МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота №1**

з курсу «Платформи корпоративних інформаційних систем»

на тему:

«Вступ до Spring Boot. Контролер, сервіс, репозиторій.»

**Виконав:**

студент групи КН-41

Сенів П.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

викладач кафедри КНІС

к.т.н., доцент Горєлов В.О. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2021

**Мета**: сформувати теоретичний базис щодо структури та основних понять Spring Boot web-проекту. Набути практичних навичок написання класів контролера, сервісу, репозиторію.

**Завдання**: розробити проект із фейковим репозиторієм (List) для реалізації каталогу домашньої бібліотеки.

**Хід роботи**

1. Напишіть клас, що описує видання у каталозі домашньої бібліотеки. Поля класу:

а) id;

б) назва книги;

в) автор;

г) рік видання;

д) видавництво;

е) кількість сторінок.

Типи полів задайте самостійно.

2. Розробіть інтерфейс, котрий міститиме необхідні операції над даними. Перелік операцій:

а) додавання;

б) перегляд за id;

в) перегляд повного переліку;

г) видалення за id;

д) видалення повного переліку;

у) модифікація за id;

е) перегляд за автором;

є) перегляд за видавництвом.

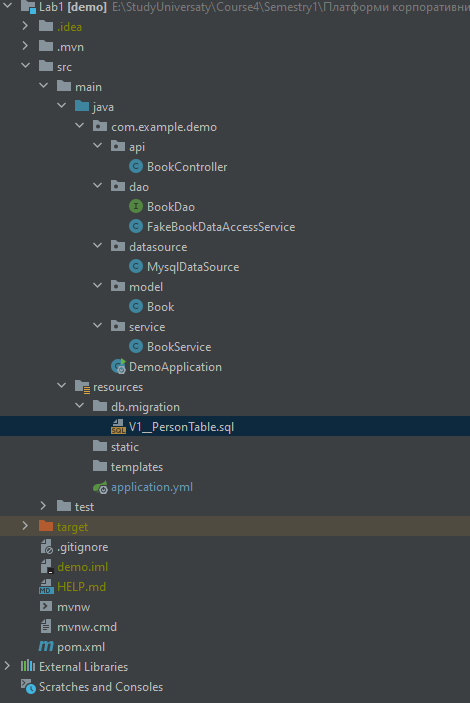
3. Розробіть репозиторій на основі інтерфейсу (п. 2). Реалізуйте його методи.

4. Побудуйте сервіс, що використовує отриманий у п. 3 репозиторій. 5. Розробіть контролер для обробки запитів користувача.

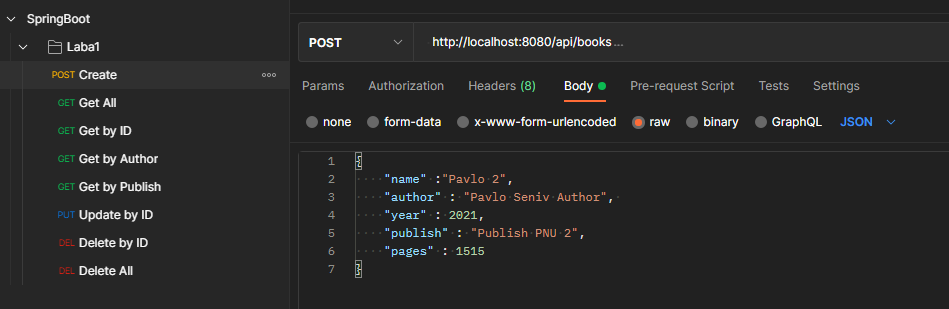
6. Продемонструйте розроблений проект у дії. Скористайтеся для надсилання запитів до сервера API-клієнтом (Postman, Insomnia тощо). Формат тіла HTTP-запитів та відповідей сервера — JSON.

7. Оформте звіт.

**Результати виконання роботи**



**Рисунок 1. Структура проекту.**



**Рисунок 2. Запити та приклад запиту.**

**Файл BookController.java**

package com.example.demo.api;  
  
import com.example.demo.model.Book;  
import com.example.demo.service.BookService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.lang.NonNull;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import javax.websocket.server.PathParam;  
import java.util.List;  
import java.util.UUID;  
  
@RequestMapping("api/books")  
@RestController  
public class BookController {  
 private final BookService bookService;  
  
 @Autowired  
 public BookController(BookService bookService) {  
 this.bookService = bookService;  
 }  
  
 @PostMapping  
 public void createBook(@Valid @NonNull @RequestBody Book book) {  
 bookService.create(book);  
 }  
  
 @GetMapping(path = "/{id}")  
 public Book getById(@PathVariable("id") UUID id) {  
 return bookService.bookDao.getById(id)  
 .orElse(null);  
 }  
  
 @GetMapping(path = "/author")  
 public Book getByAuthor(@PathParam("author") String author) {  
 return bookService.bookDao.getByAuthor(author)  
 .orElse(null);  
 }  
  
 @GetMapping(path = "/publish")  
 public Book getByPublish(@PathParam("publish") String publish) {  
 return bookService.bookDao.getByPublish(publish)  
 .orElse(null);  
 }  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookService.getAll();  
 }  
  
 @DeleteMapping(path = "/{id}")  
 public void deleteById(@PathVariable("id") UUID id) {  
 bookService.deleteById(id);  
 }  
  
 @DeleteMapping  
 public void deleteAll() {  
 bookService.deleteAll();  
 }  
  
 @PutMapping(path = "/{id}")  
 public void updateById(@PathVariable("id") UUID id, @RequestBody Book book) {  
 bookService.updateById(id, book);  
 }  
}

**Файл BookService.java**

package com.example.demo.service;  
  
import com.example.demo.dao.BookDao;  
import com.example.demo.model.Book;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.UUID;  
  
@Service  
public class BookService {  
 public final BookDao bookDao;  
 @Autowired  
 public BookService(@Qualifier("bookFakeDao") BookDao bookDao) {  
 this.bookDao = bookDao;  
 }  
  
 public int create(Book book) {  
 return bookDao.create(book);  
 }  
  
 public List<Book> getAll() {  
 return this.bookDao.getAll();  
 }  
  
 public int deleteById(UUID id) {  
 return bookDao.deleteById(id);  
 }  
  
 public int deleteAll() {  
 return bookDao.deleteAll();  
 }  
  
 public int updateById(UUID id, Book newBook) {  
 return bookDao.updateById(id, newBook);  
 }  
}

**Файл MysqlDataSource.java**

package com.example.demo.datasource;  
  
import com.zaxxer.hikari.HikariDataSource;  
import org.springframework.boot.context.properties.ConfigurationProperties;  
import org.springframework.boot.jdbc.DataSourceBuilder;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class MysqlDataSource {  
 @Bean  
 @ConfigurationProperties("app.datasource")  
 public HikariDataSource hikariDataSource() {  
 return DataSourceBuilder  
 .*create*()  
 .type(HikariDataSource.class)  
 .build();  
 }  
}

**Висновок:** на лабораторній роботі було реалізовано проект із фейковим репозиторієм (List) для реалізації каталогу домашньої бібліотеки. Я навчився користуватися API-клієнтом Postman. Надсилав запити до сервера API-клієнтом Postman.