

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение высшего
образования «Российский экономический университет имени Г.В. Плеханова»**

Московский приборостроительный техникум

ОТЧЕТ

по учебной практике

УП.04.01 Внедрение и поддержка программного обеспечения
_____.

Профессионального модуля ПМ.04 Сопровождение и обслуживание
программного обеспечения компьютерных систем _____.

Специальность 09.02.07 Информационные системы и программирование
_____.

Студент Патрикеев Глеб Викторович.
(фамилия, имя, отчество)

Группа П50-3-20

Руководитель по практической подготовке от техникума

Серяк Даниил Владимирович.
(фамилия, имя, отчество)

«__» _____ 2023 года

Содержание

Практическая работа №1	3
Практическая работа №2	10
Практическая работа №3	19
Практическая работа №4	28
Практическая работа №5	35
Практическая работа №6	42
Итоговый проект	46
Тема: магазин продажи фильмов «Кинотеатр»	46

Практическая работа №1

Цель работы: разработать приложение с использованием Spring Framework, включающее 5 страниц: основную страницу, предоставляющую доступ к страницам «Калькулятор» и «Конвертер валют», а также страницы, отображающие результаты вычислений.

Для начала мы создаем контроллер, который будет обрабатывать запросы на главную страницу через HTTP метод GET и адрес корневой URL. В начале каждого контроллера мы применяем аннотацию `@Controller`, чтобы указать, что данный класс является контроллером.

Затем мы создаем метод с именем `home()` и помечаем его аннотацией `@GetMapping("/")`, что позволяет методу обрабатывать GET-запросы на корневой адрес.

Внутри этого метода мы определяем, какое представление должно быть возвращено для отображения на главной странице.

Аналогично, создаем методы `calculator` и `currencyConverter`.

```

package com.example.calculator_spring.controller;

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

@Controller
public class MainController {
    public MainController() {
    }

    @GetMapping("/")
    public String home() {
        return "main";
    }

    @GetMapping("/calculator-form")
    public String calculator() {
        return "calc";
    }

    @GetMapping("/currency-converter-form")
    public String currencyConverter() {
        return "redirect:/convert";
    }
}

```

Рисунок 1. Контроллер главной страницы.

Контроллер калькулятора начинает свою работу с теми же методами и аннотациями, но добавляет новый метод - calculate().

Аннотация @PostMapping("/Result") указывает, что этот метод будет обрабатывать POST-запросы и определяет путь, по которому результат этого метода будет доступен.

Аннотация @RequestParam говорит о том, что метод ожидает параметры с именем «num1», «num2», «operator», из запроса, которое соответствует имени элемента в разметке страницы.

Объект Model представляет собой модель Spring, используемую для передачи данных между контроллером и представлением. Таким образом, значение, хранящееся в переменной «result», будет передано на представление под именем «result». Ниже представлена демонстрация того, как это значение будет использоваться.

```

package com.example.calculator_spring.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class CalcController {
    public CalcController() {
    }

    @GetMapping({"/calc"/})
    public String calculatorForm() { return "calc"; }

    @PostMapping({"/calculate"/})
    public String calculate(@RequestParam("num1") double num1, @RequestParam("num2") double num2, @RequestParam("operator") String operator, Model model) {
        double result = this.performCalculation(num1, num2, operator);
        model.addAttribute("result", result);
        return "result";
    }
}

```

Рисунок 2. Контроллер калькулятора. (1)

```

1 usage
private double performCalculation(double num1, double num2, String operator) {
    double result = 0.0;
    switch (operator) {
        case "add":
            result = num1 + num2;
            break;
        case "subtract":
            result = num1 - num2;
            break;
        case "multiply":
            result = num1 * num2;
            break;
        case "divide":
            if (num2 != 0.0) {
                result = num1 / num2;
            }
            break;
        case "procent":
            result = num1 * num2 / 100.0;
    }

    return result;
}
}

```

Рисунок 3. Контроллер калькулятора. (2)

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Result calculator Page</title>
</head>
<body style="text-align: center; background-color: bisque">
<center>
  <h1>Result calculator</h1>
  <p style="color: red; font-size: 35px;">
    <span th:if="{result == 0.0}">Can't divide by zero</span>
  </p>
  <p style="color: #000000; font-size: 24px;">
    Result: <span th:text="{result}"></span>
  </p>

  <form method="get" action="/">
    <input type="submit" formmethod="get" value="Main" style="width: 300px; height: 50px; padding: 10px 10px; font-size: 17px; color: #000000; background: #d050ff">
  </form>
</center>
</body>
</html>

```

Рисунок 4. Использование result.

В контроллере конвертера валют мы задаем аналогичную логику, но расширяем список параметров, которые принимаем для обработки.

```

@Controller
public class ConvertController {
    public ConvertController(){}
}

@GetMapping("/{id}/convert")
public String currencyConverterPage() {
    return "convert";
}

@PostMapping("/{id}/convert")
public String convertCurrency(@RequestParam("fromCurrency") String fromCurrency, @RequestParam("toCurrency") String toCurrency, @RequestParam("amount") String amountStr, Model m) {
    if (amountStr.isEmpty()) {
        model.addAttribute("error", "Type Quantity.");
        return "convert_result";
    } else {
        double amount = Double.parseDouble(amountStr);
        double convertedAmount;
        if (fromCurrency.equals(toCurrency)) {
            convertedAmount = amount;
        } else if ("USD".equals(fromCurrency) && "RUB".equals(toCurrency)) {
            convertedAmount = amount * 89.0;
        } else if ("USD".equals(fromCurrency) && "EUR".equals(toCurrency)) {
            convertedAmount = amount * 0.85;
        } else if ("EUR".equals(fromCurrency) && "RUB".equals(toCurrency)) {
            convertedAmount = amount * 105.0;
        } else if ("EUR".equals(fromCurrency) && "USD".equals(toCurrency)) {
            convertedAmount = amount * 1.18;
        } else if ("RUB".equals(fromCurrency) && "USD".equals(toCurrency)) {
            convertedAmount = amount * 0.0112;
        }
    }
}

```

Рисунок 4. Контроллер конвертера валют.

Результат работы:

Main Page

Calculator

Converter

Calculator Page

Number 1: 5

+ ▾

Number 2: 5

Calculate

Result calculator

Result: 10.0

Main

Currency Converter

From currency: Доллар (USD) ▾

To currency: Рубль (RUB) ▾

Quantity: 15

Convert

Result Conversions

Result: 1335.0

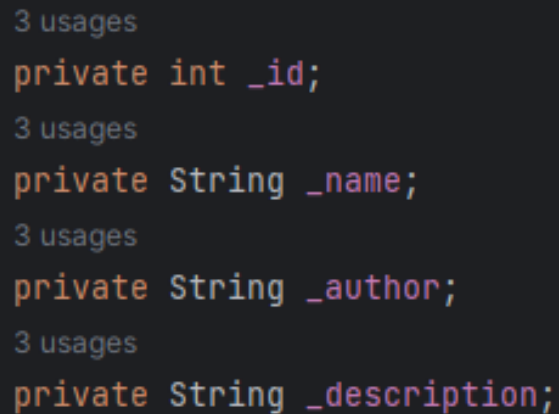
Main

Вывод: в результате данной работы было разработано приложение с использованием Spring Framework, включающее 5 страниц: основную страницу, предоставляющую доступ к страницам «Калькулятор» и «Конвертер валют», а также страницы, отображающие результаты вычислений.

Практическая работа №2

Цель работы: разработать приложение с помощью Spring Framework, в котором будет реализован паттерн DAO.

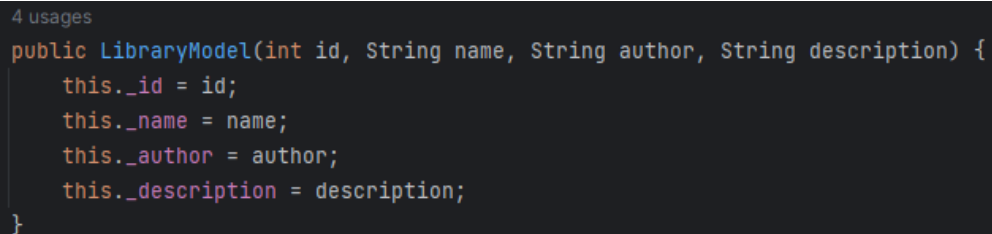
Для начала создаем модели с 4 полями.



```
3 usages
private int _id;
3 usages
private String _name;
3 usages
private String _author;
3 usages
private String _description;
```

Рисунок 5. 4 поля.

Создаем конструкторы.



```
4 usages
public LibraryModel(int id, String name, String author, String description) {
    this._id = id;
    this._name = name;
    this._author = author;
    this._description = description;
}
```

Рисунок 6. Конструктор.

Затем создаем геттеры и сеттеры, благодаря которым мы сможем получать и записывать данные модели.

```

public void setId(int id) { this._id = id; }

public void setName(String name) { this._name = name; }

1 usage
public void setAuthor(String author) { this._author = author; }

1 usage
public void setDescription(String description) { this._description = description; }

public int getId() { return _id; }

public String getName() { return _name; }

1 usage
public String getAuthor() { return _author; }

1 usage
public String getDescription() { return _description; }

```

Рисунок 7. Конструктор.

По такой же схеме создаем другие модели.

Затем мы определяем класс для работы с паттерном DAO, который обеспечивает функции создания, чтения, обновления и удаления данных.

В начале класса мы применяем аннотацию `@Component`, что сообщает Spring Framework о том, что класс `bookDAO` должен быть управляемым компонентом (bean) и подвергается управлению контейнером Spring. Это означает, что Spring будет создавать экземпляры этого класса и управлять их жизненным циклом.

Далее, мы создаем статическое поле для отслеживания количества элементов в списке. Список `List<LibraryModel>` используется для хранения объектов `LibraryModel`, и мы добавляем в него несколько данных.

```

9 usages
private List<LibraryModel> books;

{
    books = new ArrayList<>();

    books.add(new LibraryModel(++BOOK_COUNT, name: "Война и Мир", author: "Лев Николаевич Толстой", description: "роман-эпопея Льва Николаевича Толстого, описывающий русское общество в 1812 году");
    books.add(new LibraryModel(++BOOK_COUNT, name: "Мертвые души", author: "Николай Васильевич Гоголь", description: "Писать книгу Гоголь начал в 1835 году как трехтомник. Первый том закончил в 1842 году, но так и не закончил");
    books.add(new LibraryModel(++BOOK_COUNT, name: "К черту все! Берись и делай!", author: "Ричард Бренсон", description: "автобиография успешного человека который начиная с нуля создал империю");
    books.add(new LibraryModel(++BOOK_COUNT, name: "Саентология: Основы Жизни", author: "Рон Хаббард", description: "Описание состояний существования, восьми динамик, треугольника");
}

```

Рисунок 8. Список.

Метод `indexBook()` возвращает список всех книг, `showBook(int id)` возвращает книгу с указанным идентификатором, `saveBook(LibraryModel book)` добавляет объект в список.

```
1 usage
public List<LibraryModel> indexBook() { return books; }

3 usages
public LibraryModel showBook(int id) {
    return books.stream().filter(book -> book.getId() == id).findAny().orElse( other: null);
}

1 usage
public void saveBook(LibraryModel book) {
    book.setId(++BOOK_COUNT);
    books.add(book);
}

1 usage
public void updateBook(int id, LibraryModel libraryModel) {
    LibraryModel updateBook = showBook(id);
    updateBook.setName(libraryModel.getName());
    updateBook.setAuthor(libraryModel.getAuthor());
    updateBook.setDescription(libraryModel.getDescription());
}

1 usage
public void deleteBook(int id) { books.removeIf(book -> book.getId() == id); }
```

Рисунок 9. Методы.

По той же логике прописываем класс DAO для всех моделей.

Аннотация `@RequestMapping("/books")` устанавливает базовый путь, который будет применяться ко всем методам внутри данного контроллера.

Аннотация `@Autowired` указывает на то, что зависимость `bookDAO` будет автоматически внедрена при создании контроллера.

В методе `index` мы извлекаем список всех элементов и помещаем его в модель, которая затем будет использоваться для представления этого списка пользователю.

```

@Controller
@RequestMapping("/books")
public class LibraryController {

    7 usages
    private final LibraryDAO libraryDAO;

    @Autowired
    public LibraryController(LibraryDAO bookDAO) { this.libraryDAO = bookDAO; }

    @GetMapping()
    public String index(Model model) {
        model.addAttribute(attributeName: "books", libraryDAO.indexBook());
        return "Library/indexBook";
    }
}

```

Рисунок 9. Аннотации (1).

Метод show позволяет извлекать информацию о конкретном элементе, используя аннотацию `@PathVariable`, и затем добавляет эту информацию в модель.

NewBook просто открывает страницу для создания нового элемента, в то время как метод `createBook(@ModelAttribute("book") LibraryModel book, Model model)` получает данные из формы и сохраняет их в список.

EditBook также предоставляет доступ к странице для редактирования записи, передавая данные об элементе с конкретным идентификатором на эту страницу.

Метод `updateBook(@ModelAttribute("book") LibraryModel book, @PathVariable("id") int id)` обрабатывает PATCH-запросы, получает данные из формы и обновляет запись с указанным идентификатором.

`deleteBook(@PathVariable("id") int id)` удаляет запись с указанным идентификатором.

```

@GetMapping("/{id}")
public String show(@PathVariable("id") int id, Model model) {
    model.addAttribute("book", libraryDAO.showBook(id));
    return "Library/showBook";
}

@GetMapping("/{id}/new")
public String newBook(@ModelAttribute("book") LibraryModel book) { return "book/newBook"; }

@PostMapping()
public String createBook(@ModelAttribute("book") LibraryModel book) {
    libraryDAO.saveBook(book);
    return "redirect:/books";
}

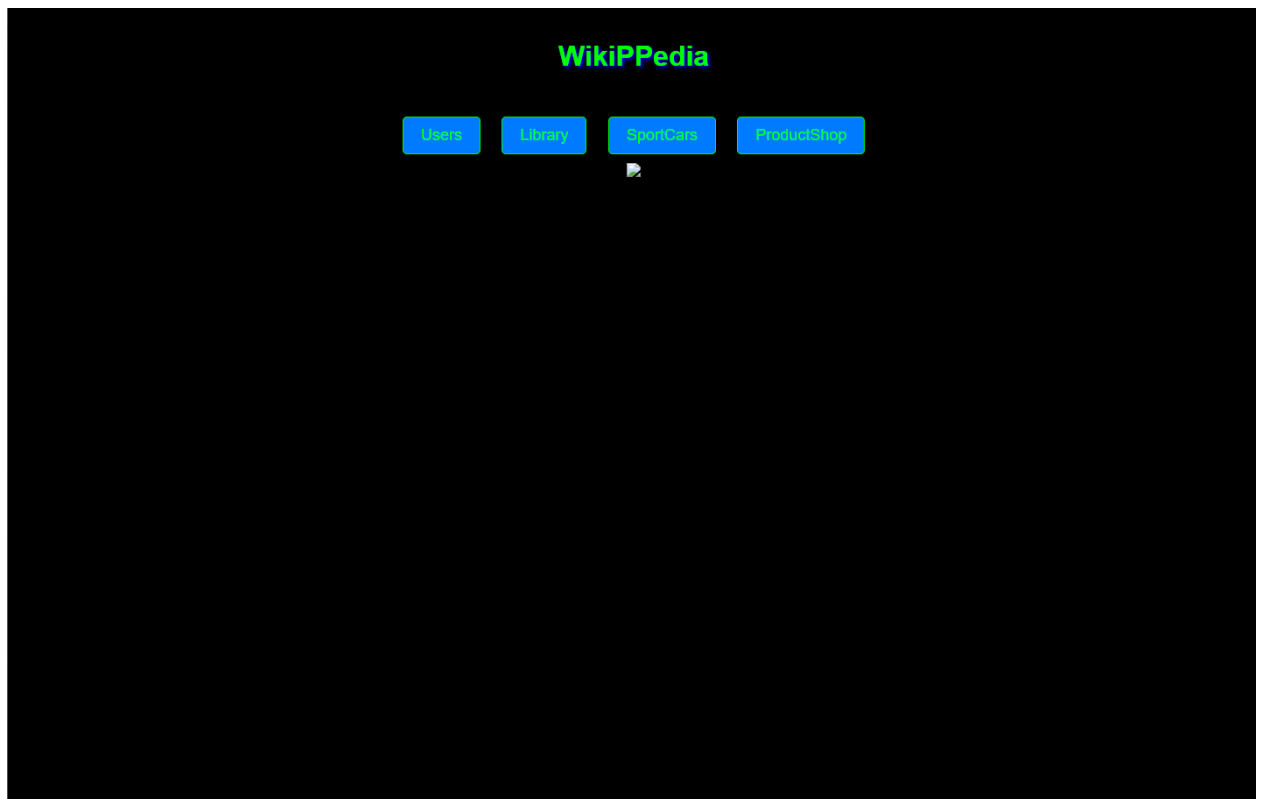
@GetMapping("/{id}/edit")
public String editBook(Model model, @PathVariable("id") int id) {
    model.addAttribute("book", libraryDAO.showBook(id));
    return "Library/editBook";
}

@PatchMapping("/{id}")
public String updateBook(@ModelAttribute("book") LibraryModel book, @PathVariable("id") int id) {
    libraryDAO.updateBook(id, book);
    return "redirect:/books";
}

```

Рисунок 10. Аннотации (2).

Результат работы:



Users

Номер	Имя	Фамилия	Отчество	Действие
1	Иван	Лунин	Игорович	Show Edit Delete
2	Алексей	Авдотьев	Михайлович	Show Edit Delete
3	Никита	Говрюшин	Иванович	Show Edit Delete
4	Глеб	Патрикеев	Викторович	Show Edit Delete

Create User

Main Page

Library

Номер	Название	Автор	Описание	Действие
1	Война и Мир	Лев Николаевич Толстой	роман-эпопея Льва Николаевича Толстого, описывающий русское общество в эпоху войн против Наполеона в 1805 —1812 годах.	Show Edit Delete
2	Мертвые души	Николай Васильевич Гоголь	Писать книгу Гоголь начал в 1835 году как трехтомник. Первый том был издан в 1842 году. Практически готовый второй том был утерян	Show Edit Delete
3	К черту все! Берись и делай!	Ричард Бренсон	автобиография успешного человека который начиная с нуля достиг самого верха	Show Edit Delete
4	Саентология: Основы Жизни	Рон Хабард	Описание состояний существования, восьми динамик, треугольника АРО, составляющих человека, полный анализ жизни как игры	Show Edit Delete

Create book

Main page

Car

Номер	Марка	Модель	Действие	
1	Lamborghini	Urus	Show Edit	Delete
2	Bugatti	Veyron	Show Edit	Delete
3	Porsche	Cayne	Show Edit	Delete
4	Alfa romeo	Giulia	Show Edit	Delete

Create car

Main page

Product List

Номер	Название	Цена	Вес	Действие	
1	Пипидастр	10.99	150.02	Show Edit	Delete
2	Банка стеклянная	19.99	240.42	Show Edit	Delete
3	Лобзик	5.99	1010.54	Show Edit	Delete
4	Станеска	7.99	1080.42	Show Edit	Delete

Create product

Main page

Edit User

Имя:

Иван

Фамилия:

Лукин

Отчество:

Игоревич

Edit

Back

Create User

Имя:

Фамилия:

Отчество:

Create

Back

Edit book

Название:

Война и Мир

Автор:

Лев Николаевич Толстой

Описание:

роман-эпопея Льва Николаевича Толстого, описывающий русское общество в эпоху войн против Наполеона в 1805—1812 годах.

Edit

Back

Library

Номер	Название	Автор	Описание
1	Война и Мир	Лев Николаевич Толстой	роман-эпопея Льва Николаевича Толстого, описывающий русское общество в эпоху войн против Наполеона в 1805—1812 годах.

Back

Show Current User			
Номер	Имя	Фамилия	Отчество
1	Иван	Лукин	Игоревич
<div>Back</div>			

И т.д.

Вывод: в результате данной работы было разработано приложение с помощью Spring Framework, в котором реализован паттерн DAO.

Практическая работа №3

Цель: переделать предыдущую работу с подключением выбранной СУБД, сделать валидацию на каждое поле с помощью аннотаций, добавить поиск определенной записи.

Ход работы:

В первую очередь подключаем зависимость для работы с postgres.

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

Рисунок 11. Зависимость

Подключаем нашу базу данных:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/pract3
spring.datasource.username=postgres
spring.datasource.password=1234
spring.datasource.driverClassName=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto=update

server.port=8081
```

Рисунок 12. Строка подключения

Начнем с модели «SportCarsModel». Для начала добавим аннотации: @Entity, которая указывает, что данный класс является сущностью, и @Table, которая используется для указания имени таблицы.

Затем применим аннотации к полю id: @Id, чтобы отметить его как первичный ключ, и @GeneratedValue, чтобы определить метод генерации первичного ключа. В данном случае я выбрал AUTO, что позволяет автоматически выбрать способ генерации.

Кроме того, для каждого поля я добавлю аннотацию `@NotBlank`, чтобы гарантировать, что поле не будет пустым. Эти шаги будут выполнены для всех моделей.

```
@Entity
@Table(name = "car")
public class SportCarsModel {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int _id;
    3 usages
    @NotBlank(message = "Mark is required")
    private String _mark;
    3 usages
    @NotBlank(message = "Model is required")
    private String _model;
    3 usages
    @NotBlank(message = "Type is required")
    private String _type;

    public SportCarsModel(){}

    no usages
    public SportCarsModel(int id, String mark, String model, String type) {
        this._id = id;
        this._mark = mark;
        this._model = model;
        this._type = type;
    }
}
```

Рисунок 13. SportCarsModel

Для каждой модели будет создан репозиторий, который обеспечит взаимодействие с базой данных и выполнение операций CRUD с сущностями, без необходимости вручную писать SQL-запросы.

Этот репозиторий предназначен для работы с сущностью `SportCarsModel`. В качестве параметра он принимает саму сущность `SportCarsModel`, а тип данных `Integer` используется для идентификатора этой сущности.

```
package com.example.databasepract.repositories;

import com.example.databasepract.models.SportCarsModel;
import org.springframework.data.jpa.repository.JpaRepository;

3 usages
public interface CarRepositories extends JpaRepository<SportCarsModel, Integer> {
}
```

Рисунок 4. SportCarsRepositories

Внедряем зависимость CarRepositories через конструктор контроллера. Это дает нам возможность использовать репозиторий для доступа к данным о пользователях.

Метод listCars извлекает всех пользователей из репозитория, добавляет их в модель и отображает список всех пользователей. Метод showAddUserForm используется для показа формы добавления нового пользователя.

Метод addCar принимает данные, введенные в форму добавления пользователя, проводит их валидацию с помощью аннотации @Valid. Если данные проходят валидацию, новый пользователь сохраняется в репозитории. Затем происходит перенаправление пользователя на страницу со списком пользователей.

Метод showEditCarForm осуществляет поиск пользователя по указанному id, добавляет его в модель и возвращает представление "user/edit-car". Этот метод используется для показа формы изменения данных пользователя.

Метод editCar выполняет изменение существующего пользователя. Метод deleteCar удаляет существующего пользователя.

```

@Controller
@RequestMapping("/cars")
public class SportCarsController {

    @Autowired
    private final CarRepositories carRepositories;

    @GetMapping("/list")
    public String listCars(Model model) {
        Iterable<SportCarsModel> cars = carRepositories.findAll();
        model.addAttribute("cars", cars);
        return "SportCars/carList";
    }

    @GetMapping("/add")
    public String showAddCarForm(Model model) {
        SportCarsModel car = new SportCarsModel();
        model.addAttribute("car", car);
        return "SportCars/addCar";
    }

    @PostMapping("/add")
    public String addCar(@Valid @ModelAttribute("car") SportCarsModel car, BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            return "SportCars/addCar";
        }
        System.out.println("Mark: " + car.getMark());
        carRepositories.save(car);
        return "redirect:/cars/list";
    }
}

```

Рисунок 5. SportCarsController (1)

```

@GetMapping("/edit/{id}")
public String showEditCarForm(@PathVariable("id") int id, Model model) {
    SportCarsModel car = carRepositories.findById(id).orElse(null);
    if (car == null) {
        return "redirect:/cars/list";
    }
    model.addAttribute("car", car);
    return "SportCars/editCar";
}

@PostMapping("/edit/{id}")
public String editCar(@PathVariable("id") int id, @Valid @ModelAttribute("car") SportCarsModel car, BindingResult bindingResult) {
    if (bindingResult.hasErrors()) {
        return "SportCars/editCar";
    }
    car.setId(id);
    carRepositories.save(car);
    return "redirect:/cars/list";
}

@GetMapping("/delete/{id}")
public String deleteCar(@PathVariable("id") int id) {
    carRepositories.deleteById(id);
    return "redirect:/cars/list";
}

```

Рисунок 6. SportCarsController (2)

Для осуществления поиска создается метод в контроллере, который принимает параметр «name» из запроса с использованием аннотации @RequestParam.

Этот параметр представляет собой строку, введенную пользователем в поле поиска. Затем метод использует репозиторий «ProductRepositories» для поиска продуктов, имена которых содержат указанную строку "name". Результаты поиска затем добавляются в модель под атрибутом «Product».

```
@GetMapping("/search")
public String searchProducts(@RequestParam(name = "name") String name, Model model) {
    Iterable<ProductModel> products = productRepositories.findByNameContainingIgnoreCase(name);
    model.addAttribute("products", products);
    return "ProductShop/productList";
}
```

Рисунок 7. searchProduct

```
public interface ProductRepositories extends JpaRepository<ProductModel, Integer> {
    1 usage
    java.lang.Iterable<ProductModel> findByNameContainingIgnoreCase(String name);
}
```

Рисунок 8. ProductRepositories

Результат:



Рисунок 9. Главная страница.

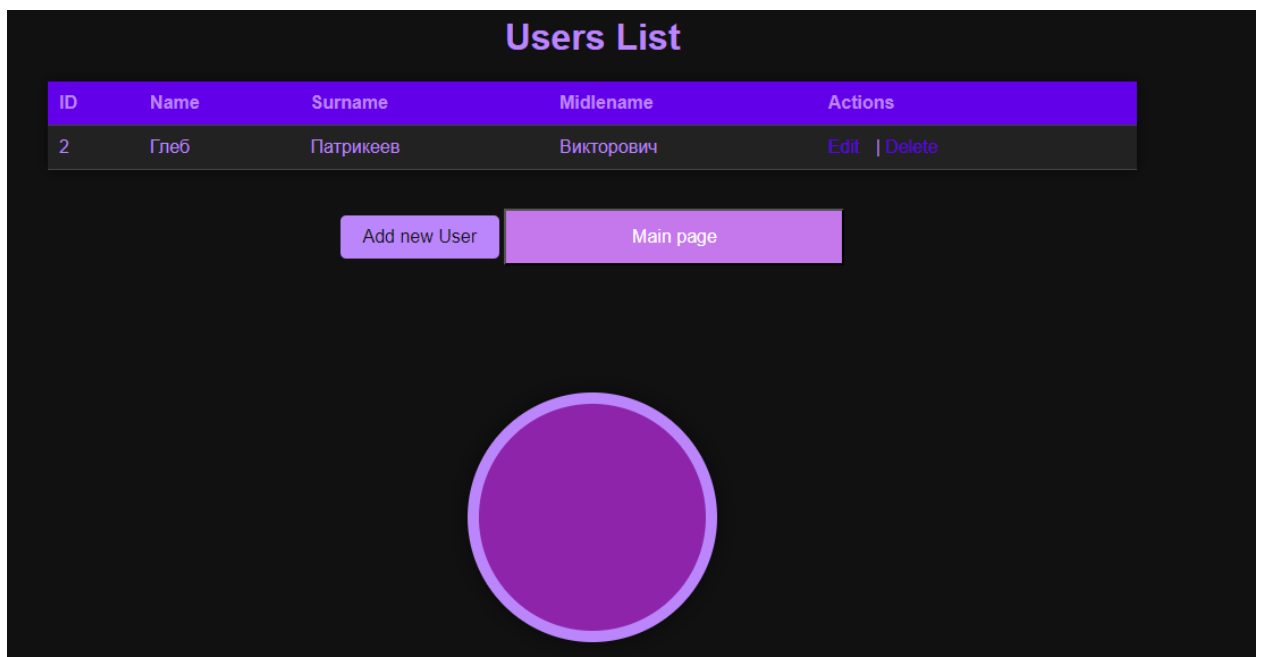


Рисунок 10. Пользователи.

Add User

Name:

АНТОН

Surname:

АНТОНКИН

Midlename:

АНТОНОВИЧ

Add User

Back

Рисунок 11. Создание пользователя.

Users List

ID	Name	Surname	Midlename	Actions
2	Глеб	Патрикеев	Викторович	Edit Delete
3	АНТОН	АНТОНКИН	АНТОНОВИЧ	Edit Delete

Add new User

Main page

Рисунок 12. Демонстрация создания.

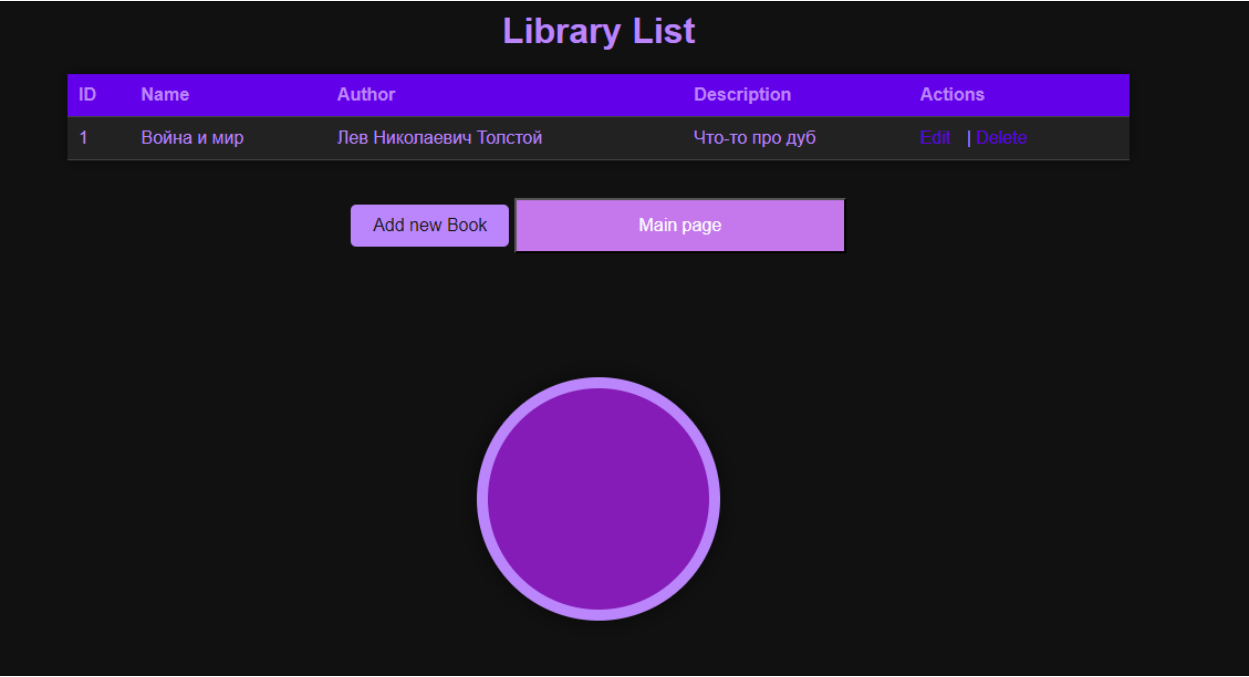


Рисунок 13. Просмотр книг

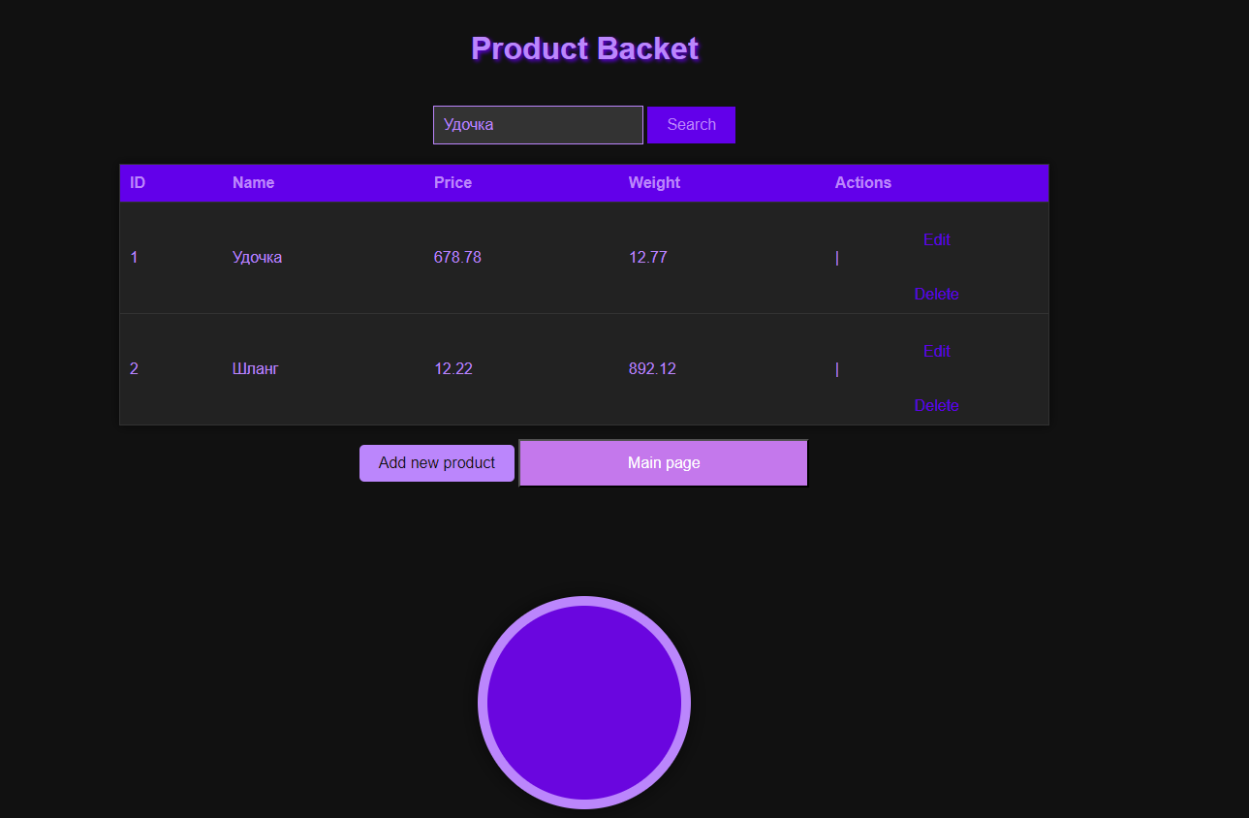


Рисунок 14. Поиск

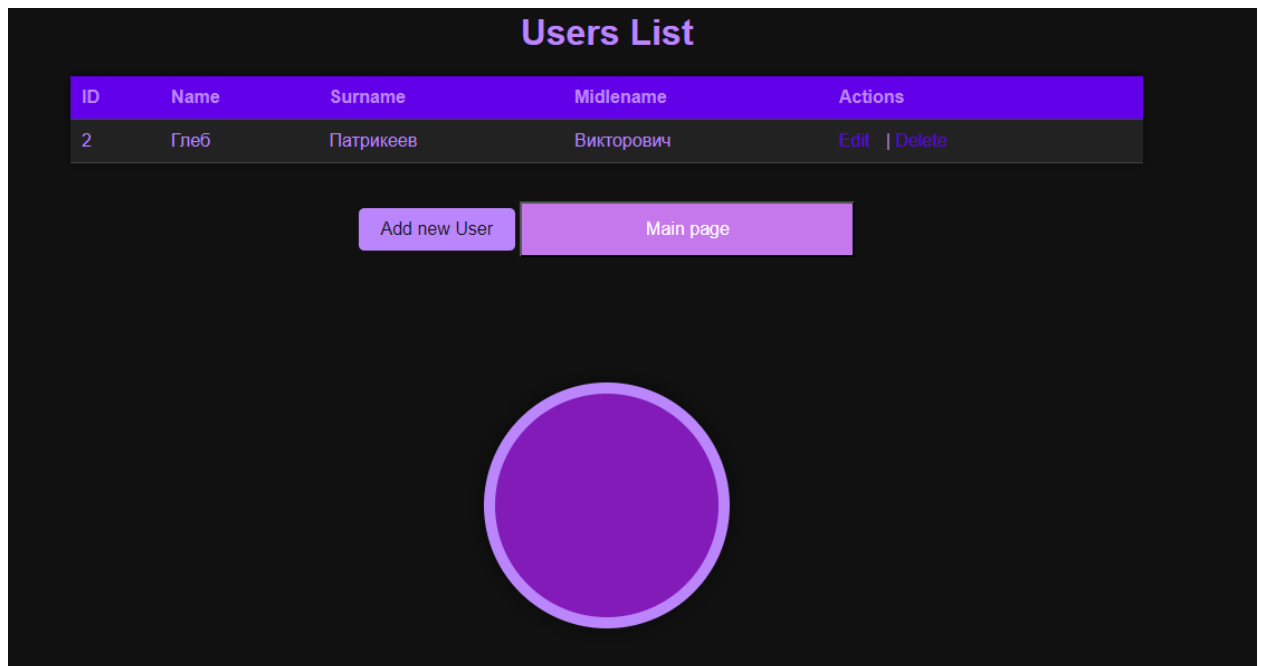


Рисунок 15. Удаление пользователя

Вывод: в процессе выполнения данной практической задачи, я внёс изменения в предыдущую работу, интегрировав выбранную систему управления базами данных (PostgreSQL). Также реализовал проверку валидности каждого поля с использованием аннотаций (`@NotBlank`) и добавил функционал поиска конкретной записи.

Практическая работа №4

Цель: реализовать 3 типа связей, описать их и продемонстрировать их работу, для этого следует создать дополнительные модели и связать с уже созданными таблицами или между собой.

Ход работы:

Прежде всего, мы устанавливаем связь OneToOne между моделями BookModel и ProductModel. Эта связь означает, что каждая книга имеет только один продукт.

Аннотация @OneToOne указывает на такую связь. Она говорит о том, что сущность имеет связь с другой сущностью таким образом, что каждая из них может быть связана только с одной записью из другой сущности. Аннотация mappedBy = "product" говорит о том, что управление этой связью осуществляется сущностью BookModel.

Также аннотация указывает название столбца в таблице ProductModel, которое будет использоваться для хранения внешнего ключа. Кроме того, аннотация @Table создаст таблицу в базе данных, соответствующую данной модели.

```
2 usages
@OneToOne(optional = true, mappedBy = "product")
private BookModel owner;
```

Рисунок 22. ProductModel.

```
2 usages
@OneToOne(optional = true, cascade = CascadeType.ALL)
@JoinColumn(name="product_id")
private ProductModel product;
```

Рисунок 23. BookModel.

Результат:

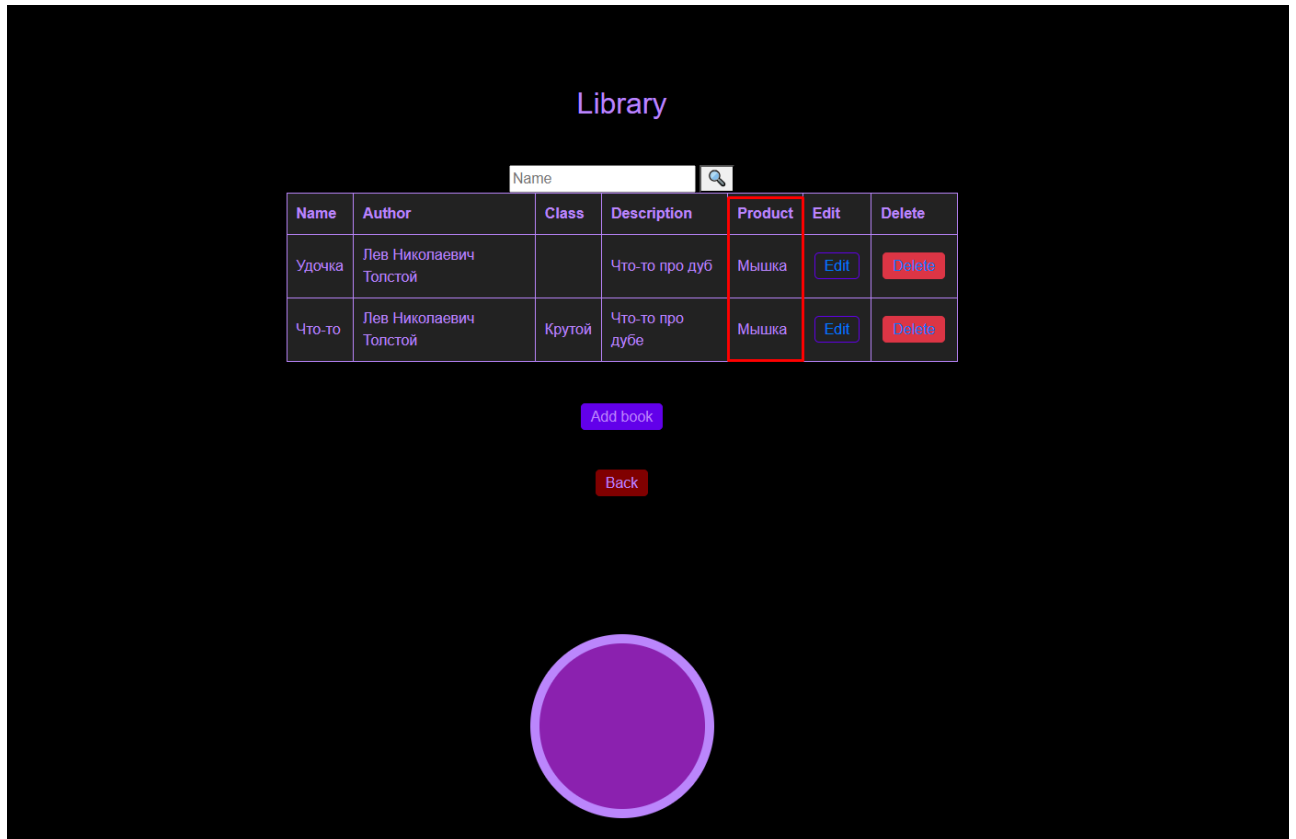


Рисунок 24. OneToOne.

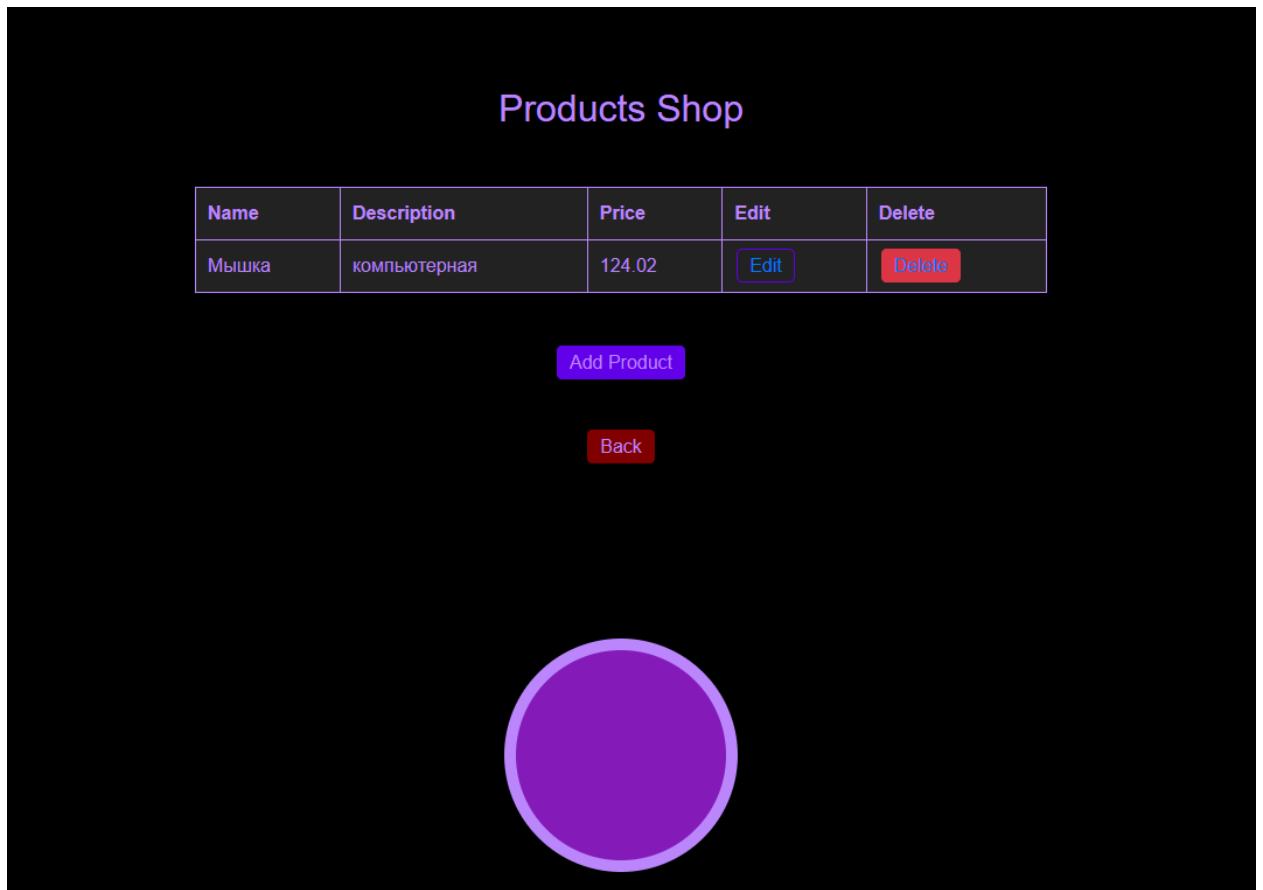
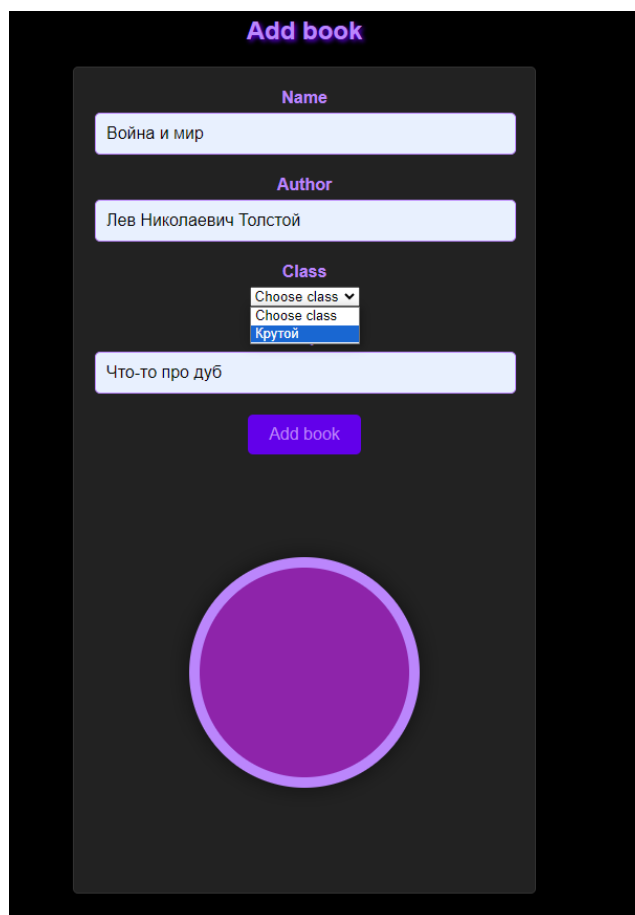


Рисунок 25. Таблица Продукт.



The image shows a web form titled "Add book" in red text. The form is set against a dark gray background. It contains four input fields: "Name" with the text "Война и мир", "Author" with "Лев Николаевич Толстой", "Class" which is a dropdown menu showing "Choose class" and "Крутой", and another text field with "Что-то про дуб". Below these fields is a red "Add book" button. At the bottom of the form is a large, light blue circle with a dark blue border.

Рисунок 26. Выбор класса.

Мы устанавливаем взаимосвязь типа "один ко многим", что означает, что одной книге может соответствовать несколько классов.

В основном, аннотации остаются неизменными, за исключением аннотации `OneToMany`, которая используется для создания связи "один ко многим". В данном случае она подразумевает, что каждой книге может соответствовать множество классов.

Аннотация `@ManyToOne` также применяется для установления связи "один ко многим", но в обратном направлении, указывая, что каждому классу соответствует одна книга.

```
@ManyToOne
@JoinColumn(name = "class_id") // Указывает на столбец, который связывает сущности
private ClassModel clas;
```

Рисунок 27. BookModel

```
3 usages
@OneToMany(mappedBy = "clas", fetch = FetchType.EAGER)
private Collection<BookModel> hero;
```

Рисунок 28. ClassModel

Результат:

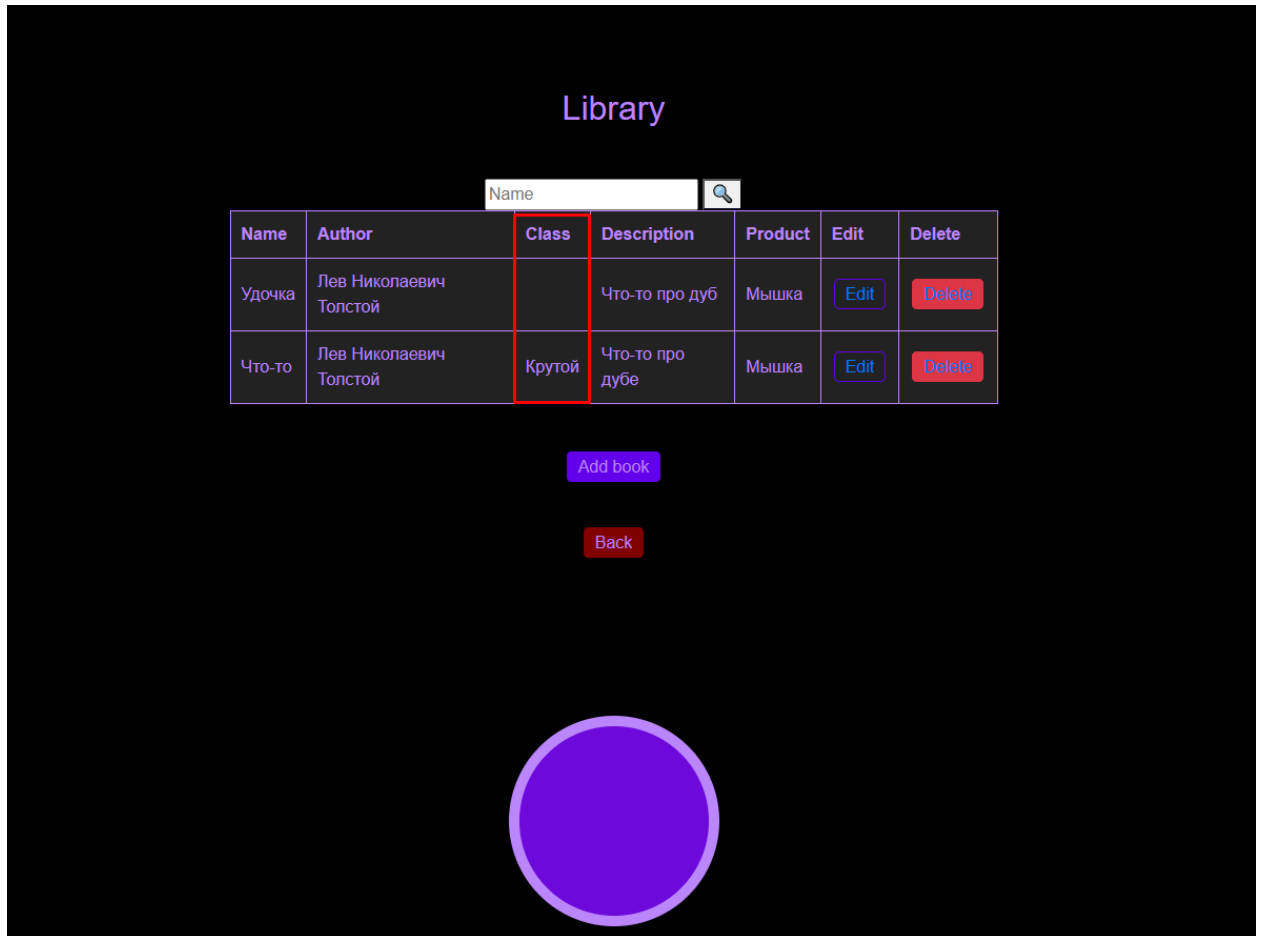


Рисунок 29. Один Ко Многим.

Это связь "многие ко многим" между UserModel и ClassModel. Аннотация @ManyToMany показывает, что существует такое отношение.

Это означает, что каждый пользователь может быть связан с несколькими классами, и каждый класс может иметь связи с несколькими пользователями.

Аннотация @JoinTable используется для определения таблицы, в которой будут храниться эти связи между UserModel и ClassModel.


```

3 usages
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable (name="users_class",
            joinColumns=@JoinColumn (name="users_id"),
            inverseJoinColumns=@JoinColumn(name="class_id"))
private List<ClassModel> clas;

```

Рисунок 30. UserController.

```

5 usages
@ManyToMany(fetch = FetchType.EAGER)
@JoinTable (name="users_class",
            joinColumns=@JoinColumn (name="class_id"),
            inverseJoinColumns=@JoinColumn(name="users_id"))
private List<UserModel> users;

```

Рисунок 31. ClassController.

Результат:

Users

Name	Surname	Midlename	Edit	Delete
Глеб		Викторович	Edit	Delete

[Add user](#)

[Back](#)

Рисунок 32. Таблица Пользователи.

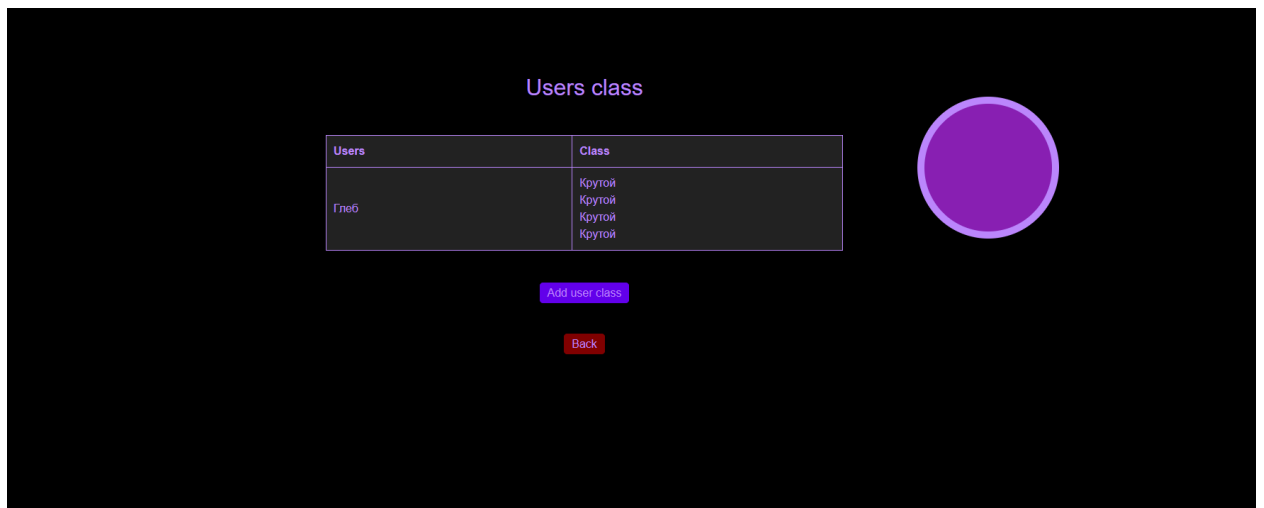


Рисунок 33.ManyToMany.

Вывод: в ходе данной практической работы, реализовал 3 типа связей, описал их и продемонстрировал их работу, для этого создал дополнительные модели и связал с уже созданными таблицами.

Практическая работа №5

Цель: реализовать механизм авторизации и регистрации.

Ход работы:

В первую очередь подключаем две новые зависимости.

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-test</artifactId>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
</dependencies>
```

Рисунок 34. Зависимости.

Мы разрабатываем модель для организации регистрации и авторизации. Аннотация `@Entity` определяет, что этот класс представляет собой объект базы данных, и его экземпляры будут сохраняться в ней.

Пометка `@Column(unique = true)` гарантирует уникальность значения поля.

Директива `@ElementCollection` применяется для хранения коллекции элементов внутри сущности, в данном случае - для хранения коллекции ролей.

Аннотация `@CollectionTable` создает таблицу, где будут храниться роли. Декларация `@Enumerated` указывает, что перечисление `Role` будет представлено в базе данных в виде строковых значений.

```

@Entity
public class UserModel {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    3 usages
    @Column(unique = true)
    private String username;
    3 usages
    private String password;
    3 usages
    private boolean active;

    3 usages
    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    public UserModel(){}

```

Рисунок 35. UserModel.

Разрабатываем `mvcConfig` для настройки отображения URL. Метод `registry.addViewController("/login")` устанавливает соответствие между URL `"/login"` и конкретным контроллером.

```

@Configuration
public class mvcConfig implements WebMvcConfigurer {

    no usages
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController(urlPathOrPattern: "/login").setViewName("login");
    }
}

```

Рисунок 36. mvcConfig.

Создаём `WebSecurityConfig`, который используется для обеспечения механизмов аутентификации и авторизации в приложении. Аннотация `@Configuration` указывает, что этот класс содержит настройки для приложения.

Аннотация `@EnableWebSecurity` указывает, что этот класс используется для настройки безопасности приложения. `@EnableGlobalMethodSecurity(prePostEnabled = true)` активирует поддержку аннотаций для методов, таких как `@PreAuthorize` и `@PostAuthorize`, что позволяет устанавливать ограничения на доступ к методам на основе ролей пользователей.

В методе `configure`, определяется способ аутентификации пользователей в приложении. В данном случае, используется JDBC-аутентификация, и информация о пользователях и их ролях извлекается из базы данных.

Метод `configure(HttpSecurity http)` задает правила доступа и авторизации для различных URL-адресов. Например, `.antMatchers("/login", "/registration").permitAll()` указывает, что страницы входа и регистрации доступны всем пользователям без аутентификации, в то время как `.anyRequest().authenticated()` означает, что для всех остальных URL-адресов требуется аутентификация.

Методы `.formLogin()` и `.logout()` настраивают форму входа и выхода.

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private DataSource dataSource;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication()
            .dataSource(dataSource)
            .passwordEncoder(passwordEncoder())
            .usersByUsernameQuery("select username, password, active from user_model where username = ?")
            .authoritiesByUsernameQuery("select u.username, ur.roles from user_model u inner join user_role ur on u.id = ur.user_id where u.username = ?");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().expressionInterceptUriRegistry()
            .antMatchers("/login", "/registration").permitAll()
            .anyRequest().authenticated()
            .and().httpSecurity()
            .formLogin().formLoginConfigurer<HttpSecurity>()
            .loginPage("/login")
            .defaultSuccessUrl("/home")
            .permitAll();
    }
}
```

Рисунок 37. WebSecurityConfig.

Разрабатываем контроллер для процесса регистрации:

Метод `registrationView()` отвечает за отображение страницы авторизации.

Метод `registrationForm()` предназначен для показа страницы регистрации новых пользователей.

Метод `registration()` осуществляет фактическую регистрацию нового пользователя. Сначала он проверяет, существует ли пользователь с таким же логином. В случае, если пользователь уже существует, он возвращает сообщение об ошибке на страницу регистрации.

Затем метод выполняет установку активности пользователя и назначает ему роль пользователя. После этого созданный пользователь сохраняется в базе данных, и пользователь перенаправляется на страницу авторизации.

```
@GetMapping("/")
public String registrationView() { return "login"; }

@GetMapping("/registration")
public String registrationForm() {
    return "registration";
}

@PostMapping("/registration")
public String registration(UserModel user, Model model) {
    UserModel userFromDB = userRepository.findByUsername(user.getUsername());
    if (userFromDB != null) {
        model.addAttribute("message", "Пользователь с таким логином уже существует");
        return "registration";
    }
    user.setActive(true);
    user.setRoles(Collections.singleton(Role.USER));
    user.setPassword(passwordEncoder.encode(user.getPassword()));
    userRepository.save(user);
    return "redirect:/login";
}
```

Рисунок 38. Контроллер для регистрации.

```

    </style>
</head>
<body>
<div class="auth-title">Авторизация</div>
<div th:if="${param.error}">
    Неверное имя пользователя или пароль.
</div>
<form th:action="@{/login}" method="post">
    <div><label> Логин : <input type="text" name="username"/> </label></div>
    <div><label> Пароль : <input type="password" name="password"/> </label></div>
    <div><input type="submit" value="Авторизоваться"/></div>
</form>
<a href="/registration">Нет аккаунта? Регистрируйтесь</a>
</body>
</html>

```

Рисунок 39. Login.

```

    </style>
</head>
<body>
<div class="auth-title">Регистрация</div>
<div th:if="${message}">
    Пользователь уже существует
</div>
<form th:action="@{/registration}" method="post">
    <div><label> Логин : <input type="text" name="username"/> </label></div>
    <div><label> Пароль : <input type="password" name="password"/> </label></div>
    <div><input type="submit" value="Зарегистрироваться"/></div>
    <a href="/login">Уже есть аккаунт? Войти</a>
</form>
</body>
</html>

```

Рисунок 40. Registration.

Результат:

The registration form is centered on a green background. It features a title 'Регистрация' in white. Below the title is a white box containing two input fields: 'Логин :' with the value 'gleb' and 'Пароль :' with three dots. A green button labeled 'Зарегистрироваться' is positioned below the password field. At the bottom of the white box, there is a link 'Уже есть аккаунт? Войти'.

Рисунок 41. Регистрация.

The authorization form is centered on a green background. It features a title 'Авторизация' in white. Below the title is a white box containing two input fields: 'Логин :' with the value 'gleb' and 'Пароль :' with three dots. A green button labeled 'Авторизоваться' is positioned below the password field. Below the white box, there is a link 'Нет аккаунта? Регистрируйтесь'.

Рисунок 42. Авторизация.

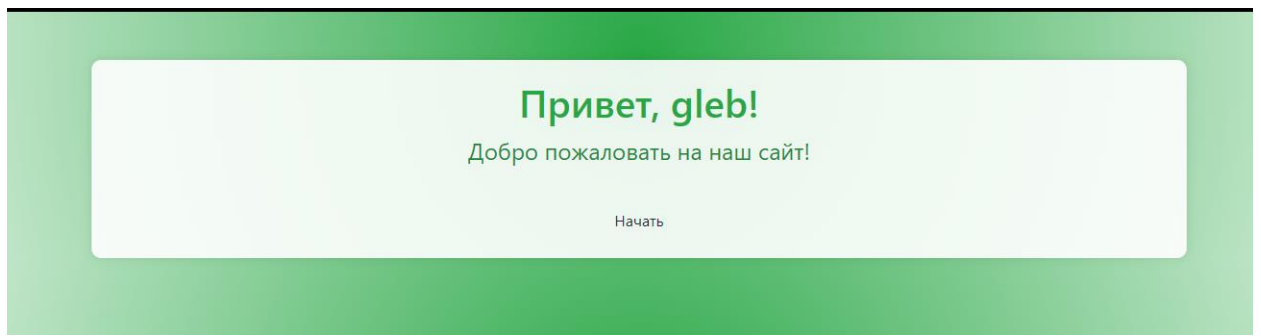


Рисунок 43. Успешная авторизация.

Вывод: в ходе данной практической работы, реализовал механизм регистрации и авторизации.

Практическая работа №6

Цель работы: реализовать механизм шифрование пароля пользователя.
Добавить разграничение прав доступа для пользователей.

Ход работы:

Создаем метод шифрования пароля.

```
@Autowired
private DataSource dataSource;

@Bean
public PasswordEncoder getPasswordEncoder(){
    return new BCryptPasswordEncoder( strength: 8);
}
```

Рисунок 44. Метод шифрования пароля.

Добавляем 2 новые роли.

```
9 usages
public enum RoleEnum implements GrantedAuthority {
    1 usage
    USER,
    no usages
    ADMIN,
    no usages
    MANAGER;
    @Override
    public String getAuthority() { return name(); }
}
```

Рисунок 45. Новые роли.

Создаем контроллер для редактирования пользователей и их ролей.

@PreAuthorize(hasAnyAuthority('MANAGER')) - аннотация
применяется к классу и ограничивает доступ к методам этого контроллера
только для пользователей с ролью 'MANAGER'

`public String editAccess(Model model)` - метод принимает объект `Model`, который предоставляет доступ к модели данных. Внутри метода он получает список всех пользователей из репозитория `userRepository`. Затем он добавляет этот список в модель данных с именем `"users"`, чтобы передать его в представление. Наконец, метод возвращает имя представления `"edit-access"`, которое будет отображаться после успешной обработки запроса.

`public String updateAccess(@PathVariable long userId, @RequestParam Set<RoleEnum> roles)` - Этот метод принимает параметры из URL-адреса (`userId`) и параметры из тела запроса (`roles`). Внутри метода он ищет пользователя по `userId` в репозитории и, если пользователь найден, обновляет его роли с использованием переданных `roles`.

```
@Autowired
private UserRepo userRepository;

@GetMapping("/edit-access")
public String editAccess(Model model) {
    // Получите список всех пользователей из репозитория
    List<ModelUser> allUsers = (List<ModelUser>) userRepository.findAll();

    // Передайте полученные данные в модель и отобразите их на странице редактирования доступа
    model.addAttribute("users", allUsers);
    return "edit-access"; // Замените на имя вашего шаблона
}

@PostMapping("/edit-access/{userId}/edit")
public String updateAccess(@PathVariable long userId, @RequestParam Set<RoleEnum> roles) {
    // Получите пользователя по ID
    Optional<ModelUser> optionalUser = userRepository.findById(userId);

    if (optionalUser.isPresent()) {
        ModelUser user = optionalUser.get();
        user.setRoles(roles);
        userRepository.save(user);
    }

    return "redirect:/edit-access";
}
```

Рисунок 46. UserAccessController.

На разные контроллеры задаем разные роли.

```
@Controller
@RequestMapping("/cars")
@PreAuthorize("hasAnyAuthority('USER', 'ADMIN')")
public class CarController {
```

Рисунок 47. Пример настройки (1).

```
@Controller
@PreAuthorize("hasAnyAuthority('MANAGER')")
public class userAccessController {
```

Рисунок 48. Пример настройки (2).

Результаты:

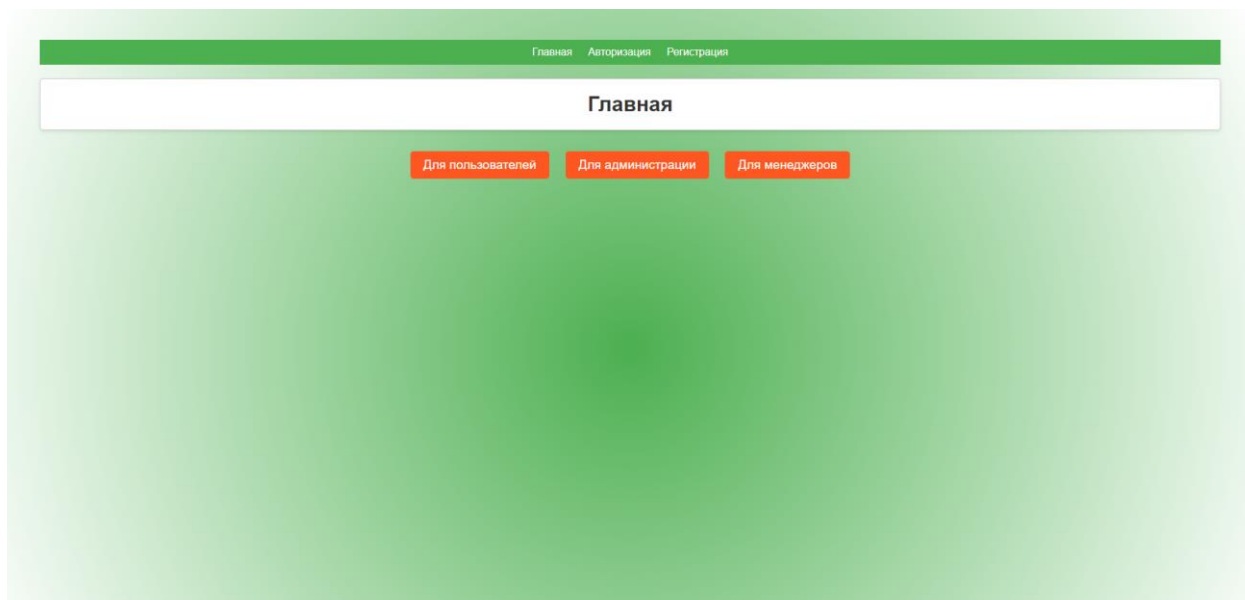


Рисунок 49. Главная страница.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Mon Sep 18 15:19:51 MSK 2023

There was an unexpected error (type=Forbidden, status=403).

Forbidden

org.springframework.security.access.AccessDeniedException: Доступ запрещен

```
at org.springframework.security.access.vote.AffirmativeBase.decide(AffirmativeBase.java:77)
at org.springframework.security.access.intercept.AbstractSecurityInterceptor.attemptAuthorization(AbstractSecurityInterceptor.java:253)
at org.springframework.security.access.intercept.AbstractSecurityInterceptor.beforeInvocation(AbstractSecurityInterceptor.java:222)
at org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke(MethodSecurityInterceptor.java:64)
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750)
at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:692)
at com.example.authreg.controller.userAccessController$$EnhancerBySpringCGLIB$$ef20746f.editAccess(<generated>)
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.base/java.lang.reflect.Method.invoke(Method.java:568)
at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:197)
at org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:141)
at org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:106)
at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)
at org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)
at org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1064)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:963)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:655)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:764)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:227)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:162)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

Рисунок 50. Попытка входа на страницу для менеджера, под аккаунтом пользователя.

Пользователь	Роль	Изменение
giebmanager	MANAGER	<div>Пользователь</div> <div>Изменить роль</div>

Рисунок 50. Изменение ролей.

Вывод: Реализовал механизм шифрование пароля пользователя.
Добавить разграничение прав доступа для пользователей.

Итоговый проект

Тема: магазин продажи фильмов «Кинотеатр»

Описание предметной области:

Информационная система представляет из себя веб-приложение для продажи фильмов. В её основе лежат три основные роли: пользователь, кассир и администратор.

Пользователь имеет возможность просматривать каталог фильмов, оформлять заказы, просматривать и редактировать свои заказы.

Кассир осуществляет операции с каталогом фильмов, редактирует информацию о режиссерах, добавляет и редактирует данные о фильмах, указывает и изменяет их жанры. Кроме того, кассир работает с заказами, управляет их статусами.

Администратор обладает полными правами доступа ко всей информации в системе. В его обязанности входит редактирование ролей всех пользователей, а также мониторинг и администрирование всех аспектов работы системы. Кроме того, администратор может добавлять, редактировать и удалять любые данные в системе, включая информацию о пользователях, фильмах, режиссерах и жанрах.

Диаграмма базы данных:

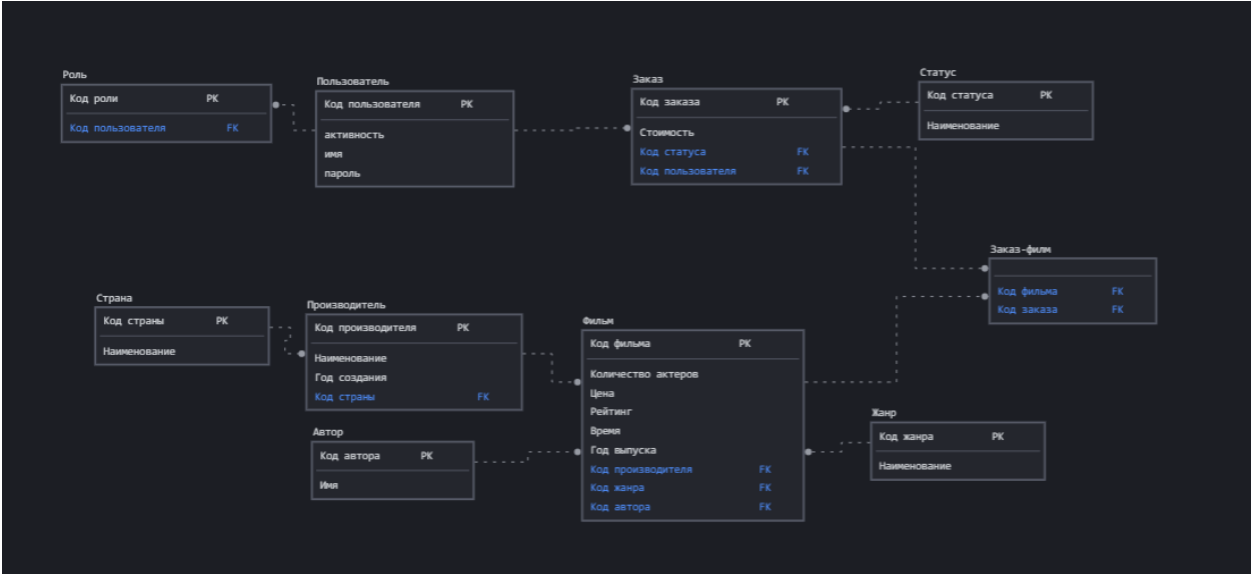


Рисунок 51 – Инфологическая модель базы данных.

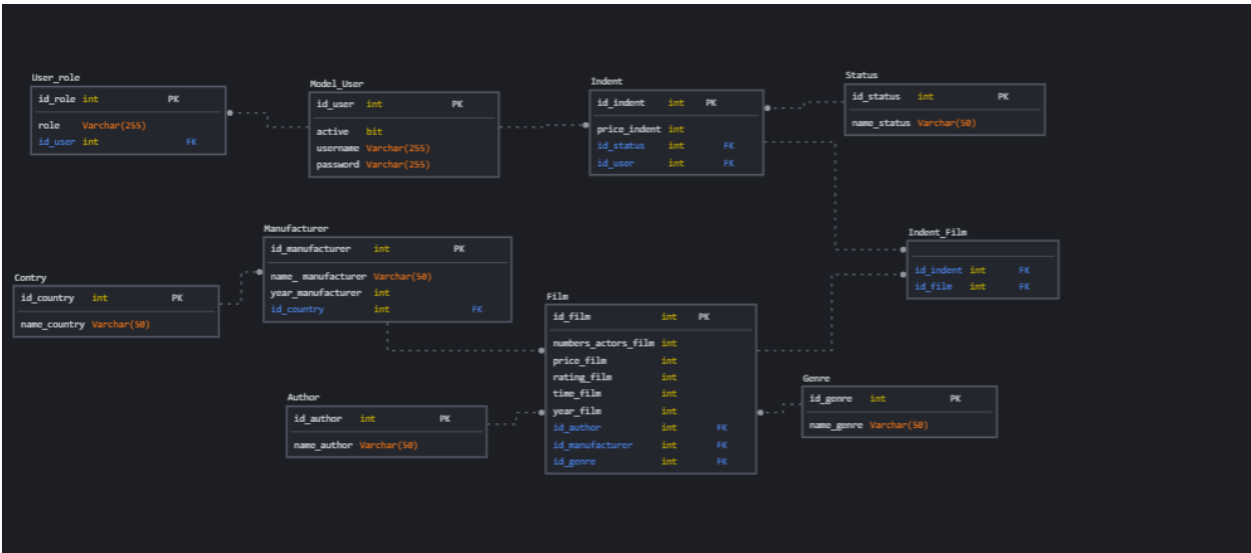


Рисунок 52 - Даталогическая модель базы данных.

Словарь данных:

Таблица 1 - Словарь данных

Ключ	Наименование	Тип данных	Примечание
Author			
PK	id_author	INT	Первичный ключ таблицы «Author»

	name_author	Varchar(50)	
Contry			
PK	id_country	INT	Первичный ключ таблицы «Country»
	name_country	Varchar(50)	
User_role			
PK	id_role	INT	Первичный ключ таблицы «Role»
	role	Varchar(255)	
FK	user_id	INT	Внешний ключ таблицы «User»
Status			
PK	id_status	INT	Первичный ключ таблицы «Status»
	name_status	Varchar(50)	
Model_User			
PK	id_user	INT	Первичный ключ таблицы «User»
	active	Boolean	
	username	Varchar(255)	
	password	Varchar(255)	
Manufacturer			
PK	Id_manufacturer	INT	Первичный ключ таблицы «Manufacturer»

	name_manufacturer	Varchar(50)	
	year_manufacturer	INT	
FK	country_id	INT	Внешний ключ таблицы «Country»
Indent_Film			
FK	indent_id	INT	Внешний ключ таблицы «Indent»
FK	film_id	INT	Внешний ключ таблицы «Film»
Indent			
PK	id_indent	INT	Первичный ключ таблицы «Indent»
	price_indent	INT	
FK	user_id	INT	Внешний ключ таблицы «User»
FK	status_id	INT	Внешний ключ таблицы «Status»
Genre			
PK	id_genre	INT	Первичный ключ таблицы «Genre»
	name_genre	Varchar(50)	
Film			
PK	id_film	INT	Первичный ключ таблицы «Film»
	numbers_actors_film	INT	
	price_film	INT	

	rating_film	INT	
	time_film	INT	
	year_film	INT	
FK	manufacturer_id	INT	Внешний ключ таблицы «Manufacturer»
FK	genre_id	INT	Внешний ключ таблицы «Genre»
FK	authror_id	INT	Внешний ключ таблицы «Author»

Скрипт базы данных:

```
CREATE TABLE IF NOT EXISTS public.author
```

```
(
    id_author bigint NOT NULL,
    name_author character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT author_pkey PRIMARY KEY (id_author)
)
```

```
CREATE TABLE IF NOT EXISTS public.country
```

```
(
    id_country bigint NOT NULL,
    name_country character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT country_pkey PRIMARY KEY (id_country)
)
```

```
CREATE TABLE IF NOT EXISTS public.film
```

```
(
    id_film bigint NOT NULL,
    name character varying(50) COLLATE pg_catalog."default",
    numbers_actors_film integer NOT NULL,
    price_film double precision NOT NULL,
    rating_film integer NOT NULL,
    time_film integer NOT NULL,
    year_film integer NOT NULL,
    id_authror bigint,
```

```

id_genre bigint,

id_manufacturer bigint,

CONSTRAINT film_pkey PRIMARY KEY (id_film),

CONSTRAINT fk32i59h9wca27rdt61ld4vbhd FOREIGN KEY (id_manufacturer)

REFERENCES public.manufacturer (id_manufacturer) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fkepqq0d9lvjpmdd64hr1u1ossuk FOREIGN KEY (id_author)

REFERENCES public.author (id_author) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fknwqkdkdqxrymsqj9lm7h64r2b FOREIGN KEY (id_genre)

REFERENCES public.genre (id_genre) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT film_numbers_actors_film_check CHECK (numbers_actors_film >= 1 AND numbers_actors_film <= 200),

CONSTRAINT film_rating_film_check CHECK (rating_film >= 1 AND rating_film <= 10),

CONSTRAINT film_time_film_check CHECK (time_film <= 1000 AND time_film >= 5),

CONSTRAINT film_year_film_check CHECK (year_film <= 2023 AND year_film >= 2000)

)

CREATE TABLE IF NOT EXISTS public.genre

(

id_genre bigint NOT NULL,

name_genre character varying(50) COLLATE pg_catalog."default",

CONSTRAINT genre_pkey PRIMARY KEY (id_genre)

)

CREATE TABLE IF NOT EXISTS public.indent

(

id_indent bigint NOT NULL,

price_indent double precision NOT NULL,

id_status bigint,

id_user bigint,

CONSTRAINT indent_pkey PRIMARY KEY (id_indent),

CONSTRAINT fk2ci4cvwlomxtmw60klpcxgn3u FOREIGN KEY (id_user)

REFERENCES public.model_user (id_user) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

```

```

CONSTRAINT fk8rske5d66t62qluvmbtbavbou FOREIGN KEY (id_status)

REFERENCES public.status (id_status) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

CREATE TABLE IF NOT EXISTS public.indent_film

(

indent_id bigint NOT NULL,

film_id bigint NOT NULL,

CONSTRAINT fkep7anflmg08i4t4xpdw1s2foh FOREIGN KEY (indent_id)

REFERENCES public.indent (id_indent) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fkfkm93bqwu9aw0epsa26enfq51 FOREIGN KEY (film_id)

REFERENCES public.film (id_film) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

CREATE TABLE IF NOT EXISTS public.manufacturer

(

id_manufacturer bigint NOT NULL,

name_genre character varying(50) COLLATE pg_catalog."default",

year_manufacturer integer NOT NULL,

id_country bigint,

CONSTRAINT manufacturer_pkey PRIMARY KEY (id_manufacturer),

CONSTRAINT fks4p4eftcxvj7s1jl7t86lu6d FOREIGN KEY (id_country)

REFERENCES public.country (id_country) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT manufacturer_year_manufacturer_check CHECK (year_manufacturer >= 1900 AND year_manufacturer <= 2023)

)

CREATE TABLE IF NOT EXISTS public.model_user

(

id_user bigint NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE

9223372036854775807 CACHE 1 ),

active boolean NOT NULL,

password character varying(255) COLLATE pg_catalog."default",

```

```

        username character varying(255) COLLATE pg_catalog."default",

        CONSTRAINT model_user_pkey PRIMARY KEY (id_user)

    )

CREATE TABLE IF NOT EXISTS public.status

(

    id_status bigint NOT NULL,

    name_status character varying(50) COLLATE pg_catalog."default",

    CONSTRAINT status_pkey PRIMARY KEY (id_status)

)

CREATE TABLE IF NOT EXISTS public.user_role

(

    user_id bigint NOT NULL,

    roles character varying(255) COLLATE pg_catalog."default",

    CONSTRAINT fkhnk3nw6rsvkly3ww7umdq7ys1 FOREIGN KEY (user_id)

        REFERENCES public.model_user (id_user) MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

)

```

Код программы:

MvcConfig

```

package com.example.kinoteatr.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class MvcConfig implements WebMvcConfigurer {

    public void addViewControllers(ViewControllerRegistry registry) {

        registry.addViewController("/login").setViewName("login");

    }

}

```

WebSecurityConfig

```

package com.example.kinoteatr.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

import javax.sql.DataSource;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private DataSource dataSource;

    @Bean
    public PasswordEncoder getPasswordEncoder(){
        return new BCryptPasswordEncoder(8);
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.jdbcAuthentication().dataSource(dataSource).passwordEncoder(getPasswordEncoder())
            .usersByUsernameQuery("select username, password, active from model_user where username =?")
            .authoritiesByUsernameQuery("select u.username, ur.roles from model_user u inner join user_role ur on u.id_user =
ur.user_id where u.username=?");
    }

    @Override

```

```
protected void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests()

        .antMatchers("/login", "/registration").permitAll()

        .anyRequest()

        .authenticated()

        .and()

        .formLogin()

        .loginPage("/login")

        .defaultSuccessUrl("/")

        .permitAll()

        .and()

        .logout()

        .permitAll()

        .and().csrf().disable().cors().disable();

    }

}
```

AuthorController

```
package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.Author;
import com.example.kinoteatr.repo.AuthorRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@Controller
@RequestMapping("/authors")
@PreAuthorize("hasAnyAuthority('ADMIN')")
public class AuthorController {

    private final AuthorRepo authorRepo;
```

@Autowired

```
public AuthorController(AuthorRepo authorRepo) {  
    this.authorRepo = authorRepo;  
}
```

@GetMapping()

```
public String listAuthors(Model model) {  
    Iterable<Author> authors = authorRepo.findAll();  
    model.addAttribute("authors", authors);  
    return "authors/allAuthors";  
}
```

@GetMapping("/addAuthor")

```
public String showAddClassForm(Model model) {  
    Author author = new Author();  
    //clas.setPublicationYear(0);  
    model.addAttribute("author", author);  
    return "authors/addAuthor";  
}
```

@PostMapping("/addAuthor")

```
public String addAuthor(@Valid @ModelAttribute("author") Author author, BindingResult bindingResult) {  
    if (bindingResult.hasErrors()) {  
        return "authors/addAuthor";  
    }  
    System.out.println("Name: " + author.getName_author());  
    authorRepo.save(author);  
    return "redirect:/authors";  
}
```

@GetMapping("/editAuthor/{id}")

```
public String showEditAuthorForm(@PathVariable("id") long id, Model model) {  
    Author author = authorRepo.findById(id).orElse(null);  
    if (author == null) {  
        return "redirect:/authors";  
    }  
}
```



```

        model.addAttribute("author", author);

        return "authors/editAuthor";
    }

@PostMapping("/editAuthor/{id}")

public String editAuthor(@PathVariable("id") int id, @Valid @ModelAttribute("author") Author author, BindingResult
bindingResult) {

    if (bindingResult.hasErrors()) {

        return "authors/editAuthor"; // Останется на странице редактирования с отображением ошибок
    }

    author.setId_author(id);

    authorRepo.save(author);

    return "redirect:/authors"; // Перенаправление на страницу со всеми героями
}

@GetMapping("/deleteAuthor/{id}")

public String deleteAuthor(@PathVariable("id") long id) {

    authorRepo.deleteById(id);

    return "redirect:/authors";
}
}

```

CountryController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.Country;

import com.example.kinoteatr.repo.CountryRepo;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

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

import javax.validation.Valid;

@Controller

@RequestMapping("/countries")

```

```

@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN')")

public class CountryController {

    private final CountryRepo countryRepo;

    @Autowired

    public CountryController(CountryRepo countryRepo) {

        this.countryRepo = countryRepo;

    }

    @GetMapping()

    public String listCountries(Model model) {

        Iterable<Country> countries = countryRepo.findAll();

        model.addAttribute("countries", countries);

        return "countries/allCountries";

    }

    @GetMapping("/addCountry")

    public String showAddClassForm(Model model) {

        Country country = new Country();

        //clas.setPublicationYear(0);

        model.addAttribute("country", country);

        return "countries/addCountry";

    }

    @PostMapping("/addCountry")

    public String addCountry(@Valid @ModelAttribute("country") Country country, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {

            return "countries/addCountry";

        }

        System.out.println("Name: " + country.getName_country());

        countryRepo.save(country);

        return "redirect:/countries";

    }

    @GetMapping("/editCountry/{id}")

    public String showEditCountryForm(@PathVariable("id") long id, Model model) {

```

```

        Country country = countryRepo.findById(id).orElse(null);

        if (country == null) {

            return "redirect:/countries";

        }

        model.addAttribute("country", country);

        return "countries/editCountry";

    }

    @PostMapping("/editCountry/{id}")

    public String editCountry(@PathVariable("id") int id, @Valid @ModelAttribute("country") Country country, BindingResult
bindingResult) {

        if (bindingResult.hasErrors()) {

            return "countries/editCountry"; // Останется на странице редактирования с отображением ошибок

        }

        country.setId_country(id);

        countryRepo.save(country);

        return "redirect:/countries";

    }

    @GetMapping("/deleteCountry/{id}")

    public String deleteCountry(@PathVariable("id") long id) {

        countryRepo.deleteById(id);

        return "redirect:/countries";

    }

}

```

FilmController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.*;

import com.example.kinoteatr.repo.AuthorRepo;

import com.example.kinoteatr.repo.FilmRepo;

import com.example.kinoteatr.repo.GenreRepo;

import com.example.kinoteatr.repo.ManufacturerRepo;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

```

```

import org.springframework.validation.BindingResult;

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

import javax.validation.Valid;

@Controller

@RequestMapping("/films")

@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN')")

public class FilmController {

    @Autowired

    public ManufacturerRepo manufacturerRepo;

    @Autowired

    private FilmRepo filmRepo;

    @Autowired

    private GenreRepo genreRepo;

    @Autowired

    private AuthorRepo authorRepo;

    @GetMapping()

    public String listManu(Model model) {

        Iterable<Film> films = filmRepo.findAll();

        model.addAttribute("films", films);

        return "films/allFilms";

    }

    @GetMapping("/addFilm")

    public String showAddClassForm(Model model) {

        Film film = new Film();

        film.setYear_film(2000);

        film.setTime_film(2);

        film.setNumbers_actors_film(1);

        film.setRating_film(1);

        Iterable<Manufacturer> manufacturers = manufacturerRepo.findAll();

        model.addAttribute("manufacturers", manufacturers);

        Iterable<Genre> genres = genreRepo.findAll();

```

```

        model.addAttribute("genres", genres);

        Iterable<Author> authors = authorRepo.findAll();

        model.addAttribute("authors", authors);

        model.addAttribute("film", film);

        return "films/addFilm";
    }

    @PostMapping("/addFilm")
    public String addManu(@Valid @ModelAttribute("film") Film film, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {

            return "films/addFilm";

        }

        System.out.println("Name: " + film.getName_film());

        filmRepo.save(film);

        return "redirect:/films";

    }

    @GetMapping("/editFilm/{id}")
    public String showEditManuForm(@PathVariable("id") long id, Model model) {

        Film film = filmRepo.findById(id).orElse(null);

        if (film == null) {

            return "redirect:/films";

        }

        Iterable<Manufacturer> manufacturers = manufacturerRepo.findAll();

        model.addAttribute("manufacturers", manufacturers);

        Iterable<Genre> genres = genreRepo.findAll();

        model.addAttribute("genres", genres);

        Iterable<Author> authors = authorRepo.findAll();

        model.addAttribute("authors", authors);

        model.addAttribute("film", film);

        return "films/editFilm";
    }

```

```

    }

    @PostMapping("/editFilm/{id}")

    public String editManu(@PathVariable("id") long id, @Valid @ModelAttribute("film") Film film, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {

            return "films/editFilm"; // Останется на странице редактирования с отображением ошибок

        }

        film.setId_film(id);

        filmRepo.save(film);

        return "redirect:/films";

    }

    @GetMapping("/deleteFilm/{id}")

    public String deleteManu(@PathVariable("id") long id) {

        filmRepo.deleteById(id);

        return "redirect:/films";

    }

}

```

GenreController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.Genre;

import com.example.kinoteatr.repo.GenreRepo;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

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

import javax.validation.Valid;

@Controller

@RequestMapping("/genres")

@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN')")

```

```

public class GenreController {

    private final GenreRepo genreRepo;

    @Autowired

    public GenreController(GenreRepo genreRepo) {

        this.genreRepo = genreRepo;
    }

    @GetMapping()

    public String listGenres(Model model) {

        Iterable<Genre> genres = genreRepo.findAll();

        model.addAttribute("genres", genres);

        return "genres/allGenres";
    }

    @GetMapping("/addGenre")

    public String showAddClassForm(Model model) {

        Genre genre = new Genre();

        model.addAttribute("genre", genre);

        return "genres/addGenre";
    }

    @PostMapping("/addGenre")

    public String addGenre(@Valid @ModelAttribute("genre") Genre genre, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {

            return "genres/addGenre";
        }

        System.out.println("Name: " + genre.getName_genre());

        genreRepo.save(genre);

        return "redirect:/genres";
    }

    @GetMapping("/editGenre/{id}")

    public String showEditGenreForm(@PathVariable("id") long id, Model model) {

        Genre genre = genreRepo.findById(id).orElse(null);

        if (genre == null) {

            return "redirect:/genres";
        }
    }
}

```

```

    }

    model.addAttribute("genre", genre);

    return "genres/editGenre";
}

@PostMapping("/editGenre/{id}")

public String editGenre(@PathVariable("id") int id, @Valid @ModelAttribute("genre") Genre genre, BindingResult bindingResult)
{
    if (bindingResult.hasErrors()) {

        return "genres/editGenre"; // Останется на странице редактирования с отображением ошибок
    }

    genre.setId_genre(id);

    genreRepo.save(genre);

    return "redirect:/genres";
}

@GetMapping("/deleteGenre/{id}")

public String deleteGenre(@PathVariable("id") long id) {

    genreRepo.deleteById(id);

    return "redirect:/genres";
}
}

```

IndentController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.Film;

import com.example.kinoteatr.model.Indent;

import com.example.kinoteatr.model.ModelUser;

import com.example.kinoteatr.model.Status;

import com.example.kinoteatr.repo.FilmRepo;

import com.example.kinoteatr.repo.IndentRepo;

import com.example.kinoteatr.repo.StatusRepo;

import com.example.kinoteatr.repo.UserRepo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.security.access.prepost.PreAuthorize;

```



```

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

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

import org.springframework.web.servlet.mvc.support.RedirectAttributes;


import javax.validation.Valid;


@Controller

@RequestMapping("/indents")

@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN', 'USER')")

public class IndentController {


    private static final Logger logger = LoggerFactory.getLogger(IndentController.class);


    @Autowired

    public IndentRepo indentRepo;

    @Autowired

    private UserRepo userRepo;

    @Autowired

    private StatusRepo statusRepo;

    @Autowired

    private FilmRepo filmRepo;


    @GetMapping()

    public String listManu(Model model) {

        Iterable<Indent> indents = indentRepo.findAll();

        model.addAttribute("indents", indents);

        return "indents/allIndents";

    }


    @GetMapping("/addIndent")

    public String showAddClassForm(Model model) {

        Indent indent = new Indent();


        Iterable<Status> statuses = statusRepo.findAll();

        model.addAttribute("statuses", statuses);

```

```

Iterable<ModelUser> users = userRepo.findAll();

model.addAttribute("users", users);


model.addAttribute("indent", indent);

return "indents/addIndent";
}

```

```

@PostMapping("/addIndent")

public String addManu(@Valid @ModelAttribute("indent") Indent indent, BindingResult bindingResult) {

    if (bindingResult.hasErrors()) {

        return "indents/addIndent";

    }

    indentRepo.save(indent);

    return "redirect:/indents";

}

```

```

@GetMapping("/editIndent/{id}")

public String showEditManuForm(@PathVariable("id") long id, Model model) {

    Indent indent = indentRepo.findById(id).orElse(null);

    if (indent == null) {

        return "redirect:/indents";

    }

}

```

```

Iterable<Status> statuses = statusRepo.findAll();

model.addAttribute("statuses", statuses);

```

```

Iterable<ModelUser> users = userRepo.findAll();

model.addAttribute("users", users);


model.addAttribute("indent", indent);

return "indents/editIndent";

}

```

```

@PostMapping("/editIndent/{id}")

public String editManu(@PathVariable("id") long id, @Valid @ModelAttribute("indent") Indent indent, BindingResult bindingResult) {

```

```

if (bindingResult.hasErrors()) {

    return "indents/editIndent"; // Останется на странице редактирования с отображением ошибок

}

indent.setId_indent(id);

indentRepo.save(indent);

return "redirect:/indents";

}

```

```

@GetMapping("/deleteIndent/{id}")

public String deleteManu(@PathVariable("id") long id) {

    indentRepo.deleteById(id);

    return "redirect:/indents";

}

```

```

@GetMapping("/addFilmToIndent")

public String showAddFilmToIndentPage(Model model) {

    Iterable<Film> film = filmRepo.findAll();

    model.addAttribute("film", film);

    Iterable<Indent> indent = indentRepo.findAll();

    model.addAttribute("indent", indent);

    return "indents/addFilmToIndent"; // Название HTML-шаблона для этой страницы

}

```

```

@PostMapping("/addFilmToIndent")

public String addFilmToIndent(@RequestParam long indent, @RequestParam String film, RedirectAttributes redirectAttributes)

{

    Indent indent2 = indentRepo.findByName(indent);

    Film film2 = filmRepo.findByName(film);

    indent2.getFilm().add(film2);

    film2.getIndent().add(indent2);

    indentRepo.save(indent2);

    // Добавьте атрибут "success" для сообщения об успешном добавлении

    redirectAttributes.addFlashAttribute("success", "Фильм успешно добавлено в заказ");

    return "redirect:/indents";

}

```

```
}
```

```
}
```

ManufacturerController

```
package com.example.kinoteatr.controller;
```

```
import com.example.kinoteatr.model.Country;
```

```
import com.example.kinoteatr.model.Manufacturer;
```

```
import com.example.kinoteatr.repo.CountryRepo;
```

```
import com.example.kinoteatr.repo.ManufacturerRepo;
```

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

```
import org.springframework.security.access.prepost.PreAuthorize;
```

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
```

```
import org.springframework.validation.BindingResult;
```

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

```
import javax.validation.Valid;
```

```
@Controller
```

```
@RequestMapping("/manufacturers")
```

```
@PreAuthorize("hasAnyAuthority('ADMIN')")
```

```
public class ManufacturerController {
```

```
    @Autowired
```

```
    public ManufacturerRepo manufacturerRepo;
```

```
    @Autowired
```

```
    private CountryRepo countryRepo;
```

```
    @GetMapping()
```

```
    public String listManu(Model model) {
```

```
        Iterable<Manufacturer> manufacturers = manufacturerRepo.findAll();
```

```
        model.addAttribute("manufacturers", manufacturers);
```

```
        return "manufacturers/allManufacturers";
```

```
}
```

```
@GetMapping("/addManufacturer")
```

```
public String showAddClassForm(Model model) {  
  
    Manufacturer manufacturer = new Manufacturer();  
  
    manufacturer.setYear_manufacturer(2000);  
  
  
    Iterable<Country> countries = countryRepo.findAll();  
  
    model.addAttribute("countries", countries);  
  
  
    model.addAttribute("manufacturer", manufacturer);  
  
    return "manufacturers/addManufacturer";  
}
```

```
@PostMapping("/addManufacturer")
```

```
public String addManu(@Valid @ModelAttribute("manufacturer") Manufacturer manufacturer, BindingResult bindingResult) {  
  
    if (bindingResult.hasErrors()) {  
  
        return "manufacturers/addManufacturer";  
    }  
  
    System.out.println("Name: " + manufacturer.getName_manufacturer());  
  
    manufacturerRepo.save(manufacturer);  
  
    return "redirect:/manufacturers";  
}
```

```
@GetMapping("/editManufacturer/{id}")
```

```
public String showEditManuForm(@PathVariable("id") long id, Model model) {  
  
    Manufacturer manufacturer = manufacturerRepo.findById(id).orElse(null);  
  
    if (manufacturer == null) {  
  
        return "redirect:/manufacturers";  
    }  
  
  
    Iterable<Country> countries = countryRepo.findAll();  
  
    model.addAttribute("countries", countries);  
  
  
    model.addAttribute("manufacturer", manufacturer);  
  
    return "manufacturers/editManufacturer";  
}
```

```

        @PostMapping("/editManufacturer/{id}")

        public String editManu(@PathVariable("id") int id, @Valid @ModelAttribute("manufacturer") Manufacturer manufacturer,
BindingResult bindingResult) {

            if (bindingResult.hasErrors()) {

                return "manufacturers/editManufacturer"; // Останется на странице редактирования с отображением ошибок
            }

            manufacturer.setId_manufacturer(id);

            manufacturerRepo.save(manufacturer);

            return "redirect:/manufacturers";

        }

        @GetMapping("/deleteManufacturer/{id}")

        public String deleteManu(@PathVariable("id") long id) {

            manufacturerRepo.deleteById(id);

            return "redirect:/manufacturers";

        }

    }
}

```

registerController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.ModelUser;

import com.example.kinoteatr.model.RoleEnum;

import com.example.kinoteatr.repo.UserRepo;

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

import java.util.Collections;

@Controller

public class registerController {

```

```
@Autowired
```

```
private UserRepo userRepository;
```

```
@Autowired
```

```
private PasswordEncoder passwordEncoder;
```

```
@GetMapping("/registration")
```

```
private String RegView()
```

```
{  
  
    return "regis";  
}
```

```
@PostMapping("/registration")
```

```
private String Reg(ModelUser user, Model model)
```

```
{  
  
    ModelUser user_from_db = userRepository.findByUsername(user.getUsername());  
  
    if (user_from_db != null)  
  
    {  
  
        model.addAttribute("message", "Пользователь с таким логином уже существует");  
  
        return "regis";  
  
    }  
  
    user.setActive(true);  
  
    user.setRoles(Collections.singleton(RoleEnum.USER));  
  
    user.setPassword(passwordEncoder.encode(user.getPassword()));  
  
    userRepository.save(user);  
  
    return "redirect:/login";  
}  
}
```

StatusController

```
package com.example.kinoteatr.controller;
```

```
import com.example.kinoteatr.model.Status;
```

```
import com.example.kinoteatr.repo.StatusRepo;
```

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

```
import org.springframework.security.access.prepost.PreAuthorize;
```

```
import org.springframework.stereotype.Controller;
```

```

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

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

import javax.validation.Valid;

@Controller

@RequestMapping("/status")

@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN')")

public class StatusController {

    private final StatusRepo statusRepo;

    @Autowired

    public StatusController(StatusRepo statusRepo) {

        this.statusRepo = statusRepo;

    }

    @GetMapping()

    public String listStatus(Model model) {

        Iterable<Status> status = statusRepo.findAll();

        model.addAttribute("status", status);

        return "status/allStatus";

    }

    @GetMapping("/addStatus")

    public String showAddClassForm(Model model) {

        Status status = new Status();

        //clas.setPublicationYear(0);

        model.addAttribute("status", status);

        return "status/addStatus";

    }

    @PostMapping("/addStatus")

    public String addStatus(@Valid @ModelAttribute("status") Status status, BindingResult bindingResult) {

        if (bindingResult.hasErrors()) {

            return "status/addStatus";

        }

    }

```



```

        System.out.println("Name: " + status.getName_status());

        statusRepo.save(status);

        return "redirect:/status";
    }

    @GetMapping("/editStatus/{id}")
    public String showEditStatusForm(@PathVariable("id") long id, Model model) {

        Status status = statusRepo.findById(id).orElse(null);

        if (status == null) {

            return "redirect:/status";

        }

        model.addAttribute("status", status);

        return "status/editStatus";
    }

    @PostMapping("/editStatus/{id}")
    public String editStatus(@PathVariable("id") int id, @Valid @ModelAttribute("status") Status status, BindingResult bindingResult)
    {

        if (bindingResult.hasErrors()) {

            return "status/editStatus"; // Останется на странице редактирования с отображением ошибок

        }

        status.setId_status(id);

        statusRepo.save(status);

        return "redirect:/status";
    }

    @GetMapping("/deleteStatus/{id}")
    public String deleteStatus(@PathVariable("id") long id) {

        statusRepo.deleteById(id);

        return "redirect:/status";
    }
}

```

userAccessController

```

package com.example.kinoteatr.controller;

import com.example.kinoteatr.model.ModelUser;

import com.example.kinoteatr.model.RoleEnum;

```

```

import com.example.kinoteatr.repo.UserRepo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
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.RequestParam;


import java.util.List;
import java.util.Optional;
import java.util.Set;


@Controller
@PreAuthorize("hasAnyAuthority('ADMIN')")
public class userAccessController {


    @Autowired
    private UserRepo userRepository;


    @GetMapping("/edit-access")
    public String editAccess(Model model) {

        // Получите список всех пользователей из репозитория
        List<ModelUser> allUsers = (List<ModelUser>) userRepository.findAll();


        // Передайте полученные данные в модель и отобразите их на странице редактирования доступа
        model.addAttribute("users", allUsers);

        return "edit-access"; // Замените на имя вашего шаблона
    }


    @PostMapping("/edit-access/{userId}/edit")
    public String updateAccess(@PathVariable long userId, @RequestParam Set<RoleEnum> roles) {

        // Получите пользователя по ID
        Optional<ModelUser> optionalUser = userRepository.findById(userId);


        if (optionalUser.isPresent()) {

```

```

        ModelUser user = optionalUser.get();

        user.setRoles(roles);

        userRepository.save(user);
    }

    return "redirect:/edit-access";
}

}

```

indexController

```

package com.example.kinoteatr.controller;

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

@Controller

public class indexController {

    @GetMapping("/")

    public String home() {

        return "index";
    }

}

```

Author

```

package com.example.kinoteatr.model;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import java.util.Collection;

@Entity

@Table(name = "author")

public class Author {

```

```
@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id_author;

@NotBlank(message = "Введите имя")

@Size(max = 50, message = "Длина имени должна быть не больше 200 символов")

@Column(name = "name_author")

private String name_author;


@OneToMany(mappedBy = "author", fetch = FetchType.LAZY)

private Collection<Film> film;


public Author(){}


public Author(long id_author, String name_author, Collection<Film> film) {

    this.id_author = id_author;

    this.name_author = name_author;

    this.film = film;

}


public long getId_author() {

    return id_author;

}


public void setId_author(long id_author) {

    this.id_author = id_author;

}


public String getName_author() {

    return name_author;

}


public void setName_author(String name_author) {

    this.name_author = name_author;

}


public Collection<Film> getFilm() {

    return film;

}
```

```

    }

    public void setFilm(Collection<Film> film) {

        this.film = film;

    }

}

```

Country

```

package com.example.kinoteatr.model;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import java.util.Collection;

@Entity
@Table(name = "country")
public class Country {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id_country;

    @NotBlank(message = "Введите название")
    @Size(max = 50, message = "Длина названия должна быть не больше 50 символов")
    @Column(name = "name_country")
    private String name_country;

    @OneToMany(mappedBy = "country", fetch = FetchType.LAZY)
    private Collection<Manufacturer> manufacturer;

    public Country(){}

    public Country(long id_country, String name_country, Collection<Manufacturer> manufacturer) {

        this.id_country = id_country;

        this.name_country = name_country;

        this.manufacturer = manufacturer;
    }
}

```

```

    }

    public long getId_country() {

        return id_country;

    }

    public void setId_country(long id_country) {

        this.id_country = id_country;

    }

    public String getName_country() {

        return name_country;

    }

    public void setName_country(String name_country) {

        this.name_country = name_country;

    }

    public Collection<Manufacturer> getManufacturer() {

        return manufacturer;

    }

    public void setManufacturer(Collection<Manufacturer> manufacturer) {

        this.manufacturer = manufacturer;

    }

}

```

Film

```

package com.example.kinoteatr.model;

import javax.persistence.*;

import javax.validation.constraints.*;

import java.util.List;

@Entity

@Table(name = "film")

```

```
public class Film {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

    private long id_film;

    @NotBlank(message = "Введите название")

    @Size(max = 50, message = "Длина названия должна быть не больше 50 символов")

    @Column(name = "name")

    private String name_film;

    @Min(value = 1, message = "Значение должно быть больше 0")

    @Max(value = 200, message = "Значение должно быть меньше 200")

    private int numbers_actors_film;

    @Min(value = 1, message = "Значение должно быть больше 0")

    @Max(value = 10, message = "Значение должно быть меньше 10")

    private int rating_film;

    @DecimalMin(value = "1.0", message = "Значение должно быть больше 0")

    @DecimalMax(value = "100000.0", message = "Значение должно быть меньше 100000")

    private double price_film;

    @Min(value = 1800, message = "Значение должно быть больше 2000")

    @Max(value = 2030, message = "Значение должно быть меньше 2023")

    private int year_film;

    @Min(value = 1, message = "Значение должно быть больше 5")

    @Max(value = 100, message = "Значение должно быть меньше 1000")

    private int time_film;

    @ManyToOne

    @JoinColumn(name = "id_genre") // Указывает на столбец, который связывает сущности

    private Genre genre;

    @ManyToOne

    @JoinColumn(name = "id_manufacturer") // Указывает на столбец, который связывает сущности

    private Manufacturer manufacturer;
```

```

@ManyToOne

@JoinColumn(name = "id_author") // Указывает на столбец, который связывает сущности

private Author author;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable (name="indent_film",

            joinColumns=@JoinColumn (name="film_id"),

            inverseJoinColumns=@JoinColumn(name="indent_id"))

private List<Indent> indent;

public Film(){ }

    public Film(long id_film, String name_film, int numbers_actors_film, int rating_film, double price_film, int year_film, int time_film,
Genre genre, Manufacturer manufacturer, Author author, List<Indent> indent) {

        this.id_film = id_film;

        this.name_film = name_film;

        this.numbers_actors_film = numbers_actors_film;

        this.rating_film = rating_film;

        this.price_film = price_film;

        this.year_film = year_film;

        this.time_film = time_film;

        this.genre = genre;

        this.manufacturer = manufacturer;

        this.author = author;

        this.indent = indent;

    }

    public List<Indent> getIndent() {

        return indent;

    }

    public void setIndent(List<Indent> indent) {

        this.indent = indent;

    }

    public long getId_film() {

```



```
        return id_film;
    }

    public void setId_film(long id_film) {
        this.id_film = id_film;
    }

    public String getName_film() {
        return name_film;
    }

    public void setName_film(String name_film) {
        this.name_film = name_film;
    }

    public int getNumbers_actors_film() {
        return numbers_actors_film;
    }

    public void setNumbers_actors_film(int numbers_actors_film) {
        this.numbers_actors_film = numbers_actors_film;
    }

    public int getRating_film() {
        return rating_film;
    }

    public void setRating_film(int rating_film) {
        this.rating_film = rating_film;
    }

    public double getPrice_film() {
        return price_film;
    }

    public void setPrice_film(double price_film) {
        this.price_film = price_film;
    }
}
```

```
}
```

```
public int getYear_film() {  
    return year_film;  
}
```

```
public void setYear_film(int year_film) {  
    this.year_film = year_film;  
}
```

```
public int getTime_film() {  
    return time_film;  
}
```

```
public void setTime_film(int time_film) {  
    this.time_film = time_film;  
}
```

```
public Genre getGenre() {  
    return genre;  
}
```

```
public void setGenre(Genre genre) {  
    this.genre = genre;  
}
```

```
public Manufacturer getManufacturer() {  
    return manufacturer;  
}
```

```
public void setManufacturer(Manufacturer manufacturer) {  
    this.manufacturer = manufacturer;  
}
```

```
public Author getAuthor() {  
    return author;  
}
```

```
    public void setAuthor(Author author) {  
        this.author = author;  
    }  
}
```

Genre

```
package com.example.kinoteatr.model;
```

```
import javax.persistence.*;
```

```
import javax.validation.constraints.NotBlank;
```

```
import javax.validation.constraints.Size;
```

```
import java.util.Collection;
```

```
@Entity
```

```
@Table(name = "genre")
```

```
public class Genre {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private long id_genre;
```

```
    @NotBlank(message = "Введите название")
```

```
    @Size(max = 50, message = "Длина названия должна быть не больше 50 символов")
```

```
    @Column(name = "name_genre")
```

```
    private String name_genre;
```

```
    @OneToMany(mappedBy = "genre", fetch = FetchType.LAZY)
```

```
    private Collection<Film> film;
```

```
    public Genre(){}
```

```
    public Genre(long id_genre, String name_genre, Collection<Film> film) {
```

```
        this.id_genre = id_genre;
```

```
        this.name_genre = name_genre;
```

```
        this.film = film;
```

```
    }
```

```

public long getId_genre() {

    return id_genre;

}

public void setId_genre(long id_genre) {

    this.id_genre = id_genre;

}

public String getName_genre() {

    return name_genre;

}

public void setName_genre(String name_genre) {

    this.name_genre = name_genre;

}

public Collection<Film> getFilm() {

    return film;

}

public void setFilm(Collection<Film> film) {

    this.film = film;

}
}

```

Indent

```

package com.example.kinoteatr.model;

import javax.persistence.*;

import javax.validation.constraints.DecimalMax;

import javax.validation.constraints.DecimalMin;

import java.util.List;

@Entity

@Table(name = "indent")

public class Indent {

```

```

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private long id_indent;

@DecimalMin(value = "0.0", message = "Значение должно быть больше 0")

@DecimalMax(value = "100000.0", message = "Значение должно быть меньше 100000")

private double price_indent;


@ManyToOne

@JoinColumn(name = "id_status") // Указывает на столбец, который связывает сущности

private Status status;


@ManyToMany(fetch = FetchType.EAGER)

@JoinTable (name="indent_film",

            joinColumns=@JoinColumn (name="indent_id"),

            inverseJoinColumns=@JoinColumn(name="film_id"))

private List<Film> film;


@ManyToOne

@JoinColumn(name = "id_user") // Указывает на столбец, который связывает сущности

private ModelUser user;


public Indent(){ }


public Indent(long id_indent, double price_indent, Status status, List<Film> film, ModelUser user) {

    this.id_indent = id_indent;

    this.price_indent = price_indent;

    this.status = status;

    this.film = film;

    this.user = user;

}


public ModelUser getUser() {

    return user;

}


public void setUser(ModelUser user) {

```

```
        this.user = user;
    }

    public long getId_indent() {
        return id_indent;
    }

    public void setId_indent(long id_indent) {
        this.id_indent = id_indent;
    }

    public double getPrice_indent() {
        return price_indent;
    }

    public void setPrice_indent(double price_indent) {
        this.price_indent = price_indent;
    }

    public Status getStatus() {
        return status;
    }

    public void setStatus(Status status) {
        this.status = status;
    }

    public List<Film> getFilm() {
        return film;
    }

    public void setFilm(List<Film> film) {
        this.film = film;
    }
}
```

Manufacturer

```
package com.example.kinoteatr.model;

import javax.persistence.*;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import java.util.Collection;

@Entity
@Table(name = "manufacturer")
public class Manufacturer {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id_manufacturer;

    @NotBlank(message = "Введите название")
    @Size(max = 50, message = "Длина названия должна быть не больше 50 символов")
    @Column(name = "name_genre")
    private String name_manufacturer;

    @Min(value = 1900, message = "Значение должно быть больше 1900")
    @Max(value = 2023, message = "Значение должно быть меньше 2023")
    private int year_manufacturer;

    @ManyToOne
    @JoinColumn(name = "id_country") // Указывает на столбец, который связывает сущности
    private Country country;

    @OneToMany(mappedBy = "manufacturer", fetch = FetchType.LAZY)
    private Collection<Film> film;

    public Manufacturer(long id_manufacturer, String name_manufacturer, int year_manufacturer, Country country, Collection<Film>
film) {

        this.id_manufacturer = id_manufacturer;

        this.name_manufacturer = name_manufacturer;

        this.year_manufacturer = year_manufacturer;
```

```
        this.country = country;

        this.film = film;
    }

    public Manufacturer() {

    }

    public long getId_manufacturer() {

        return id_manufacturer;
    }

    public void setId_manufacturer(long id_manufacturer) {

        this.id_manufacturer = id_manufacturer;
    }

    public String getName_manufacturer() {

        return name_manufacturer;
    }

    public void setName_manufacturer(String name_manufacturer) {

        this.name_manufacturer = name_manufacturer;
    }

    public int getYear_manufacturer() {

        return year_manufacturer;
    }

    public void setYear_manufacturer(int year_manufacturer) {

        this.year_manufacturer = year_manufacturer;
    }

    public Country getCountry() {

        return country;
    }

    public void setCountry(Country country) {
```



```

        this.country = country;
    }

    public Collection<Film> getFilm() {

        return film;
    }

    public void setFilm(Collection<Film> film) {

        this.film = film;
    }
}

```

ModelUser

```

package com.example.kinoteatr.model;

import javax.persistence.*;
import java.util.Set;

@Entity

public class ModelUser {

    public ModelUser(){ }

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long ID_User;

    private String username;

    private String password;

    private boolean active;

    @ElementCollection(targetClass = RoleEnum.class, fetch = FetchType.EAGER)

    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))

    @Enumerated(EnumType.STRING)

    private Set<RoleEnum> roles;

    public Long getID_User() {

        return ID_User;
    }
}

```

```
public void setID_User(Long ID_User) {  
    this.ID_User = ID_User;  
}
```

```
public String getUsername() {  
    return username;  
}
```

```
public void setUsername(String username) {  
    this.username = username;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```
public boolean isActive() {  
    return active;  
}
```

```
public void setActive(boolean active) {  
    this.active = active;  
}
```

```
public Set<RoleEnum> getRoles() {  
    return roles;  
}
```

```
public void setRoles(Set<RoleEnum> roles) {  
    this.roles = roles;  
}
```

```
public void modelUser(String username, String password, boolean active, Set<RoleEnum> roles) {
```

```
        this.username = username;

        this.password = password;

        this.active = active;

        this.roles = roles;
    }
}
```

RoleEnum

```
package com.example.kinoteatr.model;

import org.springframework.security.core.GrantedAuthority;

public enum RoleEnum implements GrantedAuthority {

    USER,

    ADMIN,

    CASHIER;

    @Override

    public String getAuthority()

    {

        return name();

    }

}
```

Status

```
package com.example.kinoteatr.model;

import javax.persistence.*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

import java.util.Collection;

@Entity

@Table(name = "status")

public class Status {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

    private long id_status;

    @NotBlank(message = "Введите название")
```

```
@Size(max = 50, message = "Длина названия должна быть не больше 50 символов")
```

```
@Column(name = "name_status")
```

```
private String name_status;
```

```
@OneToMany(mappedBy = "status", fetch = FetchType.LAZY)
```

```
private Collection<Indent> order;
```

```
public Status(){ }
```

```
public Status(long id_status, String name_status, Collection<Indent> order) {
```

```
    this.id_status = id_status;
```

```
    this.name_status = name_status;
```

```
    this.order = order;
```

```
}
```

```
public long getId_status() {
```

```
    return id_status;
```

```
}
```

```
public void setId_status(long id_status) {
```

```
    this.id_status = id_status;
```

```
}
```

```
public String getName_status() {
```

```
    return name_status;
```

```
}
```

```
public void setName_status(String name_status) {
```

```
    this.name_status = name_status;
```

```
}
```

```
public Collection<Indent> getOrder() {
```

```
    return order;
```

```
}
```

```
public void setOrder(Collection<Indent> order) {
```

```
        this.order = order;
    }
}
```

UserRepo

```
package com.example.kinoteatr.repo;

import com.example.kinoteatr.model.ModelUser;
import org.springframework.data.repository.CrudRepository;

public interface UserRepo extends CrudRepository<ModelUser, Long> {

    ModelUser findByUsername(String username);

}
```

IndentRepo

```
package com.example.kinoteatr.repo;

import com.example.kinoteatr.model.Indent;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface IndentRepo extends JpaRepository<Indent, Long> {

    @Query("SELECT g FROM Indent g WHERE g.id_indent = :id")
    Indent findByName(@Param("id") long id);

}
```

FilmRepo

```
package com.example.kinoteatr.repo;

import com.example.kinoteatr.model.Film;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
```

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

import org.springframework.stereotype.Repository;

@Repository

public interface FilmRepo extends JpaRepository<Film, Long> {

    @Query("SELECT g FROM Film g WHERE g.name_film = :name_film")
    Film findByName(@Param("name_film") String name_film);

}
```

Edit-acess.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Изменение ролей</title>

<style>

/*Main*/

body {

    font-family: Arial, sans-serif;

    background-color: #000;

    color: #fff;

    text-align: center;

    margin: 0;

    display: flex;

    flex-direction: column;

    min-height: 100vh;

}

h1 {

    color: #ff6f00;

    background-color: #000;

    border: 1px solid #ff6f00;

    padding: 20px;

    border-radius: 5px;
```

```
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
  }
```

```
table {  
  width: 80%;  
  margin: 20px auto;  
  border-collapse: collapse;  
  background-color: #000;  
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
}
```

```
th, td {  
  padding: 10px;  
  text-align: left;  
}
```

```
th {  
  background-color: #ff6f00;  
  color: #000;  
}
```

```
tr:nth-child(even) {  
  background-color: #222;  
}
```

```
ul {  
  padding: 0;  
}
```

```
select {  
  width: 100%;  
  padding: 5px;  
}
```

```
button {  
  background-color: #ff6f00;  
  color: #000;
```

```
border: none;

padding: 8px 16px;

cursor: pointer;

}
```

```
button:hover {

background-color: #ff4500;

}
```

```
/*Header*/
```

```
.kinopoisk-header {

background-color: #000;

padding: 30px;

border-bottom: 2px solid #FF6F00;

display: flex;

justify-content: space-between;

align-items: center;

}
```

```
.logo img {

max-height: 50px;

}
```

```
.menu ul {

list-style-type: none;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

}
```

```
.menu li {

margin: 0 15px;

}
```

```
.menu a {
```



```
text-decoration: none;

color: #fff;

font-size: 19px;

transition: color 0.3s;

}
```

```
.menu a:hover {

color: #FF6F00;

}
```

```
/*Fotter*/
```

```
.kinopoisk-footer {

background-color: #FF6F00;

padding: 20px;

color: #fff;

text-align: center;

margin-top: auto;

}
```

```
.footer-content {

display: flex;

justify-content: space-between;

align-items: center;

}
```

```
.footer-logo img {

max-height: 50px;

}
```

```
.footer-links ul {

list-style-type: none;

margin: 0;

padding: 0;

display: flex;

}
```

```
.footer-links li {  
  
    margin: 0 15px;  
  
}
```

```
.footer-links a {  
  
    text-decoration: none;  
  
    color: #fff;  
  
    font-size: 19px;  
  
}
```

```
.copyright {  
  
    margin-top: 20px;  
  
    font-size: 14px;  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<header class="kinopoisk-header">
```

```
<div class="logo">
```

```

```

```
</div>
```

```
<nav class="menu">
```

```
<ul>
```

```
<li><a href="/login">Выйти из аккаунта</a></li>
```

```
<li><a href="/registration">Зарегистрироваться</a></li>
```

```
<li><a href="/">Главная</a></li>
```

```
</ul>
```

```
</nav>
```

```
</header>
```

```
<h1>Изменение ролей</h1>
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Пользователь</th>
```

```
<th>Роль</th>
```

```
<th>Изменение</th>
```

```

</tr>

</thead>

<tbody>

<tr th:each="user : ${users}">

    <td th:text="${user.username}"></td>

    <td>

        <ul>

            <h5 th:each="role : ${user.roles}" th:text="${role}"></h5>

        </ul>

    </td>

    <td>

        <form method="post" th:action="@{/edit-access/{userId}/edit(userId=${user.ID_User})}">

            <select name="roles">

                <option value="USER">Пользователь</option>

                <option value="ADMIN">Администратор</option>

                <option value="MODERATOR">Кассир</option>

            </select>

            <button type="submit">Изменить роль</button>

        </form>

    </td>

</tr>

</tbody>

</table>

<footer class="kinopoisk-footer">

<div class="footer-content">

<div class="footer-logo">

</div>

<div class="footer-links">

<ul>

    <li><a href="https://www.aviasales.ru/">Реклама</a></li>

    <li><a href="http://duma.gov.ru/contacts/citizen/">Контакты</a></li>

</ul>

</div>

</div>

<div class="copyright">

```

© 2023 KinoTeatr. Все права защищены.

</div>

</footer>

</body>

</html>

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>AUTH</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #000;

color: #fff;

display: flex;

flex-direction: column;

min-height: 100vh;

}

/*Header*/

.kinopoisk-header {

background-color: #000;

padding: 30px;

border-bottom: 2px solid #FF6F00;

display: flex;

justify-content: space-between;

align-items: center;

}

```
.logo img {  
    max-height: 50px;  
}
```

```
.menu ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    display: flex;  
    justify-content: center;  
}
```

```
.menu li {  
    margin: 0 15px;  
}
```

```
.menu a {  
    text-decoration: none;  
    color: #fff;  
    font-size: 19px;  
    transition: color 0.3s;  
}
```

```
.menu a:hover {  
    color: #FF6F00;  
}
```

```
/*Fotter*/
```

```
.kinopoisk-footer {  
    background-color: #FF6F00;  
    padding: 20px;  
    color: #fff;  
    text-align: center;  
    margin-top: auto;  
}
```

```
.footer-content {  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
}
```

```
.footer-logo img {  
    max-height: 50px;  
}
```

```
.footer-links ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    display: flex;  
}
```

```
.footer-links li {  
    margin: 0 15px;  
}
```

```
.footer-links a {  
    text-decoration: none;  
    color: #fff;  
    font-size: 19px;  
}
```

```
.copyright {  
    margin-top: 20px;  
    font-size: 14px;  
}
```

```
/*Main*/
```

```
.main-content {  
    background-color: #000000;  
    padding: 30px;
```

```
    color: #fff;
}
```

```
.featured-movies,
.latest-news {
    background-color: #FF7F00;
    padding: 20px;
    margin-bottom: 20px;
}
```

```
.featured-movies h2,
.latest-news h2 {
    color: #fff;
}
```

```
.featured-movies ul,
.latest-news p {
    color: #fff;
}
```

```
/* Закругление блоков */
```

```
.kinopoisk-header, .kinopoisk-footer, .featured-movies, .latest-news {
    border-radius: 10px; /* Добавляем закругление углов */
    overflow: hidden; /* Убираем возможность выхода контента за пределы блока с закруглением */
}
```

```
/* Выделение кнопок */
```

```
.menu a[href="/login"], .menu a[href="/registration"] {
    background-color: #FF6F00;
    padding: 10px 20px;
    border-radius: 5px;
    color: #000;
    font-weight: bold;
}
```

```
.menu a[href="/login"]:hover, .menu a[href="/registration"]:hover {
    background-color: #FF4500;
}
```

```
/* Дополнительные стили для описания фильмов */

.featured-movies ul {

    list-style-type: none;

    padding: 0;

}


.featured-movies li {

    display: flex;

    align-items: center;

    margin-bottom: 20px;

}


.featured-movies li img {

    margin-right: 20px;

    max-width: 150px; /* Установите желаемую ширину для изображения */

    border-radius: 10px;

    overflow: hidden;

}


.featured-movies li p {

    font-size: 18px;

    line-height: 1.5;

}


/*Trailers*/

.new-trailers {

    background-color: #FF7F00;

    padding: 20px;

    margin-bottom: 20px;

    border-radius: 10px;

    overflow: hidden;

}


.new-trailers h2 {

    color: #fff;
```



```
text-align: center;

margin-bottom: 20px;
}
```

```
.trailer-item {

text-align: center;

margin-bottom: 20px;
}
```

```
.trailer-item img {

max-width: 450px; /* Увеличиваем ширину изображения */

border-radius: 10px;

overflow: hidden;

margin-right: 10px;
}
```

```
.trailer-item a {

color: #fff;

text-decoration: none;

background-color: #FF6F00;

padding: 10px 20px;

border-radius: 5px;
}
```

```
.trailer-item a:hover {

background-color: #FF4500;
}
```

```
.n{

display: flex;

justify-content: space-between;
}
```

</style>

</head>

<body>

<header class="kinopoisk-header">

<div class="logo">


```
</div>

<nav class="menu">

  <ul>

    <li><a href="/authors">Режиссеры</a></li>

    <li><a href="/countries">Страны</a></li>

    <li><a href="/films">Фильмы</a></li>

    <li><a href="/genres">Жанры</a></li>

    <li><a href="/indents">Заказы</a></li>

    <li><a href="/manufacturers">Производители</a></li>

    <li><a href="/status">Статус</a></li>

    <li><a href="/login">Выйти из аккаунта</a></li>

    <li><a href="/registration">Зарегистрироваться</a></li>

  </ul>

</nav>

</header>
```

```
<main class="main-content">

  <h1>Добро пожаловать на КиноТеатр!</h1>

  <p>Здесь вы найдете лучшие фильмы и сериалы.</p>

  <section class="featured-movies">

    <h2>Рекомендуемые фильмы</h2>

    <ul>

      <li>

        <p>Зеленая миля - Пол Эджкомб — начальник блока смертников в тюрьме «Холодная гора», каждый из узников которого однажды проходит «зеленую милю» по пути к месту казни. Пол повидал много заключённых и надзирателей за время работы.

          Однако гигант Джон Коффи, обвинённый в страшном преступлении, стал одним из самых необычных обитателей блока.</p></li>

      <li>

        <p>Побег из Шоушенка - Бухгалтер Энди Дюфрейн обвинён в убийстве собственной жены и её любовника. Оказавшись в тюрьме под названием Шоушенк, он сталкивается с жестокостью и беззаконием, царящими по обе стороны решётки. Каждый, кто попадает в эти стены, становится их рабом до конца жизни.

          Но Энди, обладающий живым умом и доброй душой, находит подход как к заключённым, так и к охранникам, добиваясь их особого к себе расположения.</p>

      </li>

      <li>

        
```

<p>Форрест Гамп - Сидя на автобусной остановке, Форрест Гамп — не очень умный, но добрый и открытый парень — рассказывает случайным встречным историю своей необыкновенной жизни.

С самого малолетства парень страдал от заболевания ног, соседские мальчишки дразнили его, но в один прекрасный день Форрест открыл в себе невероятные способности к бегу.

Подруга детства Джени всегда его поддерживала и защищала, но вскоре дороги их разошлись.</p>

</section>

<section class="latest-news">

<h2>Последние новости</h2>

<p>Одна из главных новостей — запуск сериала по известному циклу фантастических романов Сергея Лукьяненко «Дозоры».

Подробностей пока нет. Напомним, что в рамках этой франшизы ранее вышли

два полнометражных фильма во главе с Константином Хабенским — «Ночной Дозор» (2004) и «Дневной Дозор» (2006).</p>

<p>Еще один многосерийный проект «Студии Плюс» и «Кинопоиска» на основе книг в жанре фэнтези — сериал «Этерна».

В прошлом году вышел полнометражный фильм «Этерна: Часть первая», однако создатели экранизации решили полностью перезапустить эту киновселенную в формате сериала, у которого будет новый актерский состав и творческая команда.

К шоу уже прикреплен режиссер Сергей Трофимов («Нулевой пациент»). Премьера намечена на 2024 год.</p>

<p>Третий крупный фэнтезийный проект «Кинопоиска» — сериал «Иные», события которого будут иметь место в альтернативной Европе 1930-х годов.

Главная героиня — владеющая суперспособностями женщина по имени Аня,

которая противостоит спецслужбам и таинственному злодею в лице Нойманна. Премьера состоится в 2024 году.</p>

</section>

<section class="new-trailers">

<h2>Новые трейлеры</h2>

<div class="n">

<div class="trailer-item">

Смотреть трейлер

</div>

<div class="trailer-item">

Смотреть трейлер

</div>

<div class="trailer-item">

```
        <br><br>

        <a href="https://youtu.be/v-2V4pQIN8U" target="_blank">Смотреть трейлер</a>

    </div>

</div>

</section>

</main>

<footer class="kinopoisk-footer">

    <div class="footer-content">

        <div class="footer-logo">

        </div>

        <div class="footer-links">

            <ul>

                <li><a href="/edit-access">Редактирование доступа</a></li>

                <li><a href="https://www.aviasales.ru/">Реклама</a></li>

                <li><a href="http://duma.gov.ru/contacts/citizen/">Контакты</a></li>

            </ul>

        </div>

    </div>

    <div class="copyright">

        &copy; 2023 KinoTeatr. Все права защищены.

    </div>

</footer>

</body>

</html>
```

login.html

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>AUTH</title>

    <style>
```

```
/*Main*/

body {

    font-family: Arial, sans-serif;

    background-color: #f0f0f0; /* Светло-серый цвет фона */

    text-align: center;

    padding: 50px;

    color: #333; /* Темно-серый цвет текста */

}


#login-container {

    background-color: #ffffff; /* Белый цвет формы */

    border: 1px solid #ccc;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.6); /* Увеличил силу тени */

    padding: 20px;

    max-width: 400px;

    margin: 0 auto;

    animation: flicker 1s infinite; /* Уменьшил интервал анимации */

    animation-fill-mode: both;

}


.login-header {

    font-size: 24px;

    margin-bottom: 20px;

    color: #FF4900; /* Оранжевый цвет заголовка */

}


.form-group {

    margin-bottom: 10px;

    text-align: left;

}


.form-group label {

    display: block;

    margin-bottom: 5px;

    color: #FF4900; /* Оранжевый цвет текста лейблов */

}
```

```
.form-group input {  
    width: calc(100% - 22px);  
    padding: 10px;  
    border: 1px solid #ccc;  
    border-radius: 3px;  
}
```

```
.form-group input[type="submit"] {  
    background-color: #FF7640;  
    color: #fff;  
    border: none;  
    border-radius: 3px;  
    padding: 10px 20px;  
    cursor: pointer;  
    width: auto;  
}
```

```
.form-group input[type="submit"]:hover {  
    background-color: #FF9B73;  
}
```

```
.error-message {  
    color: #ff0000;  
}
```

```
.registration-link {  
    color: #007bff;  
    text-decoration: none;  
    display: block;  
    margin-top: 20px;  
}
```

```
.registration-link:hover {  
    color: #0056b3;  
}
```

```
@keyframes flicker {

  0% {

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

  }

  50% {

    box-shadow: 0 0 15px rgba(0, 0, 0, 0.6); /* Увеличил силу тени */

  }

  100% {

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

  }

}
```

```
/* Анимация загрузки */
```

```
@keyframes slideIn {

  from {

    transform: translateY(100%);

    opacity: 0;

  }

  to {

    transform: translateY(0);

    opacity: 1;

  }

}
```

```
.slide-in {

  animation: slideIn 1s forwards;

  animation-delay: 0.5s;

}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div id="login-container" class="slide-in">
```

```
<h1 class="login-header">Авторизация</h1>
```

```
<div th:if="{param.error}" class="error-message">
```

```
    Неверное имя пользователя или пароль.
```

```
</div>
```

```
<form th:action="@{/login}" method="post">
```

```
<div class="form-group"><label>Логин: <input type="text" name="username"/> </label></div>

<div class="form-group"><label>Пароль: <input type="password" name="password"/> </label></div>

<div class="form-group"><input type="submit" value="Авторизация"/></div>

</form>

<a href="/registration" class="registration-link">Нет аккаунта? Зарегистрироваться</a>

</div>

</body>

</html>
```

regis.html

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"

      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Регистрация</title>

<style>

body {

    font-family: Arial, sans-serif;

    background-color: #f0f0f0;

    text-align: center;

    padding: 50px;

}

.registration-container {

    background-color: #ffffff; /* Белый цвет формы */

    border: 1px solid #ccc;

    border-radius: 10px;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

    padding: 20px;

    width: 300px;

    margin: 0 auto;

    animation: slideIn 1s forwards; /* Добавленная анимация */

    animation-delay: 0.5s; /* Задержка анимации */

}
```



```
.registration-container h1 {  
  
    font-size: 24px;  
  
    margin-bottom: 20px;  
  
    color: #FF4900; /* Оранжевый цвет заголовка */  
  
}  
  
.registration-container label {  
  
    display: block;  
  
    margin-bottom: 10px;  
  
    color: #FF4900; /* Оранжевый цвет текста лейблов */  
  
}  
  
.registration-container input[type="text"],  
.registration-container input[type="password"] {  
  
    width: 95%;  
  
    padding: 10px;  
  
    margin-bottom: 10px;  
  
    border: 1px solid #ccc;  
  
    border-radius: 3px;  
  
}  
  
.registration-container input[type="submit"] {  
  
    background-color: #fd7540; /* Синий цвет кнопки */  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 3px;  
  
    padding: 10px 20px;  
  
    cursor: pointer;  
  
}  
  
.registration-container input[type="submit"]:hover {  
  
    background-color: #fd7540; /* Темно-синий цвет при наведении */  
  
}  
  
.error-message {  
  
    color: #ff0000;
```

```
}
```

```
.registration-link {  
  
  color: #007bff;  
  
  text-decoration: none;  
  
  display: block;  
  
  margin-top: 20px;  
  
}
```

```
.registration-link:hover {  
  
  color: #0056b3;  
  
}
```

```
#login-container {  
  
  background-color: #ffffff; /* Белый цвет формы */  
  
  border: 1px solid #ccc;  
  
  border-radius: 10px;  
  
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.6); /* Увеличил силу тени */  
  
  padding: 20px;  
  
  max-width: 400px;  
  
  margin: 0 auto;  
  
  animation: flicker 1s infinite; /* Уменьшил интервал анимации */  
  
  animation-fill-mode: both;  
  
}
```

```
@keyframes flicker {  
  
  0% {  
  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);  
  
  }  
  
  50% {  
  
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.6); /* Увеличил силу тени */  
  
  }  
  
  100% {  
  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);  
  
  }  
  
}
```

```

/* Анимация загрузки */

@keyframes slideIn {

  from {

    transform: translateY(100%);

    opacity: 0;

  }

  to {

    transform: translateY(0);

    opacity: 1;

  }

}

</style>

</head>

<body>

<div id="login-container" class="registration-container">

  <h1>Регистрация</h1>

  <div th:if="{ message}" class="error-message">

    Пользователь уже существует

  </div>

  <form th:action="@{/registration}" method="post">

    <div><label>Логин: <input type="text" name="username"/> </label></div>

    <div><label>Пароль: <input type="password" name="password"/> </label></div>

    <div><input type="submit" value="Регистрация"/></div>

    <a href="/login" class="registration-link">Уже есть аккаунт? Авторизоваться.</a>

  </form>

</div>

</body>

</html>

```

allStatus.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

  <meta charset="utf-8">

  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <title>Статусы</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">

```

```
<!-- Подключение стилей -->
```

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
<style>
```

```
/* Новые стили */
```

```
body {
```

```
    background-color: #000;
```

```
    color: #fff;
```

```
}
```

```
table {
```

```
    border-collapse: collapse;
```

```
    width: 100%;
```

```
    background-color: #111; /* Черный с оттенком */
```

```
    color: #fff;
```

```
}
```

```
table, th, td {
```

```
    border: 1px solid #333; /* Черный с оттенком */
```

```
}
```

```
th, td {
```

```
    padding: 10px;
```

```
    text-align: center;
```

```
}
```

```
.edit-button, .delete-button, .add-button, .back-button {
```

```
    text-decoration: none;
```

```
    padding: 5px 10px;
```

```
    margin: 2px;
```

```
    border-radius: 5px;
```

```
}
```

```
.edit-button {
```

```
    border: 1px solid #ff5722; /* Оранжевый */
```

```
    background-color: #ff5722; /* Оранжевый */
```

```
}
```

```
.edit-button:hover {
```

```
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */
```

```
}
```

```
.delete-button {
```

```
    border: 1px solid #f44336; /* Красный */
```

```
    background-color: #f44336; /* Красный */
```

```
}
```

```
.delete-button:hover {
```

```
    background-color: #d32f2f; /* Красный с темным оттенком при наведении */
```

```
}
```

```
.add-button, .back-button {
```

```
    padding: 10px 20px;
```

```
}
```

```
.add-button {
```

```
    background-color: #ff5722; /* Оранжевый */
```

```
}
```

```
.add-button:hover {
```

```
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */
```

```
}
```

```
.back-button {
```

```
    background-color: #800000; /* Темно-красный */
```

```
}
```

```
.back-button:hover {
```

```
    background-color: #640000; /* Темно-красный с темным оттенком при наведении */
```

```
}
```

```
/* Добавим яркие оранжевые вставки */
```

```
.accent-orange-bg {
```

```
background-color: #ff6f00; /* Яркий оранжевый */

color: #fff;

padding: 5px 10px;

border-radius: 3px;

}
```

```
/* Стили заголовков и текста */
```

```
h2, p {

    color: #ff5722; /* Оранжевый */

}
```

```
/* Остальные стили остаются без изменений */
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div th:switch="{status}" class="container my-5">
```

```
<div class="row">
```

```
<div class="col-md-8 mx-auto">
```

```
<div th:case="*">
```

```
<h2 class="my-5 text-center">Статусы</h2>
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Название</th>
```

```
<th>Изменение</th>
```

```
<th>Удаление</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
<tr th:each="status : {status}">
```

```
<td th:text="{status.name_status}"></td>
```

```
<td>
```

```
<a th:href="@{/status/editStatus/{id}(id={status.id_status})}" class="edit-button">Изменить</a>
```

```
</td>
```

```
<td>
```

```

        <a th:href="@{/status/deleteStatus/{id}(id=${status.id_status})}" class="delete-button">Удалить</a>

    </td>

</tr>

</tbody>

</table>

</div>

<p class="my-5 text-center">

    <a href="/status/addStatus" class="add-button">Добавить статус</a>

</p>

<p class="my-5 text-center">

    <a href="/" class="back-button">Назад</a>

</p>

</div>

</div>

</div>

</body>

</html>

```

addStatus.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление статуса</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #111;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

```

```
margin-top: 20px;

color: #ff5722;

}
```

```
.container {

  max-width: 400px;

  margin: 0 auto;

  padding: 20px;

  background-color: #222;

  border: 1px solid #333;

  border-radius: 5px;

  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}
```

```
.form-group {

  margin-bottom: 20px;

}
```

```
label {

  font-weight: bold;

  display: block;

  margin-bottom: 5px;

  color: #ff5722;

}
```

```
input[type="text"] {

  width: 95%;

  padding: 10px;

  border: 1px solid #333;

  border-radius: 5px;

  font-size: 16px;

  background-color: #333;

  color: #fff;

}
```

```
.error-message {

  color: #d9534f;

}
```



```
font-size: 14px;  
}
```

```
.submit-button {  
    background-color: #ff5722;  
    color: #fff;  
    border: none;  
    border-radius: 5px;  
    padding: 10px 20px;  
    font-size: 16px;  
    cursor: pointer;  
}
```

```
.submit-button:hover {  
    background-color: #f4511e;  
}
```

```
.custom-select {  
    width: 101%;  
    padding: 10px;  
    border: 1px solid #333;  
    border-radius: 5px;  
    font-size: 16px;  
    background-color: #333;  
    color: #fff;  
    cursor: pointer;  
}
```

```
.custom-select option {  
    padding: 10px;  
    background-color: #333;  
    color: #fff;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```

<h2>Создание статуса</h2>

<div class="container">

  <form action="/status/addStatus" th:object="${ status}" method="post">

    <div class="form-group">

      <label for="name_status">Название</label>

      <input type="text" th:field="*{name_status}" id="name_status" placeholder="Название">

      <span class="error-message" th:if="${#fields.hasErrors('name_status')}" th:errors="*{name_status}"></span>

    </div>

    <div>

      <input type="submit" class="submit-button" value="Создать статус">

    </div>

  </form>

</div>

</div>

</body>

</html>

```

editStatus.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

  <meta charset="utf-8">

  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <title>Изменение статуса</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <style>

    body {

      font-family: Arial, sans-serif;

      background-color: #000;

      color: #fff;

      margin: 0;

      padding: 0;

    }

    h2 {

      text-align: center;

      margin-top: 20px;

```

```
    color: #ff5722;
}
```

```
.container {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    background-color: #111;
    border: 1px solid #333;
    border-radius: 5px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
```

```
.form-group {
    margin-bottom: 20px;
}
```

```
label {
    font-weight: bold;
    display: block;
    margin-bottom: 5px;
    color: #ff5722;
}
```

```
input[type="text"] {
    width: 95%;
    padding: 10px;
    border: 1px solid #333;
    border-radius: 5px;
    font-size: 16px;
    background-color: #333;
    color: #fff;
}
```

```
.error-message {
    color: #d9534f;
    font-size: 14px;
}
```

```
}
```

```
.submit-button {  
  
    background-color: #ff5722;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    padding: 10px 20px;  
  
    font-size: 16px;  
  
    cursor: pointer;  
  
}
```

```
.submit-button:hover {  
  
    background-color: #f4511e;  
  
}
```

```
.custom-select {  
  
    width: 101%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
    cursor: pointer;  
  
}
```

```
.custom-select option {  
  
    padding: 10px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Изменение статуса</h2>
```

```

<div class="container">

    <form th:action="@{/status/editStatus/{id}(id=${status.id_status})}" th:object="${status}" method="post">

        <div class="form-group">

            <label for="name_status">Название</label>

            <input type="text" th:field="*{name_status}" id="name_status" placeholder="Название">

            <span class="error-message" th:if="${#fields.hasErrors('name_status')}" th:errors="*{name_status}"></span>

        </div>

        <div>

            <input type="submit" class="submit-button" value="Сохранить статус">

        </div>

    </form>

</div>

</div>

</body>

</html>

```

allManufactures.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Производители</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Подключение стилей -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <style>

        /* Новые стили */

        body {

            background-color: #000;

            color: #fff;

        }

        table {

            border-collapse: collapse;

```

```
width: 100%;

background-color: #111; /* Черный с оттенком */

color: #fff;

}


table, th, td {

    border: 1px solid #333; /* Черный с оттенком */

}


th, td {

    padding: 10px;

    text-align: center;

}


.edit-button, .delete-button, .add-button, .back-button {

    text-decoration: none;

    padding: 5px 10px;

    margin: 2px;

    border-radius: 5px;

}


.edit-button {

    border: 1px solid #ff5722; /* Оранжевый */

    background-color: #ff5722; /* Оранжевый */

}


.edit-button:hover {

    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */

}


.delete-button {

    border: 1px solid #f44336; /* Красный */

    background-color: #f44336; /* Красный */

}


.delete-button:hover {

    background-color: #d32f2f; /* Красный с темным оттенком при наведении */

}
```

```
}
```

```
.add-button, .back-button {  
    padding: 10px 20px;  
}
```

```
.add-button {  
    background-color: #ff5722; /* Оранжевый */  
}
```

```
.add-button:hover {  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
}
```

```
.back-button {  
    background-color: #800000; /* Темно-красный */  
}
```

```
.back-button:hover {  
    background-color: #640000; /* Темно-красный с темным оттенком при наведении */  
}
```

```
/* Добавим яркие оранжевые вставки */
```

```
.accent-orange-bg {  
    background-color: #ff6f00; /* Яркий оранжевый */  
    color: #fff;  
    padding: 5px 10px;  
    border-radius: 3px;  
}
```

```
/* Стили заголовков и текста */
```

```
h2, p {  
    color: #ff5722; /* Оранжевый */  
}
```

```
/* Остальные стили остаются без изменений */
```

```

</style>

</head>

<body>

<div th:switch="{ manufacturers}" class="container my-5">

    <div class="row">

        <div class="col-md-8 mx-auto">

            <div th:case="*">

                <h2 class="my-5 text-center">Производители</h2>

                <table>

                    <thead>

                        <tr>

                            <th>Название</th>

                            <th>Год основания</th>

                            <th>Страна</th>

                            <th>Изменение</th>

                            <th>Удаление</th>

                        </tr>

                    </thead>

                    <tbody>

                        <tr th:each="manufacturer : { manufacturers}">

                            <td th:text="{ manufacturer.name_manufacturer}"></td>

                            <td th:text="{ manufacturer.year_manufacturer}"></td>

                            <td th:text="{ manufacturer.country != null ? manufacturer.country.name_country : ""}"></td>

                            <td>

                                <a th:href="@{/manufacturers/editManufacturer/{id}(id={ manufacturer.id_manufacturer})}" class="edit-
button">Изменить</a>

                            </td>

                            <td>

                                <a th:href="@{/manufacturers/deleteManufacturer/{id}(id={ manufacturer.id_manufacturer})}" class="delete-
button">Удалить</a>

                            </td>

                        </tr>

                    </tbody>

                </table>

            </div>

            <p class="my-5 text-center">

```



```
        <a href="/manufacturers/addManufacturer" class="add-button">Добавить производителя</a>

    </p>

    <p class="my-5 text-center">

        <a href="/" class="back-button">Назад</a>

    </p>

</div>

</div>

</div>

</div>

</body>

</html>
```

addManufacturer.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление производителя</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #111;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

            margin-top: 20px;

            color: #ff5722;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;
```

```
padding: 20px;

background-color: #222;

border: 1px solid #333;

border-radius: 5px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
```

```
.form-group {

margin-bottom: 20px;
}
```

```
label {

font-weight: bold;

display: block;

margin-bottom: 5px;

color: #ff5722;
}
```

```
input[type="text"] {

width: 95%;

padding: 10px;

border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;
}
```

```
.error-message {

color: #d9534f;

font-size: 14px;
}
```

```
.submit-button {

background-color: #ff5722;

color: #fff;

border: none;
```

```
border-radius: 5px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

}
```

```
.submit-button:hover {

background-color: #f4511e;

}
```

```
.custom-select {

width: 101%;

padding: 10px;

border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;

cursor: pointer;

}
```

```
.custom-select option {

padding: 10px;

background-color: #333;

color: #fff;

}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Создание производителей</h2>
```

```
<div class="container">
```

```
<form action="/manufacturers/addManufacturer" th:object="${manufacturer}" method="post">
```

```
<div class="form-group">
```

```
<label for="name_manufacturer">Название</label>
```

```
<input type="text" th:field="*{name_manufacturer}" id="name_manufacturer" placeholder="Название">
```

```

        <span
            class="error-message"
            th:if="{#fields.hasErrors('name_manufacturer')}"
th:errors="*{name_manufacturer}"></span>

    </div>

    <div class="form-group">

        <label for="year_manufacturer">Год основания</label>

        <input type="text" th:field="*{year_manufacturer}" id="year_manufacturer" placeholder="Год основания">

        <span class="error-message" th:if="{#fields.hasErrors('year_manufacturer')}" th:errors="*{year_manufacturer}"></span>

    </div>

    <label for="country">Страна</label>

    <div class="form-group">

        <select th:field="*{country}" id="country" class="custom-select">

            <option value="">Выберите страну</option>

            <option
                th:each="countryItem
                :
                ${countries}"
                th:value="{countryItem.id_country}"
th:text="{countryItem.name_country}"></option>

            </select>

            <span class="error-message" th:if="{#fields.hasErrors('country')}" th:errors="*{country}"></span>

        </div>

    <div>

        <input type="submit" class="submit-button" value="Создать производителя">

    </div>

</form>

</div>

</div>

</body>

</html>

```

editManufactures.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Изменение производителя</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #000;

```

```
    color: #fff;

    margin: 0;

    padding: 0;
}
```

```
h2 {

    text-align: center;

    margin-top: 20px;

    color: #ff5722;
}
```

```
.container {

    max-width: 400px;

    margin: 0 auto;

    padding: 20px;

    background-color: #111;

    border: 1px solid #333;

    border-radius: 5px;

    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
```

```
.form-group {

    margin-bottom: 20px;
}
```

```
label {

    font-weight: bold;

    display: block;

    margin-bottom: 5px;

    color: #ff5722;
}
```

```
input[type="text"] {

    width: 95%;

    padding: 10px;

    border: 1px solid #333;

    border-radius: 5px;
}
```

```
font-size: 16px;

background-color: #333;

color: #fff;

}
```

```
.error-message {

    color: #d9534f;

    font-size: 14px;

}
```

```
.submit-button {

    background-color: #ff5722;

    color: #fff;

    border: none;

    border-radius: 5px;

    padding: 10px 20px;

    font-size: 16px;

    cursor: pointer;

}
```

```
.submit-button:hover {

    background-color: #f4511e;

}
```

```
.custom-select {

    width: 101%;

    padding: 10px;

    border: 1px solid #333;

    border-radius: 5px;

    font-size: 16px;

    background-color: #333;

    color: #fff;

    cursor: pointer;

}
```

```
.custom-select option {

    padding: 10px;

}
```

```

        background-color: #333;

        color: #fff;

    }

</style>

</head>

<body>

<div>

    <h2>Изменение производителя</h2>

    <div class="container">

        <form
                                th:action="@{/manufacturers/editManufacturer/{id}(id=${manufacturer.id_manufacturer})}"
th:object="${manufacturer}" method="post">

            <div class="form-group">

                <label for="name_manufacturer">Название</label>

                <input type="text" th:field="*{name_manufacturer}" id="name_manufacturer" placeholder="Название">

                <span
                                class="error-message"
                                th:if="${#fields.hasErrors('name_manufacturer')}"
th:errors="*{name_manufacturer}"></span>

            </div>

            <div class="form-group">

                <label for="year_manufacturer">Год основания</label>

                <input type="text" th:field="*{year_manufacturer}" id="year_manufacturer" placeholder="Год основания">

                <span class="error-message" th:if="${#fields.hasErrors('year_manufacturer')}" th:errors="*{year_manufacturer}"></span>

            </div>

            <label for="country">Страна</label>

            <div class="form-group">

                <select th:field="*{country}" id="country" class="custom-select">

                    <option value="">Выберите страну</option>

                    <option
                                th:each="countryItem
                                :
                                ${countries}"
                                th:value="${countryItem.id_country}"
th:text="${countryItem.name_country}"></option>

                </select>

                <span class="error-message" th:if="${#fields.hasErrors('country')}" th:errors="*{country}"></span>

            </div>

            <div>

                <input type="submit" class="submit-button" value="Сохранить производителя">

            </div>

        </form>

    </div>

</div>

</body>

```

```
</html>
```

allIndets.html

```
<!DOCTYPE html>
```

```
<html xmlns:th="http://www.thymeleaf.org">
```

```
<head>
```

```
    <meta charset="utf-8">
```

```
    <meta http-equiv="x-ua-compatible" content="ie=edge">
```

```
    <title>Заказы</title>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <!-- Подключение стилей -->
```

```
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
<style>
```

```
    /* Новые стили */
```

```
    body {
```

```
        background-color: #000;
```

```
        color: #fff;
```

```
    }
```

```
    table {
```

```
        border-collapse: collapse;
```

```
        width: 100%;
```

```
        background-color: #111; /* Черный с оттенком */
```

```
        color: #fff;
```

```
    }
```

```
    table, th, td {
```

```
        border: 1px solid #333; /* Черный с оттенком */
```

```
    }
```

```
    th, td {
```

```
        padding: 10px;
```

```
        text-align: center;
```

```
    }
```



```
.edit-button, .delete-button, .add-button, .back-button {  
  
    text-decoration: none;  
  
    padding: 5px 10px;  
  
    margin: 2px;  
  
    border-radius: 5px;  
  
}
```

```
.edit-button {  
  
    border: 1px solid #ff5722; /* Оранжевый */  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.edit-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.delete-button {  
  
    border: 1px solid #f44336; /* Красный */  
  
    background-color: #f44336; /* Красный */  
  
}
```

```
.delete-button:hover {  
  
    background-color: #d32f2f; /* Красный с темным оттенком при наведении */  
  
}
```

```
.add-button, .back-button {  
  
    padding: 10px 20px;  
  
}
```

```
.add-button {  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.add-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.back-button {

    background-color: #800000; /* Темно-красный */

}

.back-button:hover {

    background-color: #640000; /* Темно-красный с темным оттенком при наведении */

}

/* Добавим яркие оранжевые вставки */

.accent-orange-bg {

    background-color: #ff6f00; /* Яркий оранжевый */

    color: #fff;

    padding: 5px 10px;

    border-radius: 3px;

}

/* Стили заголовков и текста */

h2, p {

    color: #ff5722; /* Оранжевый */

}

/* Остальные стили остаются без изменений */
```

</style>

</head>

<body>

<div th:switch="{indents}" >

<div class="row">

<div class="col-md-8 mx-auto">

<div th:case="*" class="table-container">

<h2 class="my-5 text-center">Заказы</h2>

<table>

<thead>

<tr>

<th>Номер</th>

<th>Общая цена</th>

```

        <th>Пользователь</th>

        <th>Статус</th>

        <th>Фильмы</th>

        <th>Изменение</th>

        <th>Удаление</th>

    </tr>

</thead>

<tbody>

<tr th:each="indent : ${indents}">

    <td th:text="${indent.id_indent}"></td>

    <td th:text="${indent.price_indent}"></td>

    <td th:text="${indent.user != null ? indent.user.username : ""}"></td>

    <td th:text="${indent.status != null ? indent.status.name_status : ""}"></td>

    <td>

        <span th:each="film : ${indent.film}">

            <span th:text="${film.name_film}"></span><br> <!-- Добавлен перевод строки -->

        </span>

    </td>

    <td>

        <a th:href="@{/indents/editIndent/{id}(id=${indent.id_indent})}" class="edit-button">Изменить</a>

    </td>

    <td>

        <a th:href="@{/indents/deleteIndent/{id}(id=${indent.id_indent})}" class="delete-button">Удалить</a>

    </td>

</tr>

</tbody>

</table>

</div>

<p class="my-5 text-center">

    <a href="/indents/addIndent" class="add-button">Добавить заказ</a>

</p>

<p class="my-5 text-center">

    <a href="/indents/addFilmToIndent" class="add-button">Добавить фильм к заказу</a>

</p>

<p class="my-5 text-center">

    <a href="/" class="back-button">Назад</a>

</p>

```

</div>

</div>

</div>

</body>

</html>

addIndets.html

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<title>Добавление заказа</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body {

font-family: Arial, sans-serif;

background-color: #111;

color: #fff;

margin: 0;

padding: 0;

}

h2 {

text-align: center;

margin-top: 20px;

color: #ff5722;

}

.container {

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #222;

border: 1px solid #333;

border-radius: 5px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

```
}
```

```
.form-group {  
    margin-bottom: 20px;  
}
```

```
label {  
    font-weight: bold;  
    display: block;  
    margin-bottom: 5px;  
    color: #ff5722;  
}
```

```
input[type="text"] {  
    width: 95%;  
    padding: 10px;  
    border: 1px solid #333;  
    border-radius: 5px;  
    font-size: 16px;  
    background-color: #333;  
    color: #fff;  
}
```

```
.error-message {  
    color: #d9534f;  
    font-size: 14px;  
}
```

```
.submit-button {  
    background-color: #ff5722;  
    color: #fff;  
    border: none;  
    border-radius: 5px;  
    padding: 10px 20px;  
    font-size: 16px;  
    cursor: pointer;  
}
```

```
.submit-button:hover {  
    background-color: #f4511e;  
}
```

```
.custom-select {  
    width: 101%;  
    padding: 10px;  
    border: 1px solid #333;  
    border-radius: 5px;  
    font-size: 16px;  
    background-color: #333;  
    color: #fff;  
    cursor: pointer;  
}
```

```
.custom-select option {  
    padding: 10px;  
    background-color: #333;  
    color: #fff;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Создание заказа</h2>
```

```
<div class="container">
```

```
<form action="/indents/addIndent" th:object="${indent}" method="post">
```

```
<div class="form-group">
```

```
<label for="price_indent">Общая стоимость заказа</label>
```

```
<input type="text" th:field="*{price_indent}" id="price_indent" placeholder="Общая стоимость заказа">
```

```
<span class="error-message" th:if="${#fields.hasErrors('price_indent')}" th:errors="*{price_indent}"></span>
```

```
</div>
```

```
<label for="user">Пользователь</label>
```

```
<div class="form-group">
```

```
<select th:field="*{user}" id="user" class="custom-select">
```

```
<option value="">Выберите пользователя</option>
```

```

        <option th:each="userItem : ${users}" th:value="${userItem.getID_User()}" th:text="${userItem.username}"></option>

    </select>

    <span class="error-message" th:if="${#fields.hasErrors('user')}" th:errors="*{user}"></span>

</div>

<label for="status">Статус</label>

<div class="form-group">

    <select th:field="*{status}" id="status" class="custom-select">

        <option value="">Выберите статус</option>

        <option
            th:each="statusItem
            :
            ${statuses}"
            th:value="${statusItem.id_status}"
            th:text="${statusItem.name_status}"></option>

    </select>

    <span class="error-message" th:if="${#fields.hasErrors('status')}" th:errors="*{status}"></span>

</div>

<div>

    <input type="submit" class="submit-button" value="Создать заказ">

</div>

</form>

</div>

</div>

</body>

</html>

```

editIndets.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Изменение заказа</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #000;

            color: #fff;

            margin: 0;

            padding: 0;

        }
    
```

```
h2 {  
  
  text-align: center;  
  
  margin-top: 20px;  
  
  color: #ff5722;  
  
}
```

```
.container {  
  
  max-width: 400px;  
  
  margin: 0 auto;  
  
  padding: 20px;  
  
  background-color: #111;  
  
  border: 1px solid #333;  
  
  border-radius: 5px;  
  
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
  
}
```

```
.form-group {  
  
  margin-bottom: 20px;  
  
}
```

```
label {  
  
  font-weight: bold;  
  
  display: block;  
  
  margin-bottom: 5px;  
  
  color: #ff5722;  
  
}
```

```
input[type="text"] {  
  
  width: 95%;  
  
  padding: 10px;  
  
  border: 1px solid #333;  
  
  border-radius: 5px;  
  
  font-size: 16px;  
  
  background-color: #333;  
  
  color: #fff;  
  
}
```



```
.error-message {  
  color: #d9534f;  
  font-size: 14px;  
}
```

```
.submit-button {  
  background-color: #ff5722;  
  color: #fff;  
  border: none;  
  border-radius: 5px;  
  padding: 10px 20px;  
  font-size: 16px;  
  cursor: pointer;  
}
```

```
.submit-button:hover {  
  background-color: #f4511e;  
}
```

```
.custom-select {  
  width: 101%;  
  padding: 10px;  
  border: 1px solid #333;  
  border-radius: 5px;  
  font-size: 16px;  
  background-color: #333;  
  color: #fff;  
  cursor: pointer;  
}
```

```
.custom-select option {  
  padding: 10px;  
  background-color: #333;  
  color: #fff;  
}
```

</style>

```

</head>

<body>

<div>

  <h2>Изменение заказа</h2>

  <div class="container">

    <form th:action="@{/indents/editIndent/{id}(id=${indent.id_indent})}" th:object="${indent}" method="post">

      <div class="form-group">

        <label for="price_indent">Общая стоимость заказа</label>

        <input type="text" th:field="*{price_indent}" id="price_indent" placeholder="Общая стоимость заказа">

        <span class="error-message" th:if="${#fields.hasErrors('price_indent')}" th:errors="*{price_indent}"></span>

      </div>

      <label for="user">Пользователь</label>

      <div class="form-group">

        <select th:field="*{user}" id="user" class="custom-select">

          <option value="">Выберите пользователя</option>

          <option th:each="userItem : ${users}" th:value="${userItem.getID_User()}" th:text="${userItem.username}"></option>

        </select>

        <span class="error-message" th:if="${#fields.hasErrors('user')}" th:errors="*{user}"></span>

      </div>

      <label for="status">Статус</label>

      <div class="form-group">

        <select th:field="*{status}" id="status" class="custom-select">

          <option value="">Выберите статус</option>

          <option
            th:each="statusItem
              :
            ${statuses}"
            th:value="${statusItem.id_status}"
            th:text="${statusItem.name_status}"></option>

        </select>

        <span class="error-message" th:if="${#fields.hasErrors('status')}" th:errors="*{status}"></span>

      </div>

      <div>

        <input type="submit" class="submit-button" value="Сохранить заказ">

      </div>

    </form>

  </div>

</div>

</body>

</html>

```

addFilmToIndent.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

  <meta charset="utf-8">

  <title>Добавить фильм к заказу</title>

  <style>

    body {

      font-family: Arial, sans-serif;

      background-color: #000;

      color: #fff;

      margin: 0;

      padding: 0;

    }

    h1 {

      text-align: center;

      margin-top: 20px;

      color: #ff5722;

    }

    form {

      background-color: #222;

      border-radius: 5px;

      box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

      padding: 20px;

      width: 300px;

      margin: 0 auto;

    }

    h2 {

      text-align: center;

      color: #ff5722;

    }

    select {

      width: 100%;

      padding: 10px;

      margin-bottom: 10px;

      background-color: #333;

      color: #fff;

      border: 1px solid #333;
```

```

        border-radius: 5px;
    }

    button {

        background-color: #ff5722;

        color: #fff;

        border: none;

        padding: 10px 20px;

        border-radius: 5px;

        cursor: pointer;

    }

    button:hover {

        background-color: #f4511e;

    }

    a {

        display: block;

        text-align: center;

        margin-top: 20px;

        color: #ff5722;

    }

</style>

</head>

<body>

<h1>Добавить фильм к заказу</h1>

<form th:action="'${'/indents/addFilmToIndent'}" method="post">

    <label>

        <select name="film">

            <option value="">Выберите фильм</option>

            <th:block th:each="element : ${ film} ">

                <option th:text="${ element.name_film}" th:value="${ element.name_film} "></option>

            </th:block>

        </select>

    </label>

    <label>

        <select name="indent">

            <option value="">Выберите заказ</option>

            <th:block th:each="element : ${ indent} ">

                <option th:text="${ element.id_indent}" th:value="${ element.id_indent} "></option>

```

```
        </th:block>

    </select>

</label>

<br>

<div style="text-align: center;">

    <button type="submit">Добавить фильм к заказу</button>

</div>

</form>

<a href="/indents">Вернуться к заказам</a>

</body>

</html>
```

allGenres.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Жанры</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Подключение стилей -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <style>

        /* Новые стили */

        body {

            background-color: #000;

            color: #fff;

        }

        table {

            border-collapse: collapse;

            width: 100%;

            background-color: #111; /* Черный с оттенком */

            color: #fff;

        }
```

```
table, th, td {  
  
    border: 1px solid #333; /* Черный с оттенком */  
  
}
```

```
th, td {  
  
    padding: 10px;  
  
    text-align: center;  
  
}
```

```
.edit-button, .delete-button, .add-button, .back-button {  
  
    text-decoration: none;  
  
    padding: 5px 10px;  
  
    margin: 2px;  
  
    border-radius: 5px;  
  
}
```

```
.edit-button {  
  
    border: 1px solid #ff5722; /* Оранжевый */  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.edit-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.delete-button {  
  
    border: 1px solid #f44336; /* Красный */  
  
    background-color: #f44336; /* Красный */  
  
}
```

```
.delete-button:hover {  
  
    background-color: #d32f2f; /* Красный с темным оттенком при наведении */  
  
}
```

```
.add-button, .back-button {  
  
    padding: 10px 20px;
```

```
}
```

```
.add-button {  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.add-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.back-button {  
  
    background-color: #800000; /* Темно-красный */  
  
}
```

```
.back-button:hover {  
  
    background-color: #640000; /* Темно-красный с темным оттенком при наведении */  
  
}
```

```
/* Добавим яркие оранжевые вставки */  
  
.accent-orange-bg {  
  
    background-color: #ff6f00; /* Яркий оранжевый */  
  
    color: #fff;  
  
    padding: 5px 10px;  
  
    border-radius: 3px;  
  
}
```

```
/* Стиль заголовков и текста */  
  
h2, p {  
  
    color: #ff5722; /* Оранжевый */  
  
}
```

```
/* Остальные стили остаются без изменений */
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div th:switch="{ genres}" class="container my-5">
```

```
<div class="row">

  <div class="col-md-8 mx-auto">

    <div th:case="*">

      <h2 class="my-5 text-center">Жанры</h2>

      <table>

        <thead>

          <tr>

            <th>Название</th>

            <th>Изменение</th>

            <th>Удаление</th>

          </tr>

        </thead>

        <tbody>

          <tr th:each="genre : ${ genres }">

            <td th:text="${ genre.name_genre }"></td>

            <td>

              <a th:href="@{/genres/editGenre/{id}(id=${ genre.id_genre })}" class="edit-button">Изменить</a>

            </td>

            <td>

              <a th:href="@{/genres/deleteGenre/{id}(id=${ genre.id_genre })}" class="delete-button">Удалить</a>

            </td>

          </tr>

        </tbody>

      </table>

    </div>

    <p class="my-5 text-center">

      <a href="/genres/addGenre" class="add-button">Добавить жанр</a>

    </p>

    <p class="my-5 text-center">

      <a href="/" class="back-button">Назад</a>

    </p>

  </div>

</div>

</div>

</body>

</html>
```


addGenre.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление жанра</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #111;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

            margin-top: 20px;

            color: #ff5722;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            background-color: #222;

            border: 1px solid #333;

            border-radius: 5px;

            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

        }

        .form-group {

            margin-bottom: 20px;

        }

    </style>

</head>

<body>

    <h2>Добавление жанра</h2>

    <div class="container">

        <div class="form-group">

            <input type="text" value="Новый жанр" />

            <input type="button" value="Добавить" />

        </div>

    </div>

</body>

</html>
```

```
label {  
  
    font-weight: bold;  
  
    display: block;  
  
    margin-bottom: 5px;  
  
    color: #ff5722;  
  
}
```

```
input[type="text"] {  
  
    width: 95%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
.error-message {  
  
    color: #d9534f;  
  
    font-size: 14px;  
  
}
```

```
.submit-button {  
  
    background-color: #ff5722;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    padding: 10px 20px;  
  
    font-size: 16px;  
  
    cursor: pointer;  
  
}
```

```
.submit-button:hover {  
  
    background-color: #f4511e;  
  
}
```

```
.custom-select {

    width: 101%;

    padding: 10px;

    border: 1px solid #333;

    border-radius: 5px;

    font-size: 16px;

    background-color: #333;

    color: #fff;

    cursor: pointer;

}

.custom-select option {

    padding: 10px;

    background-color: #333;

    color: #fff;

}

</style>

</head>

<body>

<div>

<h2>Создание жанра</h2>

<div class="container">

    <form action="/genres/addGenre" th:object="${ genre}" method="post">

        <div class="form-group">

            <label for="name_genre">Название</label>

            <input type="text" th:field="*{name_genre}" id="name_genre" placeholder="Название">

            <span class="error-message" th:if="${#fields.hasErrors('name_genre')}" th:errors="*{name_genre}"></span>

        </div>

        <div>

            <input type="submit" class="submit-button" value="Создать жанр">

        </div>

    </form>

</div>

</div>

</body>

</html>
```

editGenre.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Изменение жанра</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #000;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

            margin-top: 20px;

            color: #ff5722;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            background-color: #111;

            border: 1px solid #333;

            border-radius: 5px;

            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

        }

        .form-group {

            margin-bottom: 20px;

        }

    </style>

</head>

<body>

    <h2>Изменение жанра</h2>

    <div class="container">

        <div class="form-group">

            <input type="text" value="Имя жанра" />

            <input type="button" value="Изменить" />

        </div>

    </div>

</body>

</html>
```

```
label {  
  
    font-weight: bold;  
  
    display: block;  
  
    margin-bottom: 5px;  
  
    color: #ff5722;  
  
}
```

```
input[type="text"] {  
  
    width: 95%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
.error-message {  
  
    color: #d9534f;  
  
    font-size: 14px;  
  
}
```

```
.submit-button {  
  
    background-color: #ff5722;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    padding: 10px 20px;  
  
    font-size: 16px;  
  
    cursor: pointer;  
  
}
```

```
.submit-button:hover {  
  
    background-color: #f4511e;  
  
}
```

```
.custom-select {  
  
    width: 101%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
    cursor: pointer;  
  
}
```

```
.custom-select option {  
  
    padding: 10px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Изменение жанра</h2>
```

```
<div class="container">
```

```
<form th:action="@{/genres/editGenre/{id}(id=${genre.id_genre})}" th:object="${genre}" method="post">
```

```
<div class="form-group">
```

```
<label for="name_genre">Название</label>
```

```
<input type="text" th:field="*{name_genre}" id="name_genre" placeholder="Название">
```

```
<span class="error-message" th:if="${#fields.hasErrors('name_genre')}" th:errors="*{name_genre}"></span>
```

```
</div>
```

```
<div>
```

```
<input type="submit" class="submit-button" value="Сохранить жанр">
```

```
</div>
```

```
</form>
```

```
</div>
```

```
</div>
```

```
</body>
```

```
</html>
```

allFilms.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Фильма</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">


    <!-- Подключение стилей -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">


    <style>

        /* Новые стили */

        body {

            background-color: #000;

            color: #fff;

        }


        table {

            border-collapse: collapse;

            width: 100%;

            background-color: #111; /* Черный с оттенком */

            color: #fff;

        }


        table, th, td {

            border: 1px solid #333; /* Черный с оттенком */

        }


        th, td {

            padding: 10px;

            text-align: center;

        }


        .edit-button, .delete-button, .add-button, .back-button {
```

```
text-decoration: none;

padding: 5px 10px;

margin: 2px;

border-radius: 5px;

}
```

```
.edit-button {

border: 1px solid #ff5722; /* Оранжевый */

background-color: #ff5722; /* Оранжевый */

}
```

```
.edit-button:hover {

background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */

}
```

```
.delete-button {

border: 1px solid #f44336; /* Красный */

background-color: #f44336; /* Красный */

}
```

```
.delete-button:hover {

background-color: #d32f2f; /* Красный с темным оттенком при наведении */

}
```

```
.add-button, .back-button {

padding: 10px 20px;

}
```

```
.add-button {

background-color: #ff5722; /* Оранжевый */

}
```

```
.add-button:hover {

background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */

}
```

```
.back-button {
```



```
background-color: #800000; /* Темно-красный */
}
```

```
.back-button:hover {
    background-color: #640000; /* Темно-красный с темным оттенком при наведении */
}
```

```
/* Добавим яркие оранжевые вставки */
.accent-orange-bg {
    background-color: #ff6f00; /* Яркий оранжевый */
    color: #fff;
    padding: 5px 10px;
    border-radius: 3px;
}
```

```
/* Стили заголовков и текста */
h2, p {
    color: #ff5722; /* Оранжевый */
}
```

```
/* Остальные стили остаются без изменений */
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div th:switch="{films}" >
```

```
<div class="row">
```

```
<div class="col-md-8 mx-auto">
```

```
<div th:case="*" class="table-container">
```

```
<h2 class="my-5 text-center">Фильмы</h2>
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Название</th>
```

```
<th>Цена</th>
```

```

        <th>Год выпуска</th>

        <th>Время фильма</th>

        <th>Количество актеров в фильме</th>

        <th>Рейтинг</th>

        <th>Жанр</th>

        <th>Производитель</th>

        <th>Режиссер</th>

        <th>Изменение</th>

        <th>Удаление</th>

    </tr>

</thead>

<tbody>

<tr th:each="film : ${ films }">

    <td th:text="${ film.name_film }"></td>

    <td th:text="${ film.price_film }"></td>

    <td th:text="${ film.year_film }"></td>

    <td th:text="${ film.time_film }"></td>

    <td th:text="${ film.numbers_actors_film }"></td>

    <td th:text="${ film.rating_film }"></td>

    <td th:text="${ film.genre != null ? film.genre.name_genre : }"></td>

    <td th:text="${ film.manufacturer != null ? film.manufacturer.name_manufacturer : }"></td>

    <td th:text="${ film.author != null ? film.author.name_author : }"></td>

    <td>

        <a th:href="@{/films/editFilm/{id}(id=${ film.id_film })}" class="edit-button">Изменить</a>

    </td>

    <td>

        <a th:href="@{/films/deleteFilm/{id}(id=${ film.id_film })}" class="delete-button">Удалить</a>

    </td>

</tr>

</tbody>

</table>

</div>

<p class="my-5 text-center">

    <a href="/films/addFilm" class="add-button">Добавить фильм</a>

</p>

<p class="my-5 text-center">

    <a href="/" class="back-button">Назад</a>

```

```
        </p>

    </div>

</div>

</div>

</body>

</html>
```

addFilm.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление фильма</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #111;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

            margin-top: 20px;

            color: #ff5722;

        }

        .container {

            max-width: 400px;

            margin: 0 auto;

            padding: 20px;

            background-color: #222;

            border: 1px solid #333;

            border-radius: 5px;
```

```
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
}
```

```
.form-group {  
    margin-bottom: 20px;  
}
```

```
label {  
    font-weight: bold;  
    display: block;  
    margin-bottom: 5px;  
    color: #ff5722;  
}
```

```
input[type="text"] {  
    width: 95%;  
    padding: 10px;  
    border: 1px solid #333;  
    border-radius: 5px;  
    font-size: 16px;  
    background-color: #333;  
    color: #fff;  
}
```

```
.error-message {  
    color: #d9534f;  
    font-size: 14px;  
}
```

```
.submit-button {  
    background-color: #ff5722;  
    color: #fff;  
    border: none;  
    border-radius: 5px;  
    padding: 10px 20px;  
    font-size: 16px;  
    cursor: pointer;
```

```
}
```

```
.submit-button:hover {  
    background-color: #f4511e;  
}
```

```
.custom-select {  
    width: 101%;  
    padding: 10px;  
    border: 1px solid #333;  
    border-radius: 5px;  
    font-size: 16px;  
    background-color: #333;  
    color: #fff;  
    cursor: pointer;  
}
```

```
.custom-select option {  
    padding: 10px;  
    background-color: #333;  
    color: #fff;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Создание фильма</h2>
```

```
<div class="container">
```

```
<form action="/films/addFilm" th:object="${ film}" method="post">
```

```
<div class="form-group">
```

```
<label for="name_film">Название</label>
```

```
<input type="text" th:field="*{name_film}" id="name_film" placeholder="Название">
```

```
<span class="error-message" th:if="${#fields.hasErrors('name_film')}" th:errors="*{name_film}" "></span>
```

```
</div>
```

```
<div class="form-group">
```

```
<label for="price_film">Цена</label>
```

```
<input type="text" th:field="*{price_film}" id="price_film" placeholder="Цена">
```

```

        <span class="error-message" th:if="${#fields.hasErrors('price_film')}}" th:errors="*{price_film}"></span>

    </div>

    <div class="form-group">

        <label for="year_film">Год выпуска</label>

        <input type="text" th:field="*{year_film}" id="year_film" placeholder="Год выпуска">

        <span class="error-message" th:if="${#fields.hasErrors('year_film')}}" th:errors="*{year_film}"></span>

    </div>

    <div class="form-group">

        <label for="time_film">Время фильма</label>

        <input type="text" th:field="*{time_film}" id="time_film" placeholder="Время фильма">

        <span class="error-message" th:if="${#fields.hasErrors('time_film')}}" th:errors="*{time_film}"></span>

    </div>

    <div class="form-group">

        <label for="numbers_actors_film">Количество актеров в фильме</label>

        <input type="text" th:field="*{numbers_actors_film}" id="numbers_actors_film" placeholder="Количество актеров в
фильме">

        <span
                                class="error-message"
                                th:if="${#fields.hasErrors('numbers_actors_film')}}"
                                th:errors="*{numbers_actors_film}"></span>

    </div>

    <div class="form-group">

        <label for="rating_film">Рейтинг фильма</label>

        <input type="text" th:field="*{rating_film}" id="rating_film" placeholder="Рейтинг фильма">

        <span class="error-message" th:if="${#fields.hasErrors('rating_film')}}" th:errors="*{rating_film}"></span>

    </div>

    <label for="genre">Жанр</label>

    <div class="form-group">

        <select th:field="*{genre}" id="genre" class="custom-select">

            <option value="">Выберите жанр</option>

            <option
                                th:each="genreItem
                                :
                                ${genres}"
                                th:value="${genreItem.id_genre}"
                                th:text="${genreItem.name_genre}"></option>

        </select>

        <span class="error-message" th:if="${#fields.hasErrors('genre')}}" th:errors="*{genre}"></span>

    </div>

    <label for="manufacturer">Производитель</label>

    <div class="form-group">

        <select th:field="*{manufacturer}" id="manufacturer" class="custom-select">

            <option value="">Выберите производителя</option>

```

```

        <option      th:each="manufacturerItem      :      ${manufacturers}"      th:value="${manufacturerItem.id_manufacturer}"
th:text="${manufacturerItem.name_manufacturer}"></option>

    </select>

    <span class="error-message" th:if="${#fields.hasErrors('manufacturer')}" th:errors="*{manufacturer}"></span>

</div>

<label for="genre">Режиссер</label>

<div class="form-group">

    <select th:field="*{author}" id="author" class="custom-select">

        <option value="">Выберите режиссера</option>

        <option      th:each="authorItem      :      ${authors}"      th:value="${authorItem.id_author}"
th:text="${authorItem.name_author}"></option>

    </select>

    <span class="error-message" th:if="${#fields.hasErrors('author')}" th:errors="*{author}"></span>

</div>

<div>

    <input type="submit" class="submit-button" value="Создать фильм">

</div>

</form>

</div>

</div>

</body>

</html>

```

editFilm.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Изменение фильма</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<style>

    body {

        font-family: Arial, sans-serif;

        background-color: #000;

        color: #fff;

        margin: 0;
    }

```

```
padding: 0;  
}
```

```
h2 {  
  text-align: center;  
  margin-top: 20px;  
  color: #ff5722;  
}
```

```
.container {  
  max-width: 400px;  
  margin: 0 auto;  
  padding: 20px;  
  background-color: #111;  
  border: 1px solid #333;  
  border-radius: 5px;  
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
}
```

```
.form-group {  
  margin-bottom: 20px;  
}
```

```
label {  
  font-weight: bold;  
  display: block;  
  margin-bottom: 5px;  
  color: #ff5722;  
}
```

```
input[type="text"] {  
  width: 95%;  
  padding: 10px;  
  border: 1px solid #333;  
  border-radius: 5px;  
  font-size: 16px;  
  background-color: #333;
```



```
    color: #fff;
}
```

```
.error-message {
    color: #d9534f;
    font-size: 14px;
}
```

```
.submit-button {
    background-color: #ff5722;
    color: #fff;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}
```

```
.submit-button:hover {
    background-color: #f4511e;
}
```

```
.custom-select {
    width: 101%;
    padding: 10px;
    border: 1px solid #333;
    border-radius: 5px;
    font-size: 16px;
    background-color: #333;
    color: #fff;
    cursor: pointer;
}
```

```
.custom-select option {
    padding: 10px;
    background-color: #333;
    color: #fff;
}
```

```

    }

</style>

</head>

<body>

<div>

    <h2>Изменение фильма</h2>

    <div class="container">

        <form th:action="@{/films/editFilm/{id}(id=${film.id_film})}" th:object="${film}" method="post">

            <div class="form-group">

                <label for="name_film">Название</label>

                <input type="text" th:field="*{name_film}" id="name_film" placeholder="Название">

                <span class="error-message" th:if="${#fields.hasErrors('name_film')}" th:errors="*{name_film}"></span>

            </div>

            <div class="form-group">

                <label for="price_film">Цена</label>

                <input type="text" th:field="*{price_film}" id="price_film" placeholder="Цена">

                <span class="error-message" th:if="${#fields.hasErrors('price_film')}" th:errors="*{price_film}"></span>

            </div>

            <div class="form-group">

                <label for="year_film">Год выпуска</label>

                <input type="text" th:field="*{year_film}" id="year_film" placeholder="Год выпуска">

                <span class="error-message" th:if="${#fields.hasErrors('year_film')}" th:errors="*{year_film}"></span>

            </div>

            <div class="form-group">

                <label for="time_film">Время фильма</label>

                <input type="text" th:field="*{time_film}" id="time_film" placeholder="Время фильма">

                <span class="error-message" th:if="${#fields.hasErrors('time_film')}" th:errors="*{time_film}"></span>

            </div>

            <div class="form-group">

                <label for="numbers_actors_film">Количество актеров в фильме</label>

                <input type="text" th:field="*{numbers_actors_film}" id="numbers_actors_film" placeholder="Количество актеров в
фильме">

                <span
                                class="error-message"
                                th:if="${#fields.hasErrors('numbers_actors_film')}"
th:errors="*{numbers_actors_film}"></span>

            </div>

            <div class="form-group">

```

```
<label for="rating_film">Рейтинг фильма</label>

<input type="text" th:field="**{rating_film}" id="rating_film" placeholder="Рейтинг фильма">

<span class="error-message" th:if="{#fields.hasErrors('rating_film')}}" th:errors="**{rating_film}"></span>

</div>

<label for="genre">Жанр</label>

<div class="form-group">

  <select th:field="**{genre}" id="genre" class="custom-select">

    <option value="">Выберите жанр</option>

    <option
      th:each="genreItem"
      :
      ${genres}"
      th:value="{genreItem.id_genre}"
th:text="{genreItem.name_genre}"></option>

  </select>

  <span class="error-message" th:if="{#fields.hasErrors('genre')}}" th:errors="**{genre}"></span>

</div>

<label for="manufacturer">Производитель</label>

<div class="form-group">

  <select th:field="**{manufacturer}" id="manufacturer" class="custom-select">

    <option value="">Выберите производителя</option>

    <option
      th:each="manufacturerItem"
      :
      ${manufacturers}"
      th:value="{manufacturerItem.id_manufacturer}"
th:text="{manufacturerItem.name_manufacturer}"></option>

  </select>

  <span class="error-message" th:if="{#fields.hasErrors('manufacturer')}}" th:errors="**{manufacturer}"></span>

</div>

<label for="genre">Режиссер</label>

<div class="form-group">

  <select th:field="**{author}" id="author" class="custom-select">

    <option value="">Выберите режиссера</option>

    <option
      th:each="authorItem"
      :
      ${authors}"
      th:value="{authorItem.id_author}"
th:text="{authorItem.name_author}"></option>

  </select>

  <span class="error-message" th:if="{#fields.hasErrors('author')}}" th:errors="**{author}"></span>

</div>

<div>

  <input type="submit" class="submit-button" value="Сохранить фильм">

</div>

</form>

</div>

</div>

</body>
```

```
</html>
```

allCountries.html

```
<!DOCTYPE html>
```

```
<html xmlns:th="http://www.thymeleaf.org">
```

```
<head>
```

```
    <meta charset="utf-8">
```

```
    <meta http-equiv="x-ua-compatible" content="ie=edge">
```

```
    <title>Страны</title>
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <!-- Подключение стилей -->
```

```
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
<style>
```

```
    /* Новые стили */
```

```
    body {
```

```
        background-color: #000;
```

```
        color: #fff;
```

```
    }
```

```
    table {
```

```
        border-collapse: collapse;
```

```
        width: 100%;
```

```
        background-color: #111; /* Черный с оттенком */
```

```
        color: #fff;
```

```
    }
```

```
    table, th, td {
```

```
        border: 1px solid #333; /* Черный с оттенком */
```

```
    }
```

```
    th, td {
```

```
        padding: 10px;
```

```
        text-align: center;
```

```
    }
```

```
.edit-button, .delete-button, .add-button, .back-button {  
  
    text-decoration: none;  
  
    padding: 5px 10px;  
  
    margin: 2px;  
  
    border-radius: 5px;  
  
}
```

```
.edit-button {  
  
    border: 1px solid #ff5722; /* Оранжевый */  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.edit-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.delete-button {  
  
    border: 1px solid #f44336; /* Красный */  
  
    background-color: #f44336; /* Красный */  
  
}
```

```
.delete-button:hover {  
  
    background-color: #d32f2f; /* Красный с темным оттенком при наведении */  
  
}
```

```
.add-button, .back-button {  
  
    padding: 10px 20px;  
  
}
```

```
.add-button {  
  
    background-color: #ff5722; /* Оранжевый */  
  
}
```

```
.add-button:hover {  
  
    background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */  
  
}
```

```
.back-button {

    background-color: #800000; /* Темно-красный */

}

.back-button:hover {

    background-color: #640000; /* Темно-красный с темным оттенком при наведении */

}

/* Добавим яркие оранжевые вставки */

.accent-orange-bg {

    background-color: #ff6f00; /* Яркий оранжевый */

    color: #fff;

    padding: 5px 10px;

    border-radius: 3px;

}

/* Стили заголовков и текста */

h2, p {

    color: #ff5722; /* Оранжевый */

}

/* Остальные стили остаются без изменений */
```

</style>

</head>

<body>

<div th:switch="{countries}" class="container my-5">

<div class="row">

<div class="col-md-8 mx-auto">

<div th:case="*">

<h2 class="my-5 text-center">Страны</h2>

<table>

<thead>

<tr>

<th>Название</th>

<th>Изменение</th>

```

        <th>Удаление</th>

    </tr>

</thead>

<tbody>

<tr th:each="country : ${countries}">

    <td th:text="${country.name_country}"></td>

    <td>

        <a th:href="@{/countries/editCountry/{id}(id=${country.id_country})}" class="edit-button">Изменить</a>

    </td>

    <td>

        <a th:href="@{/countries/deleteCountry/{id}(id=${country.id_country})}" class="delete-button">Удалить</a>

    </td>

</tr>

</tbody>

</table>

</div>

<p class="my-5 text-center">

    <a href="/countries/addCountry" class="add-button">Добавить страну</a>

</p>

<p class="my-5 text-center">

    <a href="/" class="back-button">Назад</a>

</p>

</div>

</div>

</div>

</body>

</html>

```

addCountry.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление страны</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

```

```
body {

    font-family: Arial, sans-serif;

    background-color: #111;

    color: #fff;

    margin: 0;

    padding: 0;

}


h2 {

    text-align: center;

    margin-top: 20px;

    color: #ff5722;

}


.container {

    max-width: 400px;

    margin: 0 auto;

    padding: 20px;

    background-color: #222;

    border: 1px solid #333;

    border-radius: 5px;

    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}


.form-group {

    margin-bottom: 20px;

}


label {

    font-weight: bold;

    display: block;

    margin-bottom: 5px;

    color: #ff5722;

}


input[type="text"] {

    width: 95%;
```



```
padding: 10px;

border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;

}
```

```
.error-message {

    color: #d9534f;

    font-size: 14px;

}
```

```
.submit-button {

    background-color: #ff5722;

    color: #fff;

    border: none;

    border-radius: 5px;

    padding: 10px 20px;

    font-size: 16px;

    cursor: pointer;

}
```

```
.submit-button:hover {

    background-color: #f4511e;

}
```

```
.custom-select {

    width: 101%;

    padding: 10px;

    border: 1px solid #333;

    border-radius: 5px;

    font-size: 16px;

    background-color: #333;

    color: #fff;

    cursor: pointer;

}
```

```

        .custom-select option {

            padding: 10px;

            background-color: #333;

            color: #fff;

        }

    </style>

</head>

<body>

<div>

    <h2>Создание страны</h2>

    <div class="container">

        <form action="/countries/addCountry" th:object="${ country}" method="post">

            <div class="form-group">

                <label for="name_country">Название</label>

                <input type="text" th:field="*{ name_country}" id="name_country" placeholder="Название">

                <span class="error-message" th:if="${ #fields.hasErrors('name_country')}" th:errors="*{ name_country}"></span>

            </div>

            <div>

                <input type="submit" class="submit-button" value="Создать страну">

            </div>

        </form>

    </div>

</div>

</body>

</html>

```

editCountry.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Изменение страны</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

```

```
font-family: Arial, sans-serif;

background-color: #000;

color: #fff;

margin: 0;

padding: 0;
}


h2 {

text-align: center;

margin-top: 20px;

color: #ff5722;

}


.container {

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #111;

border: 1px solid #333;

border-radius: 5px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}


.form-group {

margin-bottom: 20px;

}


label {

font-weight: bold;

display: block;

margin-bottom: 5px;

color: #ff5722;

}


input[type="text"] {

width: 95%;

padding: 10px;
```

```
border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;

}
```

```
.error-message {

color: #d9534f;

font-size: 14px;

}
```

```
.submit-button {

background-color: #ff5722;

color: #fff;

border: none;

border-radius: 5px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

}
```

```
.submit-button:hover {

background-color: #f4511e;

}
```

```
.custom-select {

width: 101%;

padding: 10px;

border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;

cursor: pointer;

}
```

```

        .custom-select option {

            padding: 10px;

            background-color: #333;

            color: #fff;

        }

    </style>

</head>

<body>

<div>

    <h2>Изменение страны</h2>

    <div class="container">

        <form th:action="@{/countries/editCountry/{id}(id=${country.id_country})}" th:object="${country}" method="post">

            <div class="form-group">

                <label for="name_country">Название</label>

                <input type="text" th:field="*{name_country}" id="name_country" placeholder="Название">

                <span class="error-message" th:if="${#fields.hasErrors('name_country')}" th:errors="*{name_country}"></span>

            </div>

            <div>

                <input type="submit" class="submit-button" value="Сохранить страну">

            </div>

        </form>

    </div>

</div>

</body>

</html>

```

allAuthors.html

```

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Режиссеры</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Подключение стилей -->

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

```

```
<style>

/* Новые стили */

body {

    background-color: #000;

    color: #fff;

}


table {

    border-collapse: collapse;

    width: 100%;

    background-color: #111; /* Черный с оттенком */

    color: #fff;

}


table, th, td {

    border: 1px solid #333; /* Черный с оттенком */

}


th, td {

    padding: 10px;

    text-align: center;

}


.edit-button, .delete-button, .add-button, .back-button {

    text-decoration: none;

    padding: 5px 10px;

    margin: 2px;

    border-radius: 5px;

}


.edit-button {

    border: 1px solid #ff5722; /* Оранжевый */

    background-color: #ff5722; /* Оранжевый */

}


.edit-button:hover {
```

```
background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */
}

.delete-button {
  border: 1px solid #f44336; /* Красный */
  background-color: #f44336; /* Красный */
}

.delete-button:hover {
  background-color: #d32f2f; /* Красный с темным оттенком при наведении */
}

.add-button, .back-button {
  padding: 10px 20px;
}

.add-button {
  background-color: #ff5722; /* Оранжевый */
}

.add-button:hover {
  background-color: #f4511e; /* Оранжевый с темным оттенком при наведении */
}

.back-button {
  background-color: #800000; /* Темно-красный */
}

.back-button:hover {
  background-color: #640000; /* Темно-красный с темным оттенком при наведении */
}

/* Добавим яркие оранжевые вставки */
.accent-orange-bg {
  background-color: #ff6f00; /* Яркий оранжевый */
  color: #fff;
  padding: 5px 10px;
```

```
border-radius: 3px;

}
```

```
/* Стили заголовков и текста */

h2, p {

    color: #ff5722; /* Оранжевый */

}
```

```
/* Остальные стили остаются без изменений */
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div th:switch="{authors}" class="container my-5">
```

```
<div class="row">
```

```
<div class="col-md-8 mx-auto">
```

```
<div th:case="*">
```

```
<h2 class="my-5 text-center">Режиссеры</h2>
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Имя</th>
```

```
<th>Изменение</th>
```

```
<th>Удаление</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
<tr th:each="author : {authors}">
```

```
<td th:text="{author.name_author}"></td>
```

```
<td>
```

```
<a th:href="@{/authors/editAuthor/{id}(id={author.id_author})}" class="edit-button">Изменить</a>
```

```
</td>
```

```
<td>
```

```
<a th:href="@{/authors/deleteAuthor/{id}(id={author.id_author})}" class="delete-button">Удалить</a>
```

```
</td>
```

```
</tr>
```



```

        </tbody>

    </table>

</div>

<p class="my-5 text-center">

    <a href="/authors/addAuthor" class="add-button">Добавить режиссера</a>

</p>

<p class="my-5 text-center">

    <a href="/" class="back-button">Назад</a>

</p>

</div>

</div>

</div>

</body>

</html>
```

addAuthors.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

    <meta charset="utf-8">

    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <title>Добавление автора</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #111;

            color: #fff;

            margin: 0;

            padding: 0;

        }

        h2 {

            text-align: center;

            margin-top: 20px;

            color: #ff5722;

        }
```

```
.container {  
  
    max-width: 400px;  
  
    margin: 0 auto;  
  
    padding: 20px;  
  
    background-color: #222;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
  
}
```

```
.form-group {  
  
    margin-bottom: 20px;  
  
}
```

```
label {  
  
    font-weight: bold;  
  
    display: block;  
  
    margin-bottom: 5px;  
  
    color: #ff5722;  
  
}
```

```
input[type="text"] {  
  
    width: 95%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
.error-message {  
  
    color: #d9534f;  
  
    font-size: 14px;  
  
}
```

```
.submit-button {  
  
    background-color: #ff5722;  
  
    color: #fff;  
  
    border: none;  
  
    border-radius: 5px;  
  
    padding: 10px 20px;  
  
    font-size: 16px;  
  
    cursor: pointer;  
  
}
```

```
.submit-button:hover {  
  
    background-color: #f4511e;  
  
}
```

```
.custom-select {  
  
    width: 101%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
    cursor: pointer;  
  
}
```

```
.custom-select option {  
  
    padding: 10px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Создание режиссера</h2>
```

```
<div class="container">
```

```
<form action="/authors/addAuthor" th:object="{author}" method="post">
```

```
<div class="form-group">

  <label for="name_author">Имя режиссера</label>

  <input type="text" th:field="*{name_author}" id="name_author" placeholder="Имя режиссера">

  <span class="error-message" th:if="${#fields.hasErrors('name_author')}}" th:errors="*{name_author}"></span>

</div>

<div>

  <input type="submit" class="submit-button" value="Создать автора">

</div>

</form>

</div>

</div>

</body>

</html>
```

editAthors.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

  <meta charset="utf-8">

  <meta http-equiv="x-ua-compatible" content="ie=edge">

  <title>Изменение режиссера</title>

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <style>

    body {

      font-family: Arial, sans-serif;

      background-color: #000;

      color: #fff;

      margin: 0;

      padding: 0;

    }

    h2 {

      text-align: center;

      margin-top: 20px;

      color: #ff5722;

    }

  </style>

</head>

<body>

  <div>

    <div class="form-group">

      <label for="name_author">Имя режиссера</label>

      <input type="text" th:field="*{name_author}" id="name_author" placeholder="Имя режиссера">

      <span class="error-message" th:if="${#fields.hasErrors('name_author')}}" th:errors="*{name_author}"></span>

    </div>

    <div>

      <input type="submit" class="submit-button" value="Создать автора">

    </div>

  </div>

</body>

</html>
```

```
.container {  
  
    max-width: 400px;  
  
    margin: 0 auto;  
  
    padding: 20px;  
  
    background-color: #111;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  
  
}
```

```
.form-group {  
  
    margin-bottom: 20px;  
  
}
```

```
label {  
  
    font-weight: bold;  
  
    display: block;  
  
    margin-bottom: 5px;  
  
    color: #ff5722;  
  
}
```

```
input[type="text"] {  
  
    width: 95%;  
  
    padding: 10px;  
  
    border: 1px solid #333;  
  
    border-radius: 5px;  
  
    font-size: 16px;  
  
    background-color: #333;  
  
    color: #fff;  
  
}
```

```
.error-message {  
  
    color: #d9534f;  
  
    font-size: 14px;  
  
}
```

```
.submit-button {
```

```
background-color: #ff5722;

color: #fff;

border: none;

border-radius: 5px;

padding: 10px 20px;

font-size: 16px;

cursor: pointer;

}
```

```
.submit-button:hover {

background-color: #f4511e;

}
```

```
.custom-select {

width: 101%;

padding: 10px;

border: 1px solid #333;

border-radius: 5px;

font-size: 16px;

background-color: #333;

color: #fff;

cursor: pointer;

}
```

```
.custom-select option {

padding: 10px;

background-color: #333;

color: #fff;

}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h2>Изменение режиссера</h2>
```

```
<div class="container">
```

```
<form th:action="@{/authors/editAuthor/{id}}(id=${author.id_author})" th:object="${author}" method="post">
```

```
<div class="form-group">
```

```

        <label for="name_author">Имя</label>

        <input type="text" th:field="*{ name_author}" id="name_author" placeholder="Имя">

        <span class="error-message" th:if="{ #fields.hasErrors('name_author')}" th:errors="*{ name_author}"></span>

    </div>

    <div>

        <input type="submit" class="submit-button" value="Сохранить режиссера">

    </div>

</form>

</div>

</div>

</body>

</html>

```

Application.properties

```

spring.datasource.url=jdbc:postgresql://localhost:5432/kinodb7

spring.datasource.username=postgres

spring.datasource.password=1234

spring.datasource.driverClassName=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

```

```

server.port=8081

```

Pom.xml

```

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.5.4</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.example</groupId>

    <artifactId>KinoTeatr</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>KinoTeatr</name>

    <description>KinoTeatr</description>

    <properties>

```

```
<java.version>17</java.version>

</properties>

<dependencies>

  <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-thymeleaf</artifactId>

  </dependency>

  <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-web</artifactId>

  </dependency>


  <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-devtools</artifactId>

    <scope>runtime</scope>

    <optional>true</optional>

  </dependency>

  <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-test</artifactId>

    <scope>test</scope>

  </dependency>


  <dependency>

    <groupId>org.postgresql</groupId>

    <artifactId>postgresql</artifactId>

    <scope>runtime</scope>

  </dependency>

  <dependency>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-data-jpa</artifactId>

  </dependency>

  <dependency>

    <groupId>org.hibernate.validator</groupId>

    <artifactId>hibernate-validator</artifactId>

    <version>6.2.0.Final</version>

  </dependency>

</dependencies>
```



```
        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-security</artifactId>

    </dependency>

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-test</artifactId>

        <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-config</artifactId>

        <version>5.5.2</version>

    </dependency>

    <dependency>

        <groupId>org.springframework.security</groupId>

        <artifactId>spring-security-core</artifactId>

        <version>6.1.3</version>

    </dependency>
</dependencies>
```

```
<build>

    <plugins>

        <plugin>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-maven-plugin</artifactId>

        </plugin>

    </plugins>

</build>
```

```
</project>
```

Реализация программы

Создаем модели для таблиц бд.

```
29 usages
@Entity
@Table(name = "film")
public class Film {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id_film;

    3 usages
    @NotBlank(message = "Введите название")
    @Size(max = 50, message = "Длина названия должна быть не больше 50 символов")
    @Column(name = "name")
    private String name_film;

    3 usages
    @Min(value = 1, message = "Значение должно быть больше 0")
    @Max(value = 200, message = "Значение должно быть меньше 200")
    private int numbers_actors_film;

    3 usages
    @Min(value = 1, message = "Значение должно быть больше 0")
    @Max(value = 10, message = "Значение должно быть меньше 10")
    private int rating_film;

    3 usages
    @DecimalMin(value = "1.0", message = "Значение должно быть больше 0")
    @DecimalMax(value = "100000.0", message = "Значение должно быть меньше 100000")
    private double price_film;

    3 usages
    @Min(value = 1800, message = "Значение должно быть больше 2000")
    @Max(value = 2030, message = "Значение должно быть меньше 2023")
    private int year_film;

    3 usages
    @Min(value = 1, message = "Значение должно быть больше 5")
    @Max(value = 100, message = "Значение должно быть меньше 1000")
    private int time_film;
```

Рисунок 53 - Пример модели.

Создаем к моделям репозитории.

```

package com.example.kinoteatr.repo;

import com.example.kinoteatr.model.Film;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

4 usages
@Repository
public interface FilmRepo extends JpaRepository<Film, Long> {

    1 usage
    @Query("SELECT g FROM Film g WHERE g.name_film = :name_film")
    Film findByName(@Param("name_film") String name_film);

}

```

Рисунок 54 - Пример репозитория.

Далее создаем контроллеры, и прописываем в них логику работы с данными.

```

@Controller
@RequestMapping("/films")
@PreAuthorize("hasAnyAuthority('CASHIER', 'ADMIN')")
public class FilmController {

    @Autowired
    public ManufacturerRepo manufacturerRepo;

    @Autowired
    private FilmRepo filmRepo;

    @Autowired
    private GenreRepo genreRepo;

    @Autowired
    private AuthorRepo authorRepo;

    @GetMapping()
    public String listManu(Model model) {
        Iterable<Film> films = filmRepo.findAll();
        model.addAttribute("films", films);
        return "films/allFilms";
    }

    @GetMapping("/addFilm")
    public String showAddClassForm(Model model) {
        Film film = new Film();
        film.setYear_film(2000);
        film.setTime_film(2);
        film.setNumbers_actors_film(1);
        film.setRating_film(1);

        Iterable<Manufacturer> manufacturers = manufacturerRepo.findAll();
        model.addAttribute("manufacturers", manufacturers);

        Iterable<Genre> genres = genreRepo.findAll();
        model.addAttribute("genres", genres);

        Iterable<Author> authors = authorRepo.findAll();
        model.addAttribute("authors", authors);

        model.addAttribute("film", film);
        return "films/addFilm";
    }
}

```

Рисунок 55 - Пример контроллера.

Создаем разметку для вывода данных.

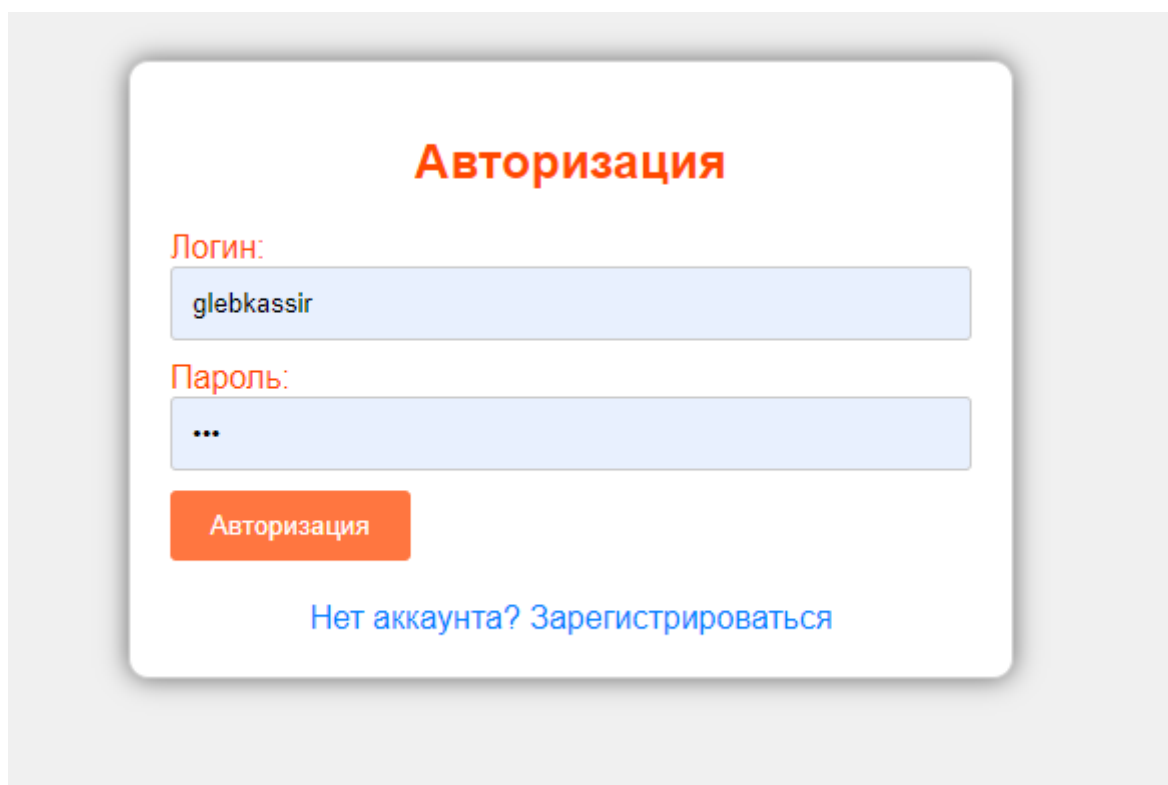
```

<div th:switch="{films}" >
  <div class="row">
    <div class="col-md-8 mx-auto">
      <div th:case="*" class="table-container">
        <h2 class="my-5 text-center">Фильмы</h2>
        <table>
          <thead>
            <tr>
              <th>Название</th>
              <th>Цена</th>
              <th>Год выпуска</th>
              <th>Время фильма</th>
              <th>Количество актеров в фильме</th>
              <th>Рейтинг</th>
              <th>Жанр</th>
              <th>Производитель</th>
              <th>Режиссер</th>
              <th>Изменение</th>
              <th>Удаление</th>
            </tr>
          </thead>
          <tbody>
            <tr th:each="film : {films}">
              <td th:text="{film.name_film}"></td>
              <td th:text="{film.price_film}"></td>
              <td th:text="{film.year_film}"></td>
              <td th:text="{film.time_film}"></td>
              <td th:text="{film.numbers_actors_film}"></td>
              <td th:text="{film.rating_film}"></td>
              <td th:text="{film.genre != null ? film.genre.name_genre : ''}"></td>
              <td th:text="{film.manufacturer != null ? film.manufacturer.name_manufacturer : ''}"></td>
              <td th:text="{film.author != null ? film.author.name_author : ''}"></td>
              <td>
                <a th:href="{@{/films/editFilm/{id}(id={film.id_film})}" class="edit-button">Изменить</a>
              </td>
              <td>
                <a th:href="{@{/films/deleteFilm/{id}(id={film.id_film})}" class="delete-button">Удалить</a>
              </td>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  </div>

```

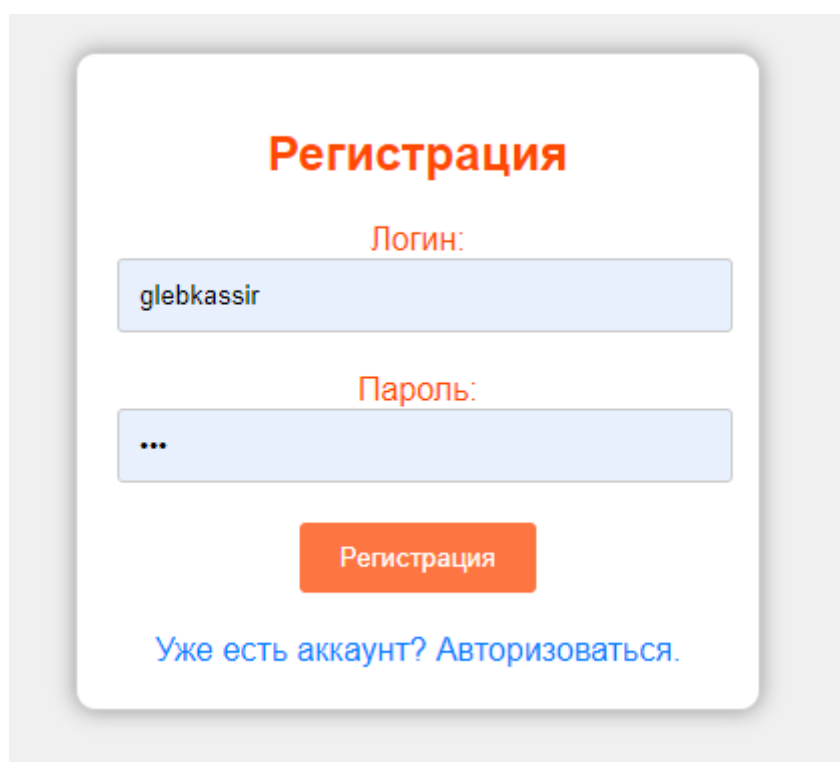
Рисунок 56 - Пример разметки.

Результат работы программы:



The image shows a web form for authorization. At the top, the title "Авторизация" is displayed in a bold, orange font. Below the title, there are two input fields. The first is labeled "Логин:" in orange text, and it contains the text "glebkassir". The second is labeled "Пароль:" in orange text, and it contains three dots "...". Below these fields is an orange button with the text "Авторизация". At the bottom of the form, there is a blue link that says "Нет аккаунта? Зарегистрироваться".

Рисунок 57 – Авторизация.



The image shows a web form for registration. At the top, the title "Регистрация" is displayed in a bold, orange font. Below the title, there are two input fields. The first is labeled "Логин:" in orange text, and it contains the text "glebkassir". The second is labeled "Пароль:" in orange text, and it contains three dots "...". Below these fields is an orange button with the text "Регистрация". At the bottom of the form, there is a blue link that says "Уже есть аккаунт? Авторизоваться".

Рисунок 58 – Регистрация.



Рисунок 59 – Главная.



Рисунок 60 – Заказы.

Создание заказа

Общая стоимость заказа

250

Пользователь

glebadmin



Статус

Создан



Создать заказ

Рисунок 61 – Добавить заказ.

Изменение заказа

Общая стоимость заказа

232.0

Пользователь

glebuser

Статус

Создан

Сохранить заказ

Рисунок 62 – Изменить заказ.

Добавить фильм к заказу

Побег

11

Добавить фильм к заказу

[Вернуться к заказам](#)

Рисунок 63– Добавить фильм к заказу.

Вывод:

В ходе данной учебной практики я получил глубокие знания и практический опыт работы с информационной системой, представляющей собой веб-приложение для продажи фильмов на основе технологии Spring Framework. Основываясь на описании системы, выделены три основные роли: пользователь, кассир и администратор.

В ходе практики я приобрел навыки работы с каждой из выделенных ролей, освоил функционал системы и научился эффективно выполнять свои обязанности в соответствии с назначенной ролью. Этот опыт позволит мне успешно применять полученные знания в будущих проектах, требующих работу с подобными информационными системами.