|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: Разработка серверных частей интернет- ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть веб-приложения “Расписание” (генерация, хранение, отображение)

Студент: Анваржонов Жавохирбек Тулкинжонович

Группа: ИКБО-20-19

Работа представлена к защите \_\_\_\_\_\_\_\_\_\_\_\_ /Анваржонов Ж. Т. /

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанов А. А.

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/Лобанов А. А/

Оценка по итогам защиты: \_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине: Разработка серверных частей интернет-ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Анваржонов Жавохирбек Тулкинжонович

Группа: ИКБО-20-19

Срок представления к защите: 23.11.2021

Руководитель: Лобанов Александр Анатольевич доцент каф. ИиППО

**Тема:** «Серверная часть веб-приложения “Расписание” (генерация, хранение, отображение)»

**Исходные данные:** используемые технологии: HTML5, CSS3, Java, Jetbrains Intellij IDEA, MySQL, наличие: межстраничной навигации, внешнего вида страниц, соответствующего современным стандартам веб-разработки, использование паттерна проектирования MVC Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:** 1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Разработать слой клиентского представления веб-приложения 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: \_\_\_\_\_\_\_\_\_\_\_/Р. Г. Болбаков/, «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Задание на КР выдал: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/А. А. Лобанов /, «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

Задание на КР получил: \_\_ \_\_\_\_\_\_\_\_\_/ Ж. Т. Анваржонов/, «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

**ОГЛАВЛЕНИЕ**

[СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЙ 4](#_Toc88051235)

[ВВЕДЕНИЕ 5](#_Toc88051236)

[1. Анализ предметной области 6](#_Toc88051237)

[1.1. Выводы к разделу 1 9](#_Toc88051238)

[2. Выбор средств и технологии ведения разработки 10](#_Toc88051239)

[3. Архитектура приложения на основе выбранного паттерна 11](#_Toc88051240)

[3.1. Выводы к разделу 3 12](#_Toc88051241)

[4. Разработка серверной части интернет- ресурса 13](#_Toc88051242)

[4.1. Структура базы данных 13](#_Toc88051243)

[4.2. Работа в среде разработки Intelij Idea 13](#_Toc88051244)

[4.3. Выводы к разделу 4 29](#_Toc88051245)

[ЗАКЛЮЧЕНИЕ 30](#_Toc88051246)

[Приложение 31](#_Toc88051247)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 32](#_Toc88051248)

# СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЙ

ПО — Программное обеспечение;

ФИО — фамилия, имя, отчество;

БД — база данных;

СУБД — система правления базами данных.

# ВВЕДЕНИЕ

В настоящее время во всех учебных заведениях используется расписание занятий в электронном формате. Да, технологии, которые появились, позволяют больше не носить с собой бумажное расписание. И это здорово, студентам порой приходится носить огромный инвентарь некоего солдата, только вместо инвентаря – учебные книжки и было бы тяжело ещё носить лишний бумажный носитель.

Невозможно представить как бы справлялись не только студенты, но и сами преподаватели, если бы не было расписания занятий. Ведь если представить, что никакого расписания люди не придумывали бы, то мозгу приходилось бы запоминать огромный багаж из названий предметов, начала и конца времени занятий, какая дисциплина идет следующим, где находится та или иная аудитория.

Целью данной курсовой работы является разработка серверной части веб- приложения «расписание занятий для групп» и изучение принципа работы компетенций и индикаторов: ПК-1, ПК-1.2, пк-1.12, ПК-1.16

# Анализ предметной области

Предметной областью для данной курсовой работы является создание веб-ресурса наподобие его аналогов – расписания для студентов/ школьников

Чтобы провести анализ этой предметной области рассмотрим основной функционал, который существует в готовых похожих программных решениях.

В приложениях, в которых уклон сделан на расписание предметов, не так много уникального функционала. В каждом приложении есть возможность создавать расписание таким, каким ты его хочешь видеть, а именно: можно выбрать аудиторию в какой будет проводиться занятие, выбрать время начала дисциплины, а так же выбрать цвет блока для каждого типа дисциплины, что помогает различать обычные занятия от занятий, посещение которых обязательно или тех, где будет проводиться, например, итоговая работа.

У аналогов данного веб-приложения заметно одно отличие – простой дизайн для удобства пользователей. Оно то и понятно, ведь это веб-ресурс связанный с расписанием, а значит здесь не должно быть потайных кнопок, замудренного дизайна, у студентов и так со временем не ладно. Красота – в простоте, особенно для приложений, уклон которых сделан на расписание.

Рассмотрим некоторые ресурсы:

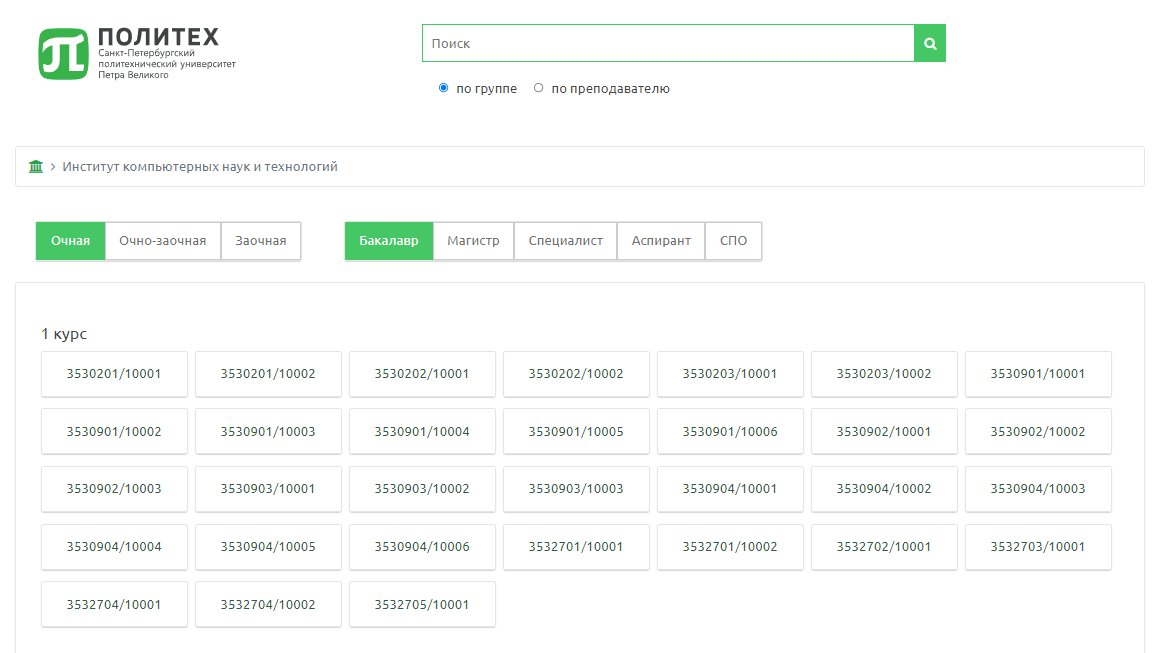


Рис. 1.1. Веб-ресурс Политеха расписания занятий (ч.1)

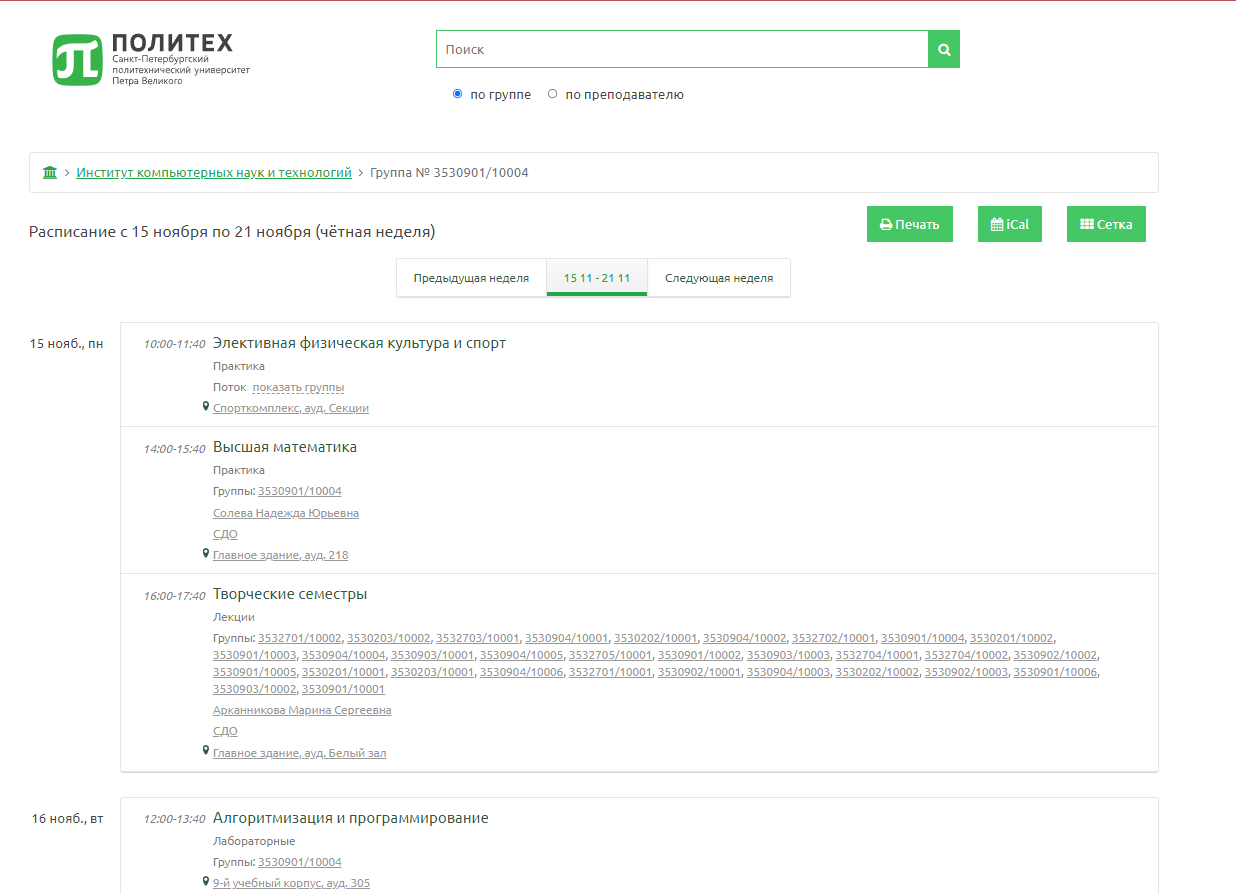


Рис. 1.2. Веб-ресурс Политеха расписания занятий (ч. 2)

На рисунке 2 изображён интерфейс расписания ВШЭ.

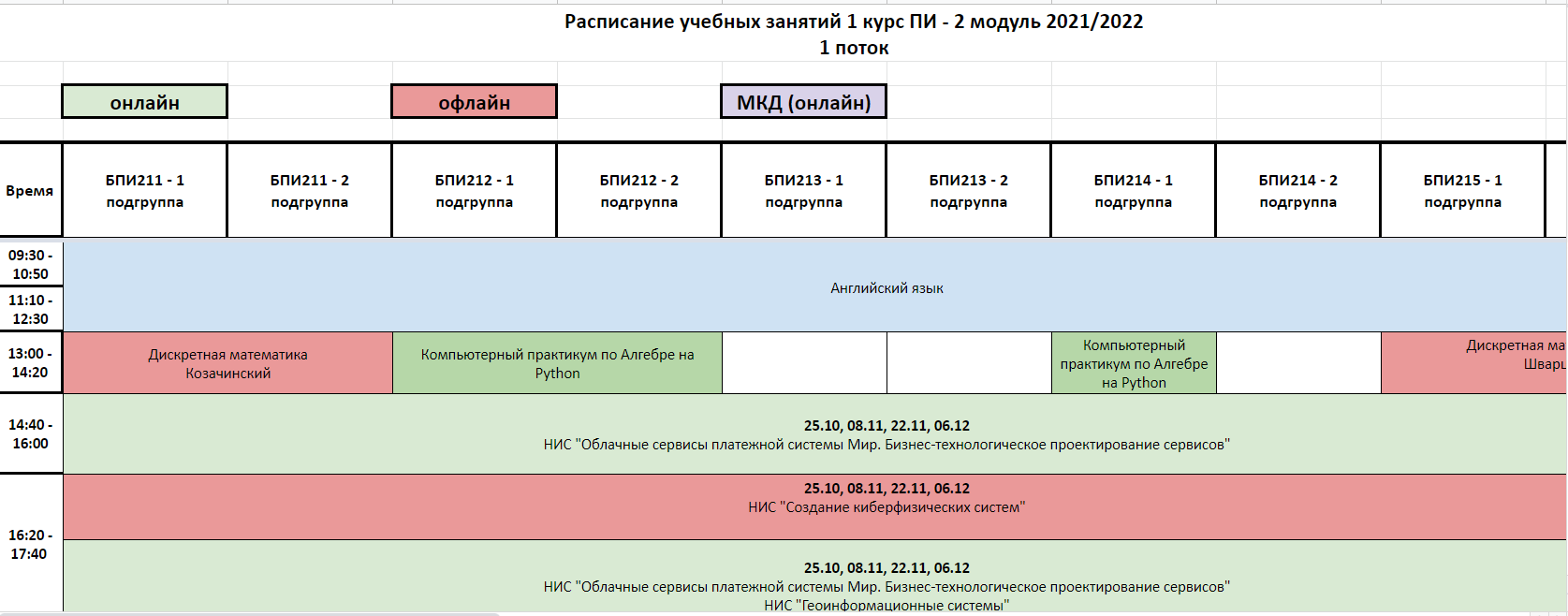


Рис. 2. Интерфейс интернет-магазина «Frenchkiss»

На рисунке 3 изображён интерфейс приложения с расписанием РТУ МИРЭА

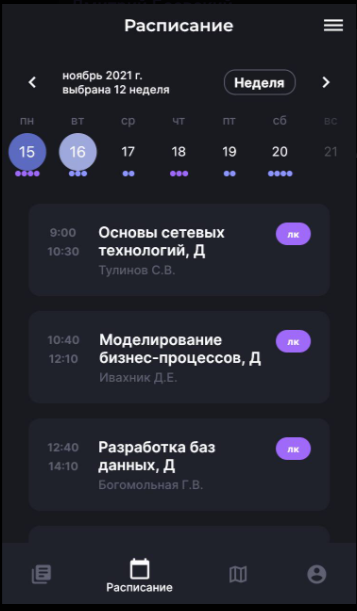


Рис. 3. Интерфейс интернет-магазина «Confaelshop»

## Выводы к разделу 1

Проведя анализ, можно сделать вывод, что для данной тематике не важен многоструктурный дизайн. Достаточно только самой важной информации – а именно:

1. Выбор расписания для конкретной группы.
2. Номер пары / Время начала- конца предмета.
3. Название предмета.
4. Аудитория, в которой будет проводиться дисциплина
5. Отображение блока с предметом, цвет которого показывает какую-то отличительную особенность.

Также не стоит забывать и про пользователей, которых захотят изменить что-то в расписании, добавить новую аудиторию, изменить расположение предметов. Для них нужно создать панель администратора, в котором они смогут:

1. Видеть список расписания
2. Добавлять/ удалять/ обновлять записи;

# Выбор средств и технологии ведения разработки

Веб- приложение будет реализовано при помощи языка программирования Java, среды разработки Inlelij IDEA и фреймворка Spring Boot со всеми его составляющими.

1. **Spring Boot** - это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.
2. **Spring Data JPA** - это библиотека, которая добавляет дополнительный уровень абстракции поверх ORM реализации JPA. По умолчанию Spring Data JPA использует Hibernate, в качестве ORM провайдера
3. **Thymeleaf** - современный серверный механизм Java-шаблонов для веб- и автономных сред, способный обрабатывать HTML, XML, JavaScript, CSS и даже простой текст. В нашем приложении он нужен для динамического отображения шаблонов.
4. **MySQL** - подключает драйвера Java Database Connectivity для выполнения SQL-запросов к базе данных.
5. **Maven -** инструмент для автоматизации сборки проектов

# Архитектура приложения на основе выбранного паттерна

В качестве паттерна проектирования архитектуры был выбран популярный MVC паттерн, потому что зачастую именно он используется в сердце Spring Boot.

Spring MVC обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними.

1. Model инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO-объектов
2. View отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.
3. Controller обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Покажем как будет выглядеть архитектура нашего приложения, а также опишем шаги от пользователя и до внесения данных в БД:

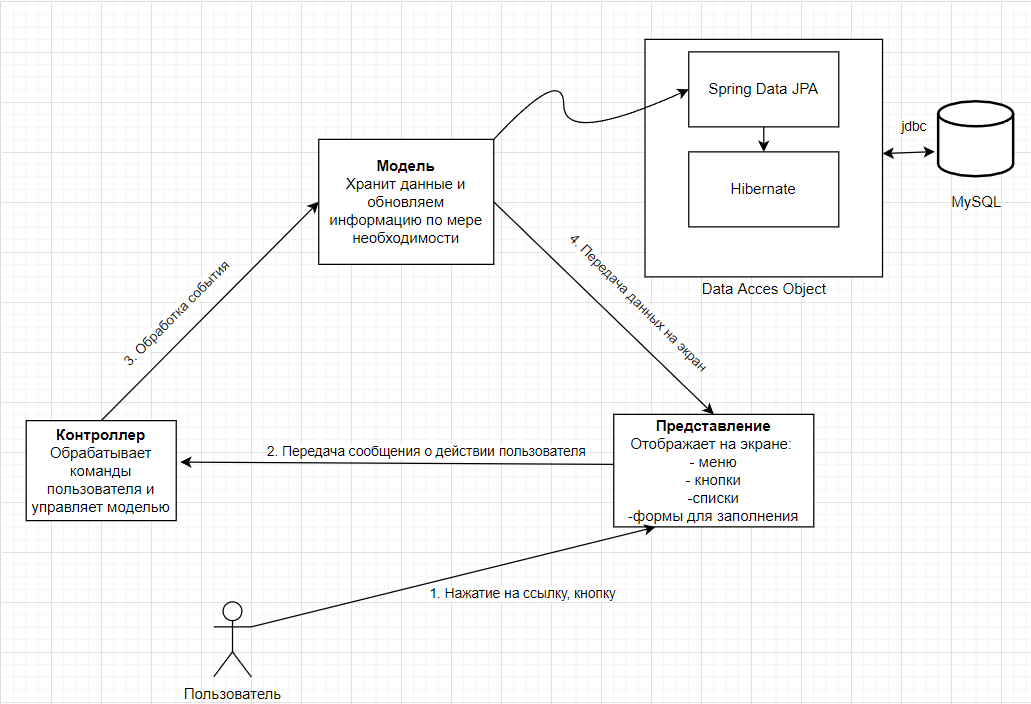


Рис. 4. Архитектура веб - приложения «расписания»

## Выводы к разделу 3

Несомненным плюсом MVC является единая глобальная архитектура приложения. Даже в сложных системах, разработчики (как те, которые разрабатывали систему, так и вновь присоединившиеся) могут легко ориентироваться в программных блоках. Например, если возникла ошибка в логике обработки данных, разработчик сразу отбрасывает 2-блока программы (controller и view) и занимается исследованием 3-го (model).

Также стоит отметить плюсом - Удобство выводить разные представления (view) для разных типов устройств, при этом пользуясь одними и теми же данными;

# Разработка серверной части интернет- ресурса

## Структура базы данных

В первую очередь составим модели сущностей и их связи в среде MySQL WorkBench, которые будем использовать в нашем приложении:

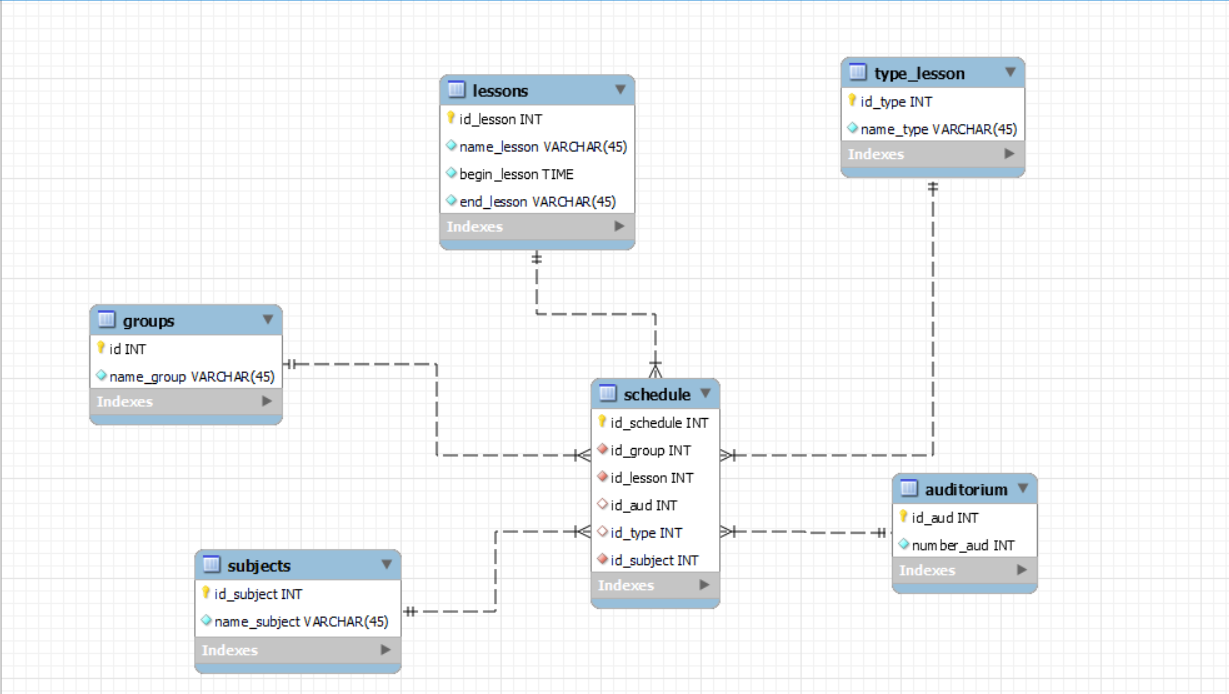
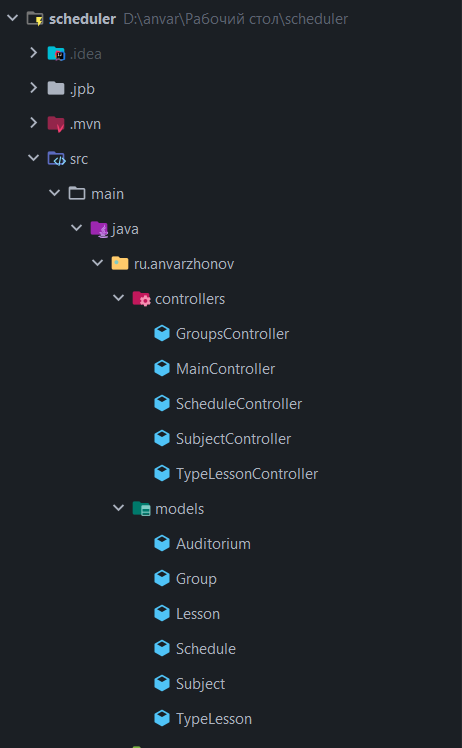
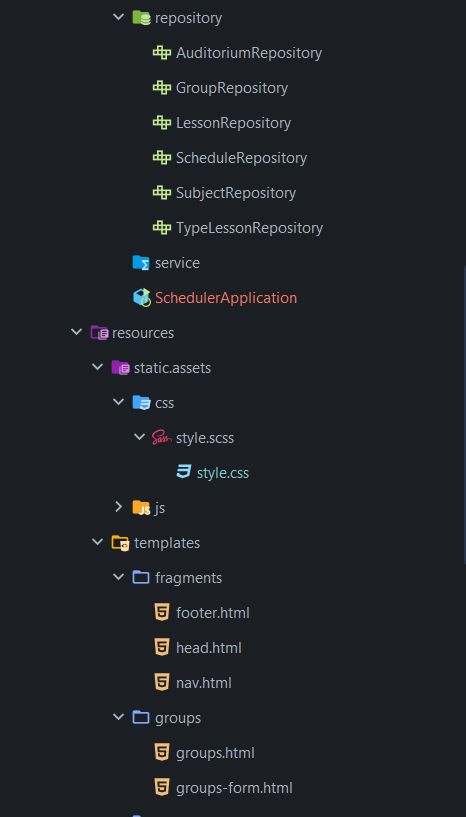


Рис. 5. Модель базы данных в виде EER Diagram

## Работа в среде разработки Intelij Idea

Покажем папочную структуру нашего интернет- ресурса:

* Контроллеры – обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.
* View - отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.
* Fragments – страницы с шаблонным кодом, чтобы исключить в отображении множество шаблонного кода.
* Model – этот блок инкапсулирует данные приложения.
* Repository – классы для взаимодействия с БД в Spring Data. Каждый репозиторий работает со своим классом-сущностью
* Test – позволяет тестировать бизнес-логику нашего приложения



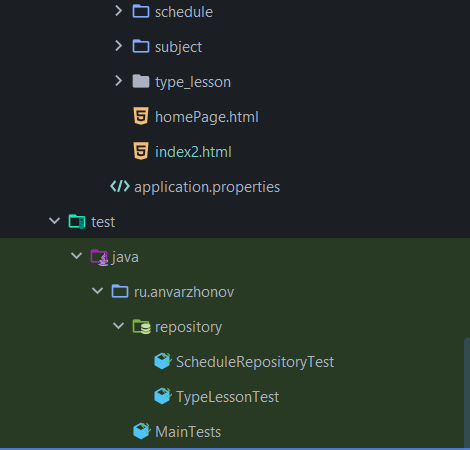


Рис. 6. Cтруктура папок веб-приложения

Ниже приведем код классов контроллеров, которые будут предназначены для перенаправления на соответствующие представления:

**GroupsController:**

*package* ru.anvarzhonov.controllers;  
  
*@Controller  
@RequestMapping*("/groups")  
*public class* GroupsController {  
 *private final* GroupRepository repository;  
  
 *public* GroupsController(GroupRepository repository) {  
 *this*.repository = repository;  
 }  
  
 *@GetMapping*()  
 *public* String showAll(Model model) {  
 model.addAttribute("groups", repository.findAll());  
 *return* "groups/groups";  
 }  
  
 *@GetMapping*("/new")  
 *public* String showScheduleNewForm(Model model) {  
 model.addAttribute("group", *new* Group());  
  
 *return* "groups/groups-form";  
 }  
  
 *@PostMapping*("/save")  
 *public* String saveSchedule(*@ModelAttribute* Group group) {  
 repository.save(group);  
  
 *return* "redirect:/groups";  
 }  
  
 *@GetMapping*("/edit/{id}")  
 *public* String editSchedule(*@PathVariable*(value = "id") Long id, Model model) {  
 Group group = repository.findById(id).get();  
 model.addAttribute("group", group);  
  
 *return* "groups/groups-form";  
 }  
 *@DeleteMapping*("/delete/{id}")  
 *public* String deleteSchedule(*@PathVariable*(value = "id") Long id, Model model) {  
 repository.deleteById(id);  
 *return* "redirect:/groups";  
 }  
}

**MainController:**

*package* ru.anvarzhonov.controllers;  
  
@Controller  
*public class* MainController {  
  
 *@Autowired* ScheduleRepository scheduleRepo;  
  
 *@Autowired* GroupRepository groupRepository;  
  
 *@GetMapping*("/schedule/{id}")  
 *public* String showIndex(*@PathVariable*("id") Long id, Model model) {  
 model.addAttribute("listSchedule",scheduleRepo.findAllByGroupId(id));  
  
 *return* "index2";  
 }  
  
 @GetMapping("/homepage")  
 *public* String homePage(Model model) {  
 model.addAttribute("groups", groupRepository.findAll());  
  
 *return* "homePage";  
 }  
}

**ScheduleControler:**

*package* ru.anvarzhonov.controllers;  
  
*@Controller  
public class* ScheduleController {  
 *private final* ScheduleRepository scheduleRepository;  
 *private final* LessonRepository lessonRepository;  
 *private final* TypeLessonRepository typeLessonRepository;  
 *private final* AuditoriumRepository auditoriumRepository;  
 *private final* GroupRepository groupRepository;  
 *private final* SubjectRepository subjectRepository;  
  
 *public* ScheduleController(ScheduleRepository scheduleRepository,  
 LessonRepository lessonRepository,  
 TypeLessonRepository typeLessonRepository,  
 AuditoriumRepository auditoriumRepository,  
 GroupRepository groupRepository,  
 SubjectRepository subjectRepository) {  
 *this*.scheduleRepository = scheduleRepository;  
 *this*.lessonRepository = lessonRepository;  
 *this*.typeLessonRepository = typeLessonRepository;  
 *this*.auditoriumRepository = auditoriumRepository;  
 *this*.groupRepository = groupRepository;  
 *this*.subjectRepository = subjectRepository;  
 }  
  
 *@GetMapping*("/schedule")  
 *public* String listSchedule(Model model) {  
 List<Schedule> listSchedule = scheduleRepository.findAll();  
 model.addAttribute("listSchedule", listSchedule);  
 *return* "schedule/schedule";  
 }  
  
 *@GetMapping*("/schedule/new")  
 *public* String showScheduleNewForm(Model model) {  
 List<Lesson> listLessons = lessonRepository.findAll();  
 List<TypeLesson> listTypeLesson = typeLessonRepository.findAll();  
 List<Auditorium> listAuditoriums = auditoriumRepository.findAll();  
 List<Group> listGroups = groupRepository.findAll();  
 List<Subject> listSubjects = subjectRepository.findAll();  
  
  
 model.addAttribute("schedule", *new* Schedule());  
 model.addAttribute("listLesson", listLessons);  
 model.addAttribute("listTypeLesson", listTypeLesson);  
 model.addAttribute("listAud", listAuditoriums);  
 model.addAttribute("listGroups", listGroups);  
 model.addAttribute("listSubjects", listSubjects);  
  
 *return* "schedule/scheduleForm";  
 }  
  
 *@PostMapping*("/schedule/save")  
 *public* String saveSchedule(*@ModelAttribute* Schedule schedule) {  
 scheduleRepository.save(schedule);  
  
 *return* "redirect:/schedule";  
 }  
  
 *@GetMapping*("/schedule/edit/{id}")  
 *public* String editSchedule(*@PathVariable*(value = "id") Long id, Model model) {  
 Schedule schedule = scheduleRepository.findById(id).get();  
  
 List<Lesson> listLessons = lessonRepository.findAll();  
 List<TypeLesson> listTypeLesson = typeLessonRepository.findAll();  
 List<Auditorium> listAuditoriums = auditoriumRepository.findAll();  
 List<Group> listGroups = groupRepository.findAll();  
 List<Subject> listSubjects = subjectRepository.findAll();  
  
  
 model.addAttribute("schedule", schedule);  
 model.addAttribute("listLesson", listLessons);  
 model.addAttribute("listTypeLesson", listTypeLesson);  
 model.addAttribute("listAud", listAuditoriums);  
 model.addAttribute("listGroups", listGroups);  
 model.addAttribute("listSubjects", listSubjects);  
  
 *return* "schedule/scheduleForm";  
 }

*@GetMapping*("/schedule/delete/{id}")  
 *public* String deleteSchedule(*@PathVariable*(value = "id") Long id, Model model) {  
 scheduleRepository.deleteById(id);  
 *return* "redirect:/schedule";  
 }  
  
  
}

**SubjectController:**

*package* ru.anvarzhonov.controllers;  
*@Controller  
@RequestMapping*("/subject")  
*public class* SubjectController {  
 *private final* SubjectRepository repository;  
  
 *public* SubjectController(SubjectRepository repository) {  
 *this*.repository = repository;  
 }  
  
 @GetMapping()  
 *public* String showAll(Model model) {  
 model.addAttribute("subjects", repository.findAll());  
 *return* "subject/subject";  
 }  
  
 @GetMapping("/new")  
 *public* String showScheduleNewForm(Model model) {  
 model.addAttribute("subject", *new* Subject());  
  
 *return* "subject/subject-form";  
 }  
  
 @PostMapping("/save")  
 *public* String saveSchedule(@ModelAttribute Subject subject) {  
 repository.save(subject);  
  
 *return* "redirect:/subject";  
 }  
  
 @GetMapping("/edit/{id}")  
 *public* String editSchedule(@PathVariable(value = "id") Long id, Model model) {  
 Subject subject = repository.findById(id).get();  
 model.addAttribute("subject", subject);  
  
 *return* "subject/subject-form";  
 }  
 @DeleteMapping("/delete/{id}")  
 *public* String deleteSchedule(@PathVariable(value = "id") Long id, Model model) {  
 repository.deleteById(id);  
 *return* "redirect:/subject";  
 }  
}

Покажем код для создания наших сущностей:

**Lesson:**

*package* ru.anvarzhonov.models;  
  
*@Entity  
@Data  
@Table*(name = "lessons")  
*@AllArgsConstructor  
@NoArgsConstructor  
 @Builder  
public class* Lesson {  
 *@Id  
 @Column*(name = "id\_lesson", nullable = *false*)  
 *@GeneratedValue*(strategy = GenerationType.IDENTITY)  
 *private* Long id;  
  
  
 *@Column*(name = "begin\_lesson")  
 *private* String beginLesson;  
  
 @Column(name = "end\_lesson")  
 *private* String endLesson;  
  
 @OneToMany(mappedBy = "lesson", fetch = FetchType.LAZY,  
 cascade = CascadeType.ALL)  
 @ToString.Exclude  
 *private* List<Schedule> schedules;  
  
}

**Schedule:**

*package* ru.anvarzhonov.models;  
  
*@Entity  
@Table*(name = "schedule")  
*@Getter  
@Setter  
@ToString  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
public class* Schedule {  
 *@Id  
 @Column*(name = "id\_schedule", nullable = *false*)  
 *@GeneratedValue*(strategy = GenerationType.IDENTITY)  
 *private* Long id;  
  
  
 *@Column*(name = "id\_weekday")  
 *private* Long weekdayId;  
  
 *@ManyToOne*(fetch = FetchType.EAGER, optional = *false*)  
 *@JoinColumn*(name = "id\_lesson",  
 nullable = *false*)  
 *private* Lesson lesson;  
 *@ManyToOne*(fetch = FetchType.LAZY, optional = *false*)  
 *@JoinColumn*(name = "id\_type", nullable = *true*)  
 *@ToString.Exclude  
 private* TypeLesson typeLesson;  
  
 *@ManyToOne*(fetch = FetchType.LAZY, optional = *false*)  
 *@JoinColumn*(name = "id\_group", nullable = *false*)  
 *@ToString.Exclude  
 private* Group group;  
  
 *@ManyToOne*(fetch = FetchType.LAZY, optional = *false*)  
 *@JoinColumn*(name = "id\_aud", nullable = *true*)  
 *@ToString.Exclude  
 private* Auditorium auditorium;  
  
 *@ManyToOne*(fetch = FetchType.LAZY, optional = *false*)  
 *@JoinColumn*(name = "id\_subject", nullable = *false*)  
 *@ToString.Exclude  
 private* Subject subject;  
}

**Subject:**

*package* ru.anvarzhonov.models;  
  
  
*import* lombok.\*;  
  
*import* javax.persistence.\*;  
*import* java.util.List;  
  
*@Entity  
@Table*(name = "subjects")  
*@AllArgsConstructor  
@NoArgsConstructor  
@Setter  
@Getter  
@Data  
public class* Subject {  
 *@Id  
 @Column*(name = "id", nullable = *false*)  
 *@GeneratedValue*(strategy = GenerationType.IDENTITY)  
 *private* Long id;  
  
 *@Column*(name = "name\_subject")  
 *private* String nameSubject;  
  
 *@OneToMany*(mappedBy = "subject", fetch = FetchType.LAZY,  
 cascade = CascadeType.ALL)  
 *@ToString.Exclude  
 private* List<Schedule> schedules;  
  
}

Во всех классах используется Lombok для удобства работы с сеттерами, геттерами, конструкторами.

Благодаря ORM реализации с помощью наших аннотаций будет произведена автоматическая связь с базой данных и если мы будем изменять что-то в объектном виде, то в базе данных появится соответствующее изменение

Дальше покажем данные сущности в БД MySQL:

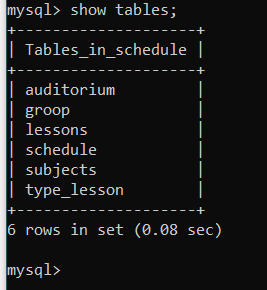


Рис. 7. Созданные таблицы наших сущностей

Покажем код репозиториев для каждой сущности для различных манипуляций базой данных:

**SubjectRepository:**

*package* ru.anvarzhonov.repository;  
  
*import* org.springframework.data.jpa.repository.JpaRepository;  
*import* org.springframework.stereotype.*Repository*;  
*import* ru.anvarzhonov.models.Subject;  
  
*@Repository  
public interface* SubjectRepository *extends* JpaRepository<Subject, Long> {  
}

**ScheduleRepository:**

*package* ru.anvarzhonov.repository;  
  
*import* org.springframework.data.jpa.repository.JpaRepository;  
*import* org.springframework.data.jpa.repository.*Query*;  
*import* org.springframework.stereotype.*Repository*;  
*import* ru.anvarzhonov.models.Auditorium;  
*import* ru.anvarzhonov.models.Lesson;  
*import* ru.anvarzhonov.models.Schedule;  
  
*import* java.util.List;  
  
*@Repository  
public interface* ScheduleRepository *extends* JpaRepository<Schedule, Long> {  
 *// select \* from schedule s inner join auditorium a on s.id\_aud=a.id\_aud where a.id\_aud= ?* Schedule findByAuditorium\_Id(Long auditorium\_id);  
  
 *@Query*("select s from Schedule s where s.lesson.id = :lesson\_id")  
 List<Schedule> findByLessonId(Long lesson\_id);  
  
 List<Schedule> findAllByGroupId(Long id);

Здесь мы прописали дополнительные методы для поиска всех расписаний для конкретной группы.

Остальные репозитории ничем не отличаются от приведенных выше, поэтому их здесь приводить не будем.

Покажем фрагменты, которые в нашем приложении используются для сокращении кода

**Nav:**

<html *lang*="en">  
<nav *class*="navbar navbar-expand-lg navbar-dark bg-dark static-top" *th:fragment*="navbar">  
 <div *class*="container">  
 <a *class*="navbar-brand" *th:href*="@{/homepage}">Главная страница</a>  
 <button *class*="navbar-toggler" *type*="button" *data-toggle*="collapse" *data-target*="#navbarResponsive"  
 *aria-controls*="navbarResponsive"  
 *aria-expanded*="false" *aria-label*="Toggle navigation">  
 <span *class*="navbar-toggler-icon"></span>  
 </button>  
 <div *class*="collapse navbar-collapse" *id*="navbarResponsive">  
 <ul *class*="navbar-nav ml-auto">  
 <li *class*="nav-item">  
 <a *class*="nav-link" *th:href*="@{/groups}">Группы  
 <span *class*="sr-only">(current)</span>  
 </a>  
 </li>  
 <li *class*="nav-item">  
 <a *class*="nav-link" *th:href*="@{/schedule}">Расписание</a>  
 </li>  
 <li *class*="nav-item">  
 <a *class*="nav-link" *th:href*="@{/subject}">Дисциплины</a>  
 </li>  
 <li *class*="nav-item">  
 <a *class*="nav-link" *th:href*="@{/type-lesson}">Типы дисциплин</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
</nav>  
</html>

**Header:**

<html *lang*="en">  
<head *th:fragment*="header">  
 <meta *charset*="UTF-8">  
 <link *rel*="stylesheet" *href*="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"  
 *integrity*="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" *crossorigin*="anonymous">  
</head>  
</html>

**Footer:**

<html *lang*="en">  
<th:block *th:fragment*="scripts">  
 <script *src*="https://code.jquery.com/jquery-3.5.1.slim.min.js"  
 *integrity*="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"  
 *crossorigin*="anonymous"></script>  
 <script *src*="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"  
 *integrity*="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"  
 *crossorigin*="anonymous"></script>  
 <script *src*="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"  
 *integrity*="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"  
 *crossorigin*="anonymous"></script>  
 <script *src*="https://kit.fontawesome.com/e8cd5e0a28.js" *crossorigin*="anonymous"></script>  
</th:block>  
</html>

Теперь нам не придется дублировать код, а просто заменять с помощью thymeleaf в нужном месте.

Покажем код главной страницы, куда попадает пользователь при переходе по адресу /homepage:

<!DOCTYPE *html*>  
<html *lang*="en" *xmlns:th*="http://www.thymeleaf.org">  
<head>  
 <meta *charset*="UTF-8">  
 <title>HomePage</title>  
 <link *rel*="stylesheet" *href*="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"  
 *integrity*="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" *crossorigin*="anonymous">  
</head>  
<body>  
<div *th:replace*="fragments/nav ::navbar"></div>  
<div *class*="container text-center">  
 <h1>Расписание групп</h1>  
 <div *th:each*="group: ${groups}">  
 <a *class*="h2" *th:href*="@{/schedule/{id}(id=${group.id})}" *th:text*="${group.nameGroup}"></a>  
 </div>  
  
</div>  
  
  
  
<div *th:replace*="fragments/footer :: scripts"></div>  
</body>  
</html>

Покажем как выглядит главная страница:

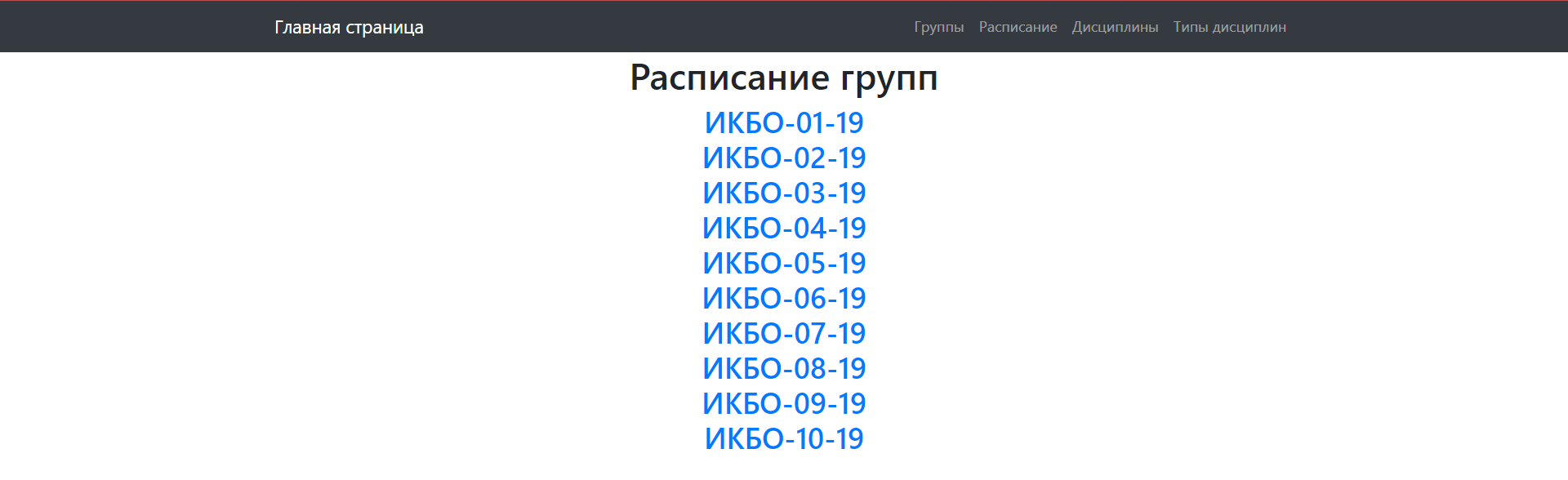


Рис. 8. Cтраница homepage.

Попробуем перейти по ссылке одной из групп:

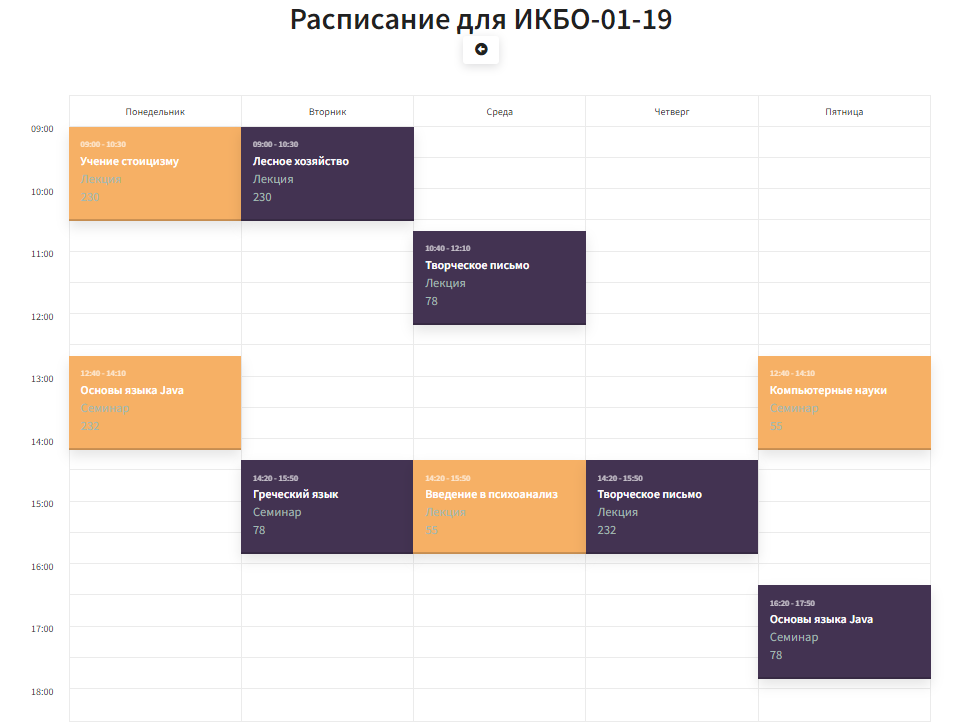


Рис. 9. Отображение страницы с расписанием группы ИКБО-01-19

Рассмотрим расписание, например, для группы ИКБО-10-19:

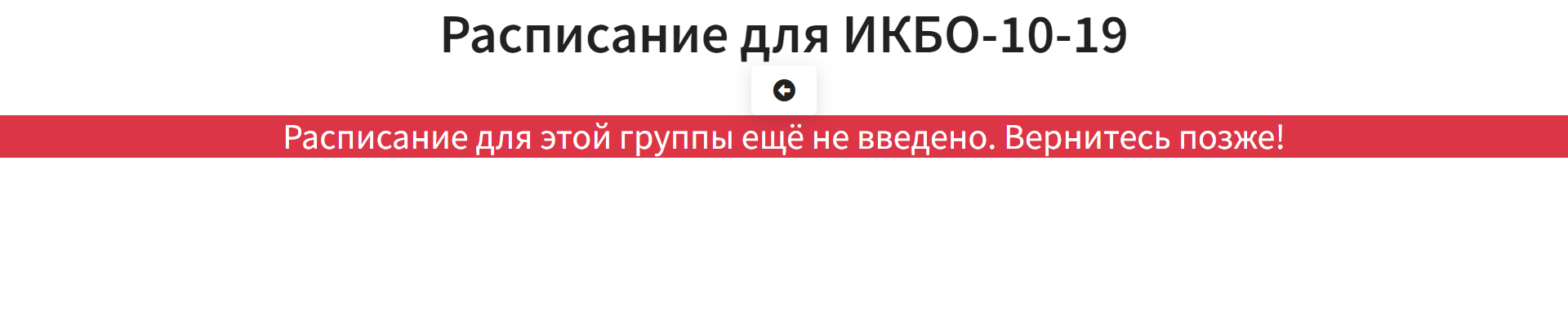


Рис. 10. Расписание для группы ИКБО-10-19

Как видим нам пишет, что расписания для этой группы ещё не ввели. Позже мы введем расписание для этой группы и заново перейдем к нему.

Посмотрим пользовательский интерфейс для работы с CRUD операциями различных сущностей:

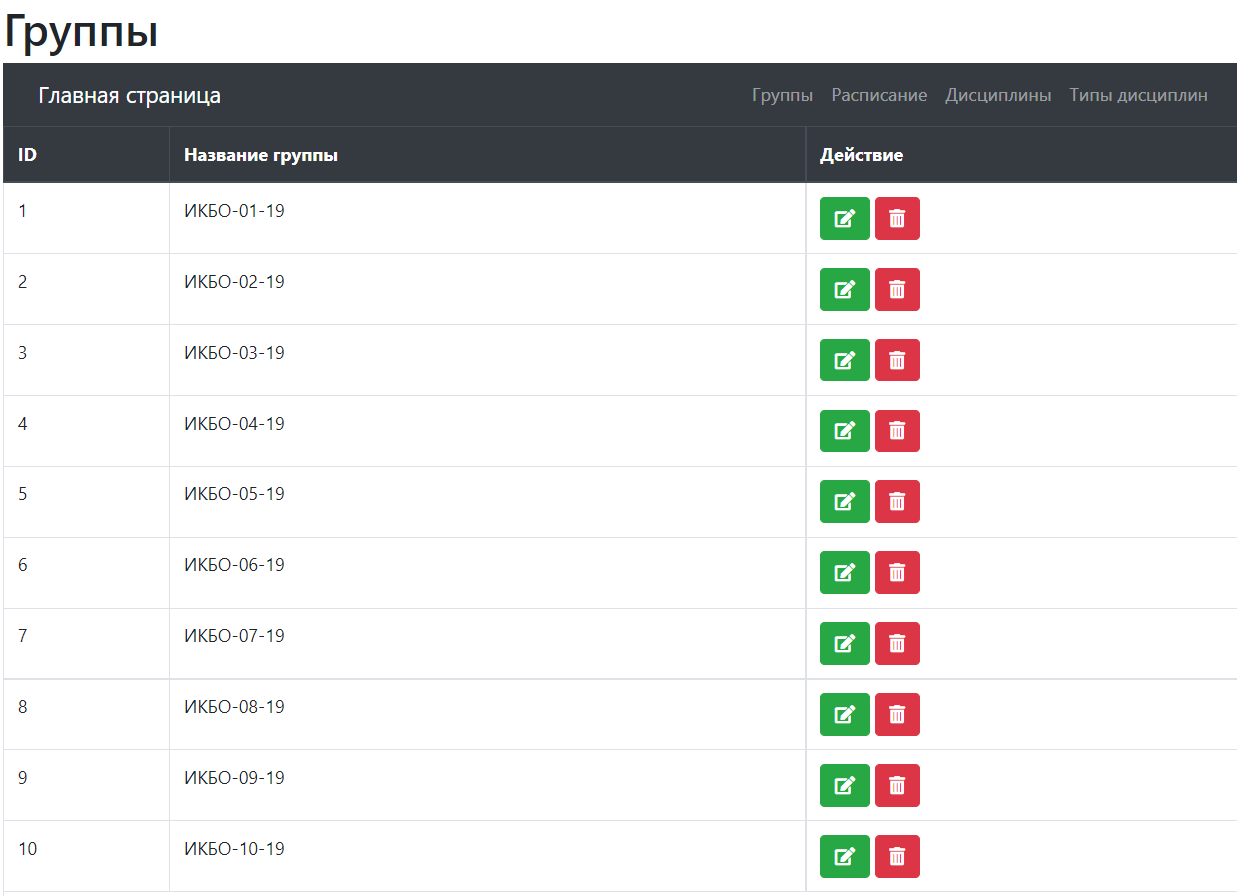


Рис. 11. Cтраница для редактирования информации о группах

Посмотрим имеющиеся дисциплины:

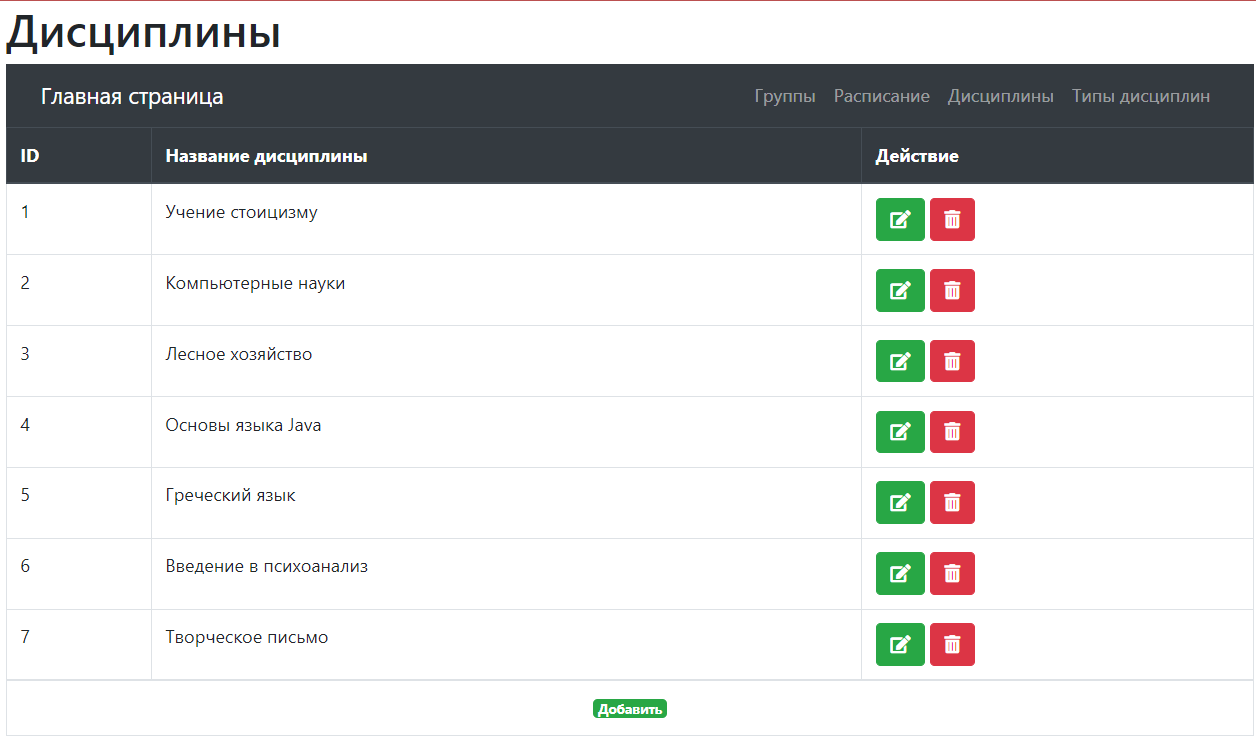


Рис. 12. Cтраница для редактирования информации о дисциплинах

Попробуем добавить новую дисциплину:

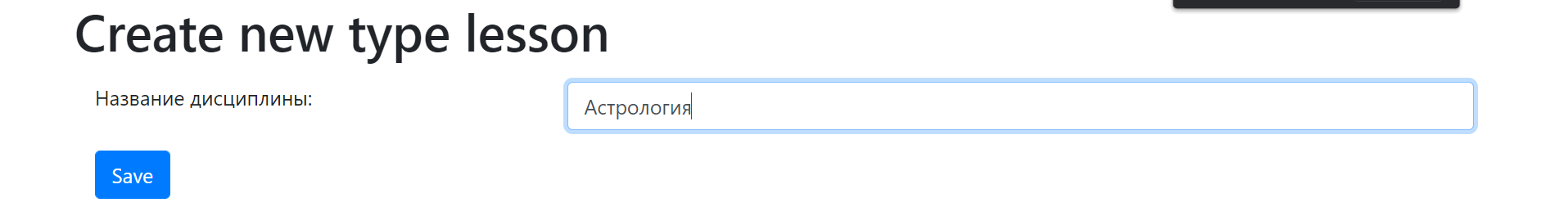


Рис. 13. Форма для ввода названия новой дисциплины

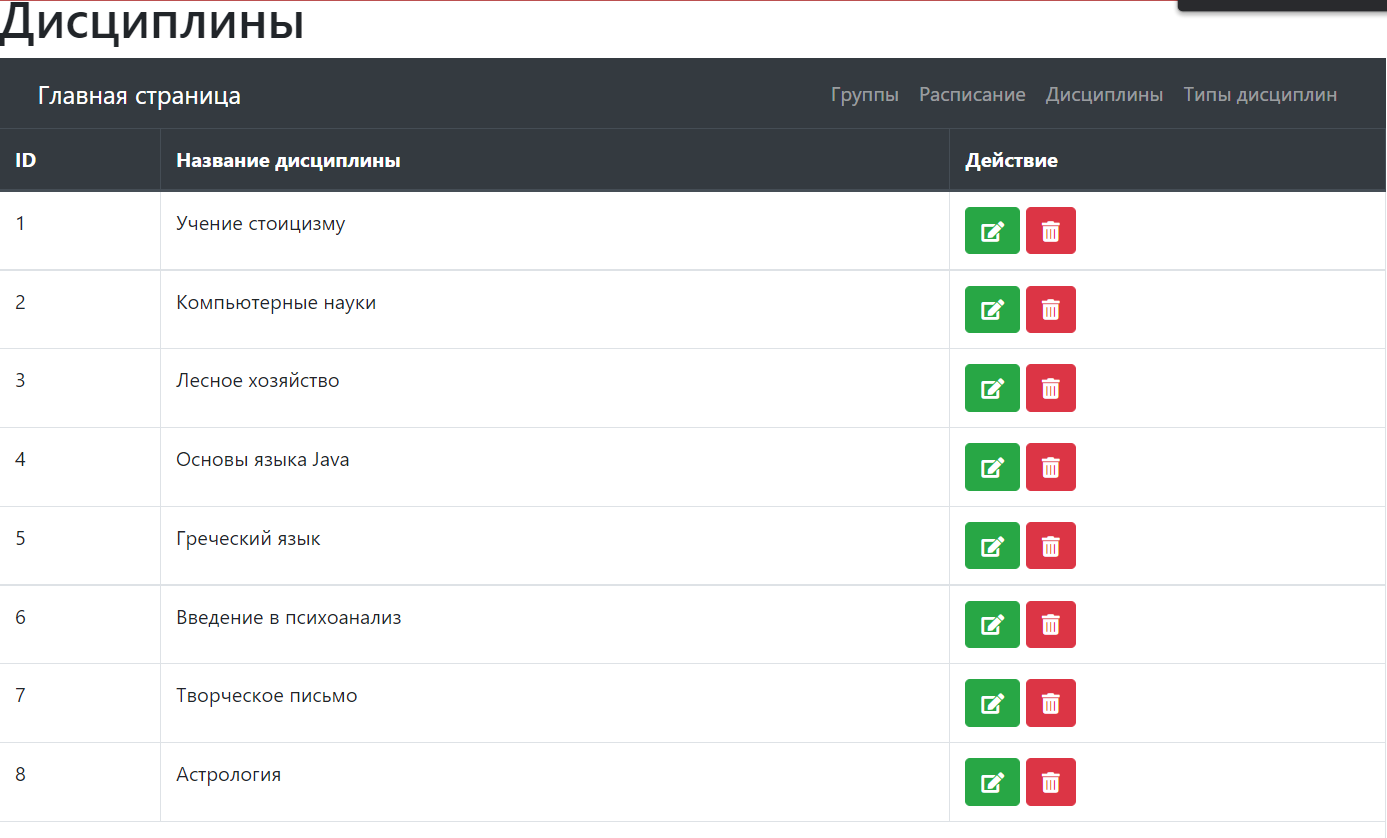


Рис. 14. Успешное введение новой дисциплины

Посмотрим вкладку с самим расписанием групп:

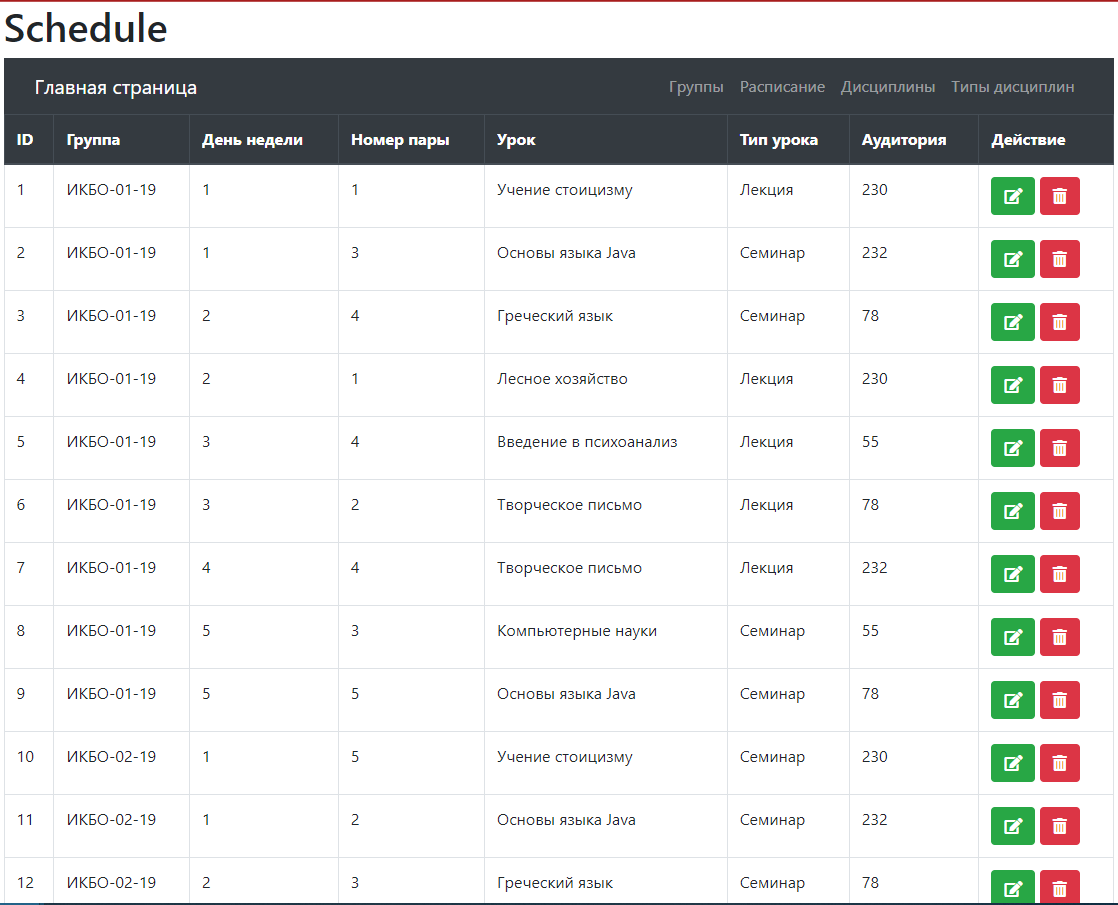
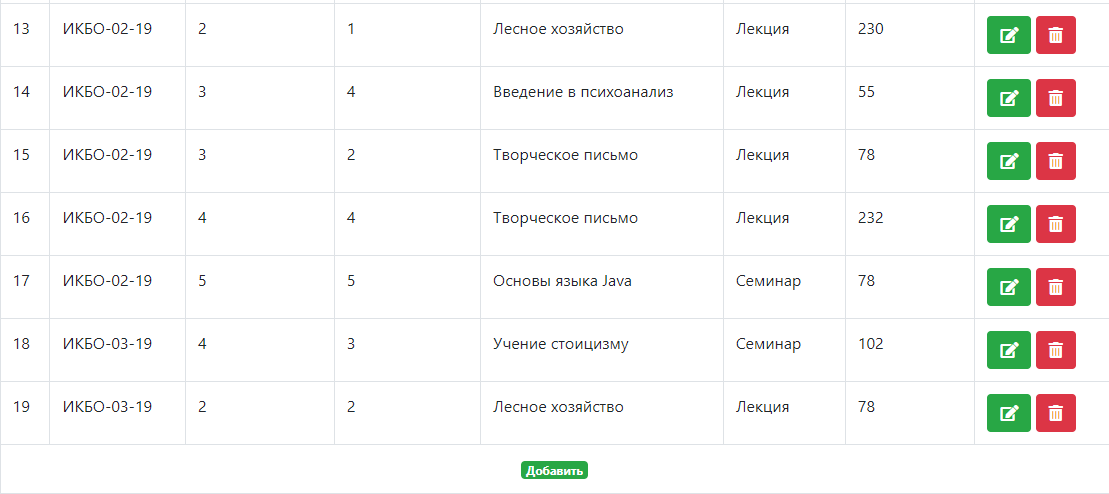


Рис. 15. Вкладка с расписанием для всех групп.

Как видим у нас пока что имеется расписание для групп: ИКБО-01-19, ИКБО-02-19 и неполное для ИКБО-03-19.

Попробуем добавить новое расписание для группы ИКБО-10-19:





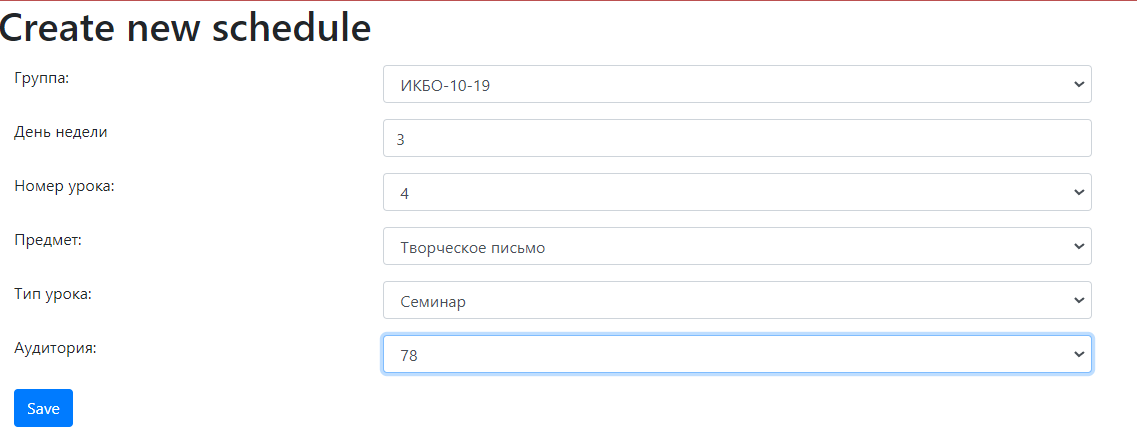


Рис. 16. Добавление нового расписания для ИКБО-10-19

Проверим отобразится ли теперь расписание для ИКБО-10-19:



Рис. 17. Расписание для ИКБО-10-19

Как видим добавление произошло успешно.

## Выводы к разделу 4

В ходе разработки был сделан такой функционал:

1. Главная страница, где пользователь может выбрать группу и посмотреть её расписание
2. Если расписание имеется в системе, то оно будет показано, иначе напишет ошибку о том, что расписания ещё не введено.
3. Были сделаны пользовательские интерфейсы для редактирования, создания и удаления записей в выбранной таблице.
4. Также же был обеспечен корректный показ в отображении новых введенных/ редактированных данных.

# Заключение

В процессе выполнения данной курсовой работы была проведена разработка серверной части веб-приложения «расписание». Спроектированное приложение дает пользователям видеть и пользоваться сайтом с минималистичным дизайном и минимумом сложных конструкций, для пользователей, которые захотят добавить/ редактировать/ удалить данные – возможность удобно редактировать содержание дисциплины, расписания, групп и тд

Был проведён анализ предметной области с выявлением базовых функций для подобного класса систем.

В ходе разработки в качестве архитектуры приложения был выбран шаблон проектирования MVC, который удачно помог создать удобную структуру без сложностей для её понимания

В процессе разработки курсовой работы были приобретены следующие

компетенции:

ПК-20 – способностью оценивать временную и емкостную сложность программного обеспечения.

ОПК-3 – готовностью применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов.

# Приложение

**Ссылка на github:** https://github.com/anvarzhonov/schedule-Spring-Boot

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Куликов С. С. Тестирование программного обеспечения. — 2017. — 356 [Дата обращения: 25.11.2021];
2. Официальный веб-сайт Spring [Электронный ресурс]: Режим доступа: https://spring.io/ [Дата обращения: 25.11.2021];
3. Веб-сайт Spring Security Reference [Электронный ресурс]: Режим доступа: https://docs.spring.io/spring security/site/docs/current/reference/html5/ [Дата обращения: 25.11.2021];
4. Видео-курс Spring Security | FULL COURSE [Электронный ресурс]: Режим доступа: https://www.youtube.com/watch?v=her\_7pa0vrg/ [Дата обращения: 25.11.2021];
5. Гэвин Кинг, Java Persistence with Hibernate [Электронный ресурс]: учебное пособие / Г. Кинг. — Электрон. дан. — М.: , 2016. — 248 с. — Режим доступа: https://www.manning.com/books/java-persistence-with-hibernate. — Загл. с экрана [Дата обращения: 25.11.2021];
6. Официальный сайт MySQL [Электронный ресурс]: Режим доступа: https://www.mysql.com/ [Дата обращения: 25.11.2021];
7. Крейг Уоллс, Spring в действии [Электронный ресурс]: учебное пособие— Электрон. дан.— 208 с. — Режим доступа: https://ru.pdfdrive.com/spring-%D0%B2-%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B8-e188685085.html— Загл. с экрана [Дата обращения: 25.11.2021];
8. Тузовский, А. Ф. Проектирование и разработка web-приложений : учебное пособие для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2021. — 218 с. — (Высшее образование). — ISBN 978-5-534-00515-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469982 [Дата обращения: 25.11.2021];
9. Хоффман Эндрю Х85 Безопасность веб-приложений. — СПб.: Питер, 2021. — 336 с.: ил. — (Серия «Бестселлеры O’Reilly») [Дата обращения: 25.11.2021];
10. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб. : Питер, 2021. — 352 c. [Дата обращения: 25.11.2021];
11. Персиваль, Г. Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура / Г. Персиваль, Б. Грегори. СПб: Питер, 2022. — 336 c [Дата обращения: 25.11.2021];
12. Раджпут Д. Spring. Все паттерны проектирования. - СПб.: Питер, 2019. [Дата обращения: 25.11.2021];
13. Меджуи М., Уайлд Э., Митра Р., Амундсен М. Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. - СПб.: Питер, 2020. [Дата обращения: 25.11.2021];
14. Никсон Р. Создаем динамические веб-сайты с помощью MySQL, JavaScript, CSS и HTML5. 5-е изд.. - СПб.: Питер, 2021. [Дата обращения: 25.11.2021];
15. Бэнкс А., Порселло Е. GraphQL: язык запросов для современных веб-приложений. - СПб.: Питер, 2019. [Дата обращения: 25.11.2021];
16. Антонова И. И., Кашкин Е. В. Разработка web-сервисов с использованием HTML, CSS, PHP и MySQL [Электронный ресурс]:учебно -методическое пособие. - М.: РТУ МИРЭА, 2019. - – Режим доступа: <http://library.mirea.ru/secret/15052019/2022.iso> [Дата обращения: 25.11.2021];
17. Диков А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 [Электронный ресурс]:учебное пособие. - Санкт-Петербург: Лань, 2019. - 188 с. – Режим доступа: <https://e.lanbook.com/book/122174> [Дата обращения: 25.11.2021];