Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа № 2

«База данных в веб-сервисе на java»

Проверил: Выполнил:

асс. каф. ЭВМ ст. гр. 250503

И.Г. Скиба М.С. Патюпин

МИНСК 2024

# 1 Цель и задачи лабораторной работы

Цель работы – освоить принципы разработки веб-сервиса с использованием стека Spring. Реализовать GET-endpoint’ы, подключить базу данных. Реализовать связи @OneToMany, @ManyToMany.

# 2 Результаты выполнения лабораторной работы

**2.1** Структура проекта

В проекте использовалась послойная архитектура:

│ Demo1Application.java

│

├───controller

│ FactController.java

│ FactInfoController.java

│

├───entity

│ CategoryEntity.java

│ FactCategoryEntity.java

│ FactEntity.java

│ NumberEntity.java

│

├───repository

│ CategoryRepository.java

│ FactCategoryRepository.java

│ FactRepository.java

│ NumberRepository.java

│

└───service

│ FactCategoryService.java

│ FactService.java

│ NumberService.java

│

└───defaultt

DefaultFactCategoryService.java

DefaultFactService.java

DefaultNumberService.java

Пакет “controller” служит для хранения классов, и реализации методов, которые обрабатывают входящие HTTP-запросы, управляют потоком выполнения и взаимодействуют с другими компонентами приложения.

Класс “FactInfoController” является контроллером REST API и содержит методы для обработки HTTP-запросов:

* Метод getInfoOne обрабатывает GET-запрос на пути /info. Он принимает два параметра: number и type, которые могут быть переданы в запросе в виде параметров. Параметры должены соответствовать регулярному выражению, указанному в аннотации @Pattern, для обеспечения безопасности. После выполняет метод getFactByFactAndCategory и возвращяет ResponseEntity<String> с фактом о числе.
* Метод getInfoAllобрабатывает GET-запрос на пути /info/all. Он принимает два параметра: number и type, которые могут быть переданы в запросе в виде параметров. Параметры должены соответствовать регулярному выражению, указанному в аннотации @Pattern, для обеспечения безопасности. После выполняет метод getFactsByFactAndCategory и возвращяет ResponseEntity<String> со всеми соответствующими фактами.
* Метод defaultMethod обрабатывает GET-запрос на любом пути, который не соответствует /info. Он возвращает ResponseEntity с сообщением о том, что необходимо указать действительный путь.

Класс “FactController” является контроллером REST API и содержит методы для обработки HTTP-запросов:

* Метод addFact является обработчиком POST-запроса для добавления факта. Он принимает значения числа (numberData), типа факта (type) и текста факта (newFact) через параметры запроса. Внутри метода происходит проверка и валидация параметров, а затем выполняются операции добавления числа в сервис чисел, нахождения или создания категории факта, создания факта и его связи с категорией. В случае успешного выполнения, метод возвращает ответ с кодом 200 OK и сообщением о успешном добавлении факта, иначе, при возникновении исключения NumberFormatException, возвращается ответ с кодом 400 Bad Request и сообщением об ошибке в формате числа или факта.
* Метод delFact представляет собой обработчик DELETE-запроса для удаления факта. Он принимает параметр number из запроса, который является идентификатором факта. Внутри метода происходит проверка и валидация параметра с помощью аннотаций @RequestParam, @Pattern и @NumberFormat. Затем метод выполняет следующие действия: находит соответствующую запись FactCategoryEntity в репозитории, удаляет связанную запись FactCategoryEntity из репозитория factCategoryRepository, удаляет факт из репозитория factRepository, проверяет, остались ли другие факты, связанные с тем же числом, и в зависимости от результата, либо удаляет число из репозитория numberRepository, либо возвращает ответ с сообщением об успешном удалении факта. Если искомая запись FactCategoryEntity не найдена, метод возвращает ответ с сообщением о том, что факт не был найден. В случае ошибки формата идентификатора, метод возвращает ответ с кодом 400 Bad Request и сообщением об ошибке в формате идентификатора.
* Метод updFact представляет собой обработчик PUT-запроса для обновления факта. Он принимает параметры из запроса, включая идентификатор факта (factId), новое значение числа (number), новый тип факта (type) и новый текст факта (newFact). Внутри метода происходит проверка и валидация параметров, а затем выполняются операции обновления значения типа факта и числа, если они указаны в запросе. В случае успешного обновления, метод возвращает ответ с кодом 200 OK и сообщением, содержащим обновленную информацию о факте. В случае возникновения ошибки при обновлении, метод возвращает ответ с кодом 500 Internal Server Error и сообщением об ошибке.

Пакет “entity” служит хранения класов отвечающих за предаставления данных, каждый класс соответствует таблице в бвзе данных.

Пакет “repository” используется для хранения классов, отвечающих за доступ к данным и выполнение операций с базой данных. Он помогает абстрагировать доступ к данным от остальных компонентов приложения, таких как сервисы или контроллеры, и предоставляет удобный интерфейс для работы с постоянным хранилищем.

Пакет "service" используется для хранения классов, которые содержат бизнес-логику и выполняют операции над данными. Сервисы служат промежуточным слоем между контроллерами и репозиториями. Они обеспечивают модульность, управление транзакциями, координацию операций и другую логику, необходимую для обработки бизнес-задач в приложении.

**2.2** Листинг компьютерной программы

1. Demo1Application.java

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Demo1Application {

public static void main(String[] args) {

SpringApplication.run(Demo1Application.class, args);

}

}

1. controller\FactController.java

package com.example.demo.controller;

import com.example.demo.entity.CategoryEntity;

import com.example.demo.entity.FactCategoryEntity;

import com.example.demo.entity.FactEntity;

import com.example.demo.entity.NumberEntity;

import com.example.demo.repository.CategoryRepository;

import com.example.demo.repository.FactCategoryRepository;

import com.example.demo.repository.FactRepository;

import com.example.demo.repository.NumberRepository;

import com.example.demo.service.FactCategoryService;

import com.example.demo.service.FactService;

import com.example.demo.service.NumberService;

import jakarta.validation.constraints.Pattern;

import lombok.AllArgsConstructor;

import org.springframework.format.annotation.NumberFormat;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

@Validated

@RestController

@AllArgsConstructor

public class FactController {

private final NumberService numberService;

private final NumberRepository numberRepository;

private final FactService factService;

private CategoryRepository categoryRepository;

private final FactCategoryRepository factCategoryRepository;

private final FactRepository factRepository;

@PostMapping(value = "/fact/add")

public ResponseEntity<String> addFact(@RequestParam(value = "number")

@Pattern(regexp = "\\d+")

@NumberFormat(style = NumberFormat.Style.NUMBER) long numberData,

@RequestParam(value = "type", defaultValue = "trivia")

@Pattern(regexp = "^(year|math|trivia)$") String type,

@RequestParam(value = "fact", defaultValue = "it's a boring number")

@Pattern(regexp = "[A-Za-z0-9\\s.,;!?\"'-]+") String newFact) {

try {

numberService.addNumber(numberData);

CategoryEntity existingCategory = categoryRepository.findCategoryByName(type);

FactEntity createdFact = factService.createFact(numberData, newFact);

FactCategoryEntity factCategory = new FactCategoryEntity();

factCategory.setCategory(existingCategory);

factCategory.setFact(createdFact);

factCategoryRepository.save(factCategory);

return ResponseEntity.ok("Fact added successfully with ID: " + createdFact.getId());

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body("Invalid number/fact format");

}

}

@DeleteMapping(value = "/fact/delete")

public ResponseEntity<String> delFact(@RequestParam(value = "number")

@Pattern(regexp = "\\d+")

@NumberFormat(style = NumberFormat.Style.NUMBER) String number) {

try {

long numberData = Long.parseLong(number);

FactCategoryEntity factCategoryEntity = factCategoryRepository.findFactCategoryEntitiesById(numberData);

if (factCategoryEntity != null) {

factCategoryRepository.delete(factCategoryRepository.getFactCategoryByFactEntity(factRepository.findById(numberData).get()));

NumberEntity delNumber = factRepository.findById(numberData).get().getNumber();

factRepository.deleteById(numberData);

long numId = factCategoryEntity.getFact().getNumber().getId();

if (factRepository.countByNumberId(numId) > 0) {

// Р•СЃР»Рё СЃРїРёСЃРѕРє РЅРµ РїСѓСЃС‚РѕР№, Р·РЅР°С‡РёС‚ РµСЃС‚СЊ СЃРѕРѕС‚РІРµС‚СЃС‚РІСѓСЋС‰РёР№ С„Р°РєС‚

return ResponseEntity.ok().body("Fact delete successfully.");

} else {

numberRepository.delete(delNumber);

return ResponseEntity.ok().body("Fact delete successfully.");

}

}

return ResponseEntity.ok("Fact not found.");

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body("Invalid id format");

}

}

@PutMapping(value = "/fact/update")

public ResponseEntity<String> updFact(@RequestParam(value = "id")

@Pattern(regexp = "\\d+")

@NumberFormat(style = NumberFormat.Style.NUMBER) long factId,

@RequestParam(value = "number", required = false)

@Pattern(regexp = "\\d+")

@NumberFormat(style = NumberFormat.Style.NUMBER) Long number,

@RequestParam(value = "type", defaultValue = "nothing")

@Pattern(regexp = "^(year|math|trivia)$") String type,

@RequestParam(value = "fact", defaultValue = "it's a boring number")

@Pattern(regexp = "[A-Za-z0-9\\s.,;!?\"'-]+") String newFact) {

try {

FactCategoryEntity factCategoryEntity = factCategoryRepository.findFactCategoryEntitiesById(factId);

if (factCategoryEntity == null) {

return ResponseEntity.badRequest().body("Fact not found for the given ID.");

}

if (!type.equals("nothing") && !factCategoryEntity.getCategory().getName().equals(type)) {

factCategoryEntity.setCategory(categoryRepository.findCategoryByName(type));

factCategoryRepository.save(factCategoryEntity);

}

if (factCategoryEntity.getFact().getNumber().getNumberData() != number.longValue()) {

factCategoryEntity.getFact().setNumber(numberService.createNumber(number.longValue()));

factCategoryRepository.save(factCategoryEntity);

}

return ResponseEntity.ok("Update fact: " + factCategoryRepository.findFactCategoryEntitiesById(factId));

} catch (Exception e) {

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("An error occurred while updating the fact.");

}

}

}

1. controller\FactInfoController.java

package com.example.demo.controller;

import com.example.demo.entity.FactCategoryEntity;

import com.example.demo.repository.CategoryRepository;

import com.example.demo.repository.FactCategoryRepository;

import com.example.demo.repository.NumberRepository;

import com.example.demo.service.FactCategoryService;

import com.example.demo.service.FactService;

import com.example.demo.service.NumberService;

import jakarta.validation.ConstraintViolationException;

import jakarta.validation.constraints.Pattern;

import lombok.AllArgsConstructor;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.annotation.\*;

import java.io.IOException;

import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.List;

@Validated

@RestController

@AllArgsConstructor

public class FactInfoController {

NumberService numberService;

NumberRepository numberRepository;

CategoryRepository categoryRepository;

FactCategoryService factCategoryService;

FactService factService;

private final FactCategoryRepository factCategoryRepository;

private final SecureRandom random = new SecureRandom ();

@GetMapping(value = "/test")

public ResponseEntity<List<String>> getTest(@RequestParam(value = "number", defaultValue = "random")

@Pattern(regexp = "\\d+|^(random)") String numberS,

@RequestParam(value = "type", defaultValue = "trivia")

@Pattern(regexp = "^(year|math|trivia)$") String type) {

long number = 0;

if (numberS.equals("random")) {

number = 500 - random.nextLong(1001);

} else {

number = Long.parseLong(numberS);

}

long numberId = numberRepository.findByNumberData(number).getId();//+

List<FactCategoryEntity> test;

test = factCategoryRepository.findFactCategoriesByFactId(numberId);//+

List<String> testS = new ArrayList<>();//Р° Р·Р°С‡РµРј РѕРЅРѕ РЅР°РґРѕ, РєР°С‚РµРіРѕСЂРёСЋ РґР°РІР°Р№

for (int i = 0; i < test.size(); i++) {

if (test.get(i).getCategory().getId() == categoryRepository.findIdByName(type)) {

testS.add("Fact id:" + test.get(i).getFact().getId() + ", " + number + " " + test.get(i).getFact().getDescription());

}

}

return ResponseEntity.ok(testS);

}

@GetMapping(value = "/info")

public ResponseEntity<String> getInfoOne(@RequestParam(value = "number", defaultValue = "random")

@Pattern(regexp = "\\d+|^(random)") String numberS,

@RequestParam(value = "type", defaultValue = "trivia")

@Pattern(regexp = "^(year|math|trivia)$") String type){

return factCategoryService.getFactByFactAndCategory(numberS, type);

}

@GetMapping(value = "/info/all", produces = "application/json")

public ResponseEntity<List<String>> getInfoAll(@RequestParam(value = "number", defaultValue = "random")

@Pattern(regexp = "\\d+|^(random)") String numberS,

@RequestParam(value = "type", defaultValue = "trivia")

@Pattern(regexp = "^(year|math|trivia)$") String type) {

return factCategoryService.getFactsByFactAndCategory(numberS, type);

}

@GetMapping(value = "/\*\*")

public ResponseEntity<String> defaultMethod() {

return new ResponseEntity<>("Please specify a valid path. For example, http://localhost:8080/info?number=5&type=math", HttpStatus.BAD\_REQUEST);

}

@ExceptionHandler(IOException.class)

public ResponseEntity<Object> handleIOException() {

return new ResponseEntity<>("STATUS: 500. Internal Server Error", HttpStatus.INTERNAL\_SERVER\_ERROR);

}

@ExceptionHandler(ConstraintViolationException.class)

public ResponseEntity<String> handleConstraintViolation(ConstraintViolationException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.BAD\_REQUEST);

}

}

1. entity\CategoryEntity.java

package com.example.demo.entity;

import jakarta.persistence.\*;

import lombok.Data;

@Entity

@Data

@Table(name = "category")

public class CategoryEntity {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id")

private long id;

@Basic

@Column(name = "name")

private String name;

}

1. entity\FactCategoryEntity.java

package com.example.demo.entity;

import jakarta.persistence.\*;

import lombok.Data;

@Entity

@Data

@Table(name = "fact\_category")

public class FactCategoryEntity {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id")

private long id;

@ManyToOne

@JoinColumn(name = "category\_id", referencedColumnName = "id")

private CategoryEntity category;

@ManyToOne

@JoinColumn(name = "fact\_id", referencedColumnName = "id")

private FactEntity fact;

}

1. entity\FactEntity.java

package com.example.demo.entity;

import jakarta.persistence.\*;

import lombok.Data;

@Entity

@Data

@Table(name = "fact")

public class FactEntity {

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Id

@Column(name = "id")

private long id;

@ManyToOne

@JoinColumn(name = "number\_id", referencedColumnName = "id")

private NumberEntity number;

@Basic

@Column(name = "description")

private String description;

}

1. entity\NumberEntity.java

package com.example.demo.entity;

import jakarta.persistence.\*;

import lombok.Data;

@Data

@Entity

@Table(name = "number")

public class NumberEntity {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private long id;

@Column(name = "number\_data")

private long numberData;

}

1. repository\CategoryRepository.java

package com.example.demo.repository;

import com.example.demo.entity.CategoryEntity;

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

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

public interface CategoryRepository extends CrudRepository<CategoryEntity, Long> {

@Query("SELECT c.id FROM CategoryEntity c WHERE c.name = :name")

long findIdByName(@Param("name") String name);

@Query("SELECT c FROM CategoryEntity c WHERE c.name = :name")

CategoryEntity findCategoryByName(@Param("name") String name);

}

1. repository\FactCategoryRepository.java

package com.example.demo.repository;

import com.example.demo.entity.FactCategoryEntity;

import com.example.demo.entity.FactEntity;

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

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface FactCategoryRepository extends CrudRepository<FactCategoryEntity, Long> {

@Query("SELECT fc FROM FactCategoryEntity fc WHERE fc.fact.number.id = :factId")

List<FactCategoryEntity> findFactCategoriesByFactId(@Param("factId") long factId);

@Query("SELECT fc FROM FactCategoryEntity fc WHERE fc.fact.id = :idFact")

FactCategoryEntity findFactCategoryEntitiesById(long idFact);

@Query("SELECT fc FROM FactCategoryEntity fc WHERE fc.fact = :factEntity")

FactCategoryEntity getFactCategoryByFactEntity(@Param("factEntity") FactEntity factEntity);

}

1. repository\FactRepository.java

package com.example.demo.repository;

import com.example.demo.entity.FactEntity;

import com.example.demo.entity.NumberEntity;

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

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface FactRepository extends CrudRepository<FactEntity, Long> {

@Query("SELECT f FROM FactEntity f WHERE f.description = :description")

FactEntity findByDescription(@Param("description") String description);

@Query( "SELECT f FROM FactEntity f WHERE f.number = :number")

List<FactEntity> findFactEntityByNumber(@Param("number") NumberEntity numId);

@Query("SELECT COUNT(f) FROM FactEntity f WHERE f.number.id = :numberId")

long countByNumberId(long numberId);

FactEntity findByNumber( NumberEntity number);

}

1. repository\NumberRepository.java

package com.example.demo.repository;

import com.example.demo.entity.NumberEntity;

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

import org.springframework.data.repository.CrudRepository;

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

@Repository

public interface NumberRepository extends CrudRepository<NumberEntity, Long> {

@Query("SELECT n FROM NumberEntity n WHERE n.numberData = :numberData")

NumberEntity findByNumberData(@Param("numberData") long numberData);

}

1. service\FactCategoryService.java

package com.example.demo.service;

import com.example.demo.entity.FactCategoryEntity;

import org.springframework.http.ResponseEntity;

import java.util.List;

public interface FactCategoryService {

FactCategoryEntity createFactCategory(long catId, long facId);

void deleteFactCategory(long id);

void updateFactCategory(long id, long catId, long facId);

ResponseEntity<List<String>> getFactsByFactAndCategory(String numberS, String type);

ResponseEntity<String> getFactByFactAndCategory(String numberS, String type);

}

1. service\FactService.java

package com.example.demo.service;

import com.example.demo.entity.FactEntity;

import org.springframework.http.ResponseEntity;

public interface FactService {

public ResponseEntity<String> deleteFact(String number);

FactEntity createFact(long number, String description);

FactEntity findFact (String description);

FactEntity getFactByNumberId(long number);

}

1. service\NumberService.java

package com.example.demo.service;

import com.example.demo.entity.NumberEntity;

import org.springframework.http.ResponseEntity;

import java.util.List;

public interface NumberService {

ResponseEntity<String> addNumber(long numberData);

ResponseEntity<String> delNumber(String number);

long findIdByNumber(long numberData);

NumberEntity createNumber(long numberData);

NumberEntity findNumber(long numberData);

void updateNumber(long id, long numberData);

void deleteNumber(long id);

public String emplyNumber(List<String> responseS, long number, String type);

}

1. service\defaultt\DefaultFactCategoryService.java

package com.example.demo.service.defaultt;

import com.example.demo.entity.CategoryEntity;

import com.example.demo.entity.FactCategoryEntity;

import com.example.demo.entity.FactEntity;

import com.example.demo.repository.CategoryRepository;

import com.example.demo.repository.FactCategoryRepository;

import com.example.demo.repository.FactRepository;

import com.example.demo.repository.NumberRepository;

import com.example.demo.service.FactCategoryService;

import com.example.demo.service.NumberService;

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

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Service;

import java.security.SecureRandom;

import java.util.\*;

@Service

public class DefaultFactCategoryService implements FactCategoryService {

private final FactCategoryRepository factCategoryRepository;

private final CategoryRepository categoryRepository;

private final FactRepository factRepository;

private final NumberRepository numberRepository;

private final NumberService numberService;

private final SecureRandom random = new SecureRandom();

private CategoryEntity getCategoryEntityById(long catId) {

// Р›РѕРіРёРєР° РїРѕР»СѓС‡РµРЅРёСЏ CategoryEntity РїРѕ РёРґРµРЅС‚РёС„РёРєР°С‚РѕСЂСѓ

Optional<CategoryEntity> categoryOptional = categoryRepository.findById(catId);

return categoryOptional.orElse(null);

}

private FactEntity getFactEntityById(long facId) {

// Р›РѕРіРёРєР° РїРѕР»СѓС‡РµРЅРёСЏ FactEntity РїРѕ РёРґРµРЅС‚РёС„РёРєР°С‚РѕСЂСѓ

Optional<FactEntity> factOptional = factRepository.findById(facId);

return factOptional.orElse(null);

}

private FactCategoryEntity getFactCategoryEntityById(long id) {

Optional<FactCategoryEntity> factCategoryOptional = factCategoryRepository.findById(id);

return factCategoryOptional.orElse(null);

}

@Autowired

public DefaultFactCategoryService(FactCategoryRepository factCategoryRepository, CategoryRepository categoryRepository, FactRepository factRepository, NumberRepository numberRepository, NumberService numberService) {

this.factCategoryRepository = factCategoryRepository;

this.categoryRepository = categoryRepository;

this.factRepository = factRepository;

this.numberRepository = numberRepository;

this.numberService = numberService;

}

@Override

public FactCategoryEntity createFactCategory(long catId, long facId) {

FactCategoryEntity factCategoryEntity = new FactCategoryEntity();

// РџРѕР»СѓС‡РµРЅРёРµ СЃРІСЏР·Р°РЅРЅС‹С… СЃСѓС‰РЅРѕСЃС‚РµР№ CategoryEntity Рё FactEntity РёР· РёС… РёРґРµРЅС‚РёС„РёРєР°С‚РѕСЂРѕРІ

CategoryEntity categoryEntity = getCategoryEntityById(catId);

FactEntity factEntity = getFactEntityById(facId);

factCategoryEntity.setCategory(categoryEntity);

factCategoryEntity.setFact(factEntity);

return factCategoryRepository.save(factCategoryEntity);

}

@Override

public void deleteFactCategory(long id) {

factCategoryRepository.deleteById(id);

}

@Override

public void updateFactCategory(long id, long catId, long facId) {

FactCategoryEntity factCategoryEntity = getFactCategoryEntityById(id);

if (factCategoryEntity != null) {

CategoryEntity categoryEntity = getCategoryEntityById(catId);

FactEntity factEntity = getFactEntityById(facId);

factCategoryEntity.setCategory(categoryEntity);

factCategoryEntity.setFact(factEntity);

factCategoryRepository.save(factCategoryEntity);

}

}

public ResponseEntity<List<String>> getFactsByFactAndCategory(String numberS, String type) {

List<String> responseS = new ArrayList<>();

long number = 0;

try {

if (numberS.equals("random")) {

number = random.nextLong(1001) - 500;

} else {

number = Long.parseLong(numberS);

}

long numberId = numberRepository.findByNumberData(number).getId();

List<FactCategoryEntity> allFactByNumber;

allFactByNumber = factCategoryRepository.findFactCategoriesByFactId(numberId);

for (int i = 0; i < allFactByNumber.size(); i++) {

if (allFactByNumber.get(i).getCategory().getId() == categoryRepository.findIdByName(type)) {

responseS.add("Fact id:" + allFactByNumber.get(i).getFact().getId() + ", " + number + " " + allFactByNumber.get(i).getFact().getDescription());

}

}

if (responseS.isEmpty()) {

responseS.add(numberService.emplyNumber(responseS, number, type));

}

return ResponseEntity.ok(responseS);

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body(Collections.singletonList("Invalid number format."));

} catch (Exception e) {

if (responseS.isEmpty()) {

responseS.add(numberService.emplyNumber(responseS, number, type));

}

return ResponseEntity.ok().body(responseS);

}

}

@Override

public ResponseEntity<String> getFactByFactAndCategory(String numberS, String type) {

ResponseEntity<List<String>> facts = this.getFactsByFactAndCategory(numberS, type);

ResponseEntity<String> fact = null;

if (facts.getBody().size() > 1) {

int factRandom = random.nextInt(facts.getBody().size());

fact = ResponseEntity.ok().body(facts.getBody().get(factRandom));

} else {

return ResponseEntity.ok().body(facts.getBody().get(0));

}

return fact;

}

}

1. service\defaultt\DefaultFactService.java

package com.example.demo.service.defaultt;

import com.example.demo.entity.FactCategoryEntity;

import com.example.demo.entity.FactEntity;

import com.example.demo.entity.NumberEntity;

import com.example.demo.repository.CategoryRepository;

import com.example.demo.repository.FactCategoryRepository;

import com.example.demo.repository.FactRepository;

import com.example.demo.repository.NumberRepository;

import com.example.demo.service.FactService;

import lombok.RequiredArgsConstructor;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class DefaultFactService implements FactService {

private final FactRepository factRepository;

private final NumberRepository numberRepository;

private final FactCategoryRepository factCategoryRepository;

private final CategoryRepository categoryRepository;

@Override

public ResponseEntity<String> deleteFact(String number) {

try {

long numberData = Long.parseLong(number);

FactCategoryEntity factCategoryEntity = factCategoryRepository.findFactCategoryEntitiesById(numberData);

if (factCategoryEntity != null) {

factCategoryRepository.delete(factCategoryRepository.getFactCategoryByFactEntity(factRepository.findById(numberData).get()));

NumberEntity delNumber = factRepository.findById(numberData).get().getNumber();

factRepository.deleteById(numberData);

if (factRepository.findByNumber(delNumber) != null) {

return ResponseEntity.ok().body("Fact delete successfully.");

}

numberRepository.delete(delNumber);

return ResponseEntity.ok().body("Fact delete successfully.");

}

return ResponseEntity.ok("Fact not found.");

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body("Invalid id format");

}

}

@Override

public FactEntity createFact(long number, String description) {

NumberEntity existingNumber = numberRepository.findByNumberData(number);

if (existingNumber == null) {

existingNumber = new NumberEntity();

existingNumber.setNumberData(number);

existingNumber = numberRepository.save(existingNumber); // Save the new NumberEntity to generate an ID

}

FactEntity factEntity = new FactEntity();

factEntity.setNumber(existingNumber);

factEntity.setDescription(description);

return factRepository.save(factEntity);

}

@Override

public FactEntity findFact(String description) {

return factRepository.findByDescription(description);

}

@Override

public FactEntity getFactByNumberId(long number) {

NumberEntity numberEntity = numberRepository.findByNumberData(number);

return factRepository.findByNumber(numberEntity);

}

}

1. service\defaultt\DefaultNumberService.java

package com.example.demo.service.defaultt;

import com.example.demo.entity.NumberEntity;

import com.example.demo.service.NumberService;

import lombok.RequiredArgsConstructor;

import com.example.demo.repository.NumberRepository;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

@RequiredArgsConstructor

public class DefaultNumberService implements NumberService {

private final NumberRepository numberRepository;

@Override

public ResponseEntity<String> addNumber(long numberData) {

try {

NumberEntity existingNumber = findNumber(numberData);

if (existingNumber != null) {

return ResponseEntity.badRequest().body("Number already exists");

}

NumberEntity createdNumber = createNumber(numberData);

return ResponseEntity.ok("Number added successfully: " + createdNumber.getNumberData());

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body("Invalid number format");

}

}

@Override

public ResponseEntity<String> delNumber(String number) {

try {

long numberData = Long.parseLong(number);

NumberEntity existingNumber = this.findNumber(numberData);

if (existingNumber != null) {

numberRepository.delete(this.findNumber(numberData));

return ResponseEntity.ok().body("Number delete successfully.");

}

return ResponseEntity.ok("Number not found.");

} catch (NumberFormatException e) {

return ResponseEntity.badRequest().body("Invalid number format");

}

}

@Override

public long findIdByNumber(long id) {

return numberRepository.findByNumberData(id).getId();

}

@Override

public NumberEntity createNumber(long numberData) {

Optional<NumberEntity> existingNumber = Optional.ofNullable(numberRepository.findByNumberData(numberData));

if (existingNumber.isPresent()) {

return existingNumber.get();

}

NumberEntity numberEntity = new NumberEntity();

numberEntity.setNumberData(numberData);

return numberRepository.save(numberEntity);

}

@Override

public NumberEntity findNumber(long numberData) {

return numberRepository.findByNumberData(numberData);

}

@Override

public void updateNumber(long id, long numberData) {

Optional<NumberEntity> existingNumber = numberRepository.findById(id);

existingNumber.ifPresent(numberEntity -> {

numberEntity.setNumberData(numberData);

numberRepository.save(numberEntity);

});

}

@Override

public void deleteNumber(long id) {

numberRepository.deleteById(id);

}

public enum Type {

TRIVIA, MATH, YEAR

}

@Override

public String emplyNumber(List<String> responseS, long number, String type) {

DefaultNumberService.Type typeEnum = DefaultNumberService.Type.valueOf(type.toUpperCase());

switch (typeEnum) {

case TRIVIA:

return number + " is an uninteresting number.";

case MATH:

return number + " is a boring number.";

case YEAR:

return number + " BC is the year that we do not know what happened.";

}

return "Xmm. Oy";

}

}

**2.3** **Результат выполнения компьютерной программы**

Результат представлен на рисунках 2.1 – 2.2.

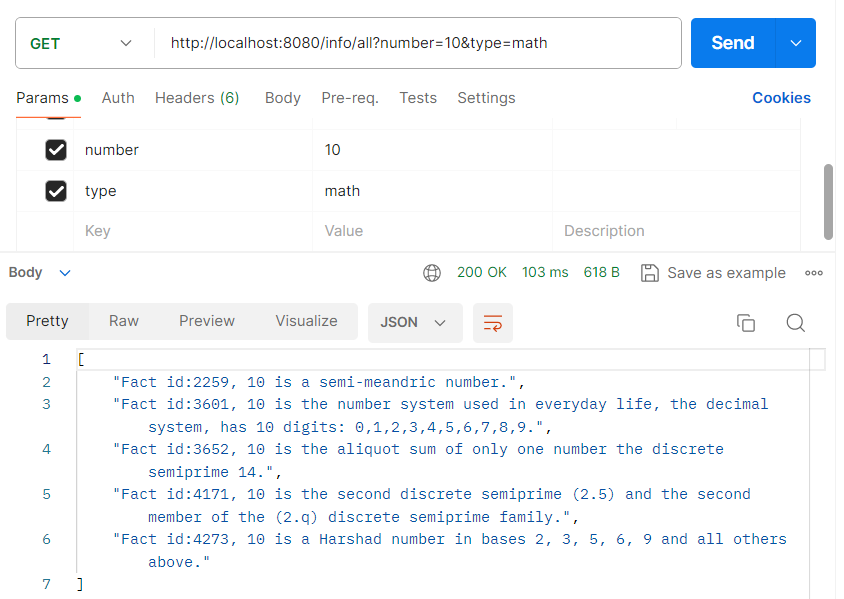


Рисунок 2.1 – Получение всех математических фактов о числе 10

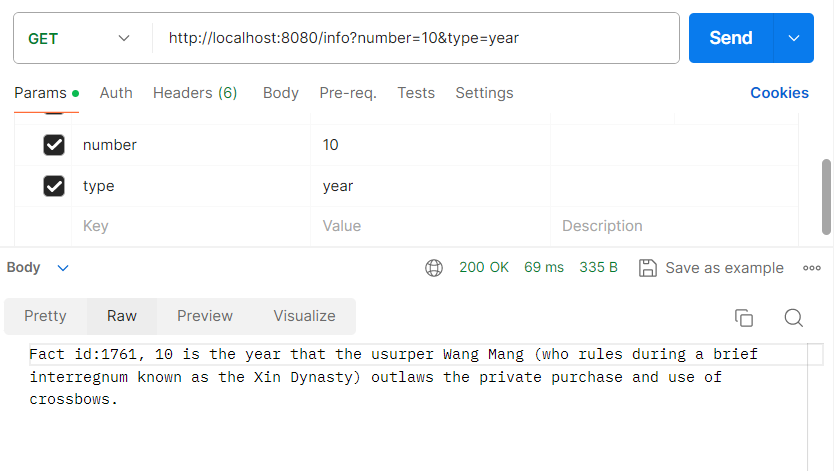


Рисунок 2.2 – Полученеие одного факта о 10 годе

При обращении к числу, факта которого нет в бвзе данных, будет выведен один из заготовленых ответов, пример приведен на рисунке 2.3.

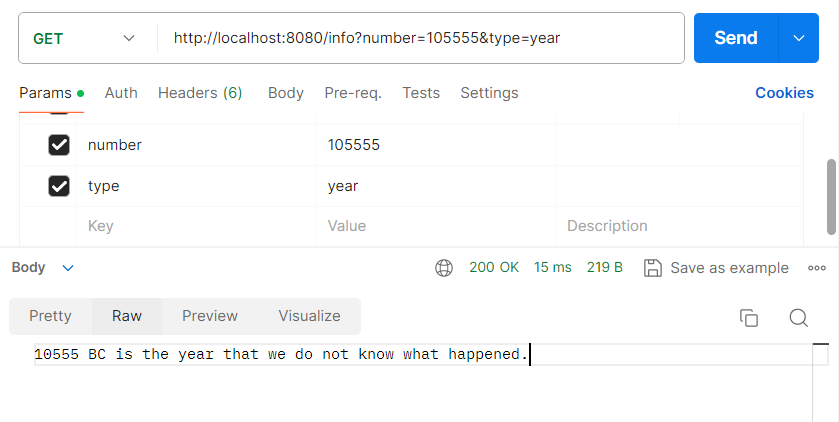


Рисунок 2.3 – Пример ответа о числе отсутствующего в бд

Результат работы добавления факта – рисунки 2.4 – 2.5; изменения факта – рисунки 2.6 – 2.7; удаления факта – рисунки 2.8 – 2.9.

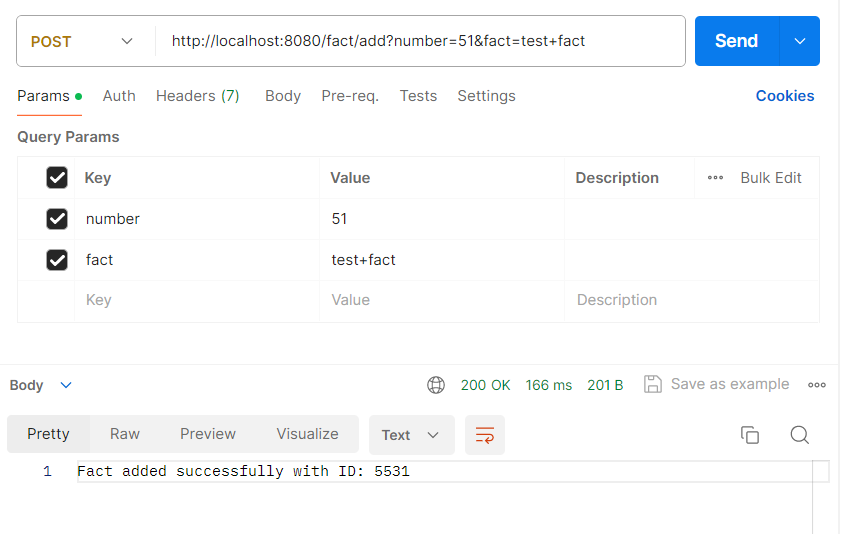


Рисунок 2.4 – Добавление факта

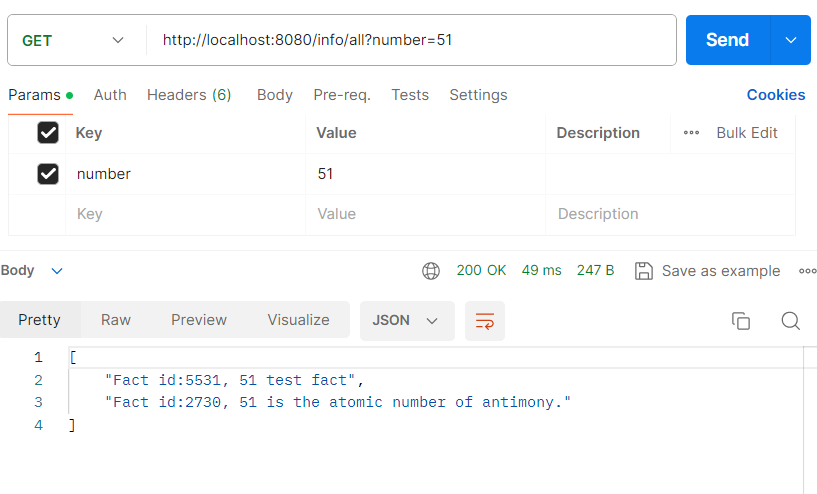


Рисунок 2.5 – Просмотр добавленного факта

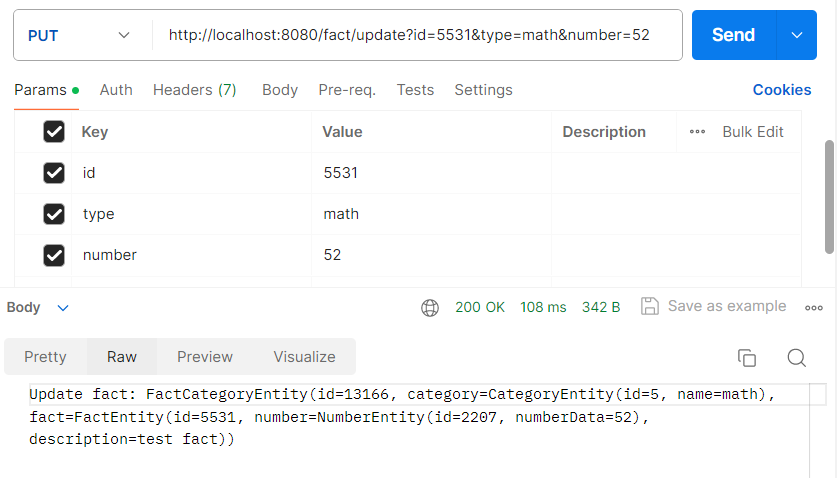


Рисунок 2.6 – Запрос на изменение факта

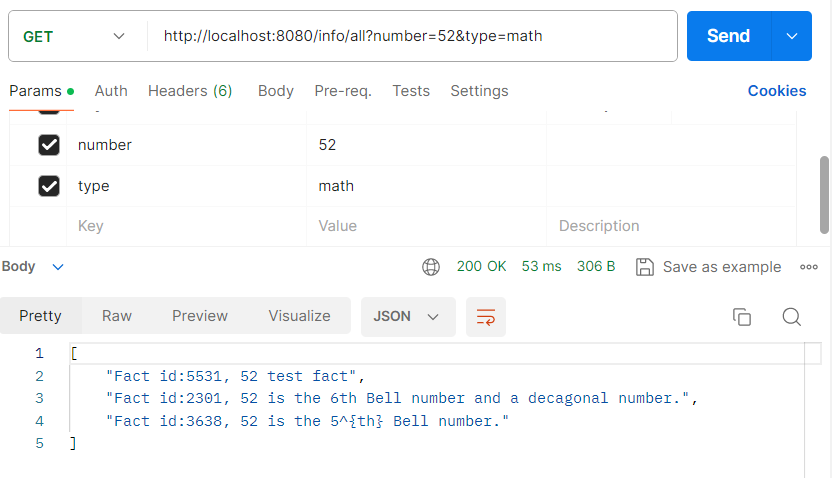


Рисунок 2.7 – Результат изменения факта

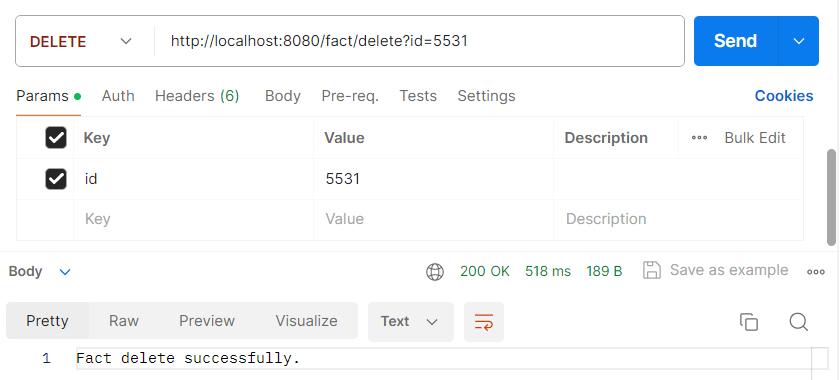


Рисунок 2.8 – Удаление факта

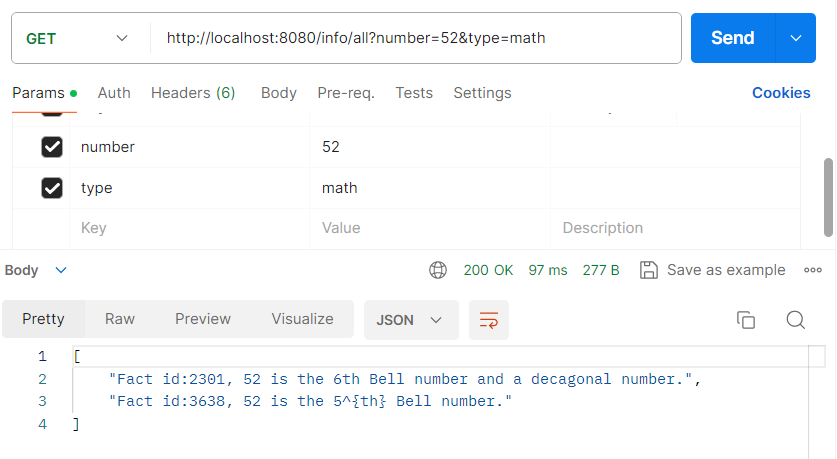


Рисунок 2.9 – Результат работы удаления факта

**2.3 Выводы по результатам выполнения ЛР**

В результате выполнения ЛР был создан простой безопасный сервис с использованием фрейморка Spring выдающий факт о числе из дазы данных PostgreSql.

Работа выполнена с использованием среды разработки Inrellj IDEA, на операционной системе Windows 10. Для тестирования API использовалось програмное обеспечение Postman. Для тестирования безопасности, и выявлении потенциально опасных фрагментов был подключен анализатор кода SonarCloud.

**3 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Лекционный материал по дисциплине “Программирование на языках высокого уровня”.