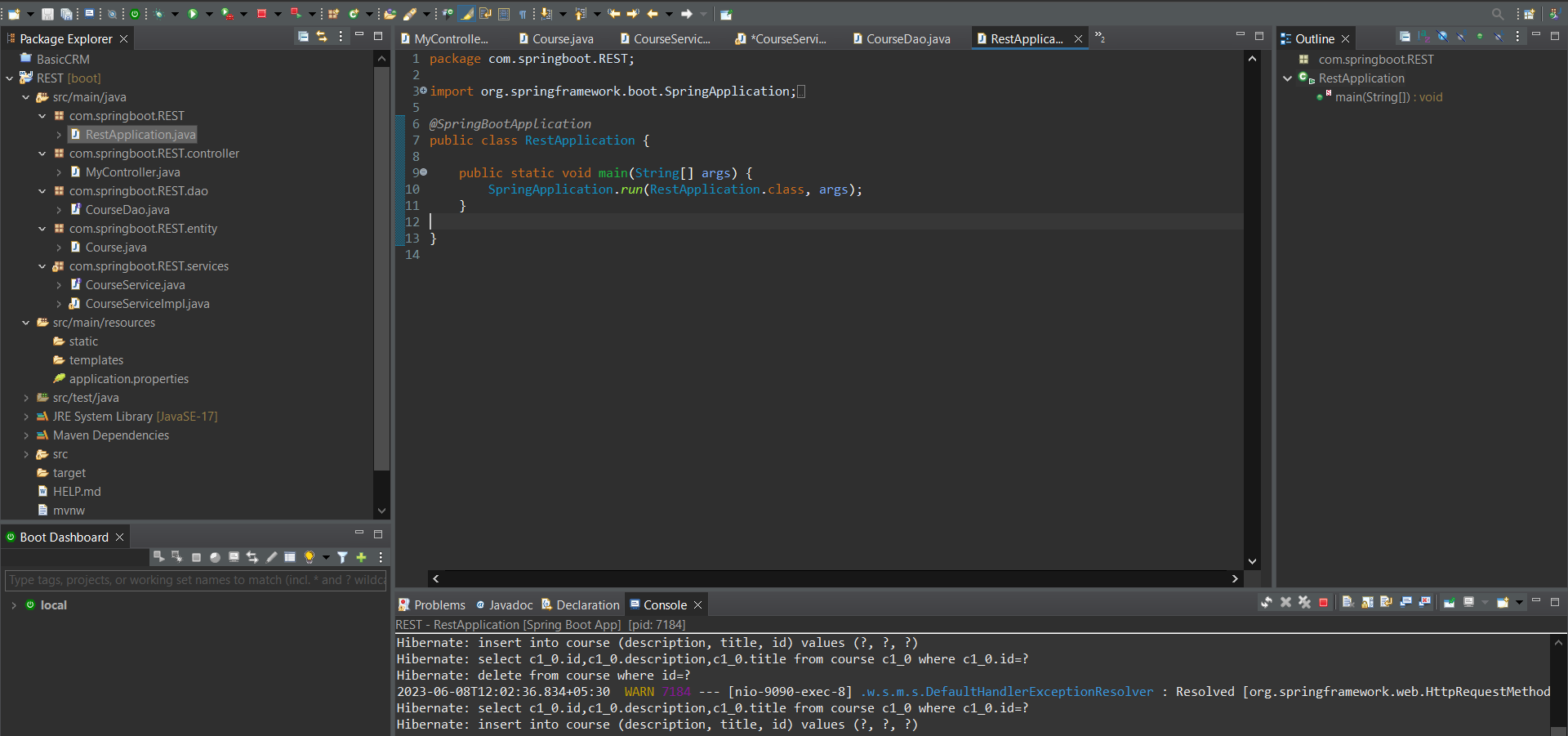


1. **/courses/{courseID} (DeleteMapping)**

****

1. **/courses (PutMapping)**





**Chapter 5**

**CONCLUSION AND**

**FUTURE ENHANCEMENTS**

In conclusion, the project aimed to create a REST API capable of performing CRUD (Create, Read, Update, Delete) operations. The development process was streamlined by utilizing Spring Boot, which provided a high level of abstraction and automation. This resulted in increased developer productivity and accelerated the overall project timeline. REST API offered a simplified and standardized approach to web service development. By adhering to REST principles and utilizing HTTP methods, the API became versatile, lightweight, and easy to maintain. Furthermore, the project emphasized code maintainability and scalability. With the modular and flexible architecture of Spring Boot, future enhancements and feature additions can be seamlessly integrated, ensuring the long-term viability and growth of the API.

While the existing REST API project has achieved its objective of creating a CRUD-capable API with streamlined development using Spring Boot, there are several future enhancements that can be considered to further improve its functionality, performance, and user experience. Some potential enhancements include:

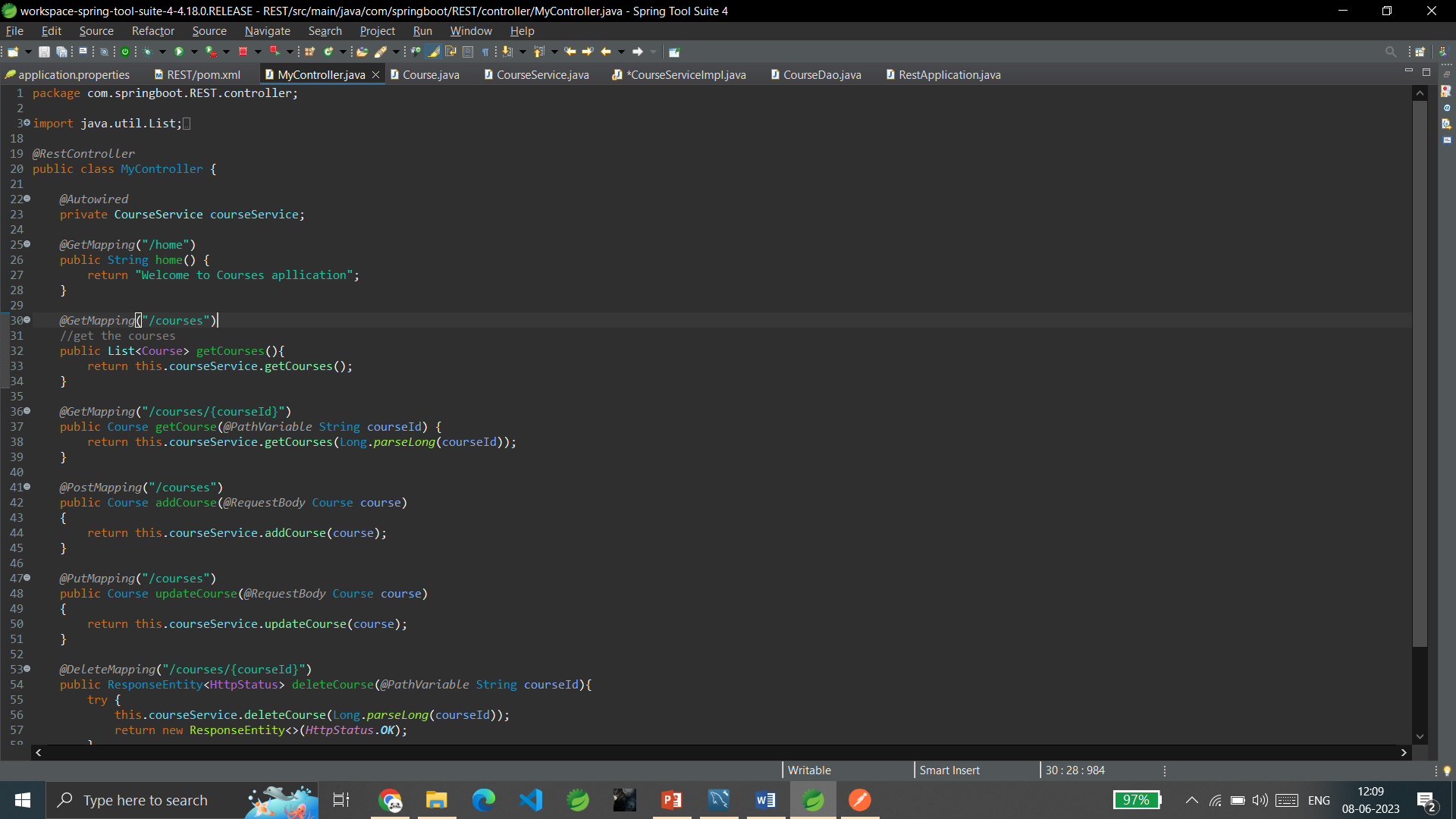
1. Authentication and Authorization: Implementing a robust authentication and authorization mechanism can enhance the security of the API. This could involve integrating authentication protocols like OAuth or JWT (JSON Web Tokens) to ensure secure access to resources and protect against unauthorized usage.
2. Caching: Introducing caching mechanisms, such as leveraging HTTP caching headers or implementing an in-memory cache, can significantly improve API performance and reduce the load on the server. Caching responses for read-heavy operations can help minimize the need for repetitive database queries and enhance the overall responsiveness of the API.
3. Monitoring and Analytics: Incorporating monitoring and analytics tools can provide valuable insights into the API's performance, usage patterns, and potential issues. By collecting metrics such as response times, error rates, and usage statistics, developers can proactively identify and address performance bottlenecks or areas for improvement.
4. Webhooks and Asynchronous Operations: If the API performs time-consuming or resource-intensive operations, introducing support for webhooks or asynchronous processing can enhance its efficiency. This allows clients to subscribe to events or receive callbacks when long-running operations are completed, reducing wait times and improving overall responsiveness.
5. Integration with External Services: Depending on the project requirements, integrating the REST API with other external services, such as third-party APIs, payment gateways, or notification systems, can enhance its functionality and provide additional value to users.

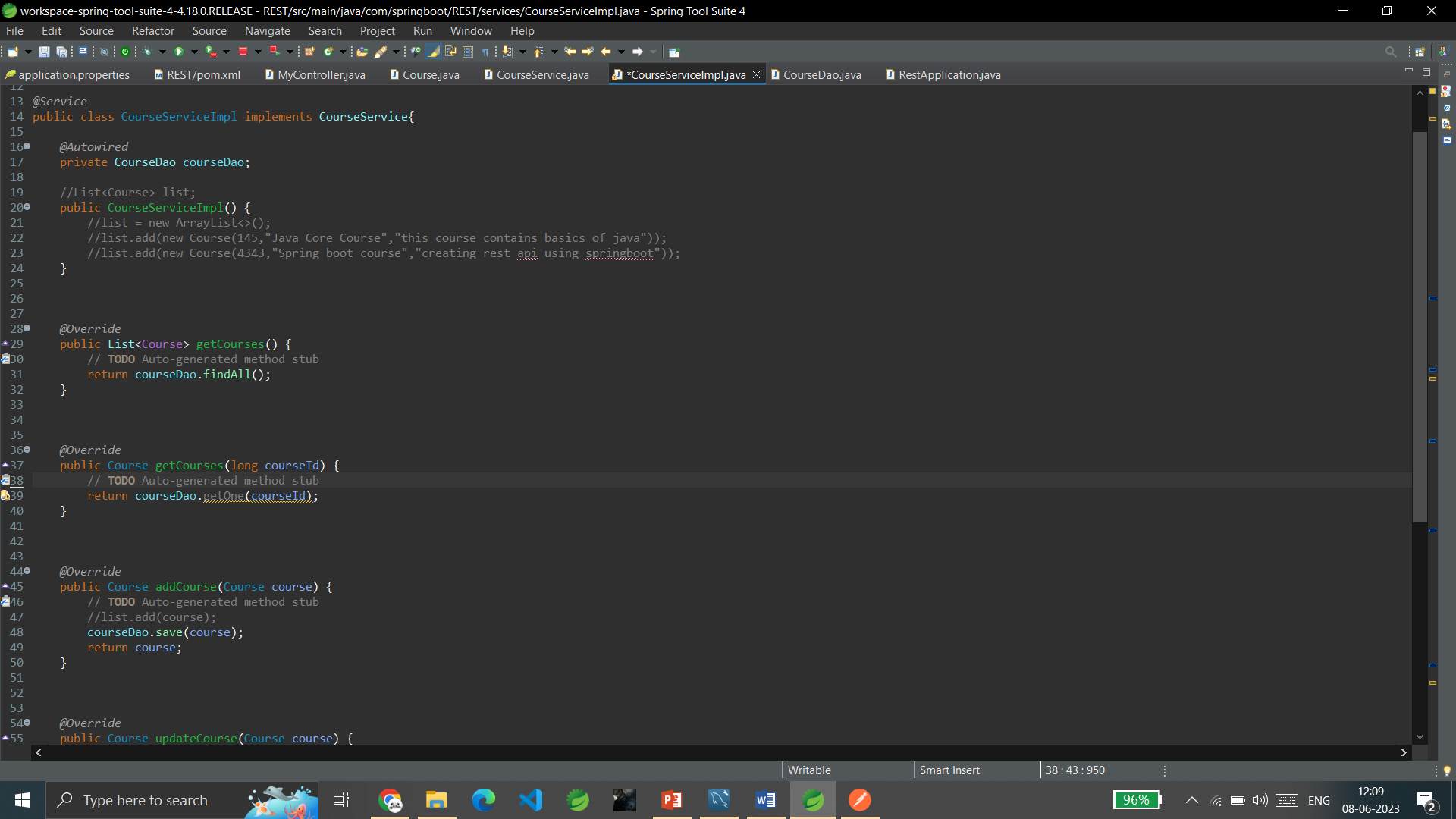
**REFERENCES**

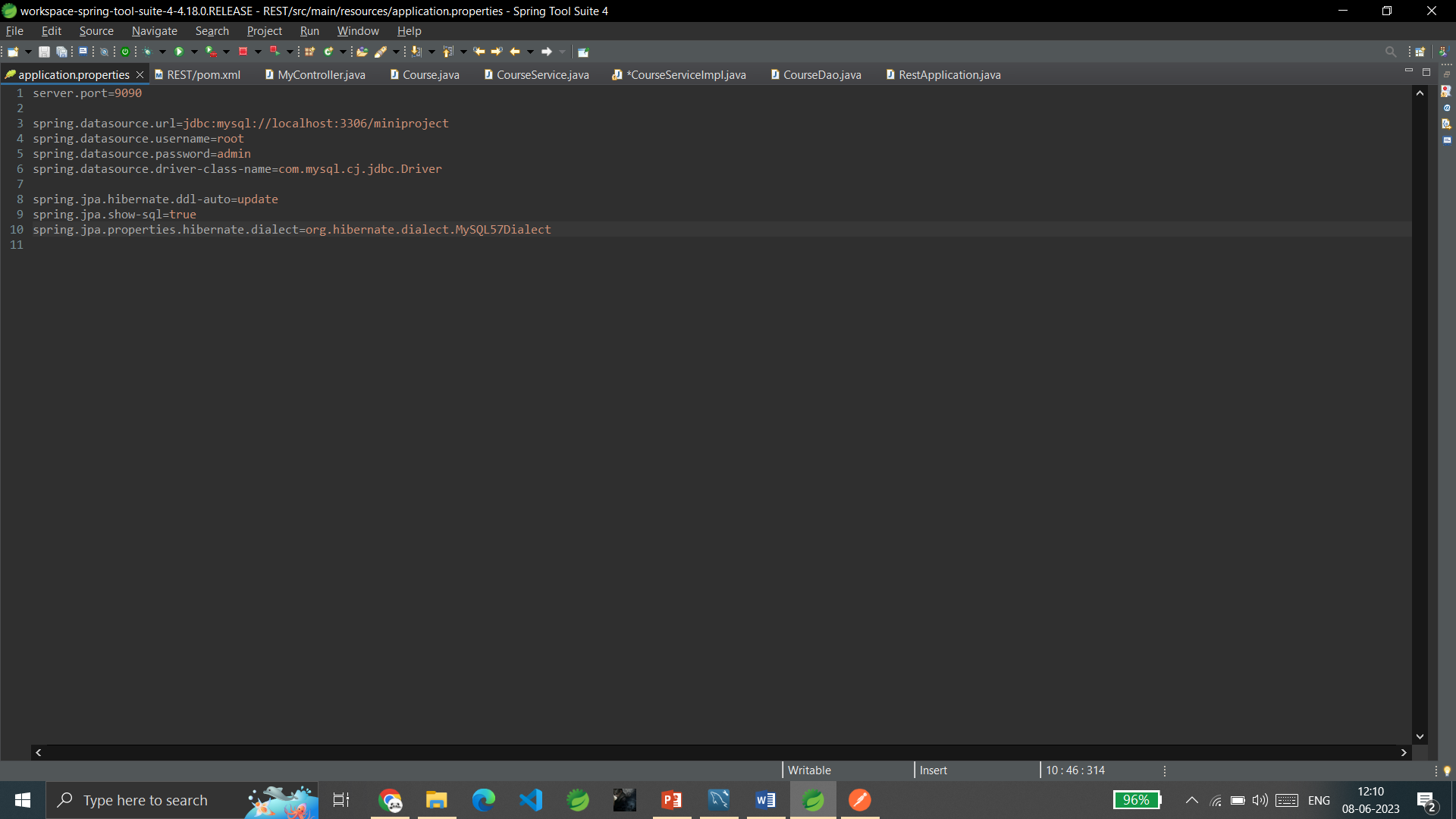
1. [**https://www.researchgate.net/publication/325770704\_An\_Analysis\_of\_Public\_REST\_Web\_Service\_APIs**](https://www.researchgate.net/publication/325770704_An_Analysis_of_Public_REST_Web_Service_APIs)
2. [**https://issuu.com/irjet/docs/irjet-v7i6207**](https://issuu.com/irjet/docs/irjet-v7i6207)
3. **https://eden.dei.uc.pt/~cnl/selected-research/2018-tsc-rest.pdf**
4. **https://www.researchgate.net/publication/271295146\_Client-Server\_Model**
5. **https://www.researchgate.net/publication/321757987\_Java\_Spring\_Boot\_Rest\_WEB\_Service\_Integration\_with\_Java\_Artificial\_Intellgence\_Weka\_Framework**
6. **https://www.researchgate.net/publication/264227921\_A\_Comparative\_study\_of\_SOAP\_vs\_REST\_web\_services\_provisioning\_techniques\_for\_mobile\_host**
7. [**https://www.akana.com/blog/what-is-rest-api**](https://www.akana.com/blog/what-is-rest-api)
8. **Dewire, D. T. (1993). Client/server computing. McGraw-Hill, Singapore**
9. **Richardson, L., & Ruby, S. (2007). RESTful Web Services: Web APIs for the Real World. O'Reilly Media. - Although not a research paper, this book by Leonard Richardson and Sam Ruby provides a comprehensive guide to building RESTful APIs and covers many practical aspects of RESTful design.**
10. **Yu, Y., et al. (2018). RESTAPI: A Unified Specification for Representing, Discovering, and Invoking APIs. IEEE Transactions on Services Computing, 11(2), 284-297. - This paper presents RESTAPI, a unified specification for representing, discovering, and invoking APIs in RESTful systems, aiming to provide better interoperability and automation of API usage.**
11. **Guinard, D., & Trifa, V. (2016). Building the Web of Things. Manning Publications. - While not focused solely on REST APIs, this book by Dominique Guinard and Vlad Trifa discusses the concept of the Web of Things and explores the use of RESTful interfaces for interacting with connected devices.**

**APPENDIX**

**Code:**







**PROJECT SOURCE CODE / LINK:**

https://github.com/nikitha-m/mini-project