|  |
| --- |
| **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](#_Toc146190759)

[Практическая работа №2 10](#_Toc146190760)

[Практическая работа №3 19](#_Toc146190761)

[Практическая работа №4 28](#_Toc146190762)

[Практическая работа №5 35](#_Toc146190763)

[Практическая работа №6 42](#_Toc146190764)

[Итоговый проект 46](#_Toc146190765)

[Тема: магазин продажи фильмов «Кинотеатр» 46](#_Toc146190766)

### 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.



Рисунок 1. Контроллер главной страницы.

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

Аннотация @PostMapping("/Resault") 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.



Рисунок 2. Контроллер калькулятора. (1)



Рисунок 3. Контроллер калькулятора. (2)

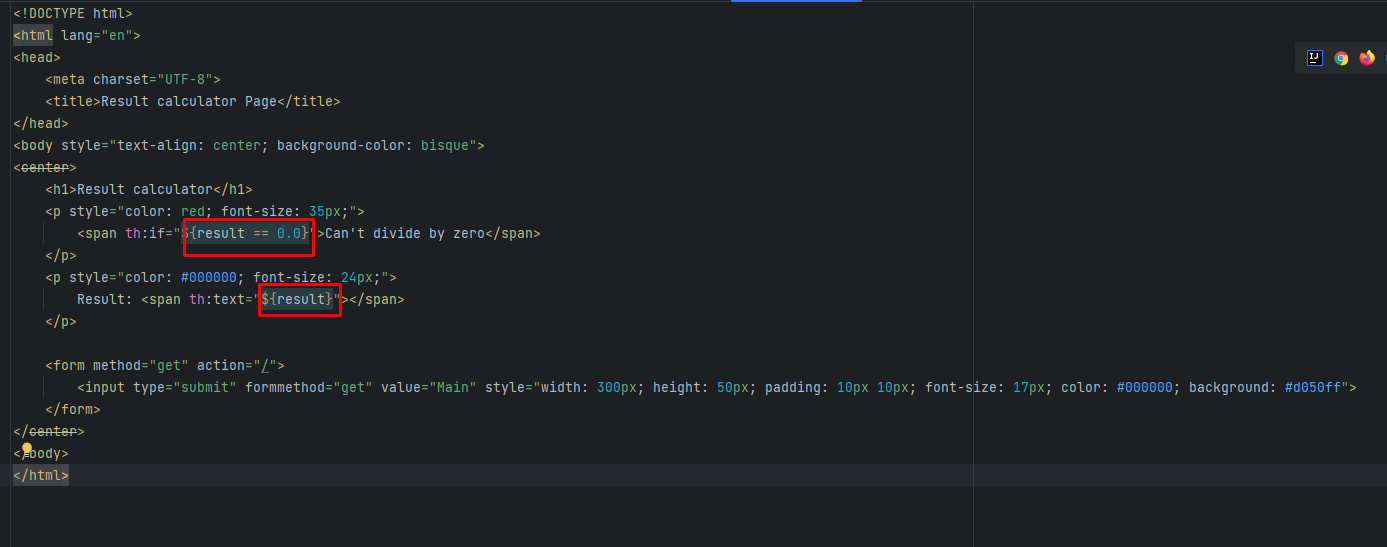


Рисунок 4. Использование result.

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

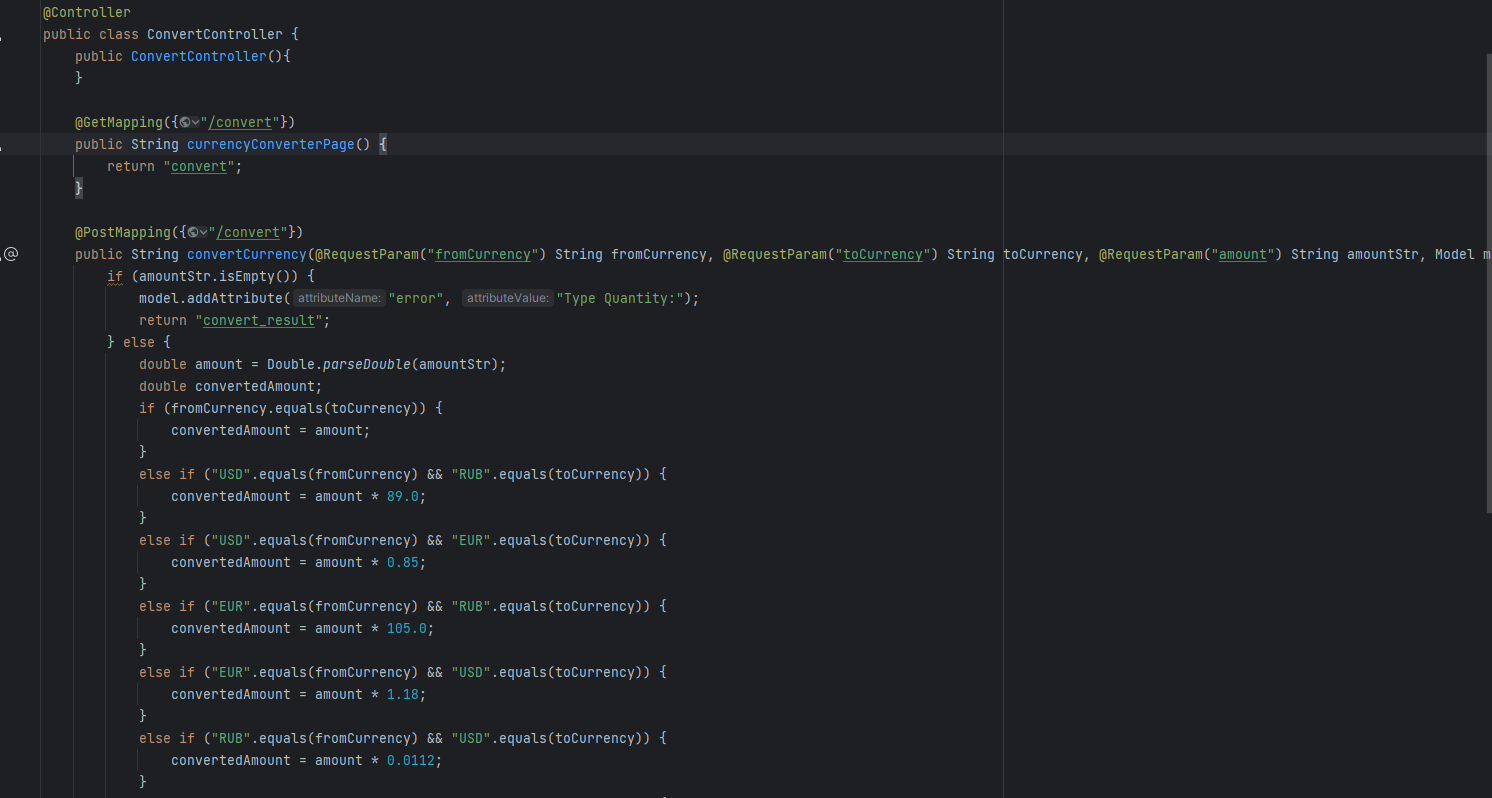
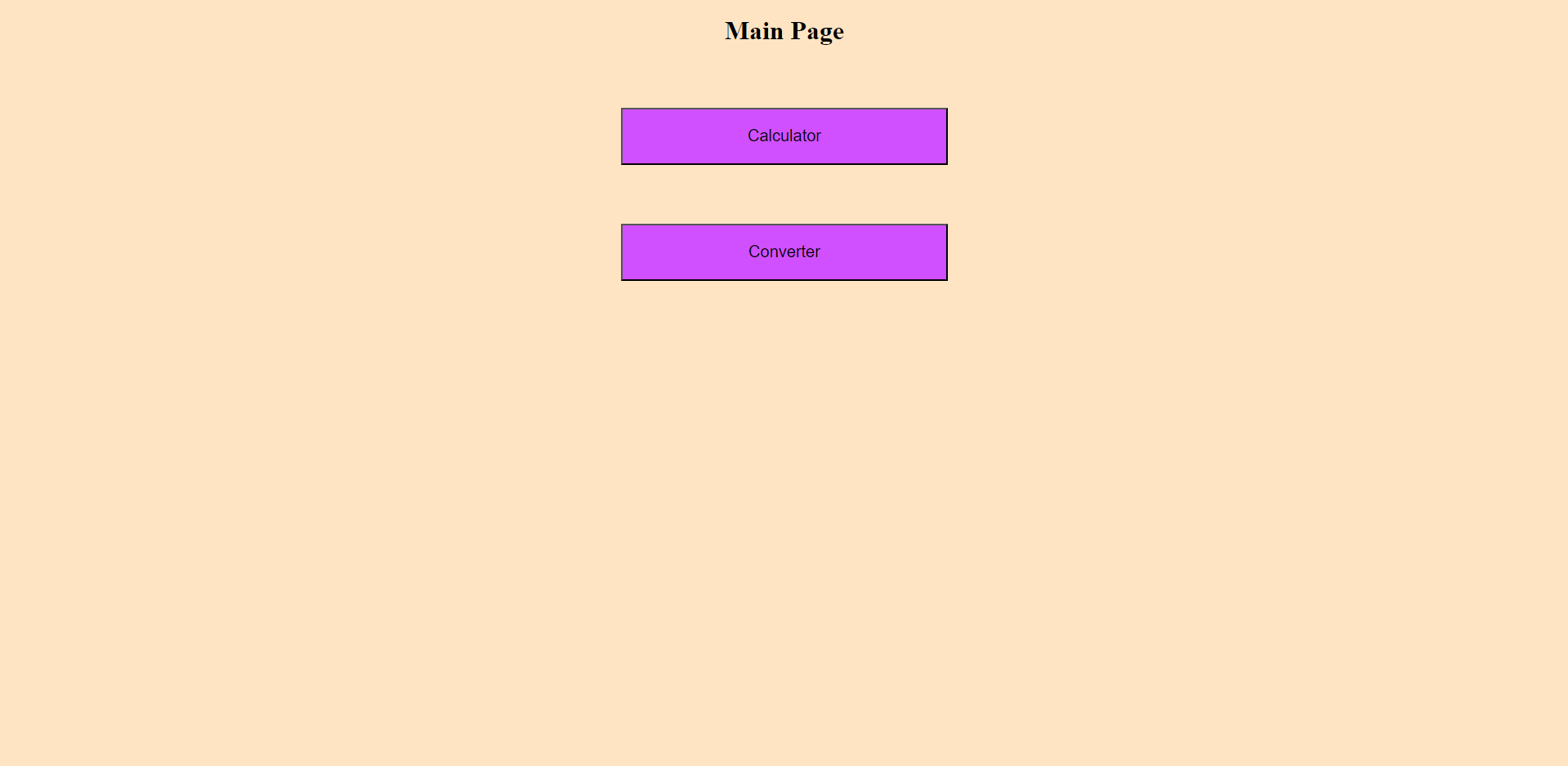
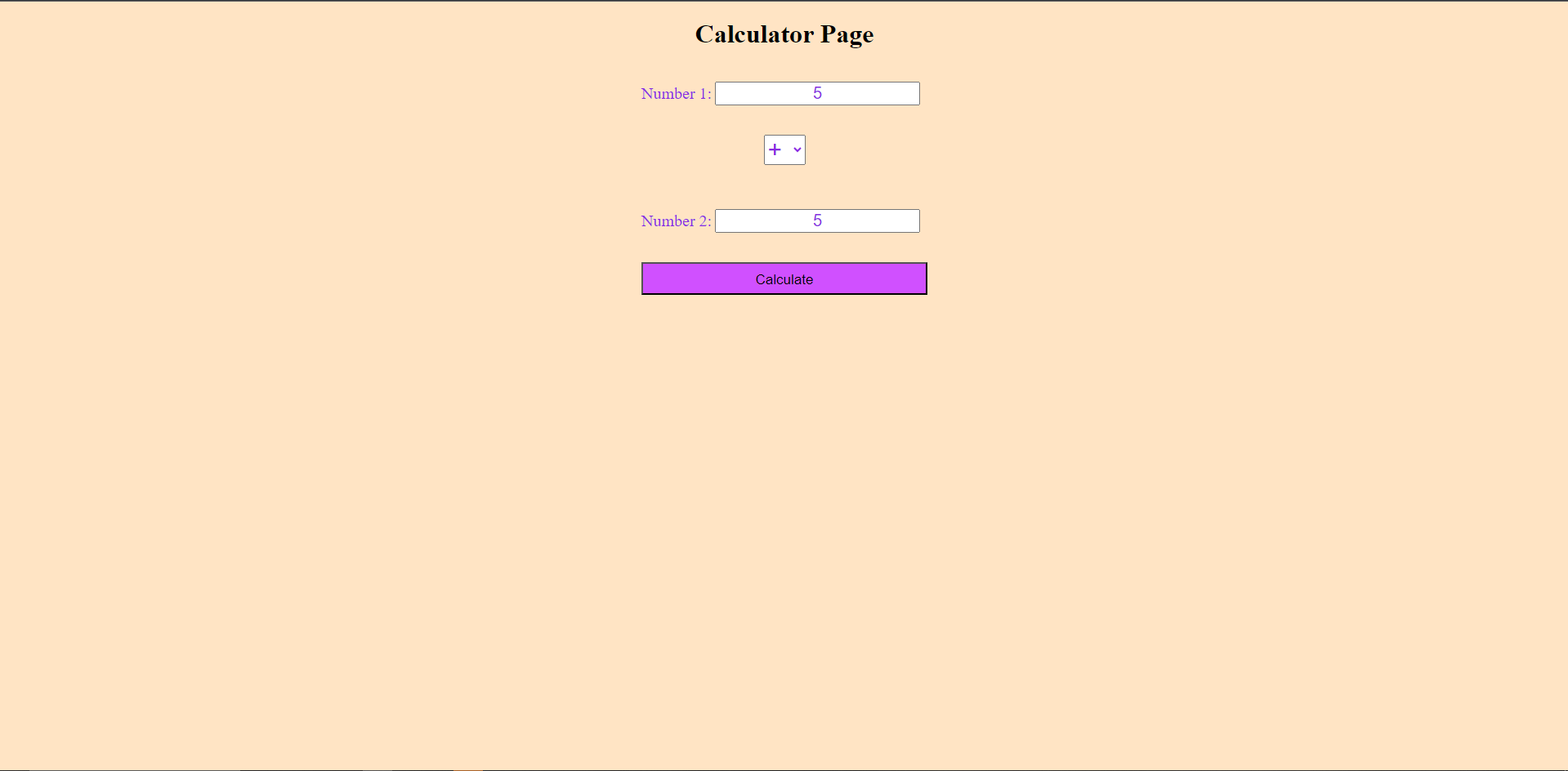
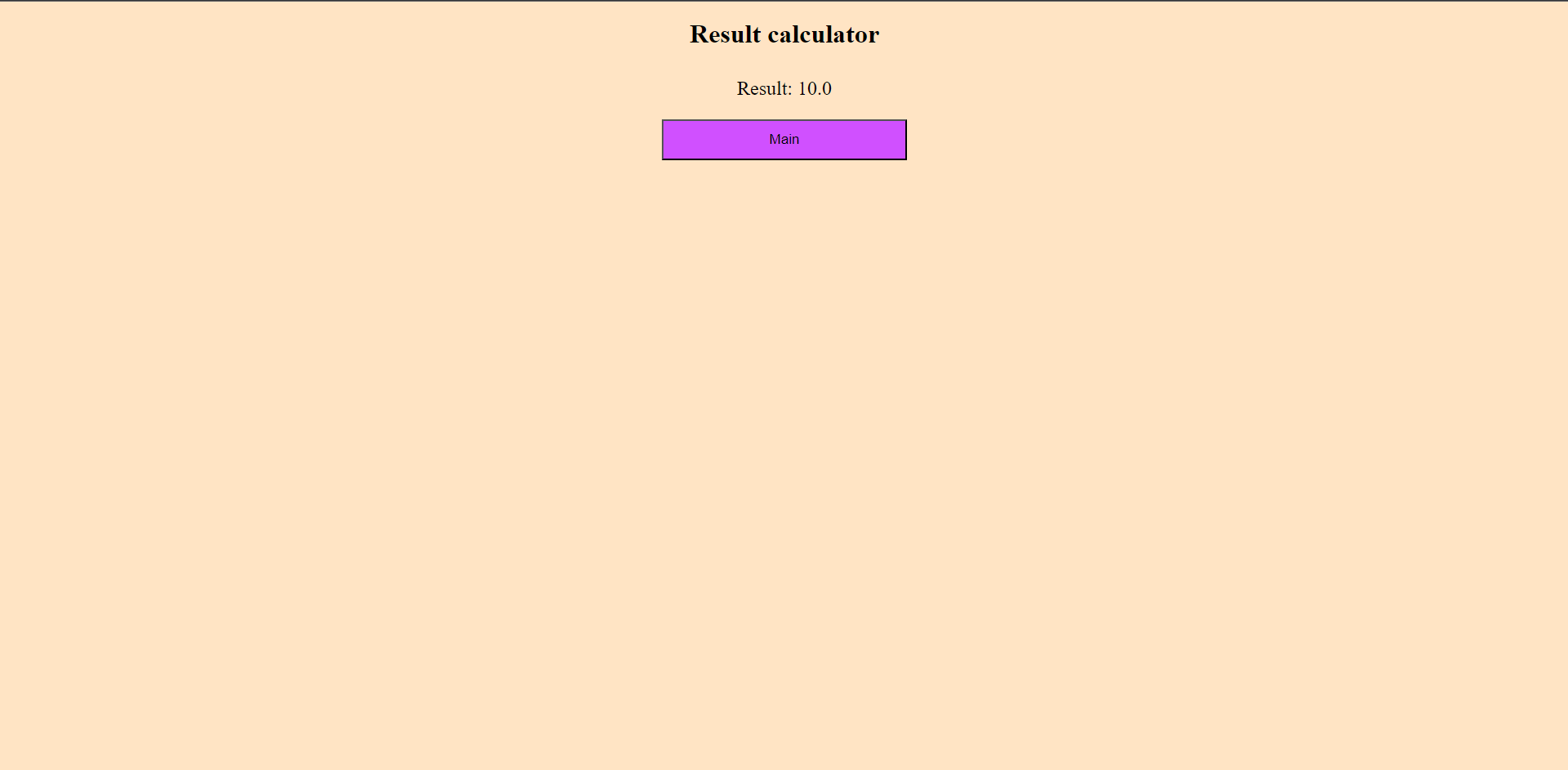


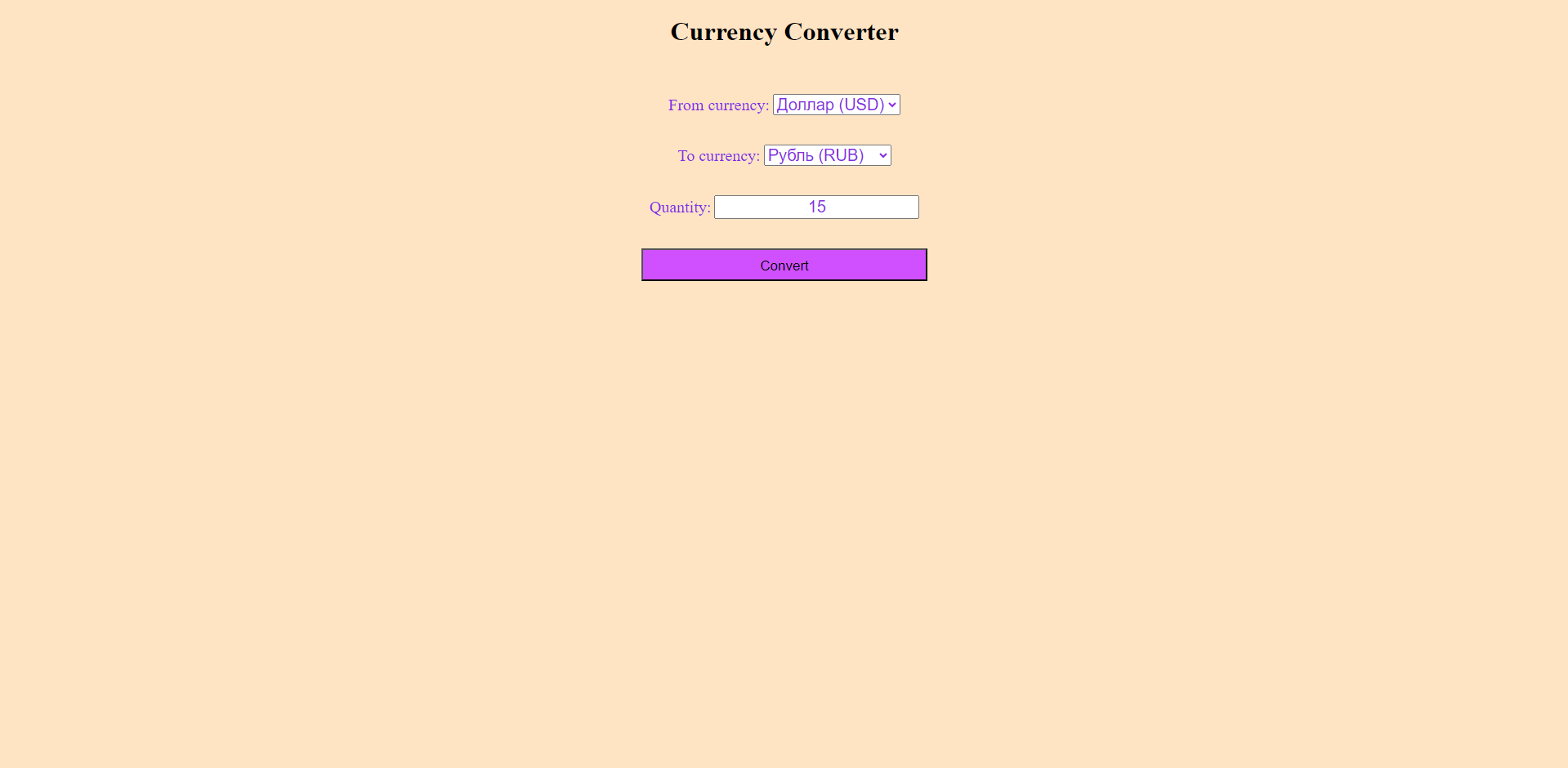
Рисунок 4. Контроллер конвертера валют.

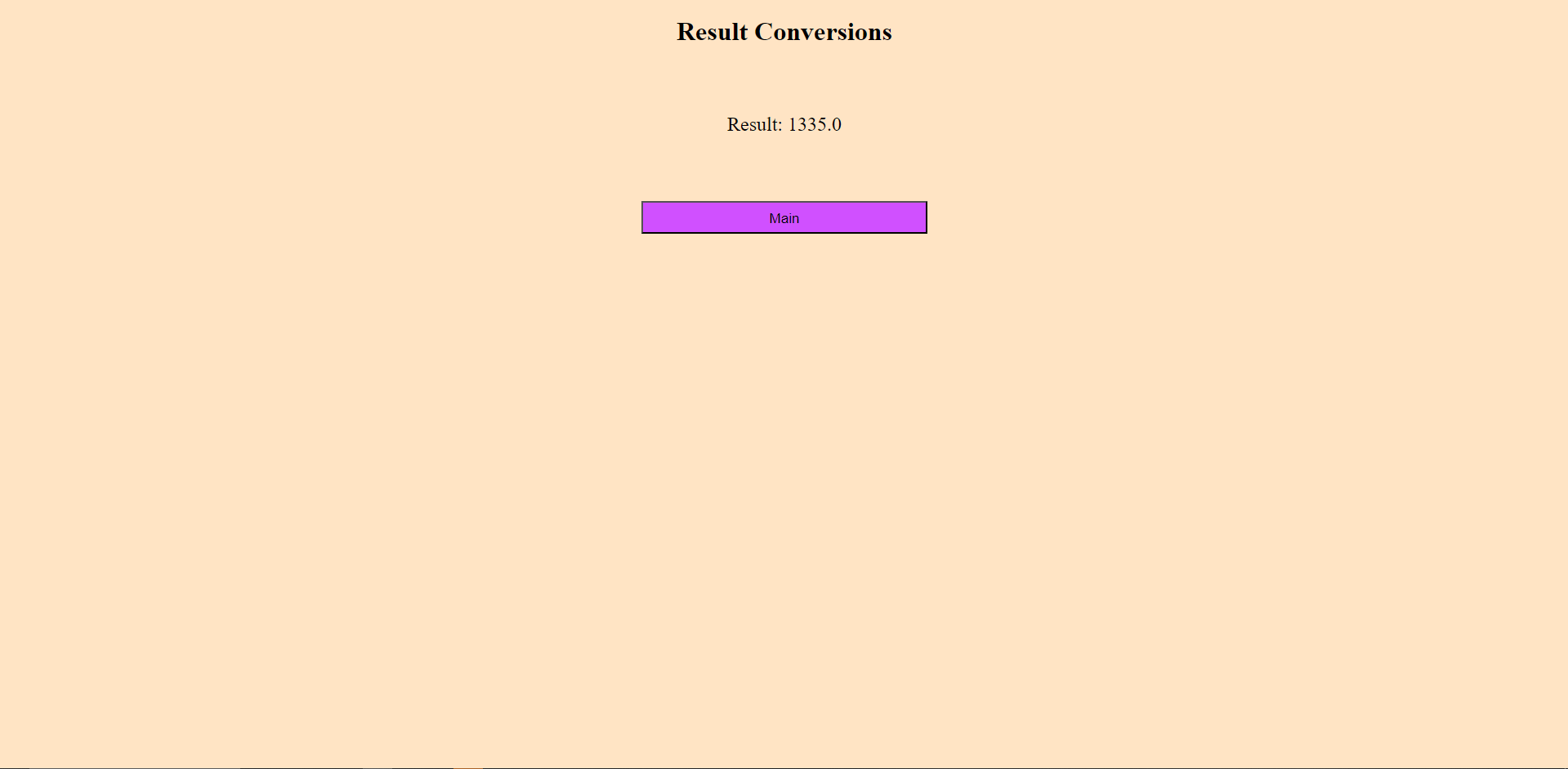
Результат работы:











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.

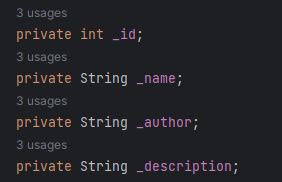


Рисунок 5. 4 поля.

Создаем конструкторы.

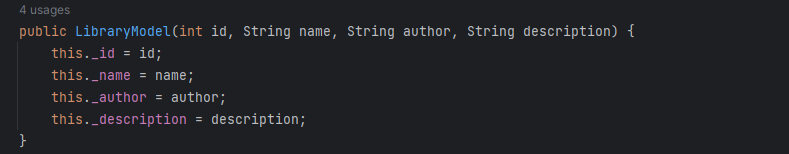


Рисунок 6. Конструктор.

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



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.

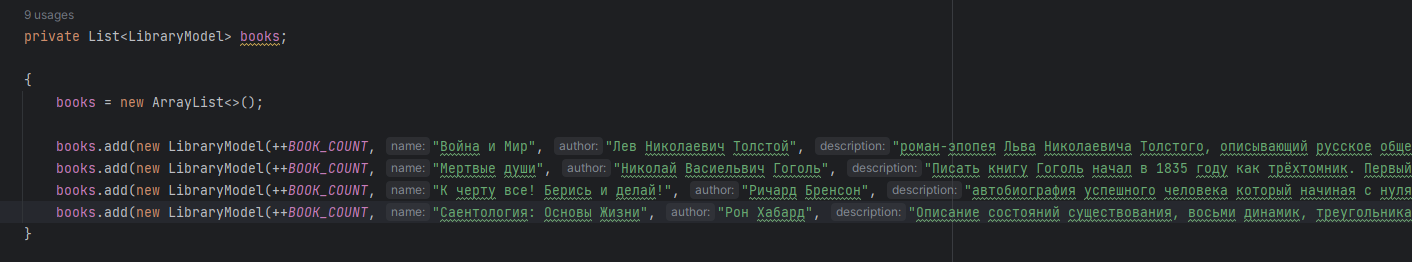


Рисунок 8. Список.

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



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.



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.

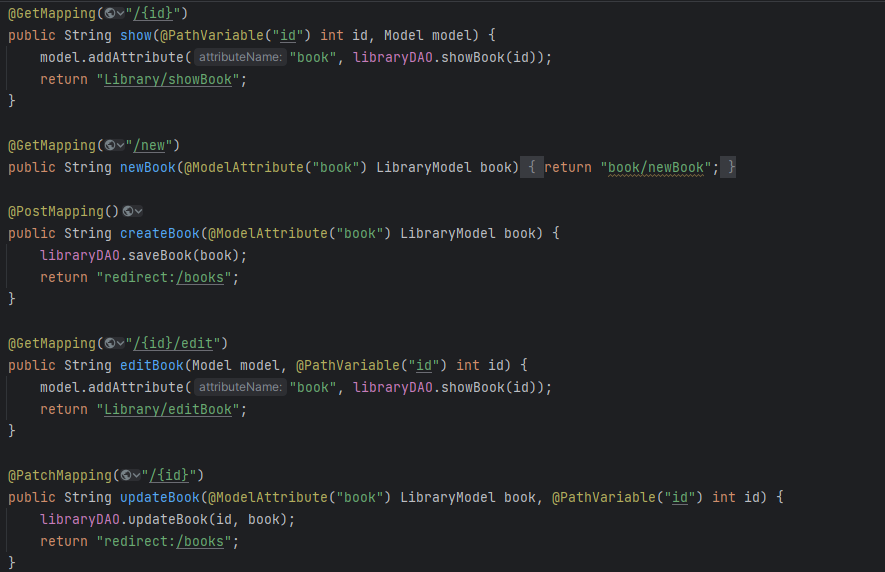
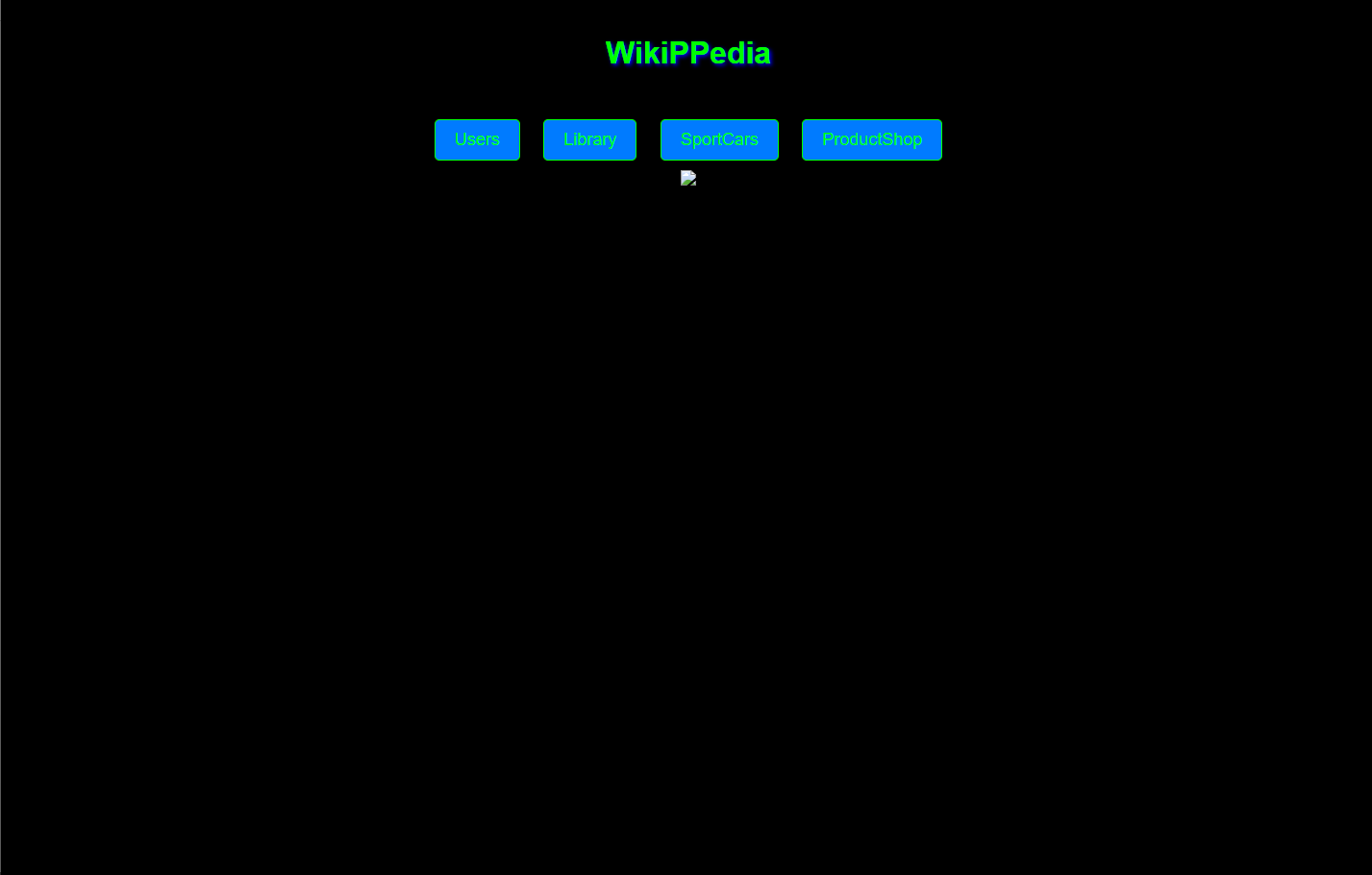
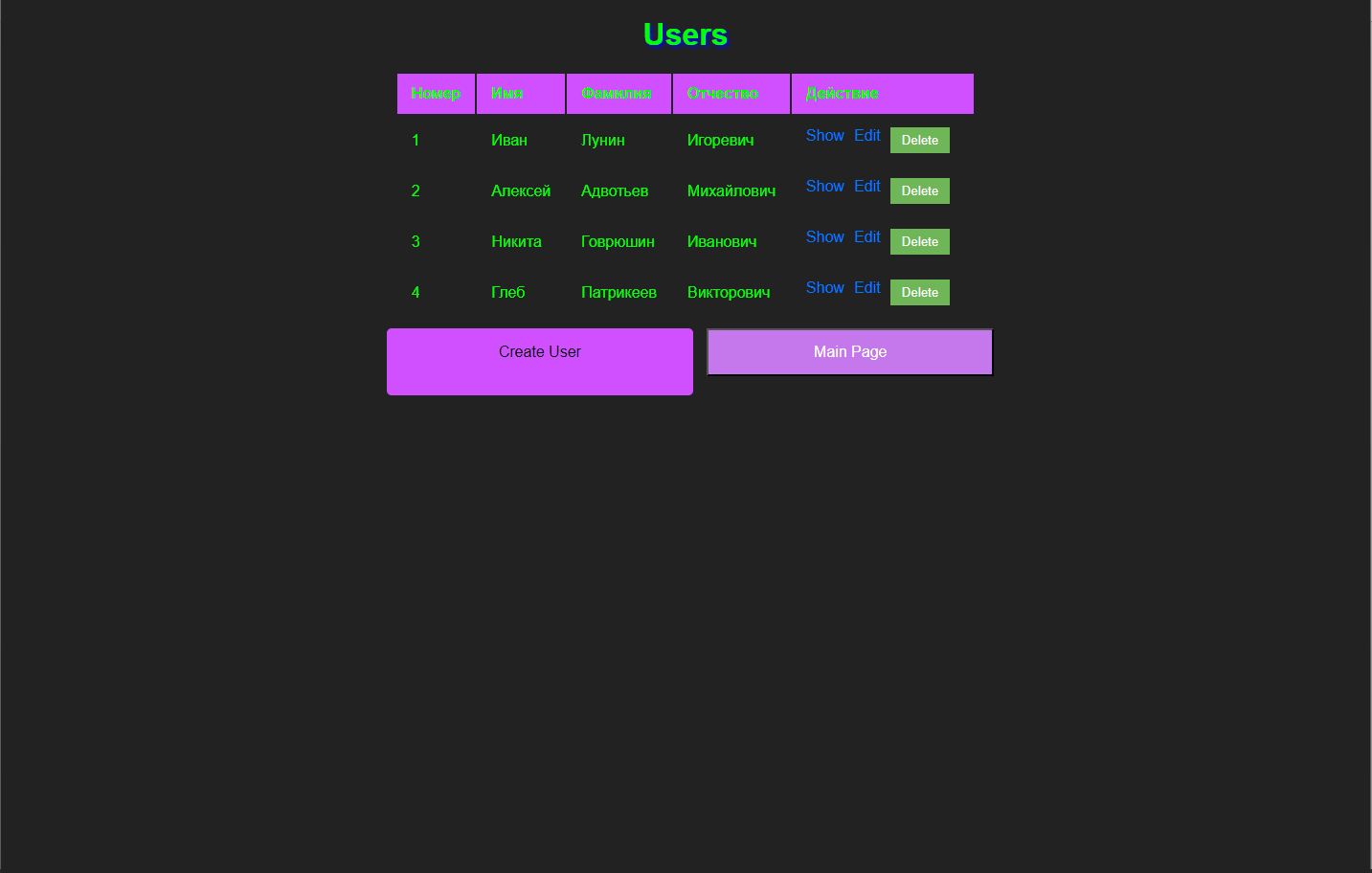


Рисунок 10. Аннотации (2).

Результат работы:

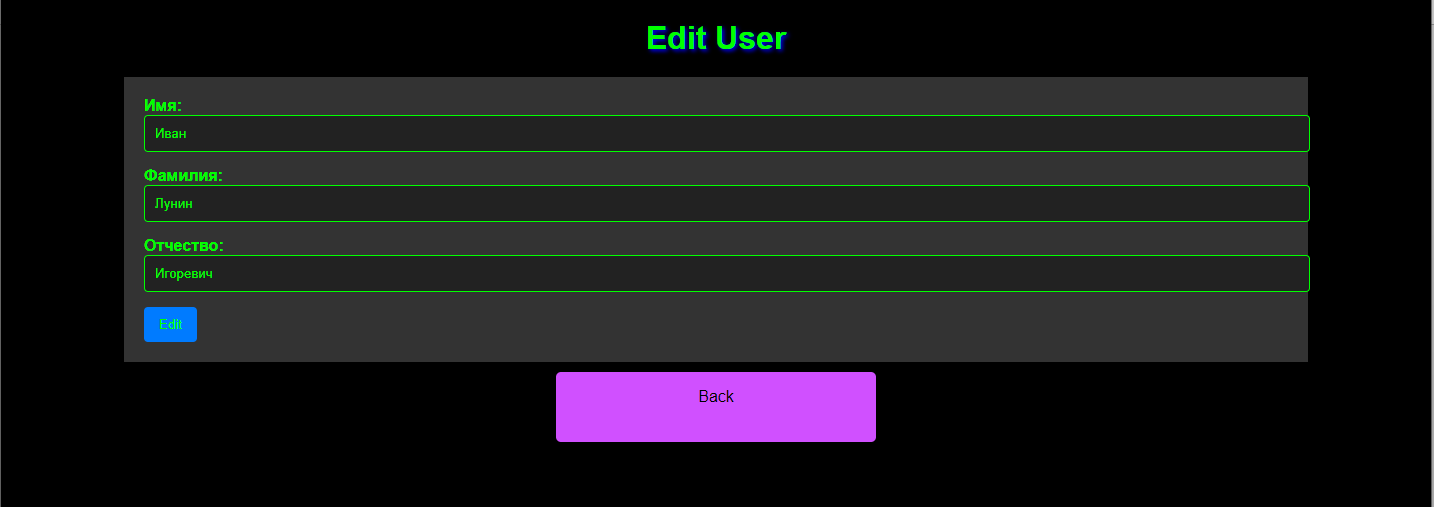


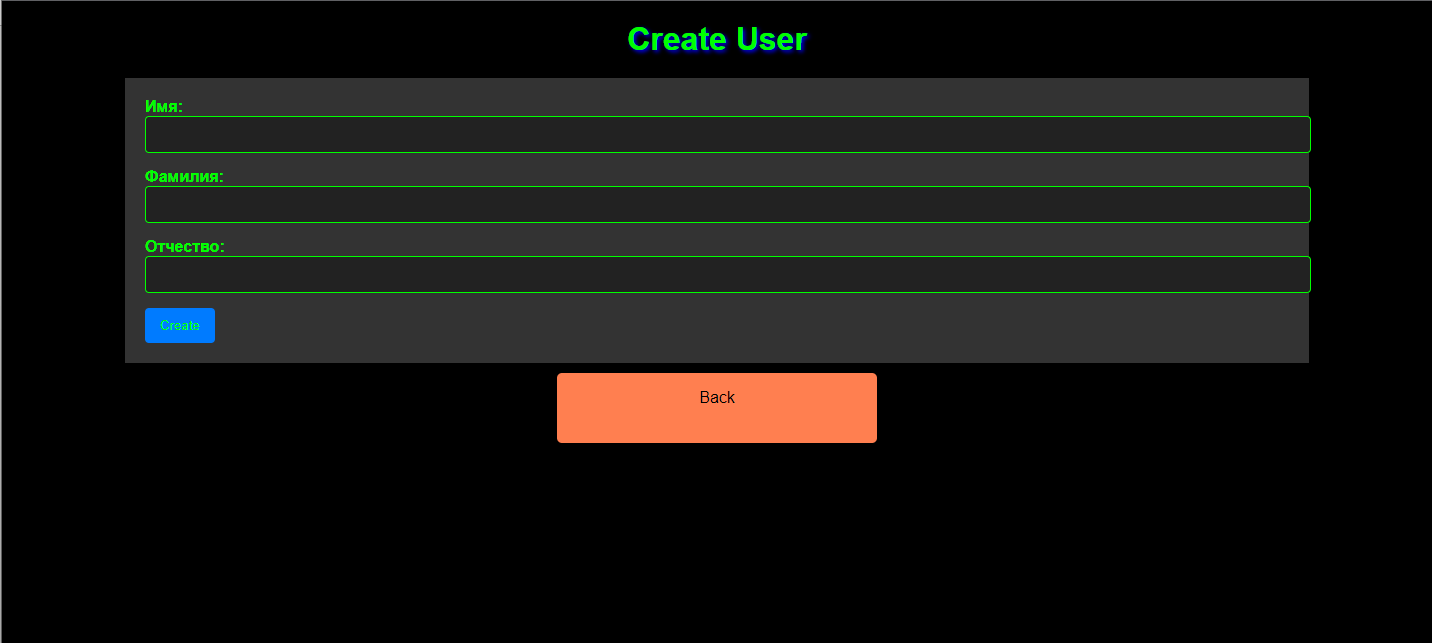


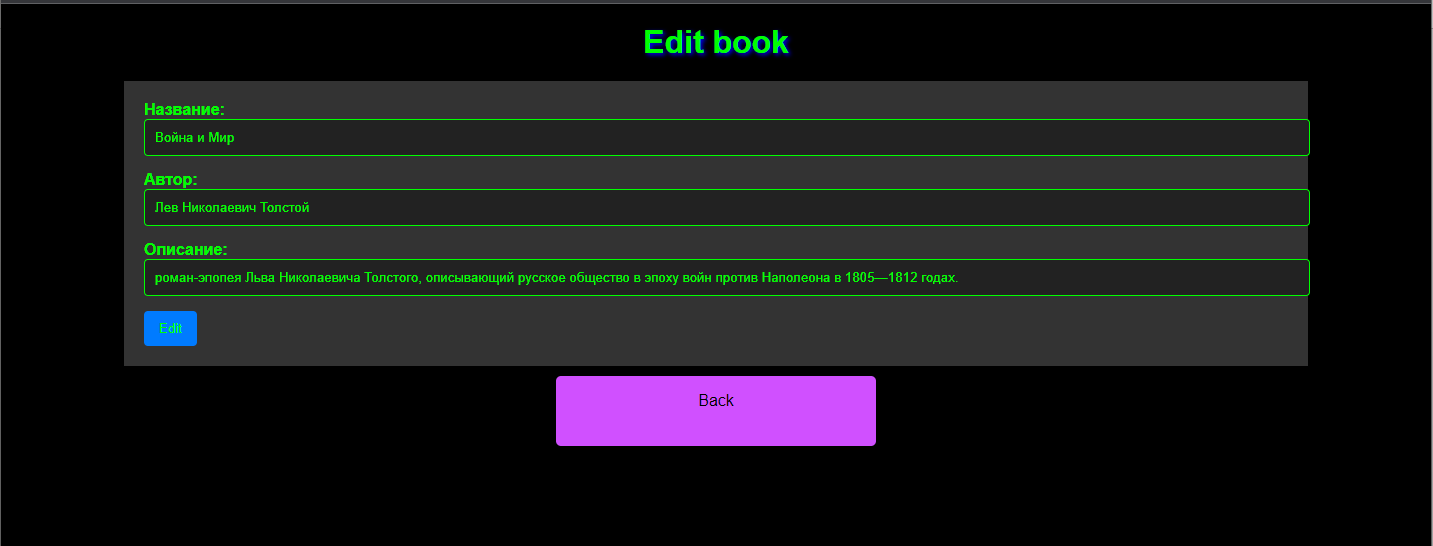


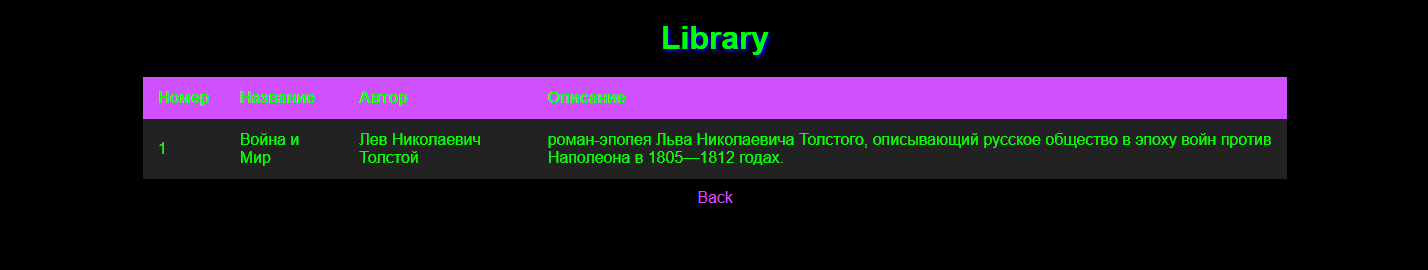


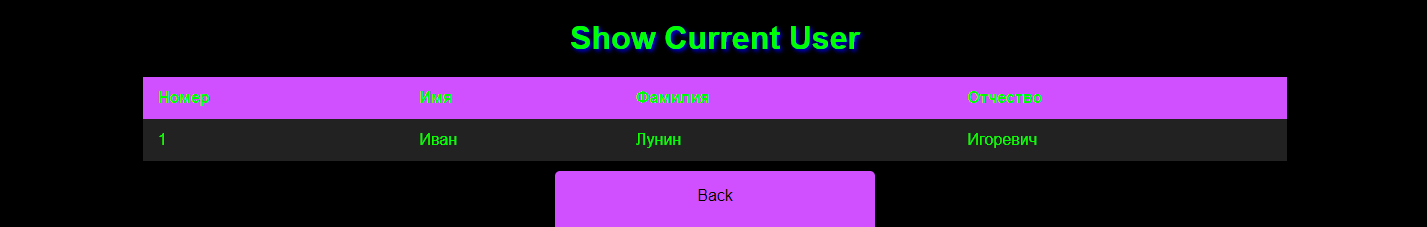












И т.д.

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 postgre.

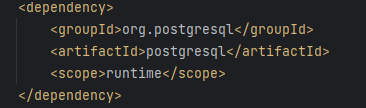


Рисунок 11. Зависимость

Подключаем нашу базу данных:

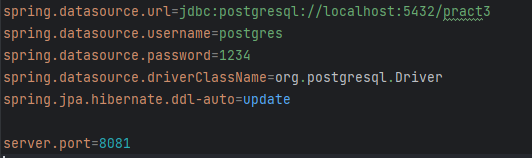


Рисунок 12. Строка подключения

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 выбирать 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.



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.

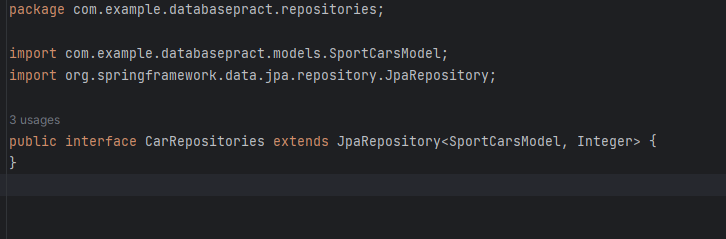


Рисунок 4. SportCarsRepositories

Внедряем зависимость CarRepositories через конструктора 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.

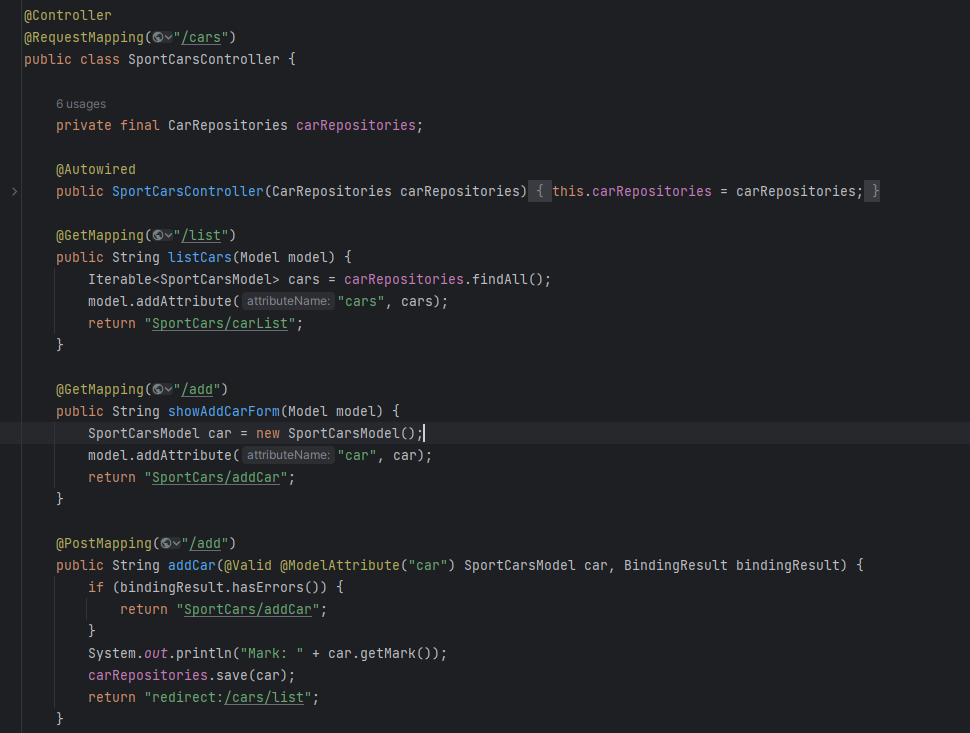


Рисунок 5. SportCarsController (1)



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».

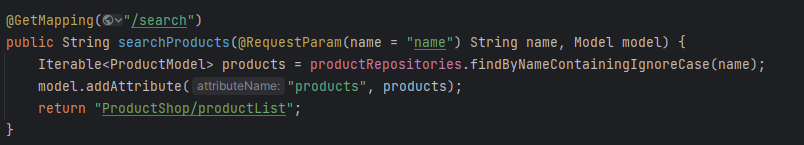


Рисунок 7. searchProduct

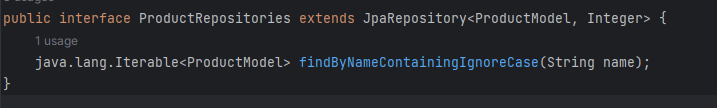


Рисунок 8. ProductRepositories

Результат:

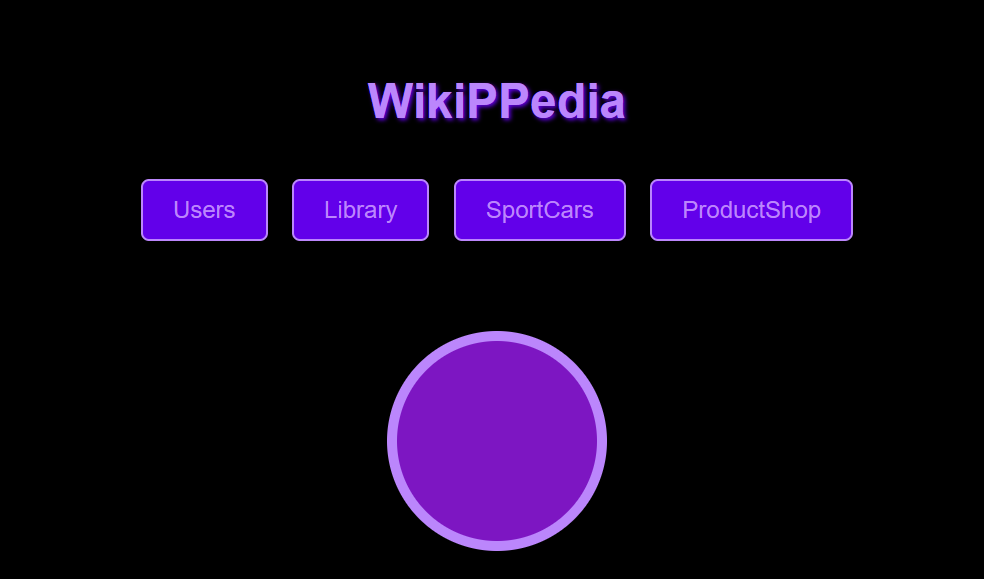


Рисунок 9. Главная страница.

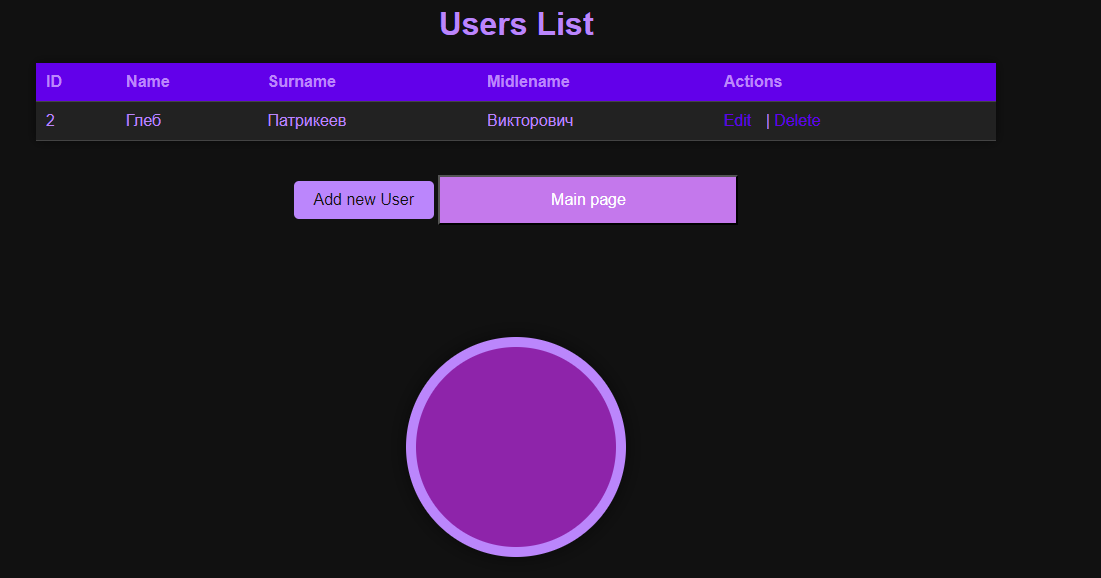


Рисунок 10. Пользователи.

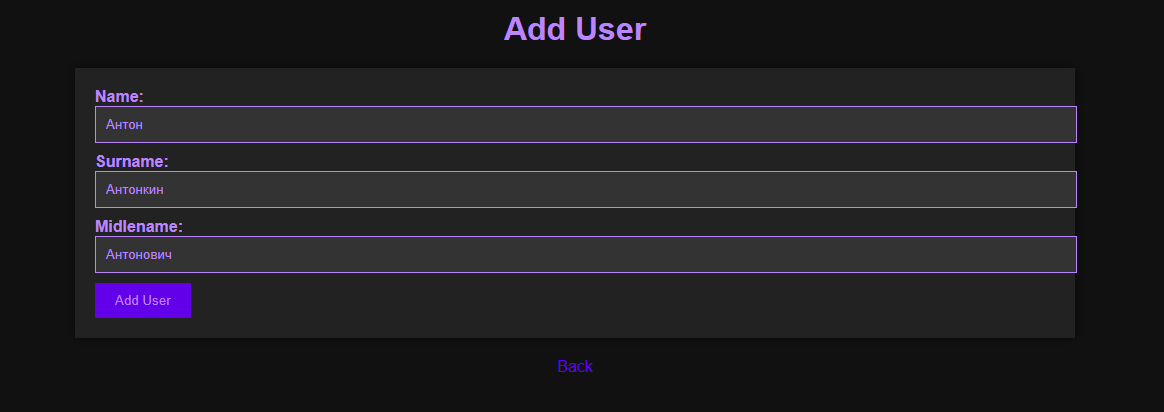


Рисунок 11. Создание пользователя.

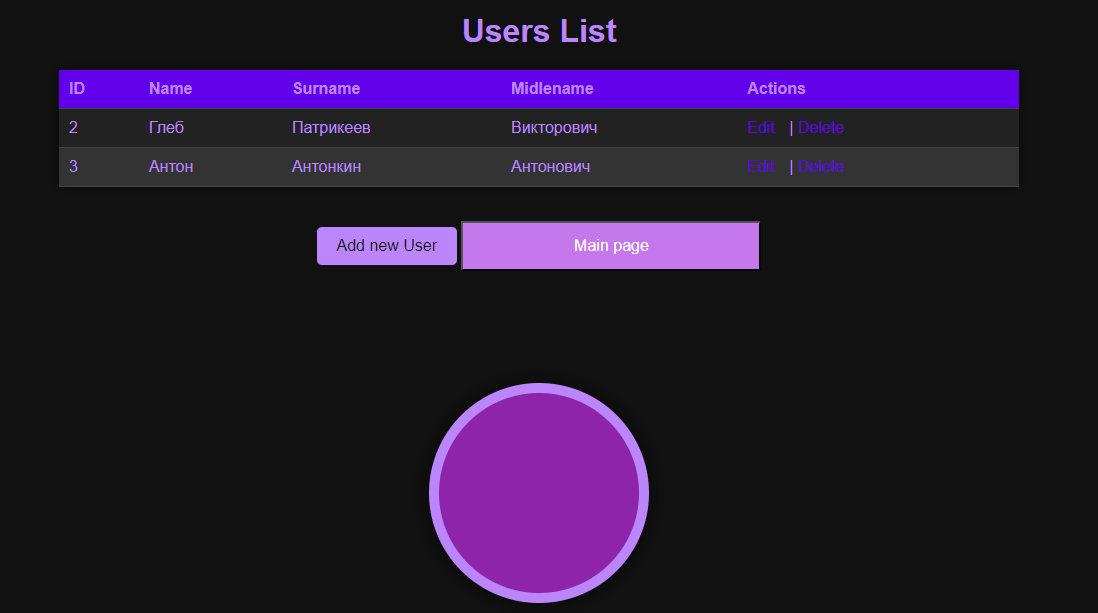


Рисунок 12. Демонстрация создания.

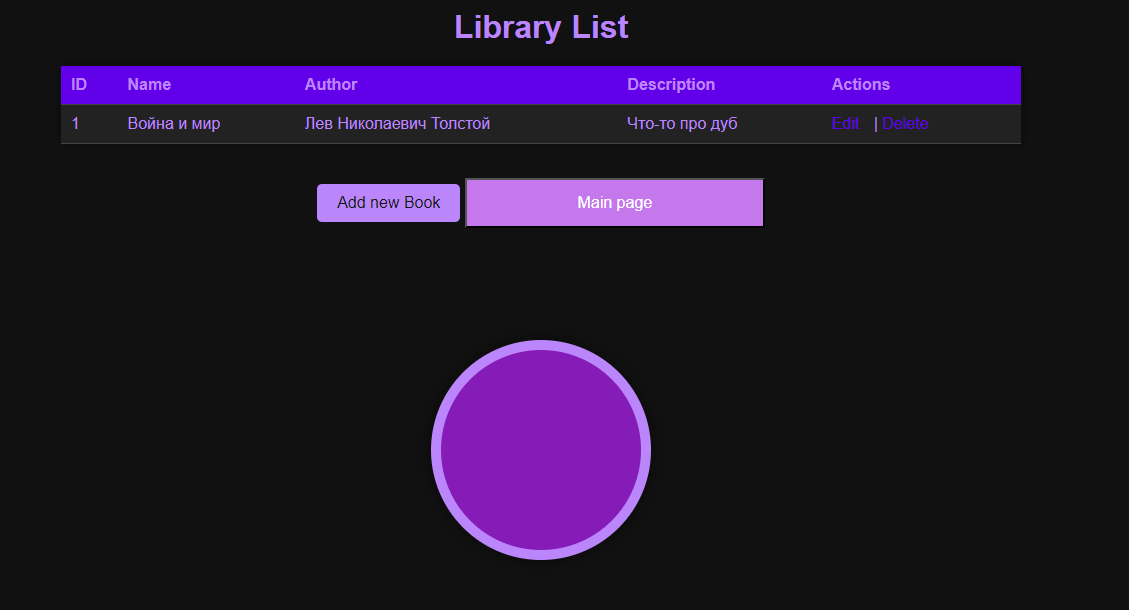


Рисунок 13. Просмотр книг

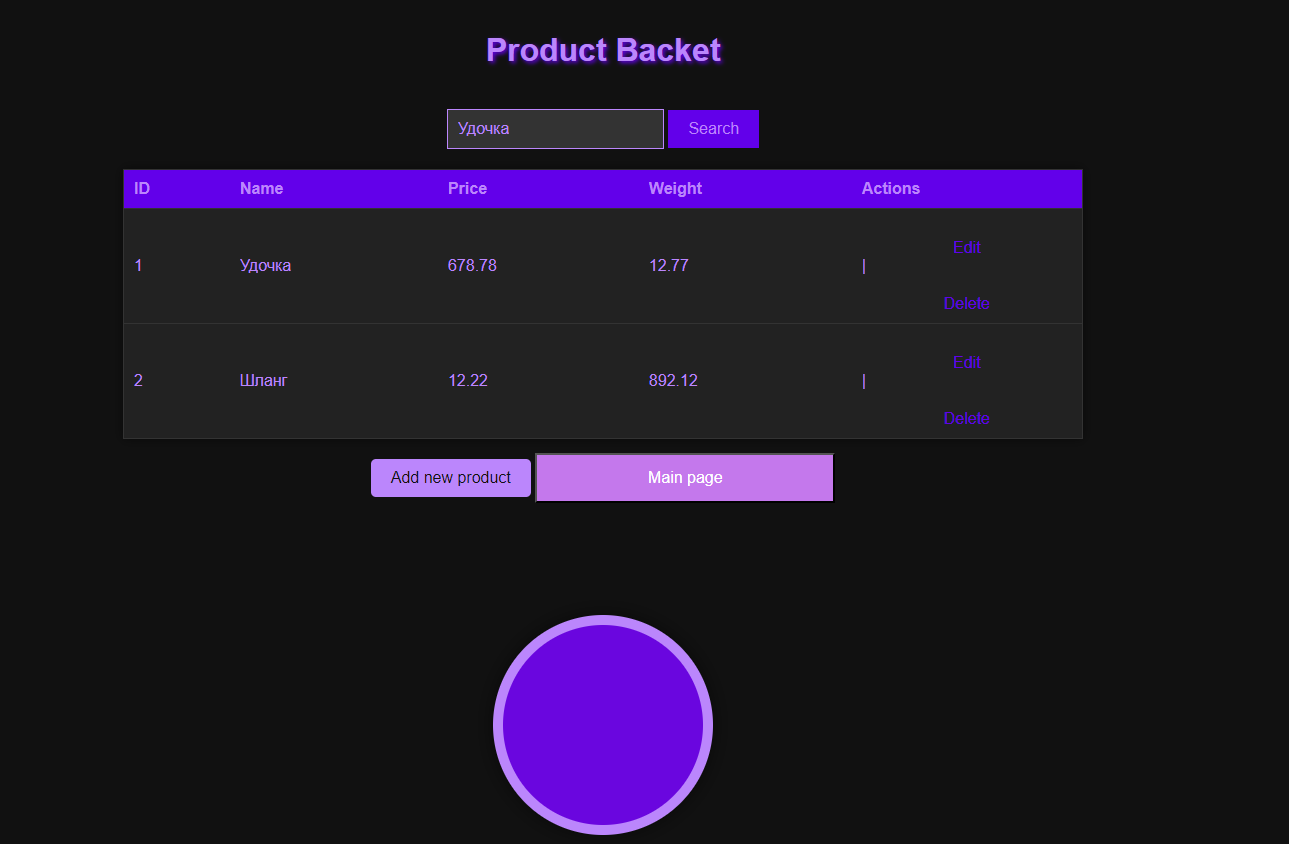


Рисунок 14. Поиск

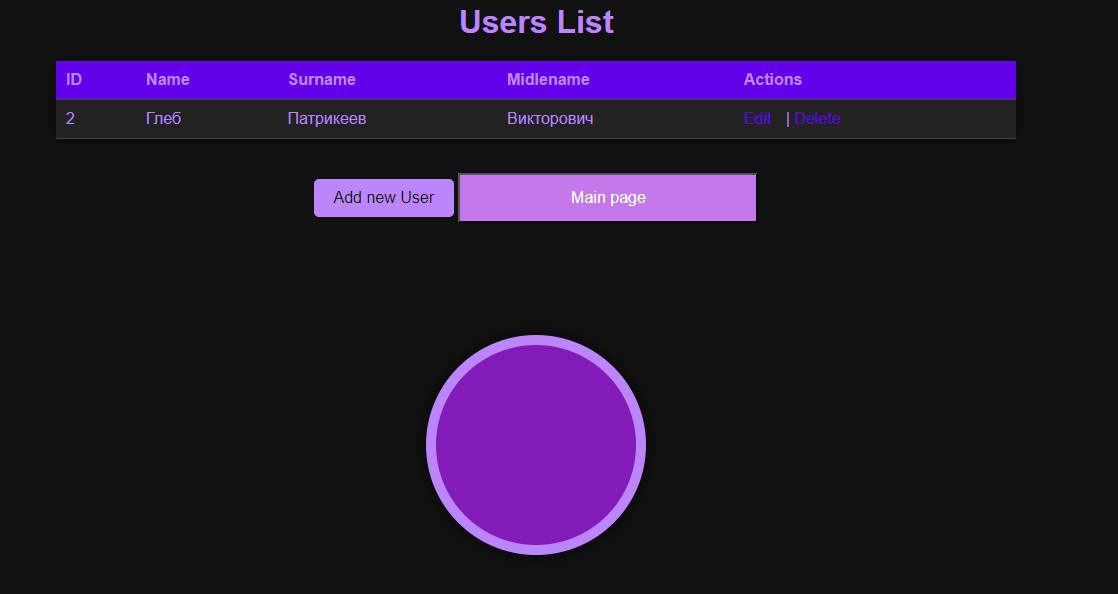


Рисунок 15. Удаление пользователя

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.

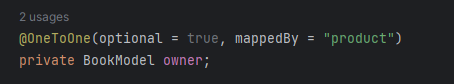


Рисунок 22. ProductModel.



Рисунок 23. BookModel.

Результат:  


Рисунок 24. OneToOne.



Рисунок 25. Таблица Продукт.

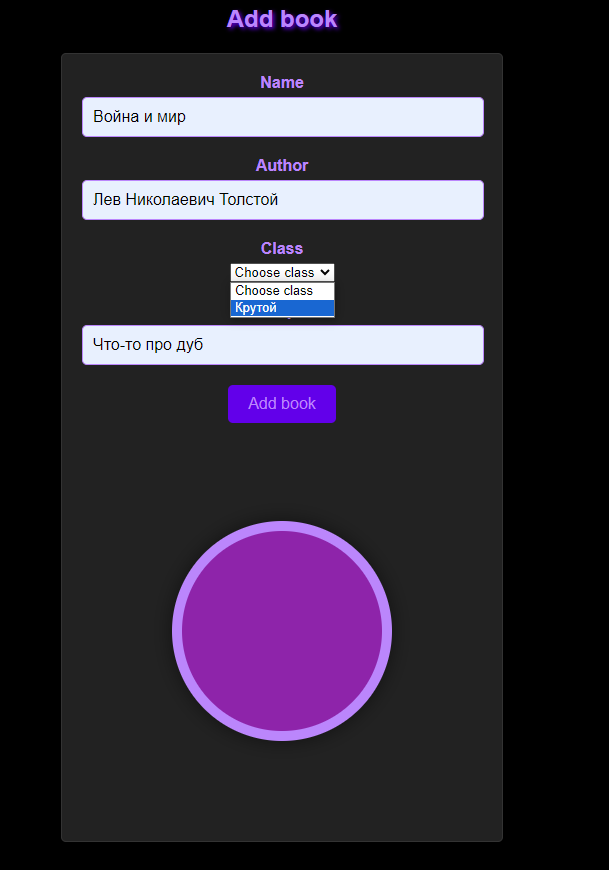


Рисунок 26. Выбор класса.

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.

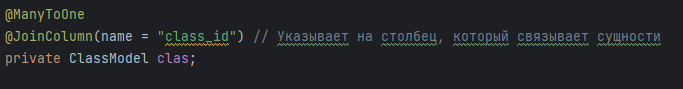


Рисунок 27. BookModel

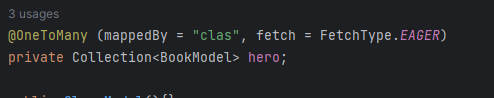


Рисунок 28. ClassModel

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.

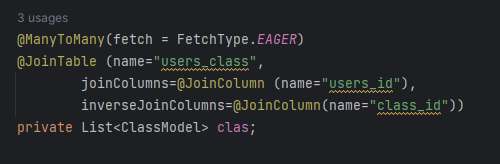


Рисунок 30. UserController.

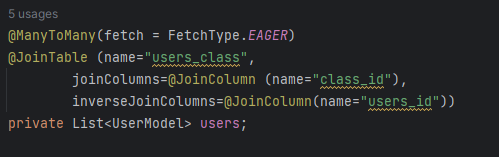


Рисунок 31. ClassController.

Результат:

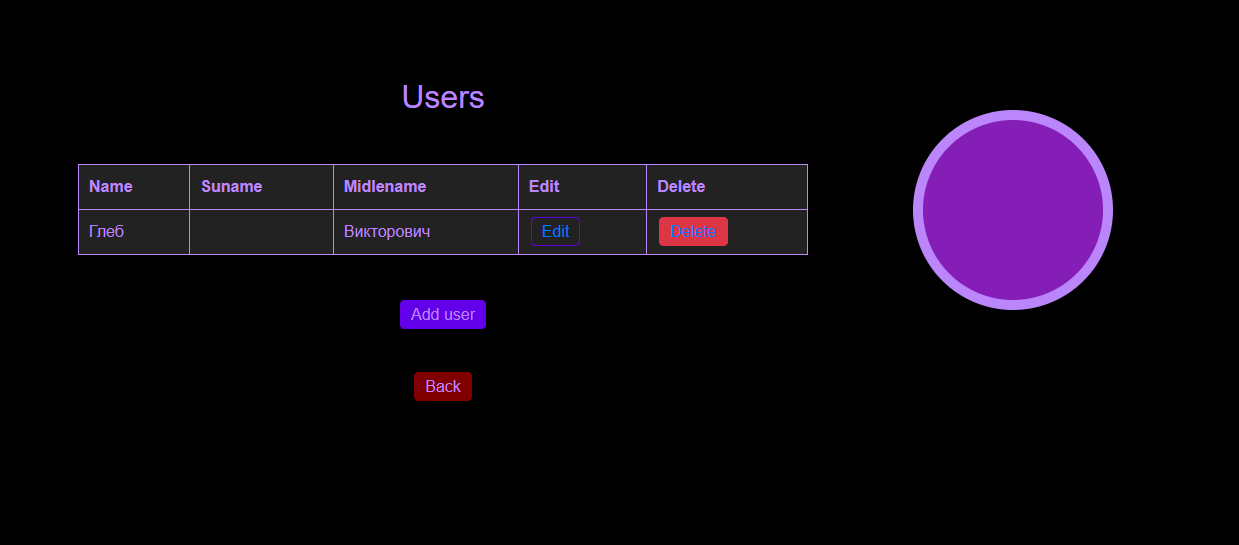


Рисунок 32. Таблица Пользователи.



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.

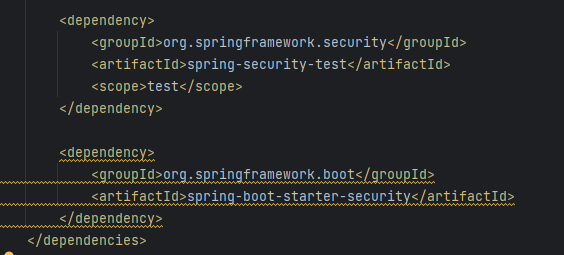


Рисунок 34. Зависимости.

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.

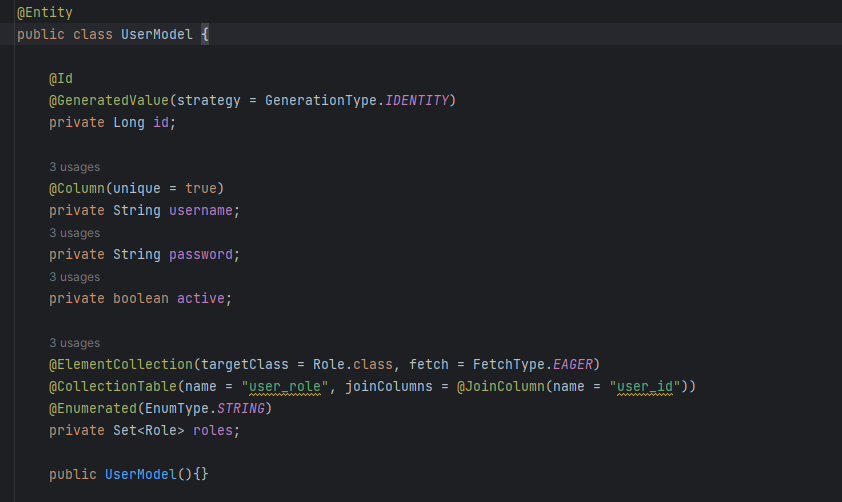


Рисунок 35. UserModel.

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

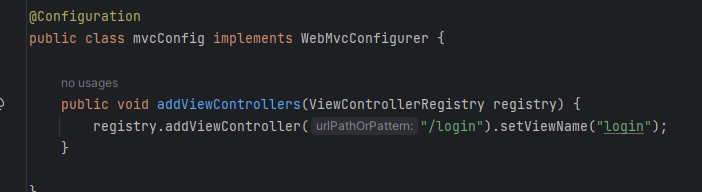


Рисунок 36. mvcConfig.

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.

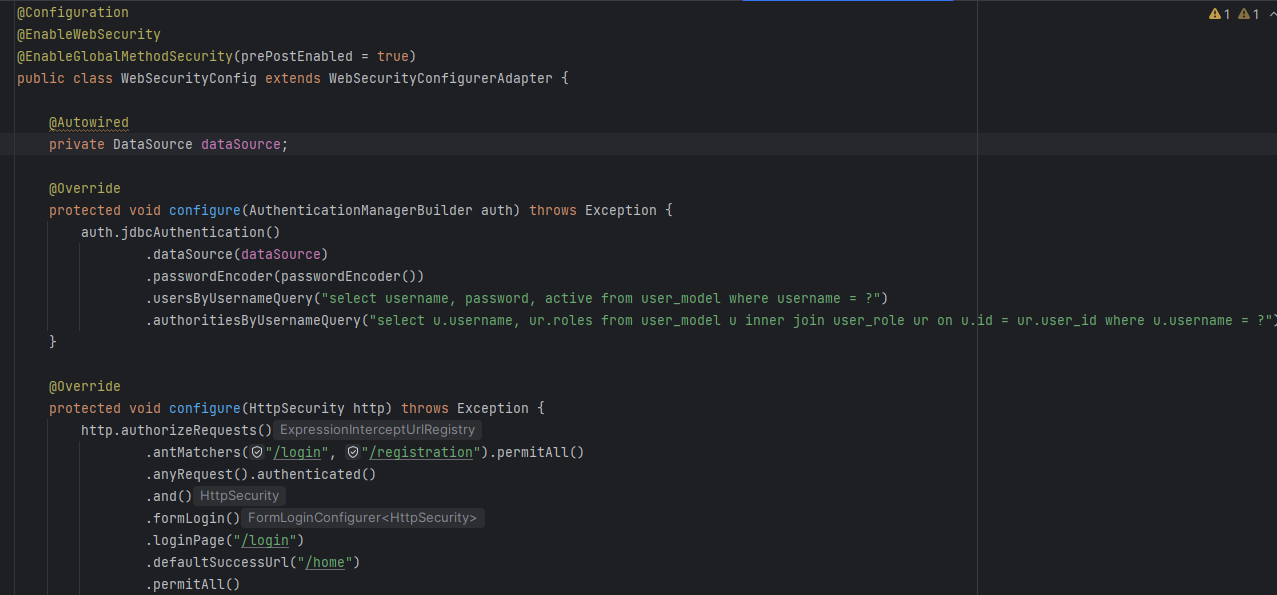


Рисунок 37. WebSecurityConfig.

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.

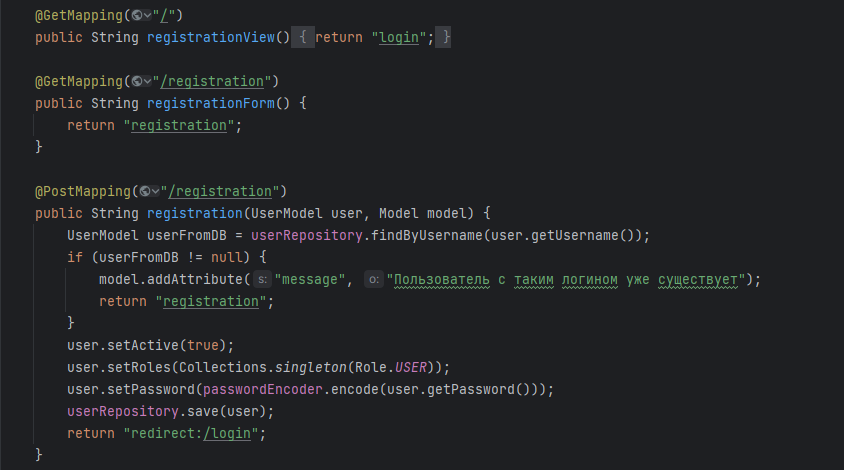


Рисунок 38. Контроллер для регистрации.

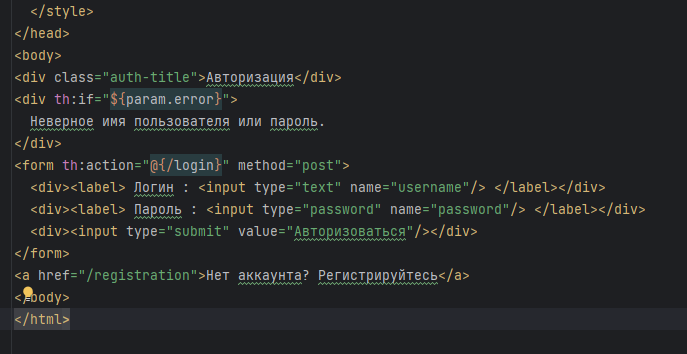


Рисунок 39. Login.

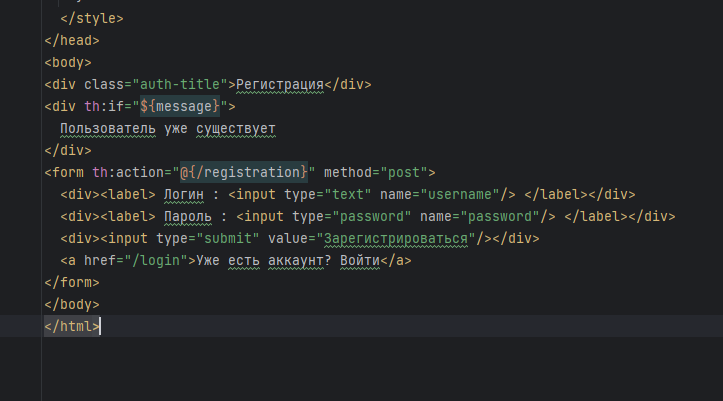


Рисунок 40. Registration.

Результат:

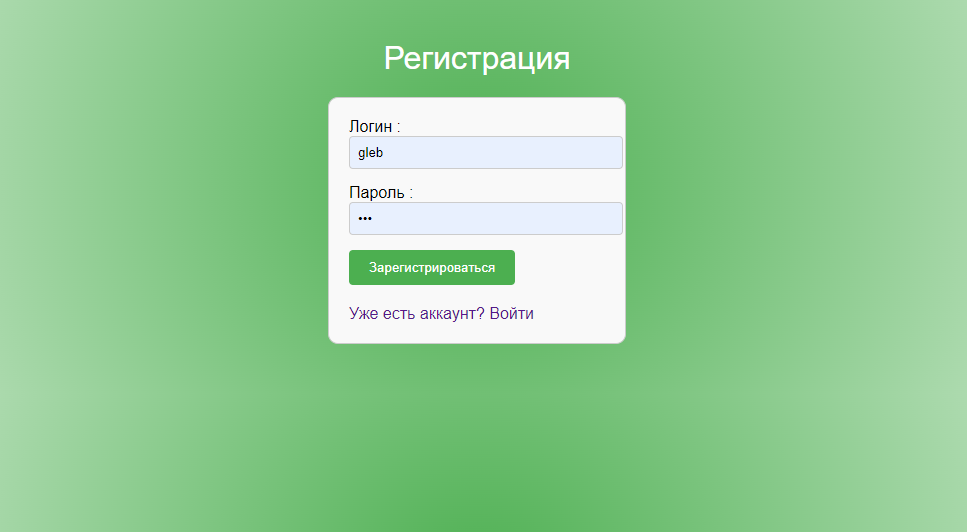


Рисунок 41. Регистрация.

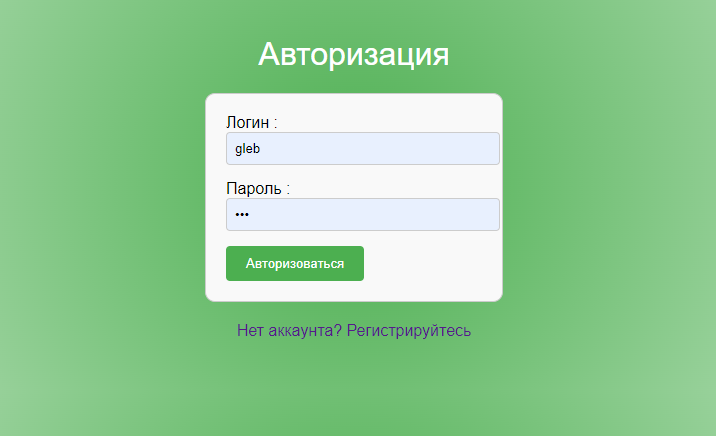


Рисунок 42. Авторизация.

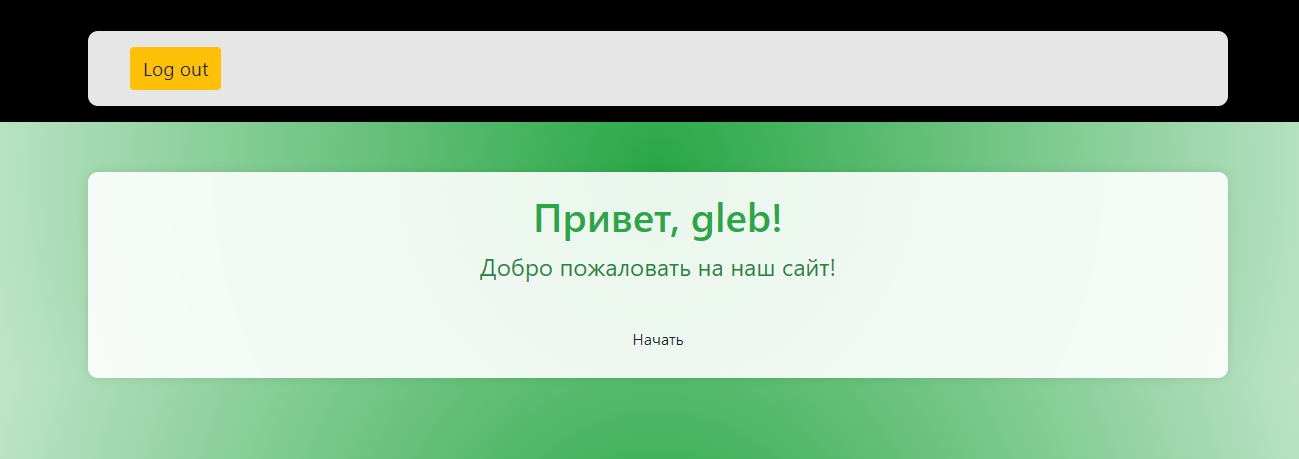


Рисунок 43. Успешная авторизация.

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.

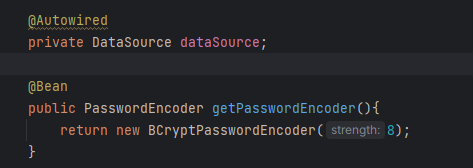


Рисунок 44. Метод шифрования пароля.

Добавляем 2 новые роли.

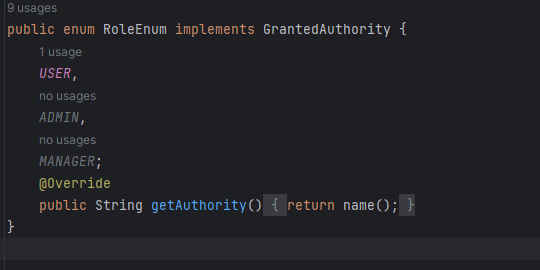


Рисунок 45. Новые роли.

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.

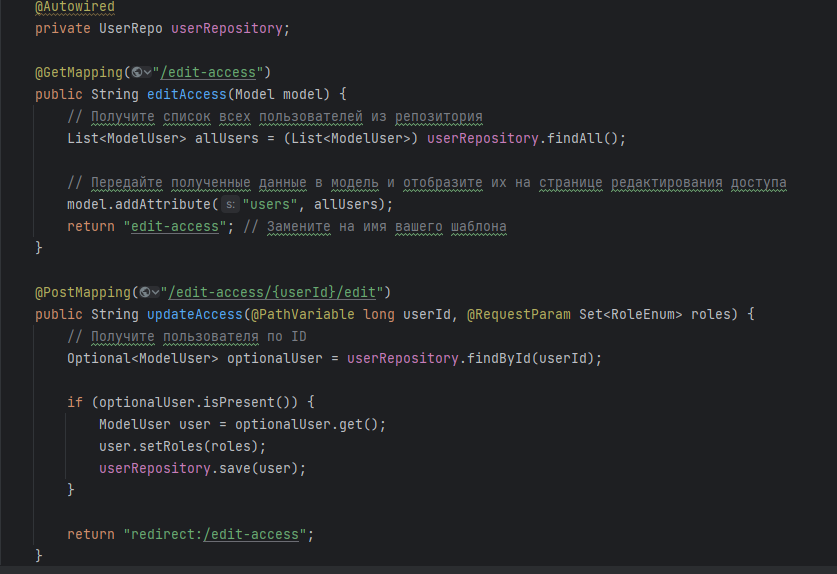


Рисунок 46. UserAcessController.

На разные контроллеры задаем разные роли.



Рисунок 47. Пример настройки (1).

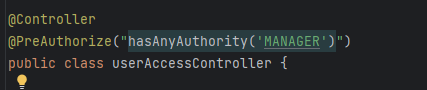


Рисунок 48. Пример настройки (2).

Результаты:

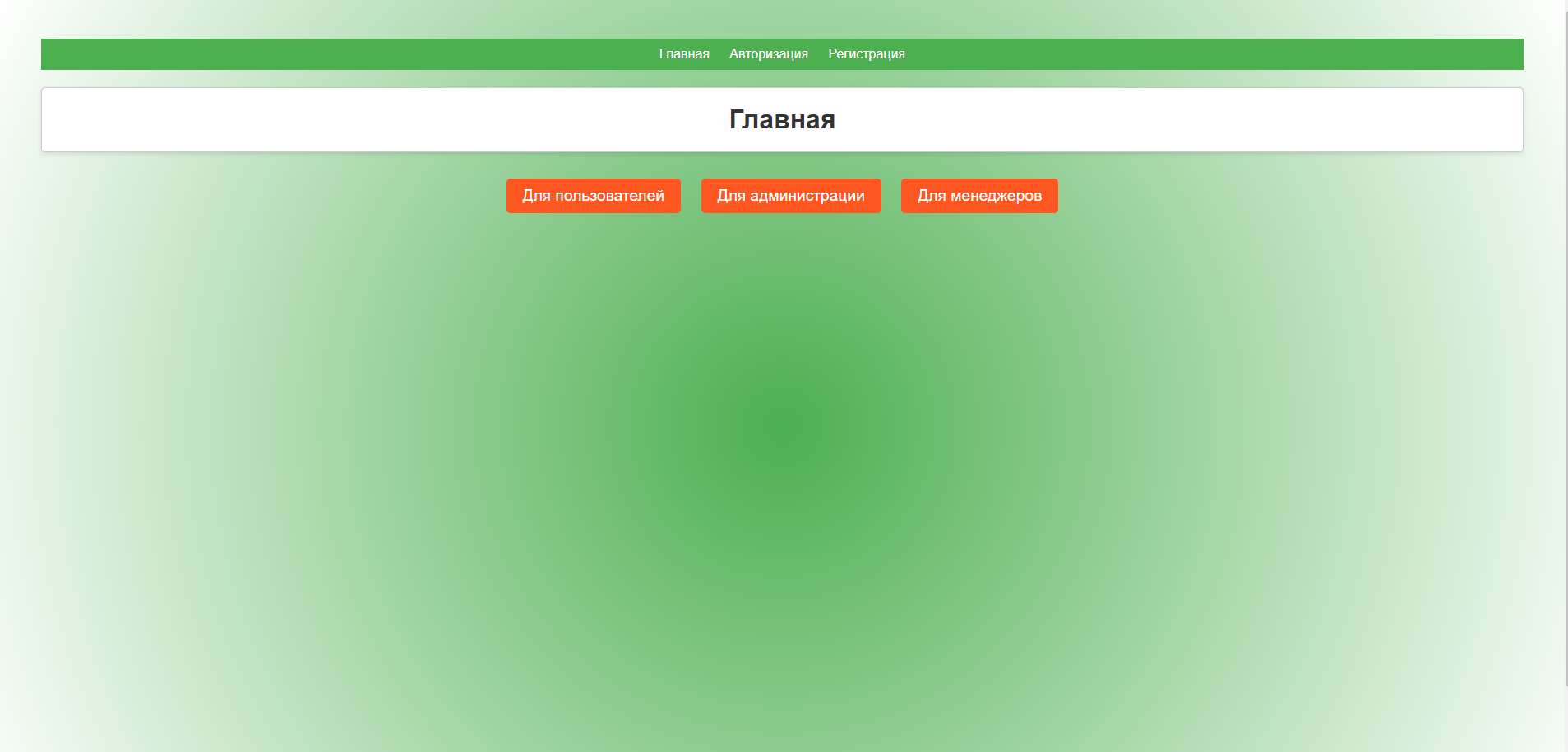


Рисунок 49. Главная страница.

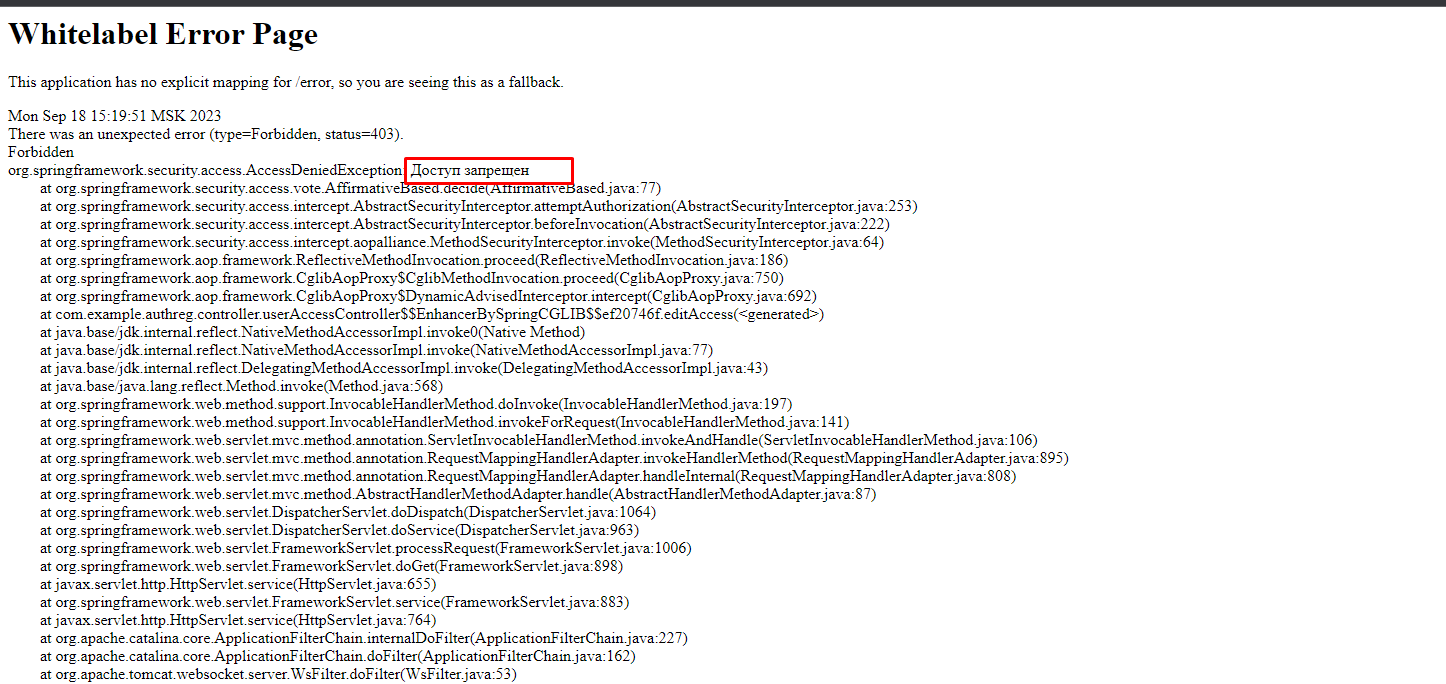


Рисунок 50. Попытка входа на страницу для менеджера, под аккаунтом пользователя.

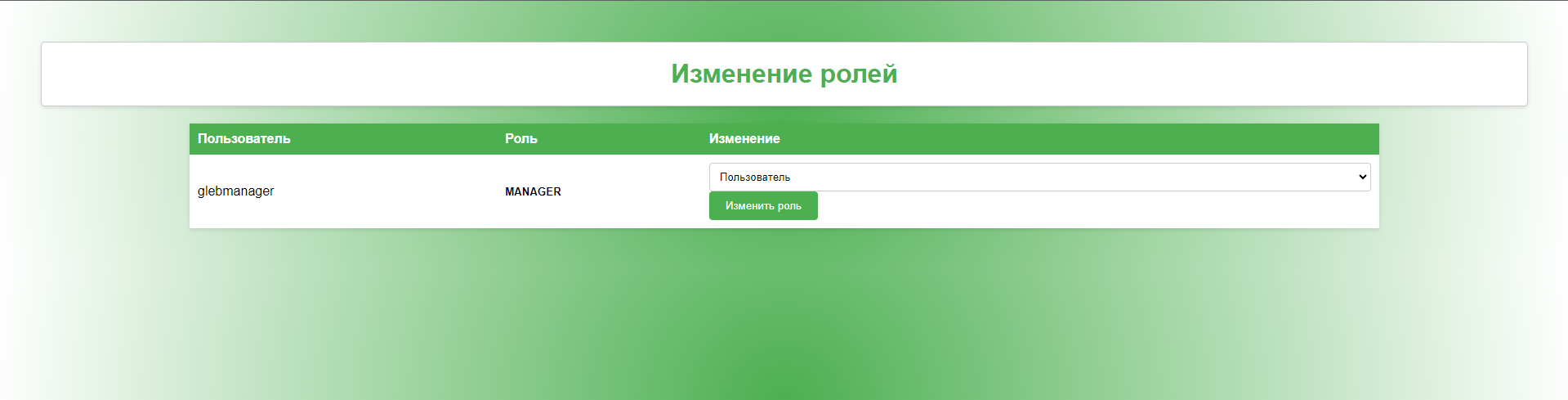


Рисунок 50. Изменение ролей.

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:

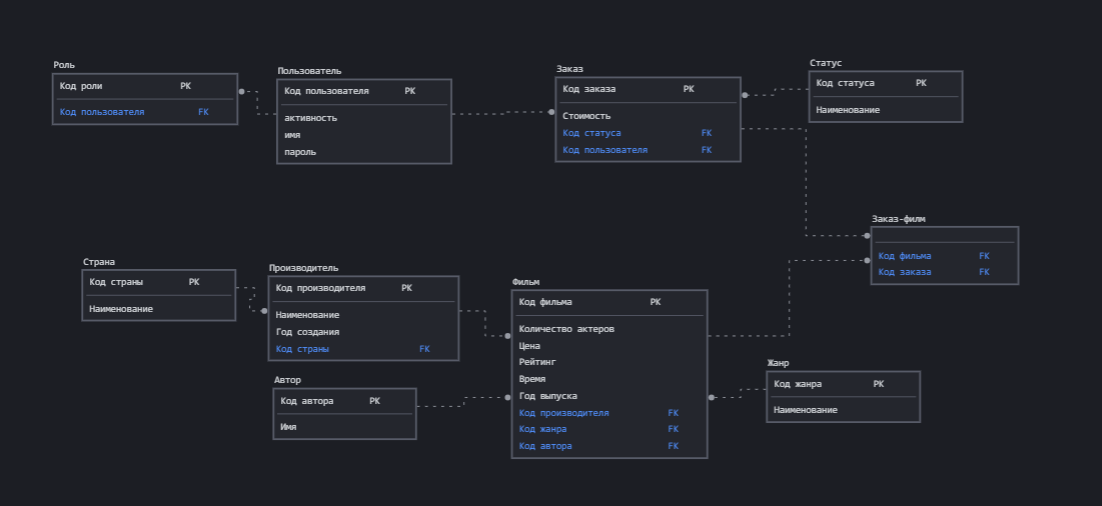


Рисунок 51 – Инфологическая модель базы данных*.*

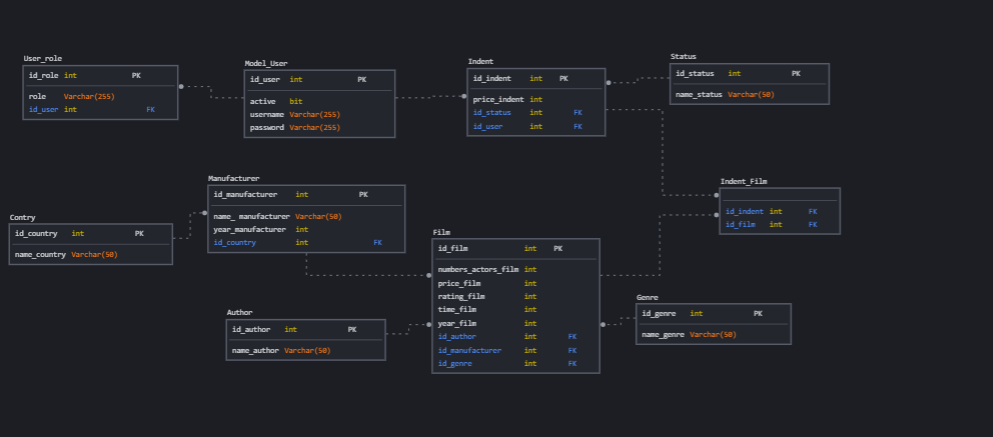


Рисунок 52 - Даталогическая модель базы данных.

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 | autrhor\_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\_author 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 fkepq0d9lvjpmd64hrl1u1ossuk FOREIGN KEY (id\_author)

REFERENCES public.author (id\_author) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fkknwqkkdqxrymsqj9lm7h64r2b 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 fkfkm93bqwu9aw0epsa26enfq5l 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 fks4p4eftcxvj7s1ljl7t86lu6d 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";

}

}

userAcessController

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">

<img src="/images/kino.png" alt="Kinopoisk 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">

<img src="/images/kino.png" alt="Kinopoisk 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">

&copy; 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">

<img src="/images/kino.png" alt="Kinopoisk 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>

<img src="/images/zelmil.jpg" alt=" ">

<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>

<img src="/images/shoushenk.jpg" alt=" ">

<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>

<img src="/images/forestgamp.jpg" alt=" ">

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

</li>

</ul>

</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">

<img src="/images/letcor.jpg" alt="Летучий корабль"> <br><br>

<a href="https://youtu.be/WHX-YBljTcE" target="\_blank">Смотреть трейлер</a>

</div>

<div class="trailer-item">

<img src="/images/papdoch.jpg" alt="Папины дочки Новые"><br><br>

<a href="https://youtu.be/4UECwKJdKF4" target="\_blank">Смотреть трейлер</a>

</div>

<div class="trailer-item">

<img src="/images/chuzay.jpg" alt="Чужая"><br><br>

<a href="https://youtu.be/v-2V4pQlN8U" target="\_blank">Смотреть трейлер</a>

</div>

</div>

</section>

</main>

<footer class="kinopoisk-footer">

<div class="footer-content">

<div class="footer-logo">

<img src="/images/kino.png" alt="Kinopoisk 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;

}

/\* Стиль 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;

}

/\* Стиль 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>

</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 производители</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;

}

/\* Стиль 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;

}

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;

}

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;

}

/\* Стиль 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">

<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;

}

/\* Стиль 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;

}

/\* Стиль 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;

}

.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>

<dependency>

<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.

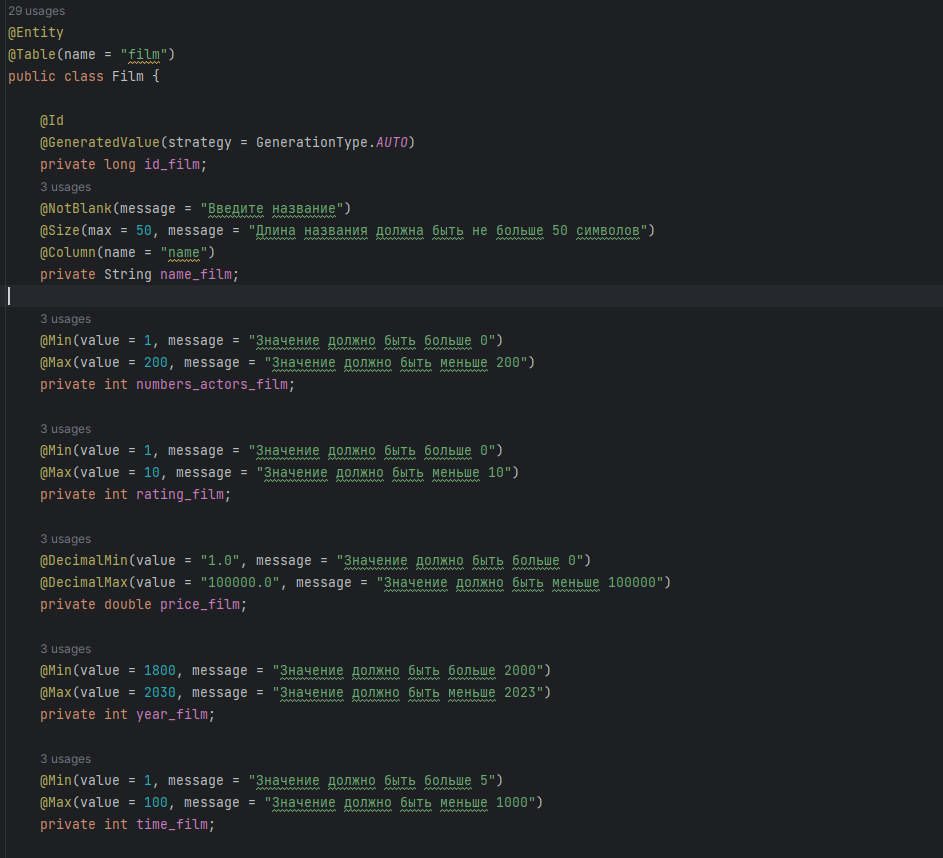


Рисунок 53 - Пример модели.

Создаем к моделям репозитории.

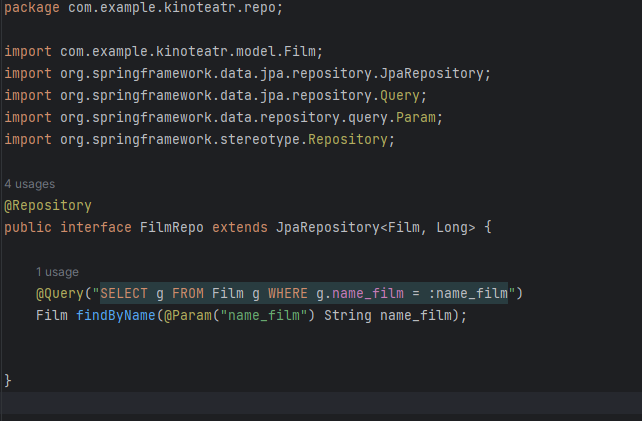


Рисунок 54 - Пример репозитория.

Далее создаем контроллеры, и прописываем в них логику работы с данными.

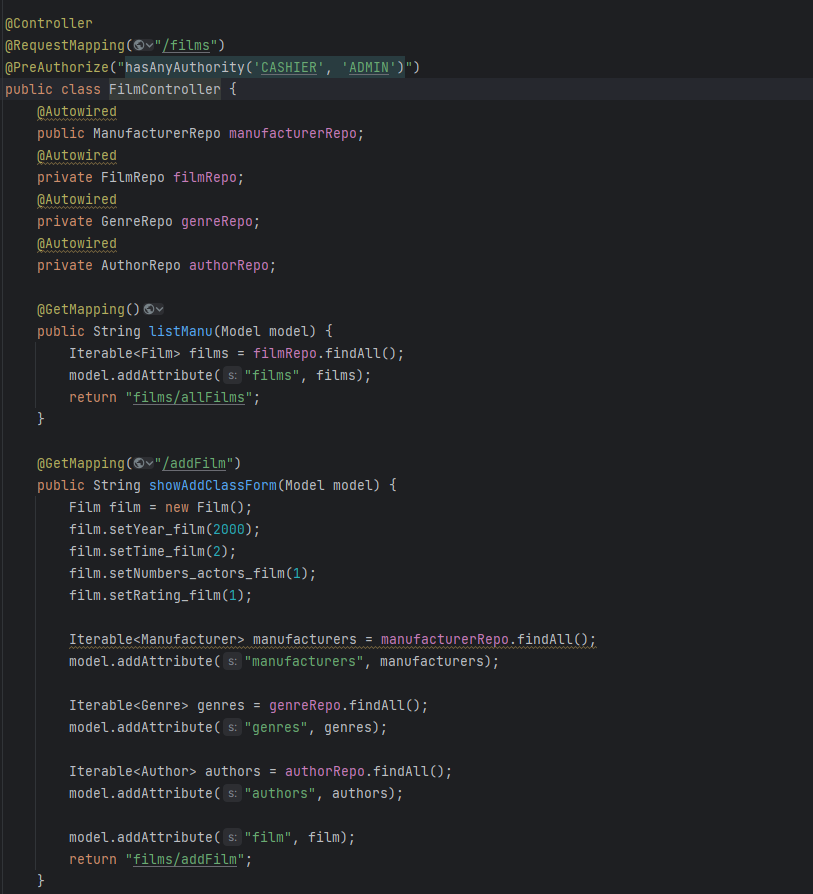


Рисунок 55 - Пример контроллера.

Создаем разметку для вывода данных.

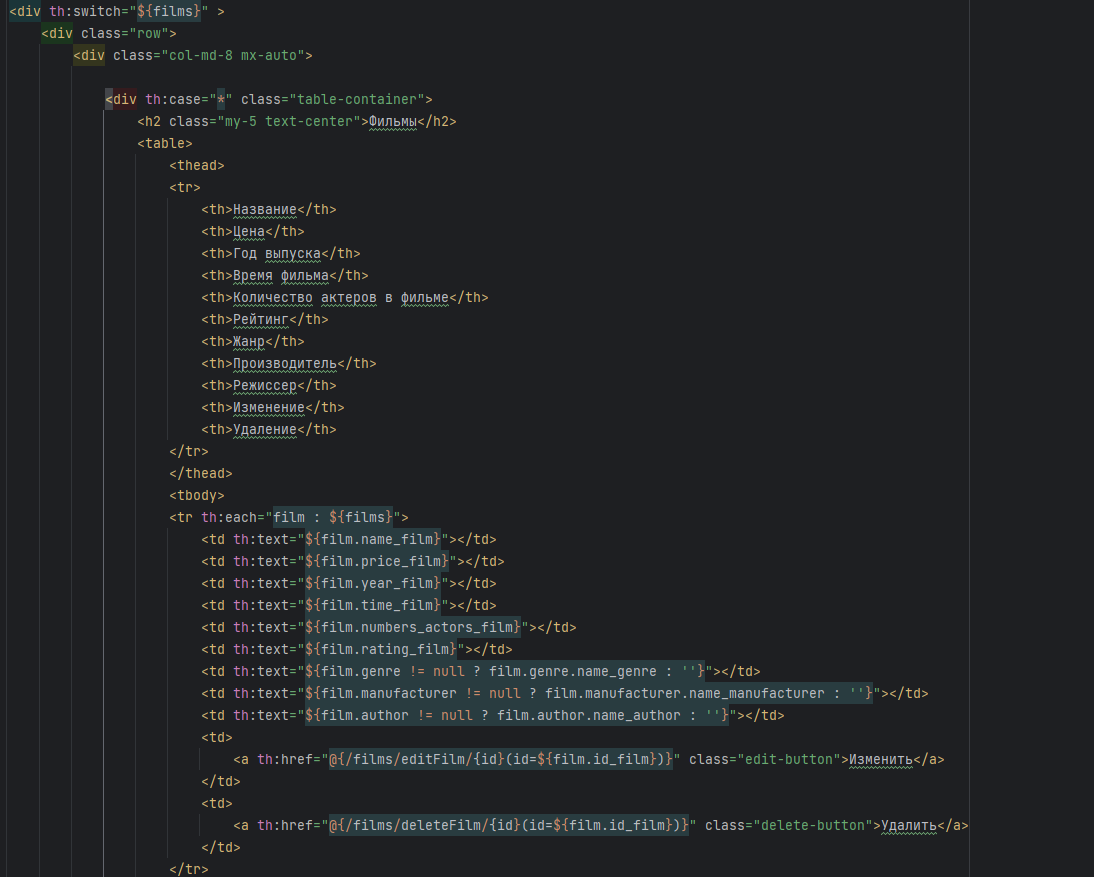


Рисунок 56 - Пример разметки.

Результат работы программы:

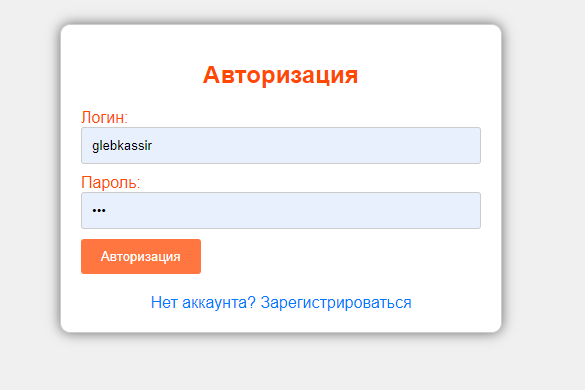


Рисунок 57 – Авторизация.

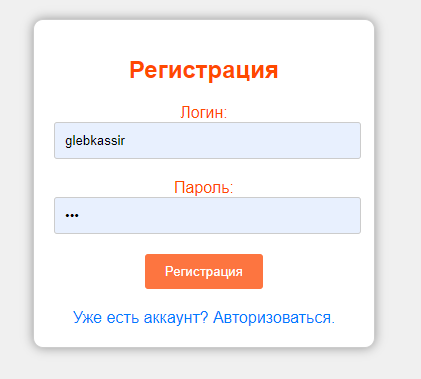


Рисунок 58 – Регистрация.



Рисунок 59 – Главная.



Рисунок 60 – Заказы.

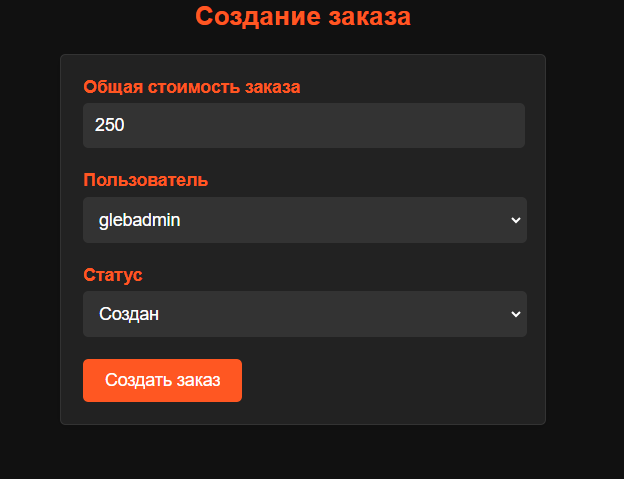


Рисунок 61 – Добавить заказ.

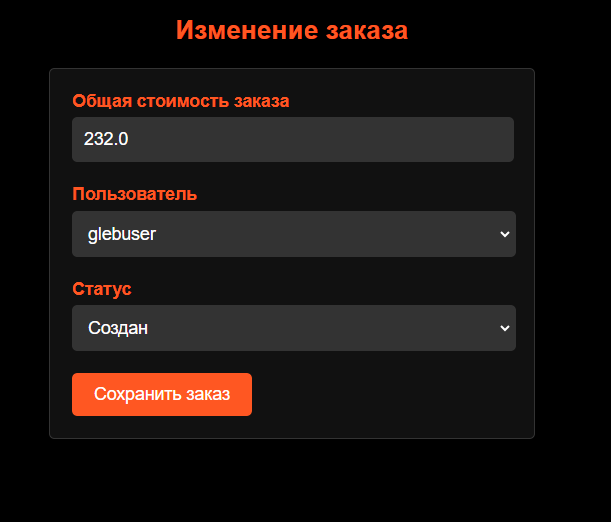


Рисунок 62 – Изменить заказ.

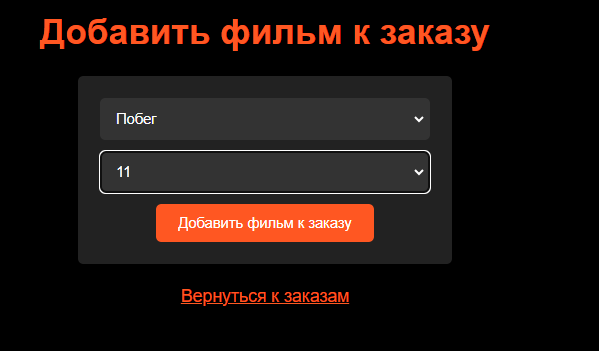


Рисунок63– Добавить фильм к заказу.

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 техлогии 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.