Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

Лабораторная работа №2

«JPA (Hibernate)»

Выполнил:

студент группы 350502

Егоренков С.С.

Проверила:

Скиба И.Г.

МИНСК 2025

# 1 ПОСТАНОВКА ЗАДАЧИ

Подключить в проект БД. Реализовать связи 1 ко многим и многие ко многим. Реализовать CRUD-операции для сущностей. Настроить каскадные операции и ленивую/жадную загрузку данных.

# 2 СТРУКТУРА ПРОЕКТА

Структура проекта представлена на рисунке 2.1.

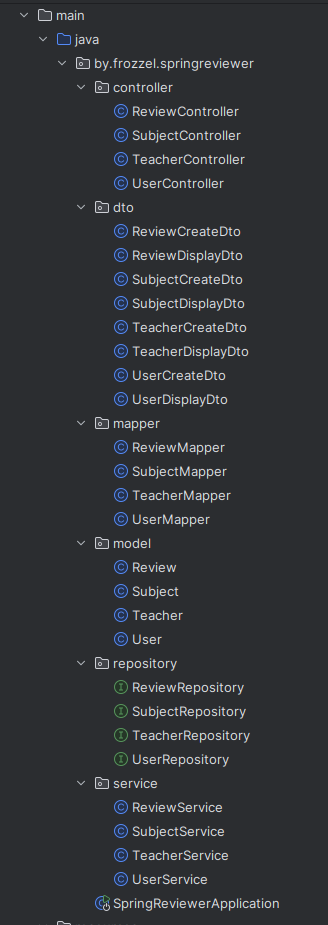


Рисунок 2.1 – структура проекта

# 3 ЛИСТИНГ КОДА

Файл ReviewController.java

package by.frozzel.springreviewer.controller;

import by.frozzel.springreviewer.dto.ReviewCreateDto;

import by.frozzel.springreviewer.dto.ReviewDisplayDto;

import by.frozzel.springreviewer.service.ReviewService;

import java.util.List;

import lombok.RequiredArgsConstructor;

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

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

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

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

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

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

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

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

@RestController

@RequestMapping("/reviews")

@RequiredArgsConstructor

public class ReviewController {

private final ReviewService reviewService;

@PostMapping

public ReviewDisplayDto createReview(@RequestBody ReviewCreateDto reviewCreateDto) {

return reviewService.saveReview(reviewCreateDto);

}

@GetMapping

public List<ReviewDisplayDto> getAllReviews() {

return reviewService.getAllReviews();

}

@GetMapping("/{id}")

public ReviewDisplayDto getReviewById(@PathVariable Integer id) {

return reviewService.getReviewById(id);

}

@GetMapping("/username/{username}")

public List<ReviewDisplayDto> getReviewsByUsername(@PathVariable String username) {

return reviewService.getReviewsByUserUsername(username);

}

@DeleteMapping("/{id}")

public void deleteReview(@PathVariable Integer id) {

reviewService.deleteReview(id);

}

@PutMapping("/{id}")

public ReviewDisplayDto updateReview(@PathVariable Integer id,

@RequestBody ReviewCreateDto reviewCreateDto) {

return reviewService.updateReview(id, reviewCreateDto);

}

@GetMapping("/teacher/{teacherId}")

public List<ReviewDisplayDto> getReviewsByTeacherId(@PathVariable Integer teacherId) {

return reviewService.getReviewsByTeacherId(teacherId);

}

@GetMapping("/userId/{userId}")

public List<ReviewDisplayDto> getReviewsByUserId(@PathVariable Integer userId) {

return reviewService.getReviewsByUserId(userId);

}

}

Файл SubjectController.java

package by.frozzel.springreviewer.controller;

import by.frozzel.springreviewer.dto.SubjectCreateDto;

import by.frozzel.springreviewer.dto.SubjectDisplayDto;

import by.frozzel.springreviewer.service.SubjectService;

import java.util.List;

import lombok.RequiredArgsConstructor;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

@RestController

@RequestMapping("/subjects")

@RequiredArgsConstructor

public class SubjectController {

private final SubjectService subjectService;

@PostMapping

public ResponseEntity<SubjectDisplayDto> createSubject(@RequestBody SubjectCreateDto dto) {

return ResponseEntity.ok(subjectService.createSubject(dto));

}

@GetMapping

public ResponseEntity<List<SubjectDisplayDto>> getAllSubjects() {

return ResponseEntity.ok(subjectService.getAllSubjects());

}

@GetMapping("/{id}")

public ResponseEntity<SubjectDisplayDto> getSubjectById(@PathVariable Integer id) {

return subjectService.getSubjectById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@GetMapping("/name/{name}")

public ResponseEntity<SubjectDisplayDto> getSubjectByName(@PathVariable String name) {

return subjectService.getSubjectByName(name)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PutMapping("/{id}")

public ResponseEntity<SubjectDisplayDto> updateSubject(@PathVariable Integer id,

@RequestBody SubjectCreateDto dto) {

return subjectService.updateSubject(id, dto)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteSubject(@PathVariable Integer id) {

return subjectService.deleteSubject(id);

}

}

Файл TeacherController.java

package by.frozzel.springreviewer.controller;

import by.frozzel.springreviewer.dto.TeacherCreateDto;

import by.frozzel.springreviewer.dto.TeacherDisplayDto;

import by.frozzel.springreviewer.service.TeacherService;

import java.util.List;

import lombok.RequiredArgsConstructor;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

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

@RestController

@RequestMapping("/teachers")

@RequiredArgsConstructor

public class TeacherController {

private final TeacherService teacherService;

@GetMapping

public List<TeacherDisplayDto> getAllTeachers() {

return teacherService.getAllTeachers();

}

@GetMapping("/{id}")

public TeacherDisplayDto getTeacherById(@PathVariable Integer id) {

return teacherService.getTeacherById(id);

}

@GetMapping("/search")

public TeacherDisplayDto getTeacherByFullName(@RequestParam String surname,

@RequestParam String name) {

return teacherService.getTeacherByFullName(surname, name);

}

@PostMapping

public TeacherDisplayDto createTeacher(@RequestBody TeacherCreateDto teacherCreateDto) {

return teacherService.createTeacher(teacherCreateDto);

}

@PutMapping("/{id}")

public TeacherDisplayDto updateTeacher(@PathVariable Integer id,

@RequestBody TeacherCreateDto teacherCreateDto) {

return teacherService.updateTeacher(id, teacherCreateDto);

}

@DeleteMapping("/{id}")

public void deleteTeacher(@PathVariable Integer id) {

teacherService.deleteTeacher(id);

}

@PostMapping("/{teacherId}/subjects/{subjectId}")

public ResponseEntity<String> assignSubjectToTeacher(

@PathVariable int teacherId,

@PathVariable int subjectId) {

teacherService.assignSubjectToTeacher(teacherId, subjectId);

return ResponseEntity.ok("Предмет успешно добавлен преподавателю");

}

@DeleteMapping("/{teacherId}/subjects/{subjectId}")

public ResponseEntity<Void> removeSubjectFromTeacher(@PathVariable int teacherId, @PathVariable int subjectId) {

teacherService.removeSubjectFromTeacher(teacherId, subjectId);

return ResponseEntity.noContent().build();

}

}

Файл UserController.java

package by.frozzel.springreviewer.controller;

import by.frozzel.springreviewer.dto.UserCreateDto;

import by.frozzel.springreviewer.dto.UserDisplayDto;

import by.frozzel.springreviewer.service.UserService;

import java.util.List;

import lombok.RequiredArgsConstructor;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

@RestController

@RequestMapping("/users")

@RequiredArgsConstructor

public class UserController {

private final UserService userService;

@PostMapping

public ResponseEntity<UserDisplayDto> createUser(@RequestBody UserCreateDto dto) {

return ResponseEntity.ok(userService.createUser(dto));

}

@GetMapping

public ResponseEntity<List<UserDisplayDto>> getAllUsers() {

return ResponseEntity.ok(userService.getAllUsers());

}

@GetMapping("/{id}")

public ResponseEntity<UserDisplayDto> getUserById(@PathVariable Integer id) {

return userService.getUserById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@GetMapping("/username/{username}")

public ResponseEntity<UserDisplayDto> getUserByUsername(@PathVariable String username) {

return userService.getUserByUsername(username)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PutMapping("/{id}")

public ResponseEntity<UserDisplayDto> updateUser(@PathVariable Integer id,

@RequestBody UserCreateDto dto) {

return userService.updateUser(id, dto)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteUser(@PathVariable Integer id) {

return userService.deleteUser(id)

? ResponseEntity.noContent().build()

: ResponseEntity.notFound().build();

}

}

Файл ReviewCreateDto.java

package by.frozzel.springreviewer.dto;

import java.time.LocalDate;

import lombok.Data;

import lombok.Setter;

@Data

@Setter

public class ReviewCreateDto {

private Integer userId;

private Integer teacherId;

private Integer subjectId;

private LocalDate date;

private Integer grade;

private String comment;

}

Файл ReviewDisplayDto.java

package by.frozzel.springreviewer.dto;

import java.time.LocalDate;

import lombok.Data;

import lombok.Setter;

@Data

@Setter

public class ReviewCreateDto {

private Integer userId;

private Integer teacherId;

private Integer subjectId;

private LocalDate date;

private Integer grade;

private String comment;

}

Файл SubjectCreateDto.java

package by.frozzel.springreviewer.dto;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class SubjectCreateDto {

private String name;

}

Файл SubjectDisplayDto.java

package by.frozzel.springreviewer.dto;

import java.util.List;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class SubjectDisplayDto {

private Integer id;

private String name;

private List<String> teacherNames;

}

Файл TeacherCreateDto.java

package by.frozzel.springreviewer.dto;

import lombok.Data;

@Data

public class TeacherCreateDto {

private String surname;

private String name;

private String patronym;

}

Файл TeacherDisplayDto.java

package by.frozzel.springreviewer.dto;

import java.util.ArrayList;

import java.util.List;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

@Data

@AllArgsConstructor

@Builder

public class TeacherDisplayDto {

private Integer id;

private String surname;

private String name;

private String patronym;

@Builder.Default

private List<String> subjects = new ArrayList<>();

}

Файл UserCreateDto.java

package by.frozzel.springreviewer.dto;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class UserCreateDto {

private String username;

}

Файл UserDisplayDto.java

package by.frozzel.springreviewer.dto;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class UserDisplayDto {

private Integer id;

private String username;

}

Файл ReviewMapper.java

package by.frozzel.springreviewer.mapper;

import by.frozzel.springreviewer.dto.ReviewCreateDto;

import by.frozzel.springreviewer.dto.ReviewDisplayDto;

import by.frozzel.springreviewer.model.Review;

import by.frozzel.springreviewer.model.Subject;

import by.frozzel.springreviewer.model.Teacher;

import by.frozzel.springreviewer.model.User;

import by.frozzel.springreviewer.repository.SubjectRepository;

import by.frozzel.springreviewer.repository.TeacherRepository;

import by.frozzel.springreviewer.repository.UserRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Component;

@Component

@RequiredArgsConstructor

public class ReviewMapper {

private final UserRepository userRepository;

private final TeacherRepository teacherRepository;

private final SubjectRepository subjectRepository;

public Review toEntity(ReviewCreateDto dto) {

Review review = new Review();

review.setDate(dto.getDate());

review.setGrade(dto.getGrade());

review.setComment(dto.getComment());

User user = userRepository.findById(dto.getUserId())

.orElseThrow(() -> new RuntimeException("User not found"));

Teacher teacher = teacherRepository.findById(dto.getTeacherId())

.orElseThrow(() -> new RuntimeException("Teacher not found"));

Subject subject = subjectRepository.findById(dto.getSubjectId())

.orElseThrow(() -> new RuntimeException("Subject not found"));

review.setUser(user);

review.setTeacher(teacher);

review.setSubject(subject);

return review;

}

public ReviewDisplayDto toDto(Review review) {

return new ReviewDisplayDto(

review.getId(),

review.getUser().getUsername(),

review.getTeacher().getSurname() + " "

+ review.getTeacher().getName() + " " + review.getTeacher().getPatronym(),

review.getSubject().getName(),

review.getDate(),

review.getGrade(),

review.getComment()

);

}

}

Файл SubjectMapper.java

package by.frozzel.springreviewer.mapper;

import by.frozzel.springreviewer.dto.SubjectCreateDto;

import by.frozzel.springreviewer.dto.SubjectDisplayDto;

import by.frozzel.springreviewer.model.Subject;

import java.util.Collections;

import java.util.stream.Collectors;

import org.springframework.stereotype.Component;

@Component

public class SubjectMapper {

public Subject toEntity(SubjectCreateDto dto) {

Subject subject = new Subject();

subject.setName(dto.getName());

subject.setTeachers(Collections.emptyList());

return subject;

}

public SubjectDisplayDto toDto(Subject subject) {

return new SubjectDisplayDto(

subject.getId(),

subject.getName(),

subject.getTeachers() != null

? subject.getTeachers().stream()

.map(teacher -> teacher.getSurname() + " " + teacher.getName() + " "

+ teacher.getPatronym())

.collect(Collectors.toList())

: Collections.emptyList()

);

}

}

Файл TeacherMapper.java

package by.frozzel.springreviewer.mapper;

import by.frozzel.springreviewer.dto.TeacherCreateDto;

import by.frozzel.springreviewer.dto.TeacherDisplayDto;

import by.frozzel.springreviewer.model.Subject;

import by.frozzel.springreviewer.model.Teacher;

import java.util.ArrayList;

import java.util.stream.Collectors;

import org.springframework.stereotype.Component;

@Component

public class TeacherMapper {

public Teacher toEntity(TeacherCreateDto dto) {

Teacher teacher = new Teacher();

teacher.setSurname(dto.getSurname());

teacher.setName(dto.getName());

teacher.setPatronym(dto.getPatronym());

teacher.setSubjects(new ArrayList<>());

return teacher;

}

public TeacherDisplayDto toDto(Teacher teacher) {

return TeacherDisplayDto.builder()

.id(teacher.getId())

.surname(teacher.getSurname())

.name(teacher.getName())

.patronym(teacher.getPatronym())

.subjects(teacher.getSubjects() != null

? teacher.getSubjects().stream()

.map(Subject::getName)

.collect(Collectors.toList())

: new ArrayList<>())

.build();

}

}

Файл UserMapper.java

package by.frozzel.springreviewer.mapper;

import by.frozzel.springreviewer.dto.UserCreateDto;

import by.frozzel.springreviewer.dto.UserDisplayDto;

import by.frozzel.springreviewer.model.User;

import org.springframework.stereotype.Component;

@Component

public class UserMapper {

public User toEntity(UserCreateDto dto) {

User user = new User();

user.setUsername(dto.getUsername());

return user;

}

public UserDisplayDto toDto(User user) {

return new UserDisplayDto(user.getId(), user.getUsername());

}

}

Файл Review.java

package by.frozzel.springreviewer.model;

import com.fasterxml.jackson.annotation.JsonBackReference;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.Table;

import java.time.LocalDate;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "reviews")

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class Review {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "user\_id", nullable = false)

@JsonBackReference

private User user;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "teacher\_id", nullable = false)

@JsonBackReference

private Teacher teacher;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "subject\_id", nullable = false)

@JsonBackReference

private Subject subject;

private LocalDate date;

private Integer grade;

private String comment;

}

Файл Subject.java

package by.frozzel.springreviewer.model;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.\*;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import java.util.List;

@Entity

@Table(name = "subjects")

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class Subject {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToMany(mappedBy = "subjects", cascade = {CascadeType.PERSIST, CascadeType.MERGE})

@JsonIgnore

private List<Teacher> teachers;

}

Файл Teacher.java

package by.frozzel.springreviewer.model;

import com.fasterxml.jackson.annotation.JsonBackReference;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.JoinTable;

import jakarta.persistence.ManyToMany;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

import java.util.ArrayList;

import java.util.List;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "teachers")

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class Teacher {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String surname;

private String name;

private String patronym;

@OneToMany(mappedBy = "teacher", cascade = CascadeType.ALL, orphanRemoval = true)

@JsonBackReference

private List<Review> reviews = new ArrayList<>();

@ManyToMany

@JoinTable(

name = "teacher\_subject",

joinColumns = @JoinColumn(name = "teacher\_id"),

inverseJoinColumns = @JoinColumn(name = "subject\_id")

)

@JsonManagedReference

private List<Subject> subjects = new ArrayList<>();

}

Файл User.java

package by.frozzel.springreviewer.model;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.FetchType;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToMany;

import jakarta.persistence.Table;

import java.util.List;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "users")

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@Column(unique = true, nullable = false)

private String username;

@OneToMany(mappedBy = "user", cascade = CascadeType.ALL,

orphanRemoval = true, fetch = FetchType.LAZY)

@JsonManagedReference

private List<Review> reviews;

}

Файл ReviewRepository.java

package by.frozzel.springreviewer.repository;

import by.frozzel.springreviewer.model.Review;

import java.util.List;

import jakarta.transaction.Transactional;

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

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

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

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

public interface ReviewRepository extends JpaRepository<Review, Integer> {

List<Review> findByTeacherId(Integer teacherId);

List<Review> findByUserId(Integer userId);

List<Review> findByUserUsername(String username);

@Transactional

@Modifying

@Query("DELETE FROM Review r WHERE r.subject.id = :subjectId")

void deleteBySubjectId(@Param("subjectId") int subjectId);

}

Файл SubjectRepository.java

package by.frozzel.springreviewer.repository;

import by.frozzel.springreviewer.model.Subject;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

@Repository

public interface SubjectRepository extends JpaRepository<Subject, Integer> {

Optional<Subject> findByName(String name);

}

Файл TeacherRepository.java

package by.frozzel.springreviewer.repository;

import by.frozzel.springreviewer.model.Teacher;

import java.util.Optional;

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

public interface TeacherRepository extends JpaRepository<Teacher, Integer> {

Optional<Teacher> findBySurnameAndName(String surname, String name);

}

Файл UserRepository.java

package by.frozzel.springreviewer.repository;

import by.frozzel.springreviewer.model.User;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

@Repository

public interface UserRepository extends JpaRepository<User, Integer> {

Optional<User> findByUsername(String username);

}

Файл ReviewService.java

package by.frozzel.springreviewer.service;

import by.frozzel.springreviewer.dto.ReviewCreateDto;

import by.frozzel.springreviewer.dto.ReviewDisplayDto;

import by.frozzel.springreviewer.mapper.ReviewMapper;

import by.frozzel.springreviewer.model.Review;

import by.frozzel.springreviewer.model.Teacher;

import by.frozzel.springreviewer.repository.ReviewRepository;

import by.frozzel.springreviewer.repository.SubjectRepository;

import by.frozzel.springreviewer.repository.TeacherRepository;

import by.frozzel.springreviewer.repository.UserRepository;

import jakarta.transaction.Transactional;

import java.util.List;

import lombok.RequiredArgsConstructor;

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

@Service

@RequiredArgsConstructor

public class ReviewService {

private final ReviewRepository reviewRepository;

private final ReviewMapper reviewMapper;

private final TeacherRepository teacherRepository;

private final SubjectRepository subjectRepository;

private final UserRepository userRepository;

@Transactional

public ReviewDisplayDto saveReview(ReviewCreateDto dto) {

userRepository.findById(dto.getUserId())

.orElseThrow(() -> new ResponseStatusException(HttpStatus

.NOT\_FOUND, "User not found"));

Teacher teacher = teacherRepository.findById(dto.getTeacherId())

.orElseThrow(() -> new ResponseStatusException(HttpStatus

.NOT\_FOUND, "Teacher not found"));

subjectRepository.findById(dto.getSubjectId())

.orElseThrow(() -> new ResponseStatusException(HttpStatus

.NOT\_FOUND, "Subject not found"));

boolean isTeaching = teacher.getSubjects().stream()

.anyMatch(s -> s.getId() == dto.getSubjectId());

if (!isTeaching) {

throw new ResponseStatusException(HttpStatus

.BAD\_REQUEST, "The teacher does not teach this subject");

}

Review review = reviewMapper.toEntity(dto);

return reviewMapper.toDto(reviewRepository.save(review));

}

public List<ReviewDisplayDto> getAllReviews() {

return reviewRepository.findAll()

.stream()

.map(reviewMapper::toDto)

.toList();

}

public ReviewDisplayDto getReviewById(Integer id) {

return reviewRepository.findById(id)

.map(reviewMapper::toDto)

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND,

"Reviews not found with id: " + id));

}

public void deleteReview(Integer id) {

reviewRepository.deleteById(id);

}

public ReviewDisplayDto updateReview(Integer id, ReviewCreateDto dto) {

Review review = reviewRepository.findById(id)

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND,

"Reviews not found with id: " + id));

review.setDate(dto.getDate());

review.setGrade(dto.getGrade());

review.setComment(dto.getComment());

return reviewMapper.toDto(reviewRepository.save(review));

}

public List<ReviewDisplayDto> getReviewsByTeacherId(Integer teacherId) {

List<Review> reviews = reviewRepository.findByTeacherId(teacherId);

if (reviews.isEmpty()) {

throw new ResponseStatusException(HttpStatus.NOT\_FOUND,

"Reviews not found for teacherId: " + teacherId);

}

return reviews.stream()

.map(reviewMapper::toDto)

.toList();

}

public List<ReviewDisplayDto> getReviewsByUserId(Integer userId) {

List<Review> reviews = reviewRepository.findByUserId(userId);

if (reviews.isEmpty()) {

throw new ResponseStatusException(HttpStatus.NOT\_FOUND,

"Reviews not found for userId: " + userId);

}

return reviews.stream()

.map(reviewMapper::toDto)

.toList();

}

public List<ReviewDisplayDto> getReviewsByUserUsername(String username) {

List<Review> reviews = reviewRepository.findByUserUsername(username);

if (reviews.isEmpty()) {

throw new ResponseStatusException(HttpStatus.NOT\_FOUND,

"Reviews not found for user: " + username);

}

return reviews.stream()

.map(reviewMapper::toDto)

.toList();

}

}

Файл SubjectService.java

package by.frozzel.springreviewer.service;

import by.frozzel.springreviewer.dto.SubjectCreateDto;

import by.frozzel.springreviewer.dto.SubjectDisplayDto;

import by.frozzel.springreviewer.mapper.SubjectMapper;

import by.frozzel.springreviewer.model.Subject;

import by.frozzel.springreviewer.model.Teacher;

import by.frozzel.springreviewer.repository.ReviewRepository;

import by.frozzel.springreviewer.repository.SubjectRepository;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collectors;

import by.frozzel.springreviewer.repository.TeacherRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.server.ResponseStatusException;

@Service

@RequiredArgsConstructor

public class SubjectService {

private final SubjectRepository subjectRepository;

private final SubjectMapper subjectMapper;

private final ReviewRepository reviewRepository;

private final TeacherRepository teacherRepository;

@Transactional

public SubjectDisplayDto createSubject(SubjectCreateDto dto) {

Subject subject = subjectMapper.toEntity(dto);

return subjectMapper.toDto(subjectRepository.save(subject));

}

public List<SubjectDisplayDto> getAllSubjects() {

return subjectRepository.findAll().stream()

.map(subjectMapper::toDto)

.toList();

}

public Optional<SubjectDisplayDto> getSubjectById(Integer id) {

return Optional.ofNullable(subjectRepository.findById(id).map(subjectMapper::toDto)

.orElseThrow(() -> new ResponseStatusException(HttpStatus

.NOT\_FOUND, "Subjects not found for id: " + id)));

}

public Optional<SubjectDisplayDto> getSubjectByName(String name) {

return subjectRepository.findByName(name).map(subjectMapper::toDto);

}

@Transactional

public Optional<SubjectDisplayDto> updateSubject(Integer id, SubjectCreateDto dto) {

return subjectRepository.findById(id)

.map(existingSubject -> {

existingSubject.setName(dto.getName());

return subjectMapper.toDto(subjectRepository.save(existingSubject));

});

}

@Transactional

public ResponseEntity<Void> deleteSubject(int subjectId) {

if (subjectId <= 0) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, "Subject ID must be a positive number");

}

Subject subject = subjectRepository.findById(subjectId)

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND, "Subject not found"));

for (Teacher teacher : subject.getTeachers()) {

teacher.getSubjects().remove(subject);

teacherRepository.save(teacher);

}

reviewRepository.deleteBySubjectId(subjectId);

subjectRepository.delete(subject);

return null;

}

}

Файл TeacherService.java

package by.frozzel.springreviewer.service;

import by.frozzel.springreviewer.dto.TeacherCreateDto;

import by.frozzel.springreviewer.dto.TeacherDisplayDto;

import by.frozzel.springreviewer.mapper.TeacherMapper;

import by.frozzel.springreviewer.model.Subject;

import by.frozzel.springreviewer.model.Teacher;

import by.frozzel.springreviewer.repository.SubjectRepository;

import by.frozzel.springreviewer.repository.TeacherRepository;

import jakarta.persistence.EntityNotFoundException;

import jakarta.transaction.Transactional;

import java.util.List;

import lombok.RequiredArgsConstructor;

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

@Service

@RequiredArgsConstructor

public class TeacherService {

private final TeacherRepository teacherRepository;

private final SubjectRepository subjectRepository;

private final TeacherMapper teacherMapper;

private static final String TEACHER\_NOT\_FOUND\_MESSAGE = "Teacher not found";

private static final String SUBJECT\_NOT\_FOUND\_MESSAGE = "Subject not found";

public List<TeacherDisplayDto> getAllTeachers() {

return teacherRepository.findAll().stream()

.map(teacherMapper::toDto)

.toList();

}

public TeacherDisplayDto getTeacherById(Integer id) {

Teacher teacher = teacherRepository.findById(id)

.orElseThrow(() -> new RuntimeException(TEACHER\_NOT\_FOUND\_MESSAGE));

return teacherMapper.toDto(teacher);

}

public TeacherDisplayDto createTeacher(TeacherCreateDto teacherCreateDto) {

Teacher teacher = teacherMapper.toEntity(teacherCreateDto);

return teacherMapper.toDto(teacherRepository.save(teacher));

}

public TeacherDisplayDto updateTeacher(Integer id, TeacherCreateDto teacherCreateDto) {

Teacher teacher = teacherRepository.findById(id)

.orElseThrow(() -> new RuntimeException(TEACHER\_NOT\_FOUND\_MESSAGE));

teacher.setSurname(teacherCreateDto.getSurname());

teacher.setName(teacherCreateDto.getName());

teacher.setPatronym(teacherCreateDto.getPatronym());

return teacherMapper.toDto(teacherRepository.save(teacher));

}

public void deleteTeacher(Integer id) {

teacherRepository.deleteById(id);

}

public TeacherDisplayDto getTeacherByFullName(String surname, String name) {

Teacher teacher = teacherRepository.findBySurnameAndName(surname, name)

.orElseThrow(() -> new RuntimeException(TEACHER\_NOT\_FOUND\_MESSAGE));

return teacherMapper.toDto(teacher);

}

@Transactional

public void assignSubjectToTeacher(int teacherId, int subjectId) {

Teacher teacher = teacherRepository.findById(teacherId)

.orElseThrow(() -> new EntityNotFoundException(TEACHER\_NOT\_FOUND\_MESSAGE));

Subject subject = subjectRepository.findById(subjectId)

.orElseThrow(() -> new EntityNotFoundException(SUBJECT\_NOT\_FOUND\_MESSAGE));

if (!teacher.getSubjects().contains(subject)) {

teacher.getSubjects().add(subject);

teacherRepository.save(teacher);

}

}

@Transactional

public void removeSubjectFromTeacher(int teacherId, int subjectId) {

Teacher teacher = teacherRepository.findById(teacherId)

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND, TEACHER\_NOT\_FOUND\_MESSAGE));

Subject subject = subjectRepository.findById(subjectId)

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND, SUBJECT\_NOT\_FOUND\_MESSAGE));

if (!teacher.getSubjects().contains(subject)) {

throw new ResponseStatusException(HttpStatus.BAD\_REQUEST, "Teacher does not teach this subject");

}

teacher.getSubjects().remove(subject);

teacherRepository.save(teacher);

}

}

Файл UserService.java

package by.frozzel.springreviewer.service;

import by.frozzel.springreviewer.dto.UserCreateDto;

import by.frozzel.springreviewer.dto.UserDisplayDto;

import by.frozzel.springreviewer.mapper.UserMapper;

import by.frozzel.springreviewer.model.User;

import by.frozzel.springreviewer.repository.UserRepository;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collectors;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

@RequiredArgsConstructor

public class UserService {

private final UserRepository userRepository;

private final UserMapper userMapper;

@Transactional

public UserDisplayDto createUser(UserCreateDto dto) {

User user = userMapper.toEntity(dto);

return userMapper.toDto(userRepository.save(user));

}

public List<UserDisplayDto> getAllUsers() {

return userRepository.findAll().stream()

.map(userMapper::toDto)

.toList();

}

public Optional<UserDisplayDto> getUserById(Integer id) {

return userRepository.findById(id).map(userMapper::toDto);

}

public Optional<UserDisplayDto> getUserByUsername(String username) {

return userRepository.findByUsername(username).map(userMapper::toDto);

}

@Transactional

public Optional<UserDisplayDto> updateUser(Integer id, UserCreateDto dto) {

return userRepository.findById(id)

.map(user -> {

user.setUsername(dto.getUsername());

return userMapper.toDto(userRepository.save(user));

});

}

@Transactional

public boolean deleteUser(Integer id) {

if (userRepository.existsById(id)) {

userRepository.deleteById(id);

return true;

}

return false;

}

}

# 4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

В результате тестирования сервиса через Postman, GET-запрос для отображения юзера по адресу [http://localhost:8080/](http://localhost:8080/reviews/1)users/frozzels выдал результат, представленный на рисунке 4.1.

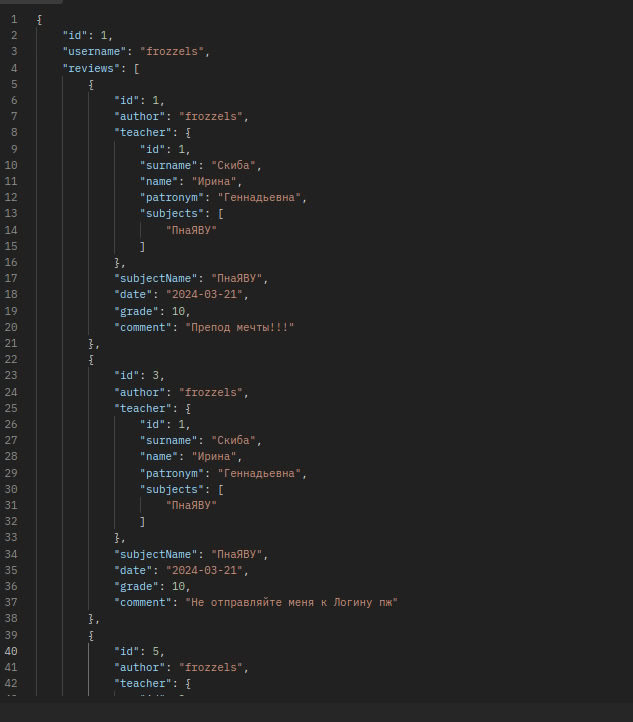


Рисунок 4.1 – JSON возвращаемый запросом на вывод юзера

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы была успешно реализована связь один ко многим (@OneToMany) и многие ко многим (@ManyToMany) с использованием базы данных PostgreSQL. Для управления данными применялся фреймворк Hibernate, что обеспечило удобство работы с объектно-реляционным отображением. Были разработаны все необходимые CRUD-операции для создания, чтения, обновления и удаления сущностей, что позволяет эффективно взаимодействовать с базой данных. Настройка каскадных операций и реализация ленивой и жадной загрузки данных также были выполнены, что добавило гибкости и производительности в работу с данными. Работа дала понимание важности правильного проектирования отношений в базах данных и их влияния на эффективность приложения.