Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №1 по**

**"Бизнес логике программных систем"**

Работу выполнил:

Тарасов А.С., P33112

Преподаватель:

Цопа Евгений Алексеевич

Санкт-Петербург

2021 год

Задание

Описать бизнес-процесс в соответствии с нотацией BPMN 2.0, после чего реализовать его в виде приложения на базе Spring Boot.  
**Вариант №210:** Coursera - <https://www.coursera.org/>

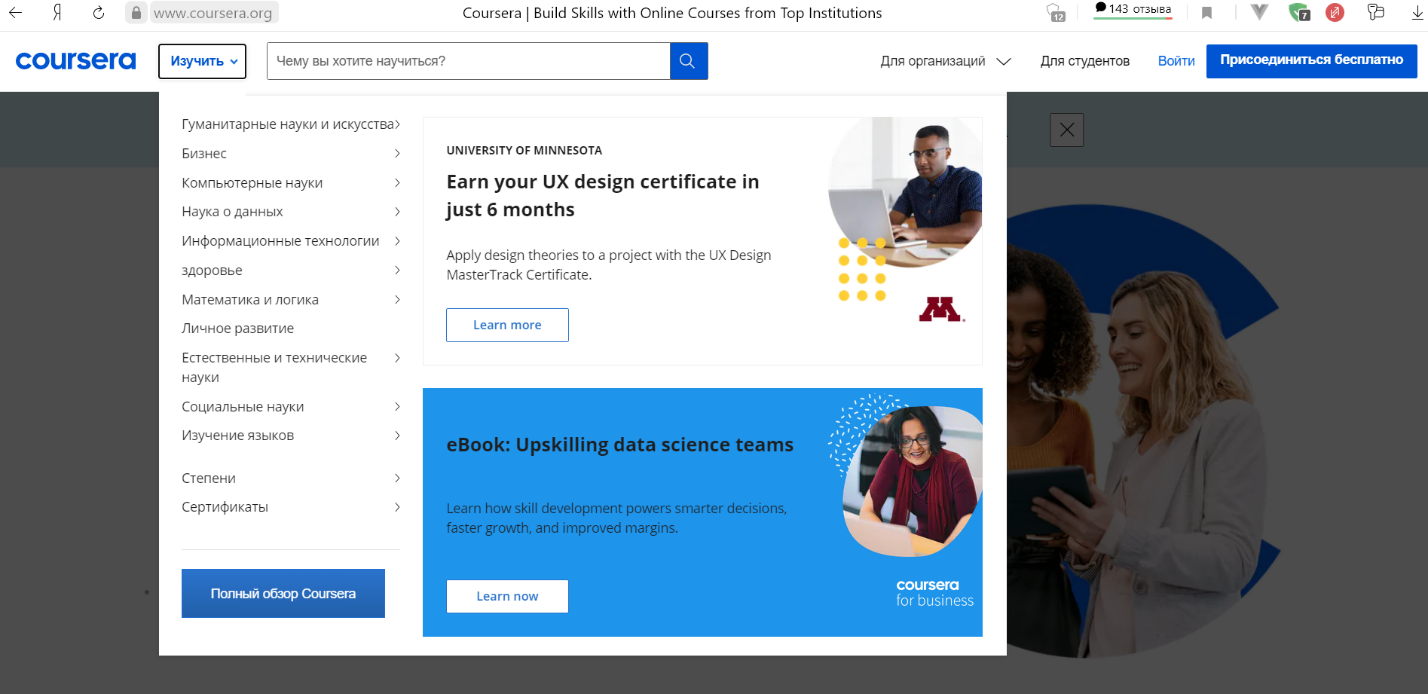
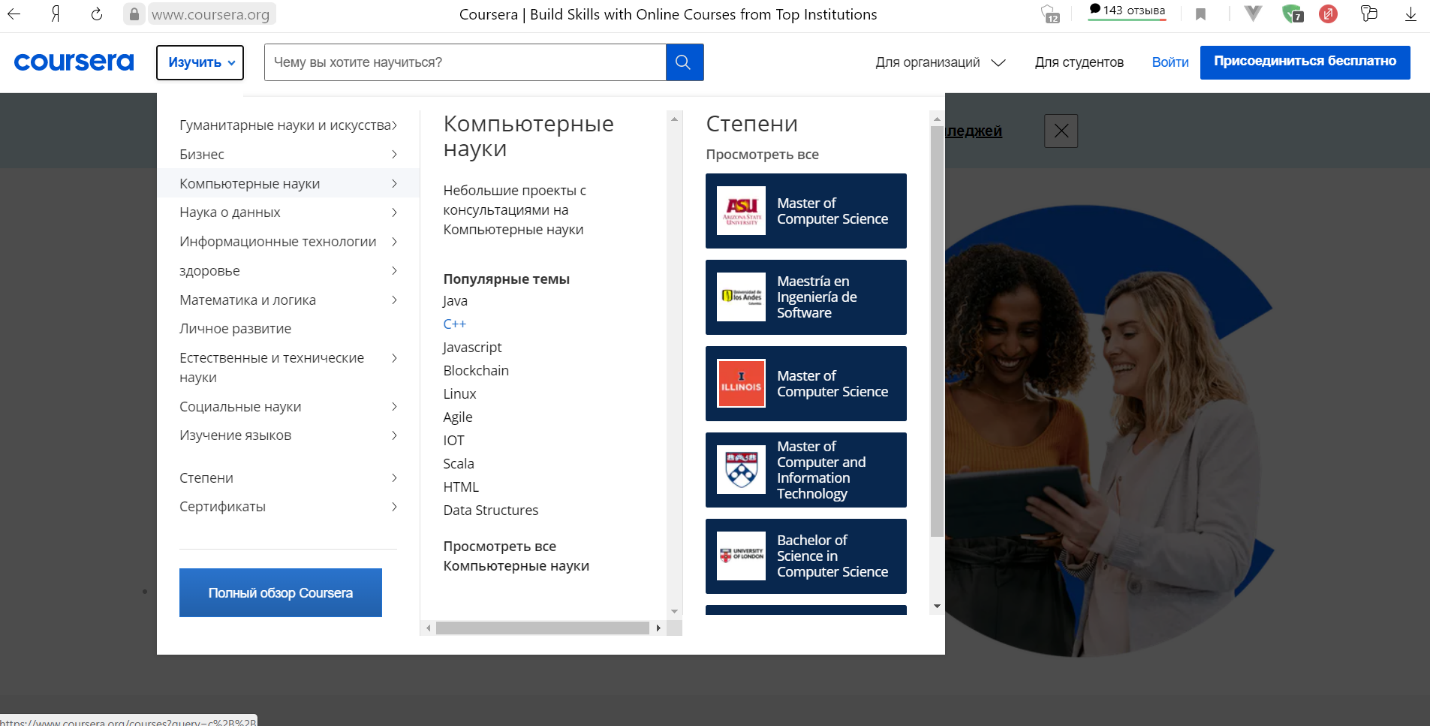
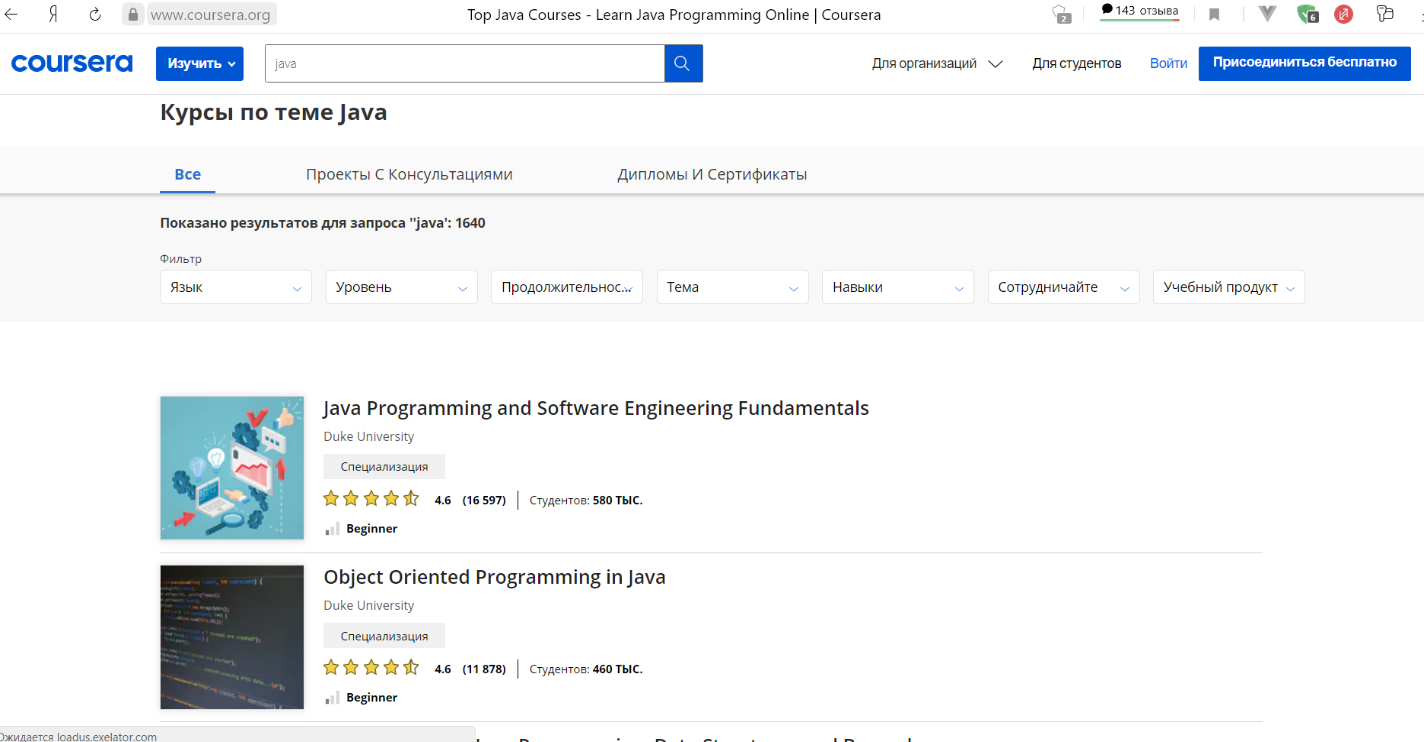
**Порядок выполнения работы:**

1. Выбрать один из бизнес-процессов, реализуемых сайтом из варианта задания.
2. Утвердить выбранный бизнес-процесс у преподавателя.
3. Специфицировать модель реализуемого бизнес-процесса в соответствии с требованиями BPMN 2.0.
4. Разработать приложение на базе Spring Boot, реализующее описанный на предыдущем шаге бизнес-процесс. Приложение должно использовать СУБД PostgreSQL для хранения данных, для всех публичных интерфейсов должны быть разработаны REST API.
5. Разработать набор curl-скриптов, либо набор запросов для REST клиента Insomnia для тестирования публичных интерфейсов разработанного программного модуля. Запросы Insomnia оформить в виде файла экспорта.
6. Развернуть разработанное приложение на сервере helios.

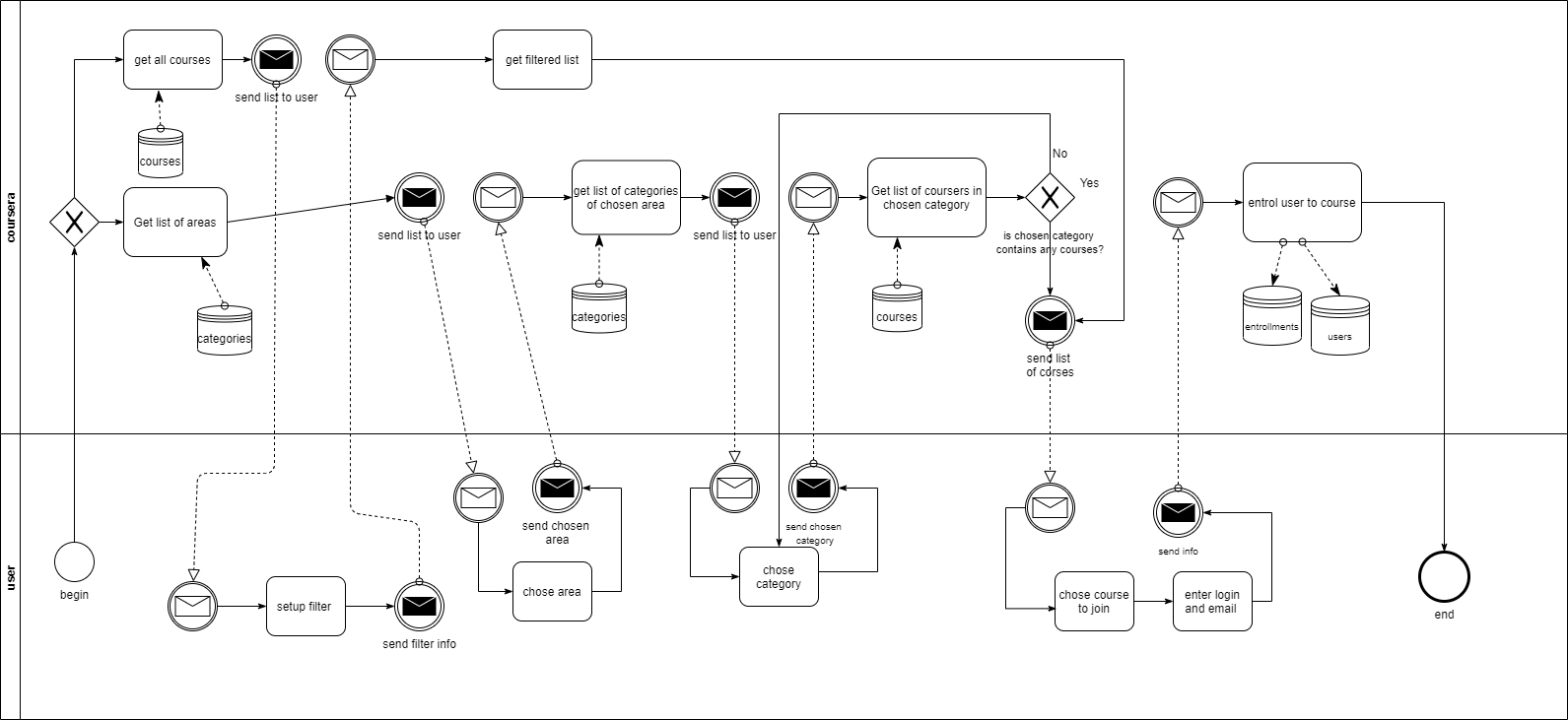
Выполнение

В качестве бизнес-процесса был выбран процесс выбора курса из существующих и запись на него.

Бизнес-процесс на исходной площадке:

1. Выбор области:
2. Выбор темы в области:
3. Получение списка курсов по выбранной области и теме:

Описание бизнес-процесса нотацией BPMN 2.0:



В ходе реализации было разработано REST приложение с помощью Spring Boot.

**Код**

1. Контроллер – уровень С в модели MVC – обрабатывает запросы пользователя и отправляет ответ.

*MainController.java:*

@RestController  
public class MainController {  
  
 @Autowired  
 StudentService studentService;  
  
 @Autowired  
 CourseService courseService;  
  
 @Autowired  
 CategoryService categoryService;  
  
  
 @GetMapping("/courses")  
 private List<Course> getAllCourses(){  
 return courseService.getAll();  
 }  
  
  
 @PostMapping("/course/area-filter")  
 private ArrayList<Course> getCoursesByArea(@RequestBody Category categoryDto){  
 ArrayList<Course> courses = new ArrayList<>();  
 ArrayList<Course> allCourses = (ArrayList<Course>) courseService.getAll();  
  
 for (Course course: allCourses) {  
 for (Category cat: course.getCategories()) {  
 if(cat.getArea().equals(categoryDto.getArea())){  
 courses.add(course);  
 break;  
 }  
 }  
 }  
 return courses;  
 }  
  
  
 @PostMapping("/course/theme-filter")  
 private ArrayList<Course> getCoursesByTheme(@RequestBody Category categoryDto){  
 ArrayList<Course> courses = new ArrayList<>();  
 ArrayList<Course> areaFilteredCourses = getCoursesByArea(categoryDto);  
  
 for (Course course: areaFilteredCourses) {  
 for (Category cat: course.getCategories()) {  
 if(cat.getTitle().equals(categoryDto.getTitle())){  
 courses.add(course);  
 break;  
 }  
 }  
 }  
 return courses;  
 }  
  
  
 @PostMapping("/course/filter")  
 private ArrayList<Course> getFilteredCourses(@RequestBody CourseFilterDto courseFilterDto) {  
 ArrayList<Course> filteredCourses = new ArrayList<>();  
 ArrayList<Course> allCourses = getCoursesByTheme(courseFilterDto.getCategory());  
  
 int duration;  
  
  
 for (Course course : allCourses) {  
 duration = course.getEndDate().getMonth() - course.getStatDate().getMonth();  
  
 if (courseFilterDto.getLeftDurationBorder() == courseFilterDto.getRightDurationBorder() ||  
 duration >= courseFilterDto.getLeftDurationBorder() && duration <= courseFilterDto.getRightDurationBorder()) {  
  
 for (int level : courseFilterDto.getLevels()) {  
 if (level ==0 || level == course.getLevel()){  
  
 for (String pltfrm: courseFilterDto.getPlatforms()) {  
 if(pltfrm.equals("") || pltfrm.equals(course.getPlatform())){  
  
 if(courseFilterDto.getMarkLeftBorder() == courseFilterDto.getMarkRightBorder() || course.getMark() >= courseFilterDto.getMarkLeftBorder() && course.getMark() <= courseFilterDto.getMarkRightBorder())  
 filteredCourses.add(course);  
 break;  
 }  
 }  
 }  
 }  
 } else break;  
 }  
  
 return filteredCourses;  
 }  
  
  
 @GetMapping("/getDto")  
 private CourseFilterDto getDto(){  
 CourseFilterDto courseFilterDto = new CourseFilterDto();  
  
 courseFilterDto.setCategory(categoryService.getById(1));  
 courseFilterDto.setLeftDurationBorder(1);  
 courseFilterDto.setRightDurationBorder(3);  
 courseFilterDto.setLevels(new int[]{1, 2});  
 courseFilterDto.setMarkLeftBorder(4.5);  
 courseFilterDto.setMarkRightBorder(4.7);  
 courseFilterDto.setPlatforms(new String[]{"Duke University"});  
  
  
 return courseFilterDto;  
 }  
  
  
 @GetMapping("/category/areas")  
 private TreeSet<String> getAllAreas(){  
 TreeSet<String> areas = new TreeSet<>();  
 for (Category cat: categoryService.getAll()) {  
 areas.add(new String(cat.getArea()));  
 }  
 return areas;  
 }  
  
  
 @PostMapping("/category/area")  
 private List<String> getAllCatByArea(@RequestBody Category categoryDto){  
 ArrayList<String> cats = new ArrayList<>();  
  
 for (Category cat: categoryService.getAll()) {  
 if (cat.getArea().equals(categoryDto.getArea()))  
 cats.add(cat.getTitle());  
 }  
 return cats;  
 }  
  
  
 @PostMapping("course/{id}/stud/add")  
 private String addStudentToCourse(@RequestBody Student newStudent, @PathVariable("id") int courseId){  
 Course course = courseService.getById(courseId);  
  
 Student student = studentService.getByEmail(newStudent.getEmail());  
 if(student == null)  
 student = newStudent;  
 course.getStudents().add(student);  
 courseService.edit(course);  
 try {  
 MailSender.*makeSend*(student.getEmail(), course);  
 return "success";  
 }catch (SendMessageExeption exception){  
 System.*out*.println("Some trouble with sending message");  
 return "failed";  
 } }}

1. Сущности – в моем сервисе я описал 3 сущности: 1 – Курс, 2 – Категория и 3 – Студент.

*Category.java:*

package buisnesslogic.entity;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
import javax.persistence.\*;  
import java.util.Collection;  
  
@Entity  
@Table(name = "category")  
public class Category {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int cat\_id;  
  
 @Column(name = "area")  
 private String area;  
  
 @Column(name = "title")  
 private String title;  
  
  
 @ManyToMany(fetch = FetchType.*LAZY*, cascade = CascadeType.*ALL*)  
 @JoinTable(name = "course\_to\_category",  
 joinColumns = @JoinColumn(name = "cat\_id"),  
 inverseJoinColumns = @JoinColumn(name = "course\_id")  
 )  
 private Collection<Course> courses;  
  
  
 public Category(){}  
  
 public Category(String area, String title){  
 this.area = area;  
 this.title = title;  
 }  
  
 @JsonIgnore  
 public int getCat\_id() {  
 return cat\_id;  
 }  
  
 public void setCat\_id(int cat\_id) {  
 this.cat\_id = cat\_id;  
 }  
  
<…>  
  
 @JsonIgnore  
 public Collection<Course> getCourses() {  
 return courses;  
 }  
  
 public void setCourses(Collection<Course> courses) {  
 this.courses = courses;  
 }  
}

*Course.java:*

package buisnesslogic.entity;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
  
import javax.persistence.\*;  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.Date;  
  
@Entity  
@Table(name = "courses")  
public class Course {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int course\_id;  
  
 @Column(name = "title")  
 private String title;  
  
 @Column(name = "level")  
 private int level;  
  
 @Column(name = "description")  
 private String description;  
  
 @Column(name = "start\_date")  
 @Temporal(TemporalType.*DATE*)  
 Date statDate;  
  
 @Column(name = "end\_date")  
 @Temporal(TemporalType.*DATE*)  
 Date endDate;  
  
 @Column(name = "platform")  
 private String platform;  
  
 @Column(name = "mark")  
 private double mark;  
  
  
*// @ManyToMany(mappedBy = "courses")  
// private Collection<Student> students;* @ManyToMany(fetch = FetchType.*LAZY*, cascade = CascadeType.*ALL*)  
 @JoinTable(name = "enrollments",  
 joinColumns = @JoinColumn(name = "course\_id"),  
 inverseJoinColumns = @JoinColumn(name = "stud\_id")  
 )  
 private Collection<Student> students;  
  
 @ManyToMany(mappedBy = "courses")  
 private Collection<Category> categories;  
  
  
 public Course(){};  
  
 public Course(String title, int level, String description,  
 Date statDate, Date endDate, String platform, double mark){  
 this.title = title;  
 this.level = level;  
 this.description = description;  
 this.statDate = statDate;  
 this.endDate = endDate;  
 this.platform = platform;  
 this.mark = mark;  
 }  
  
 public String getPlatform() {  
 return platform;  
 }  
  
 public void setPlatform(String platform) {  
 this.platform = platform;  
 }  
  
 public double getMark() {  
 return mark;  
 }  
  
 public void setMark(double mark) {  
 this.mark = mark;  
 }  
  
 @JsonIgnore  
 public Collection<Student> getStudents() {  
 return students;  
 }  
  
 public void setStudents(Collection<Student> students) {  
 this.students = students;  
 }  
  
 @JsonIgnore  
 public Collection<Category> getCategories() {  
 return categories;  
 }  
  
 public void setCategories(Collection<Category> categories) {  
 this.categories = categories;  
 }  
  
 public int getCourse\_id() {  
 return course\_id;  
 }  
  
 public void setCourse\_id(int course\_id) {  
 this.course\_id = course\_id;  
 }

. . .

}

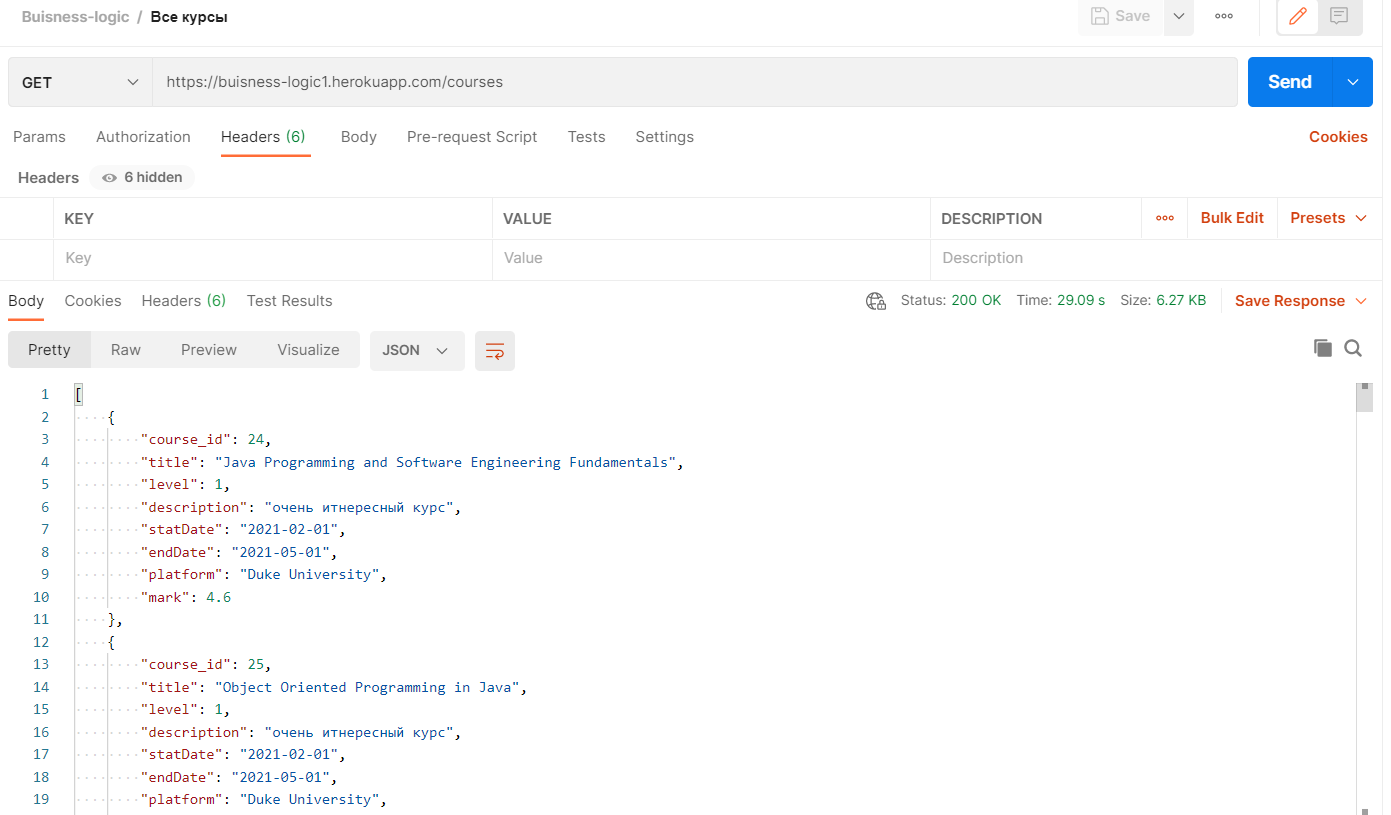
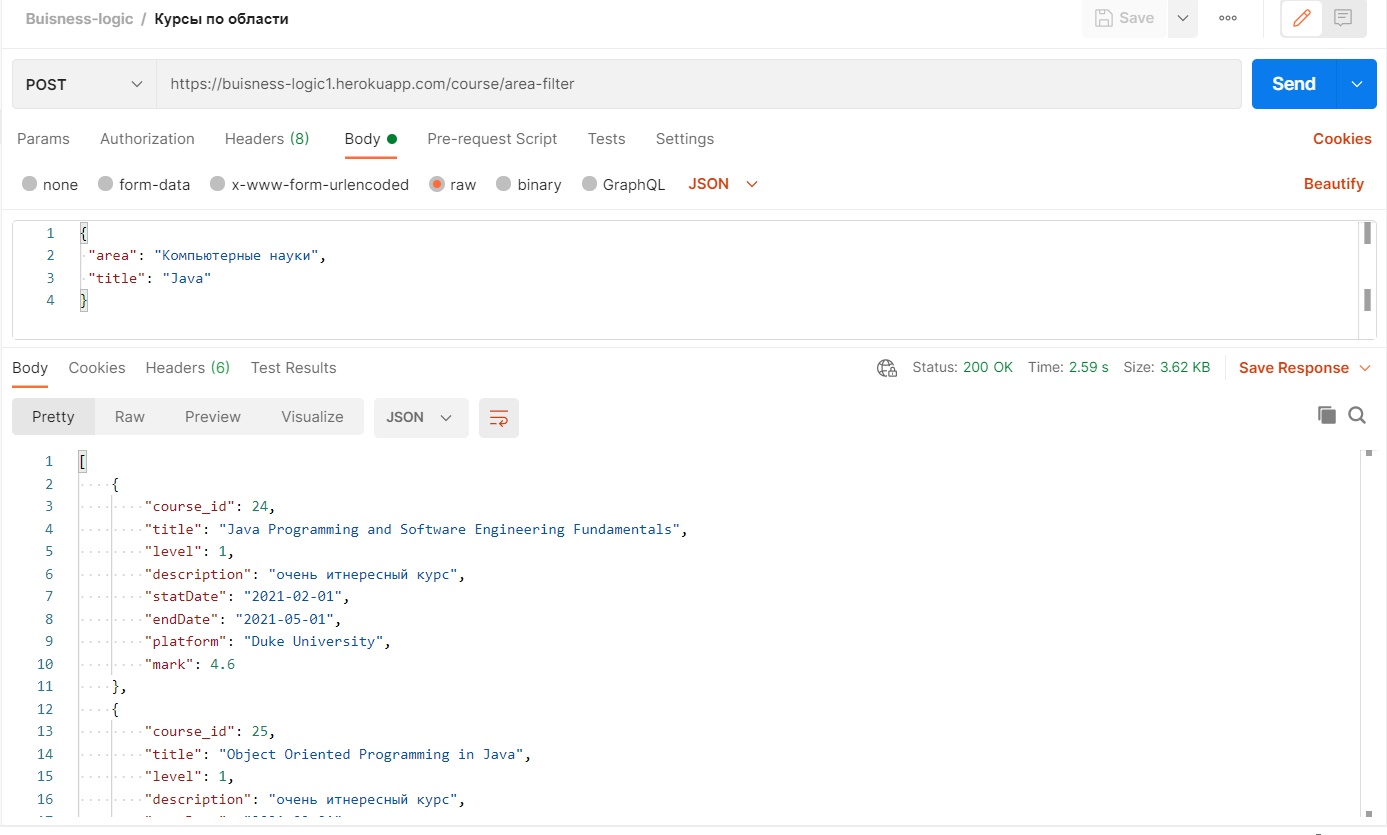
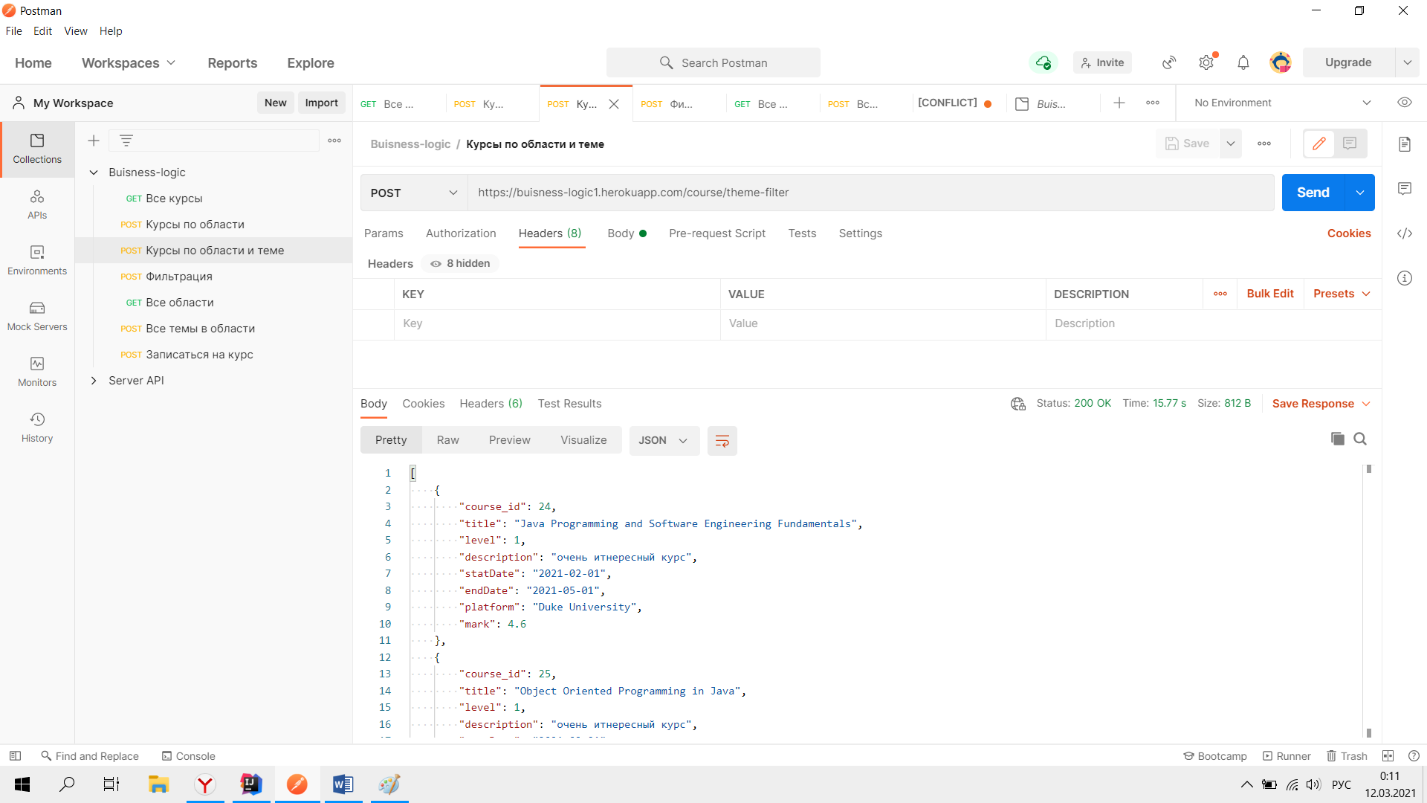
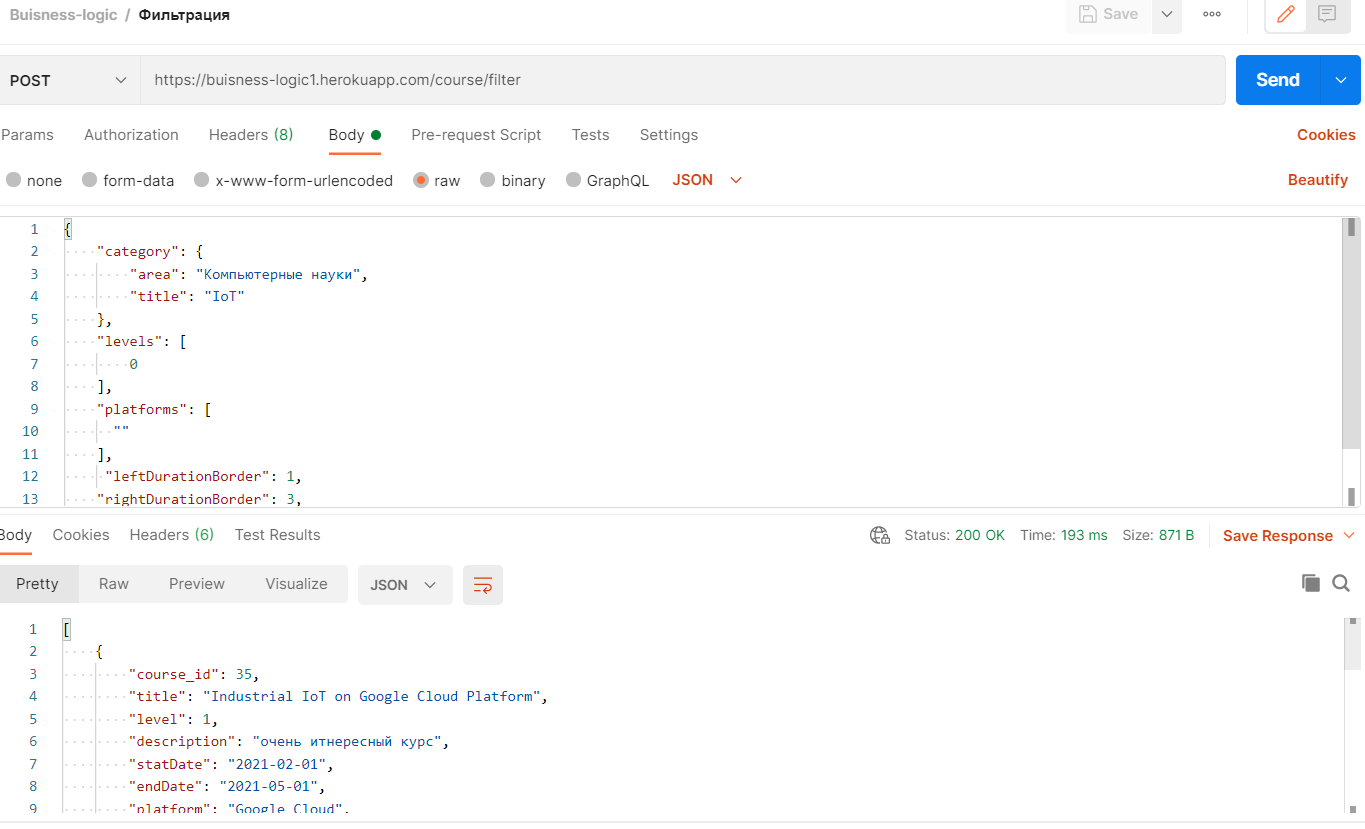
*Student.java*

package buisnesslogic.entity;  
  
import javax.persistence.\*;  
import java.util.Collection;  
  
@Entity  
@Table(name = "students")  
public class Student {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int stud\_id;  
  
 @Column(name = "name")  
 private String name;  
  
 @Column(name = "email")  
 private String email;  
  
  
  
 @ManyToMany(mappedBy = "students")  
 private Collection<Course> courses;  
  
 public void addCourse(Course course){  
 courses.add(course);  
 }  
  
 public int getStud\_id() {  
 return stud\_id;  
 }  
  
 public void setStud\_id(int stud\_id) {  
 this.stud\_id = stud\_id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Collection<Course> getCourses() {  
 return courses;  
 }  
  
 public void setCourses(Collection<Course> courses) {  
 this.courses = courses;  
 }  
}

1. Также для каждой сущности определены репозиторий и сервис – для реализации доступа к данным в бд. (Spring Data).

Приложение развернуто на PaaS-платформе Heroku.

Также был составлен набор запросов для HTTP-клиента Postman:

1. Получить все курсы:
2. Получить курсы в выбранной области:
3. Получить список курсов в выбранной категории:
4. Фильтрация по многим признакам:

И другие

Выводы по работе

Мною был выполнен анализ бизнес-процесса существующего сервиса. Выбранный процесс описан с помощью нотации BPMN 2.0 – системы специальных условных обозначений для моделирования бизнес-процессов.   
Я получил базовое представление об основных компонентах и принципах данной системы.   
  
Как уже было сказано, для реализации данного бизнес-процесса я использовал Spring Framework: использование инструментов для реализации IoC и DI, Spring MVC – для обеспечения архитектуры паттерна MVC и Spring Data для работы с базой данных.