Penugasan Praktikum 02 Pemrograman Platform Khusus

Muhammad Zacky Heta Warman\_22212775\_3SI1\_28

2.6 Penugasan

Lengkapi layanan perpustakaan di atas dengan menambahkan layanan pencarian buku berdasarkan kata kunci pencarian.

Depencency (pom.xml) :

<dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>com.mysql</groupId>

            <artifactId>mysql-connector-j</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>jakarta.validation</groupId>

            <artifactId>jakarta.validation-api</artifactId>

            <version>3.0.2</version>

            <type>jar</type>

        </dependency>

    </dependencies>

Koneksi ke database (application.properties) :

spring.application.name=perpustakaan

spring.datasource.url=jdbc:mysql://localhost:3306/perpustakaan

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

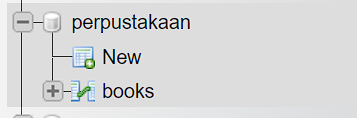
spring.datasource.username=root

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

Pembuatan database (phpMyAdmin) :



JsonRpcController.java :

Pada controller ini, ditambahkan method searchBooks untuk mencari buku berdasarkan keyword atau kata kunci pencarian.

package com.polstat.perpustakaan.controller;

import com.fasterxml.jackson.databind.JsonNode;

import com.polstat.perpustakaan.dto.BookDto;

import com.polstat.perpustakaan.rpc.JsonRpcRequest;

import com.polstat.perpustakaan.rpc.JsonRpcResponse;

import com.polstat.perpustakaan.service.BookService;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class JsonRpcController {

    @Autowired

    private BookService bookService;

    @PostMapping("/jsonrpc")

    public ResponseEntity<Object> handleJsonRpcRequest(@RequestBody JsonRpcRequest request) {

        try {

            String method = request.getMethod();

            JsonNode params = request.getParams();

            System.out.println("Method: " + method);

            switch (method) {

                case "createBook":

                    String title = params.get("title").asText();

                    String author = params.get("author").asText();

                    String description = params.get("description").asText();

                    BookDto book = BookDto.builder()

                            .title(title)

                            .description(description)

                            .author(author)

                            .build();

                    bookService.createBook(book);

                    return ResponseEntity.ok(new JsonRpcResponse("created", request.getId()));

                case "getBooks":

                    List<BookDto> books = bookService.getBooks();

                    return ResponseEntity.ok(new JsonRpcResponse(books, request.getId()));

                case "searchBooks":

                    String keyword = params.get("keyword").asText();

                    List<BookDto> result = bookService.searchBooks(keyword);

                    return ResponseEntity.ok(new JsonRpcResponse(result, request.getId()));

                default:

                    return ResponseEntity.badRequest().build();

            }

        } catch (Exception e) {

            return ResponseEntity.badRequest().build();

        }

    }

}

BookDto.java :

package com.polstat.perpustakaan.dto;

import jakarta.validation.constraints.NotEmpty;

import jakarta.validation.constraints.NotNull;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class BookDto {

    private Long id;

    @NotEmpty(message = "Judul buku wajib diisi.")

    private String title;

    @NotNull(message = "Penulis buku wajib diisi.")

    private String author;

    private String description;

}

Book.java :

package com.polstat.perpustakaan.entity;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

@Builder

@Entity

@Table(name = "books")

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column(nullable = false)

    private String title;

    @Column(nullable = false)

    private String author;

    @Column(nullable = true)

    private String description;

}

BookMapper.java :

package com.polstat.perpustakaan.mapper;

import com.polstat.perpustakaan.dto.BookDto;

import com.polstat.perpustakaan.entity.Book;

public class BookMapper {

    public static Book mapToBook(BookDto bookDto) {

        return Book.builder()

                .id(bookDto.getId())

                .title(bookDto.getTitle())

                .description(bookDto.getDescription())

                .author(bookDto.getAuthor())

                .build();

    }

    public static BookDto mapToBookDto(Book book) {

        return BookDto.builder()

                .id(book.getId())

                .title(book.getTitle())

                .description(book.getDescription())

                .author(book.getAuthor())

                .build();

    }

}

BookRepository.java :

package com.polstat.perpustakaan.repository;

import com.polstat.perpustakaan.entity.Book;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface BookRepository extends JpaRepository<Book, Long>{

    List<Book> findByTitleContainingIgnoreCase(String title);

}

JsonRpcRequest.java :

package com.polstat.perpustakaan.rpc;

import com.fasterxml.jackson.databind.JsonNode;

import lombok.\*;

@Setter

@Getter

public class JsonRpcRequest {

    private String jsonrpc;

    private String method;

    private JsonNode params;

    private String id;

}

JsonRpcResponse.java :

package com.polstat.perpustakaan.rpc;

import lombok.\*;

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

public class JsonRpcResponse {

    private String jsonrpc;

    private Object result;

    private Object error;

    private String id;

    public JsonRpcResponse(Object result, String id) {

        this.result = result;

        this.id = id;

    }

}

BookService.java :

package com.polstat.perpustakaan.service;

import com.polstat.perpustakaan.dto.BookDto;

import java.util.List;

public interface BookService {

    public void createBook(BookDto bookDto);

    public List<BookDto> getBooks();

    public List<BookDto> searchBooks(String keyword);

    }

BookServiceImpl.java :

package com.polstat.perpustakaan.service;

import com.polstat.perpustakaan.dto.BookDto;

import com.polstat.perpustakaan.entity.Book;

import com.polstat.perpustakaan.mapper.BookMapper;

import com.polstat.perpustakaan.repository.BookRepository;

import java.util.List;

import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BookServiceImpl implements BookService {

    @Autowired

    private BookRepository bookRepository;

    @Override

    public void createBook(BookDto bookDto) {

        bookRepository.save(BookMapper.mapToBook(bookDto));

    }

    @Override

    public List<BookDto> getBooks() {

        List<Book> books = bookRepository.findAll();

        List<BookDto> bookDtos = books.stream()

                .map((product) -> (BookMapper.mapToBookDto(product)))

                .collect(Collectors.toList());

        return bookDtos;

    }

    @Override

    public List<BookDto> searchBooks(String keyword){

        List<Book> books = bookRepository.findByTitleContainingIgnoreCase(keyword);

        List<BookDto> bookDtos = books.stream()

            .map((product) -> (BookMapper.mapToBookDto(product)))

            .collect(Collectors.toList());

        return bookDtos;

    }

}

PerpustakaanApplication.java :

package com.polstat.perpustakaan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class PerpustakaanApplication {

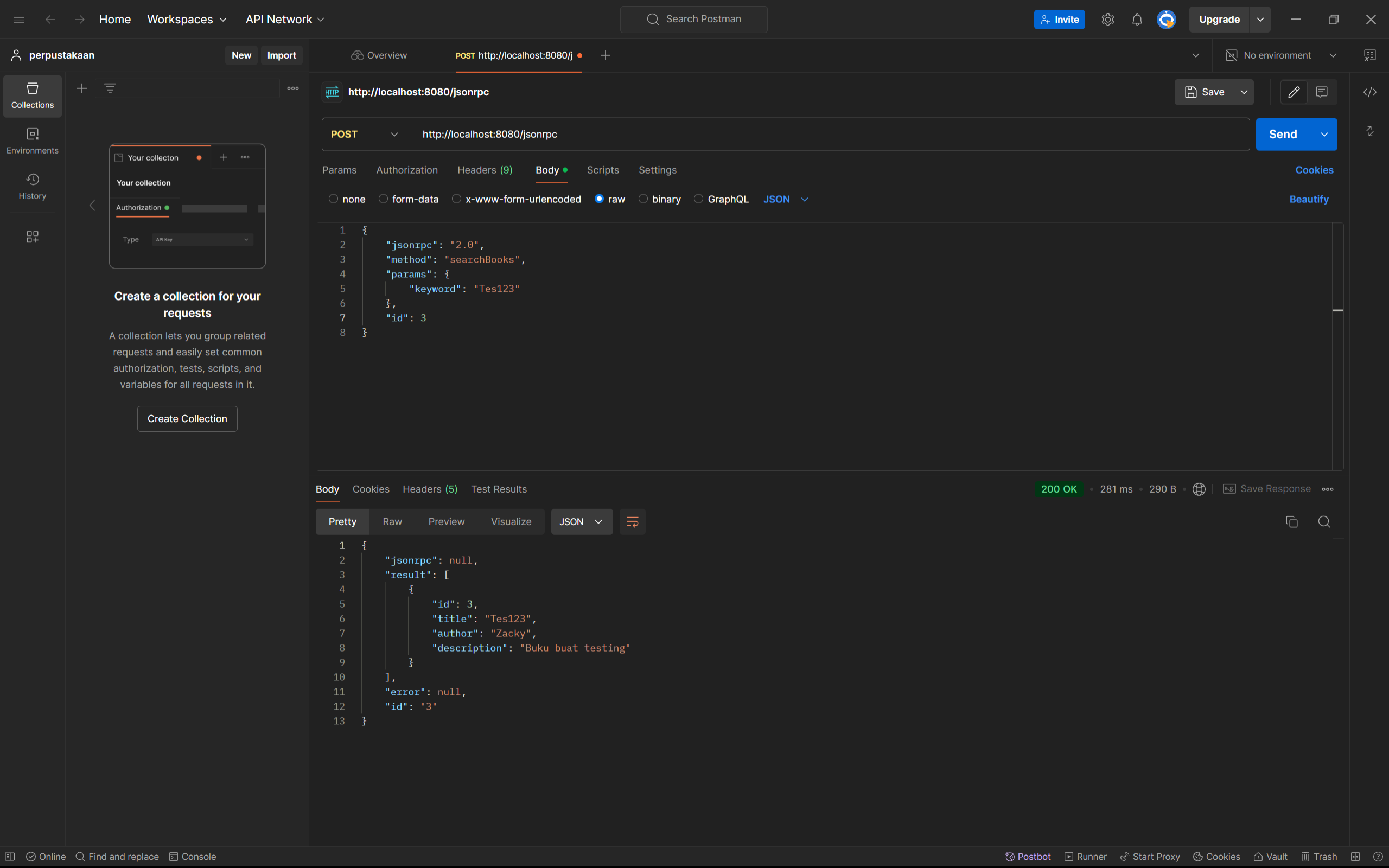
    public static void main(String[] args) {

        SpringApplication.run(PerpustakaanApplication.class, args);

    }

}

Proses permintaan dan respons JSON – RPC (Postman) :



Database perpustakaan :

