|  |
| --- |
| **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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Дрюпин

*подпись*

Группа П50-1-18

Руководитель по практической подготовке от техникума

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Д. Буканов

*подпись*

«30» ноября 2021 года

Москва 2021

Оглавление

[Практическая работа №1 3](#_Toc89079648)

[Калькулятор с Get и Post запросами 3](#_Toc89079649)

[Практическая работа №2 7](#_Toc89079650)

[Работа с БД. Добавление и вывод данных на страницу 7](#_Toc89079651)

[Практическая работа №3 14](#_Toc89079652)

[Работа с БД. Поиск и вывод данных из таблицы 14](#_Toc89079653)

[Практическая работа №4 19](#_Toc89079654)

[Работа с БД. Редактирование и удаление данных из таблицы 19](#_Toc89079655)

[Практическая работа №5 22](#_Toc89079656)

[Работа с БД. Валидация 22](#_Toc89079657)

[Практическая работа №6 27](#_Toc89079658)

[Работа с БД. Связи 27](#_Toc89079659)

[Индивидуальный проект 35](#_Toc89079660)

[Склад радиокомпонентов 35](#_Toc89079661)

# Практическая работа №1

## Калькулятор с Get и Post запросами

Цель работы: Создать простой калькулятор. Научиться использовать запросы Post и Get.

Результат работы:

1. Для создания проекта, нужно зайти на сайт https://start.spring.io/ , выбрать нужные нам параметры и зависимости, нажать кнопку Generate. После этого скачивается архив с проектом, который нужно запустить в IntelliJ IDEА

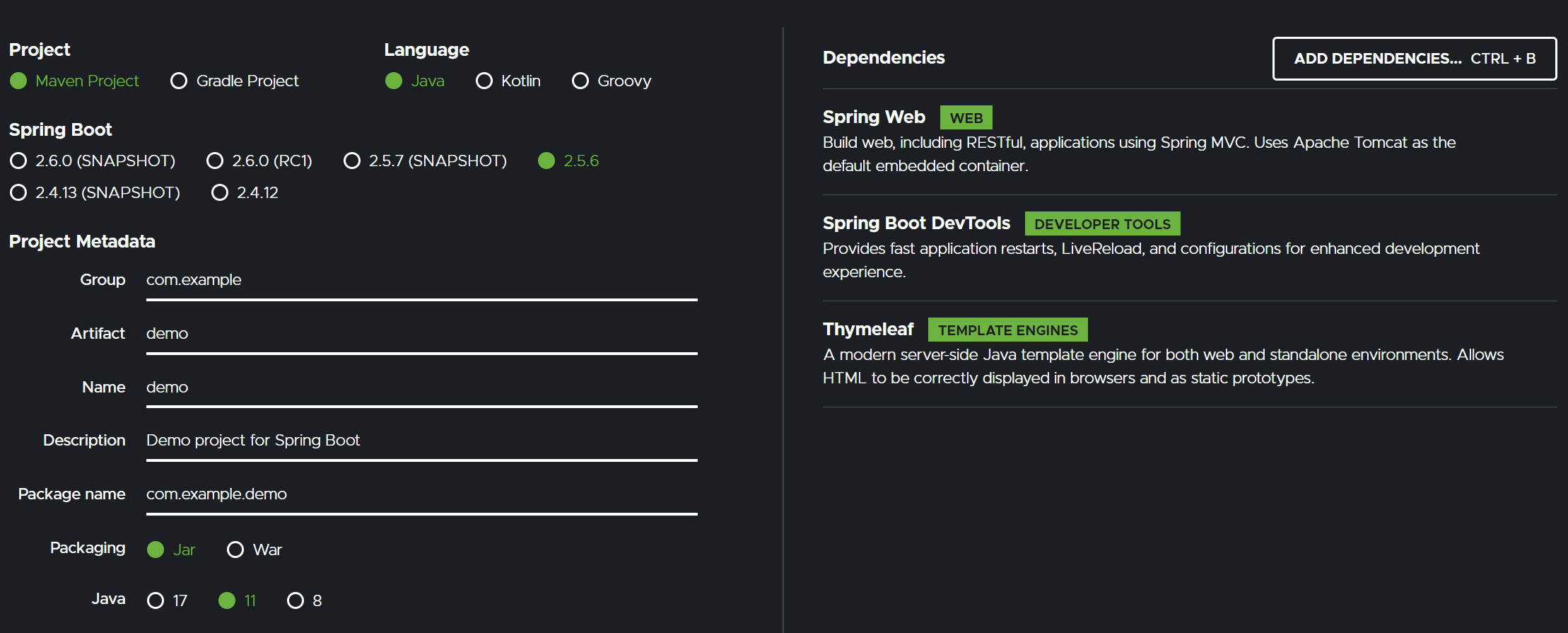


Рисунок 1 - Генерация проекта

Thymeleaf – современный серверный движок шаблонов Java как для вебсред, так и для автономных сред. Позволяет корректно отображать HTML в браузерах и в виде статических прототипов.

Spring Web – создание веб-приложения, с помощью Spring MVC.

Spring Boot DevTools – обеспечивает быстрый перезапуск приложений,

LiveReload и настройки для улучшения работы разработчиков.

1. Создаем контроллер и при помощи @GetMapping указываем, какую страницу необходимо возвращать при вызове данного контроллера.

@Controller указывает на то, что класс является контроллером.

@PostMapping указывает на то, что метод будет обрабатывать Postзапросы.

@Configuration указывает, что объект является источником определений bean-компонентов. Вызовы методов @Bean в классах @Configuration также можно использовать для определения зависимостей между компонентами.

@GetMapping указывает на то, что метод будет обрабатывать Getзапросы.

@RequestParam указывает на то, что параметры метода должны быть привязаны к параметру веб-запроса.

Первый метод @GetMapping мы используем для того, чтобы при запуске сразу открывался наш шаблон, а не вылетала ошибка.

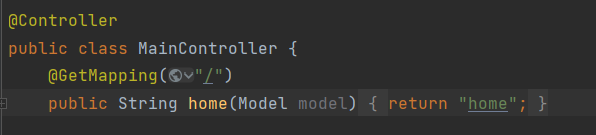


Рисунок 2 - Контроллер

1. Создаем метод, в котором будут производится вычисления значений, полученных методом Post. Чтобы значения отображались на той же странице, возвращаем текущую страницу (не производим перенаправлений на другие страницы).



Рисунок 3 - Обработка метода Post

1. Аналогично с предыдущим пунктом создаем метод для вычисления значений, но теперь полученных методом Get.



Рисунок 4 - Обработка метода Get

1. Создаем Html страницу с элементами ввода и управления, откуда будут браться значения и куда будет записываться результат. Для оформления используем CSS и Bootstrap.

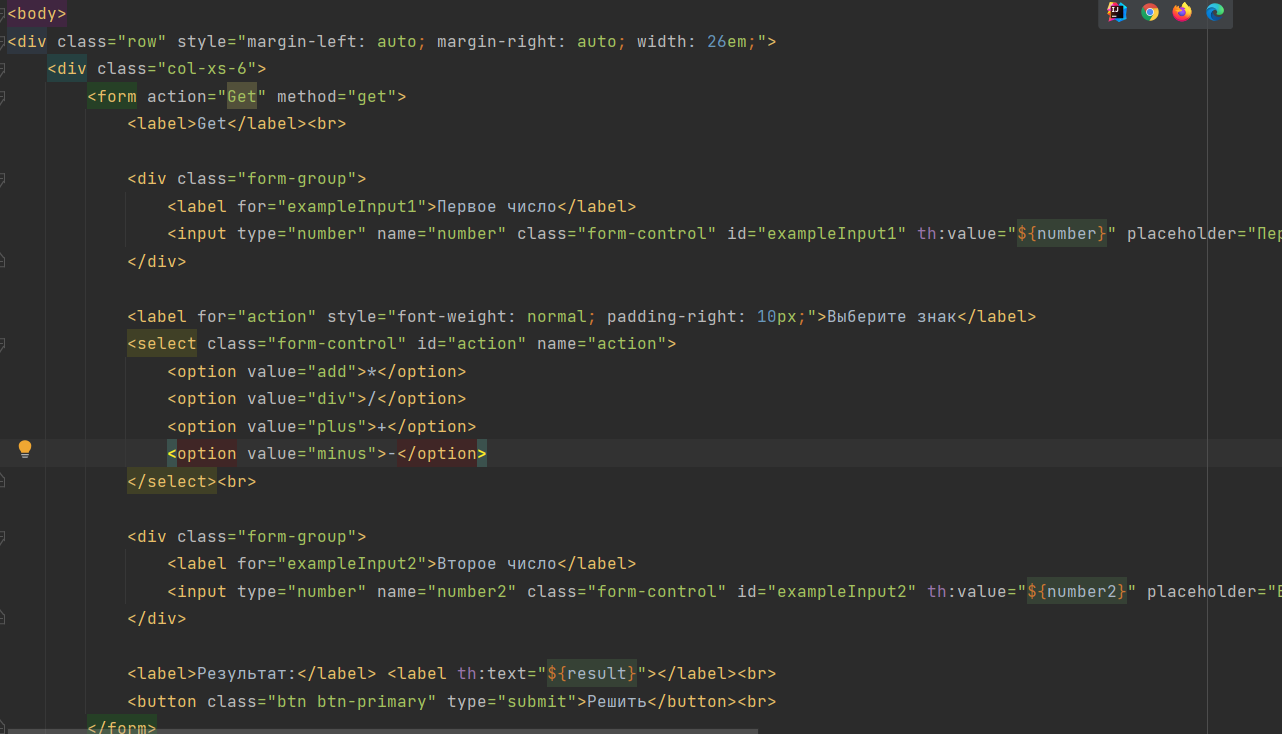


Рисунок 5 - Фрагмент файла home.html

1. Открываем в браузере «localhost:8080» и проверяем работу.

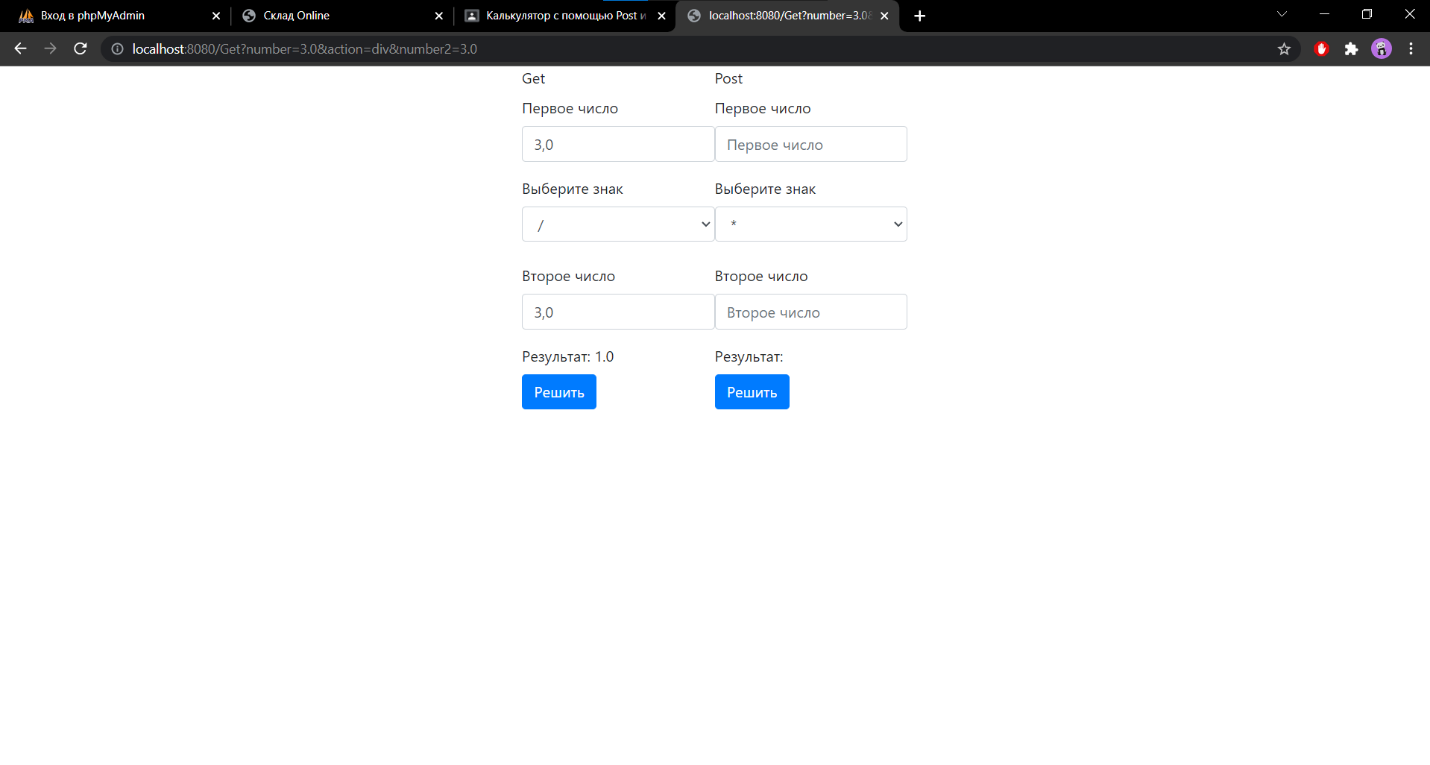


Рисунок 6 - Демонстрация метода Get

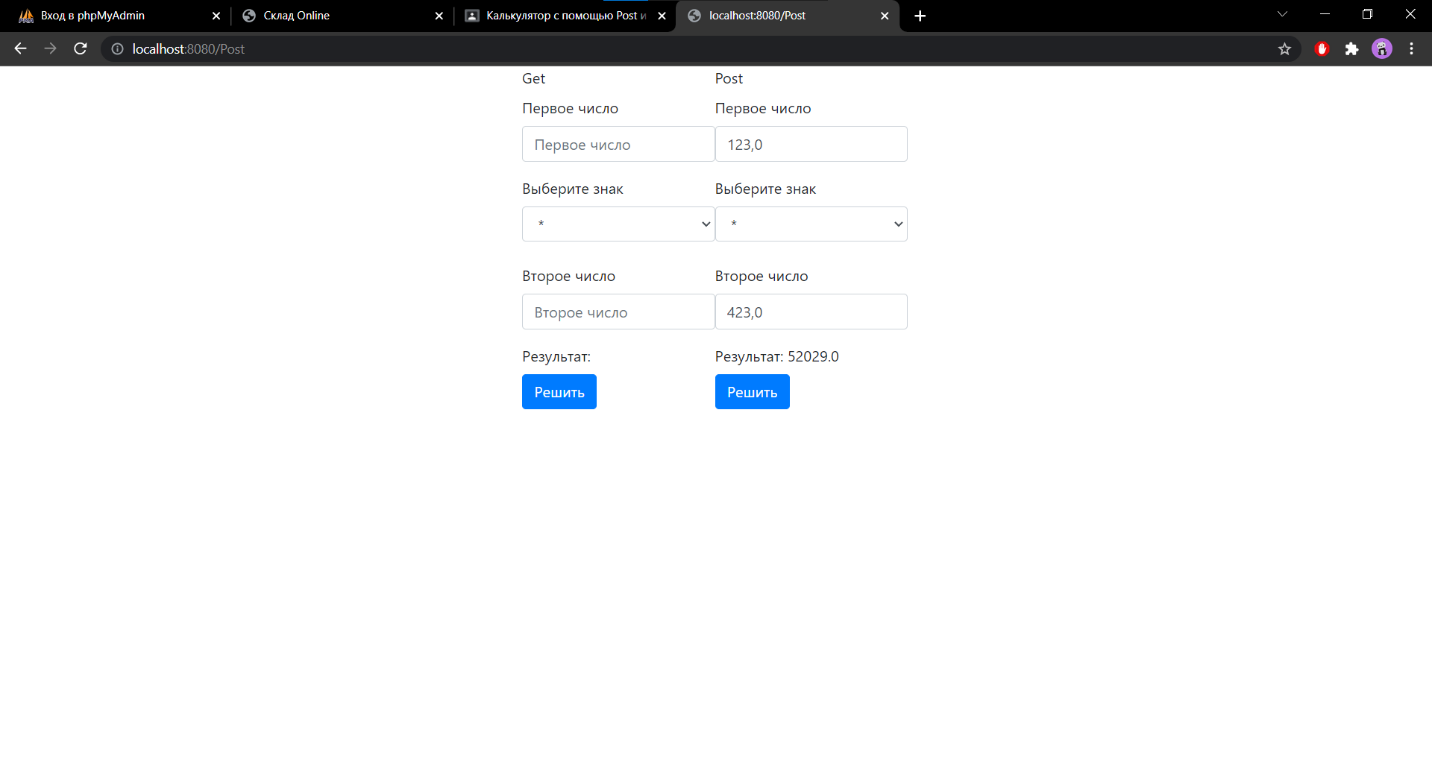


Рисунок 7 - Демонстрация метода Post

Вывод: Научились работать с методами Get и Post, и создали простейший калькулятор.

# Практическая работа №2

## Работа с БД. Добавление и вывод данных на страницу

Цель работы: Cделать добавление и вывод данных для 5 таблиц из MySQL используя Spring Data.

Результат работы:

1. Добавляем в ранее созданный шаблон проекта зависимости:
   1. Spring Web – создание веб-приложения, с помощью Spring MVC;
   2. Spring Boot DevTools – обеспечивает быстрый перезапуск приложений, LiveReload и настройки для улучшения работы разработчиков;
   3. Thymeleaf – современный серверный движок шаблонов Java как для вебсред, так и для автономных сред. Позволяет корректно отображать HTML в браузерах и в виде статических прототипов;
   4. Spring Data JPA — дополнительный удобный механизм для взаимодействия с сущностями базы данных, организации их в репозитории, извлечение данных, изменение;
   5. mysql-connector-java - предоставляет приложениям Java возможность соединения с базой данных при помощи драйвера MySQL Connector, который реализует интерфейс API Java Database Connectivity (JDBC);
   6. Commons-codecs - это «большая коллекция маленьких Java-утилит»;
2. Создаем модель Post с аннотацией @Entity. Аннотацией @Entity отмечаются классы, которые представляют собой сущности базы данных. Аннотация @Id указывает на то, что поле ниже является первичным ключом. Далее указываем поля модели и создаем Getters и Setters. Все остальные модели создаем подобным образом.

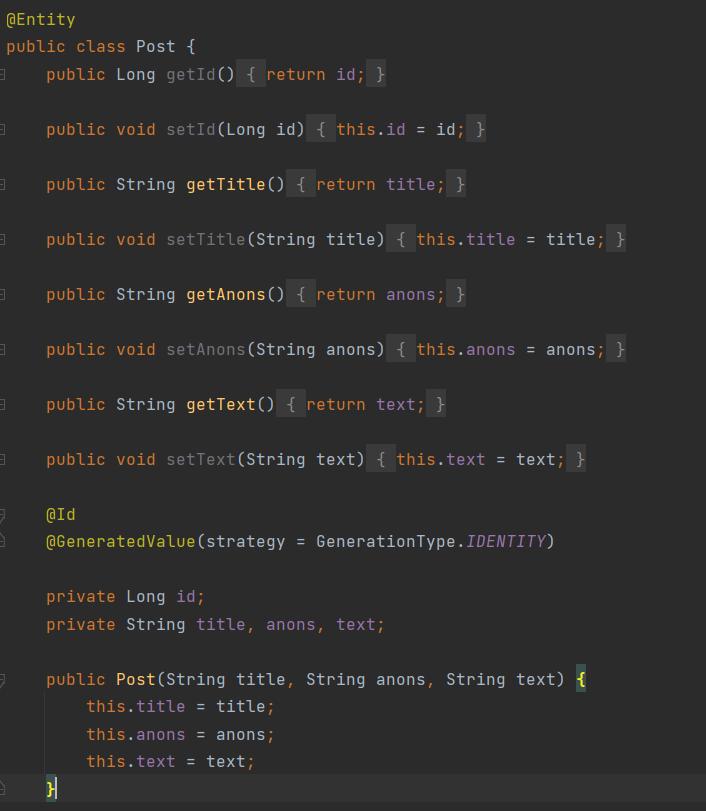


Рисунок 8 – Модель

1. Основное понятие в Spring Data — это репозиторий. Это несколько интерфейсов которые используют JPA Entity для взаимодействия с ней. Репозиторий наследует сущность.

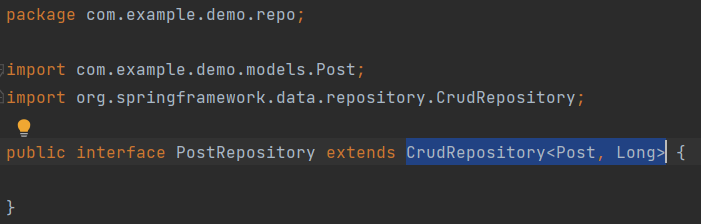


Рисунок 9 - Репозиторий

1. Создаем метод, в котором будет располагаться основная логика. Свойства класса с аннотацией @Autowired заполняются соответствующими значениями сразу после создания bean'а и перед тем, как любой из методов класса будет вызван. Далее обрабатываем @GetMapping при открытии основной страницы. Перечисляем все значения моделей и размещаем их в атрибуте, чтобы Thymeleaf мог работать с ними. Создаем @PostMapping, где при помощи Switch определяем, какие данные были отправлены и в какую таблицу их нужно записать. Сами данные получаем по ключам при помощи @RequestParam.

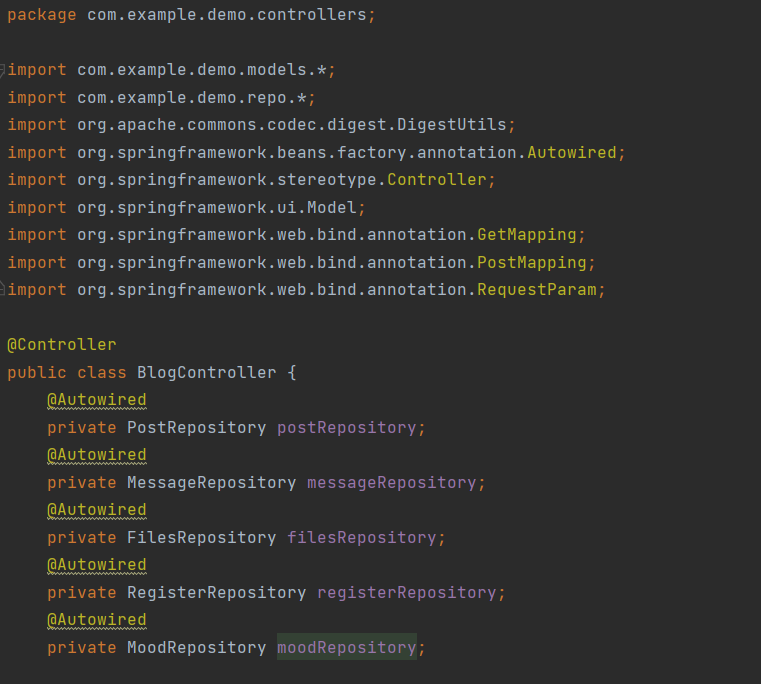


Рисунок 10 – Метод



Рисунок 11 – @GetMapping

По правилам безопасности запрещено хранить пароль в БД в открытом виде, поэтому добавляем к полученному значению пароля криптографическую соль, вычисляем хеш-сумму и полученный результат сохраняем в БД.

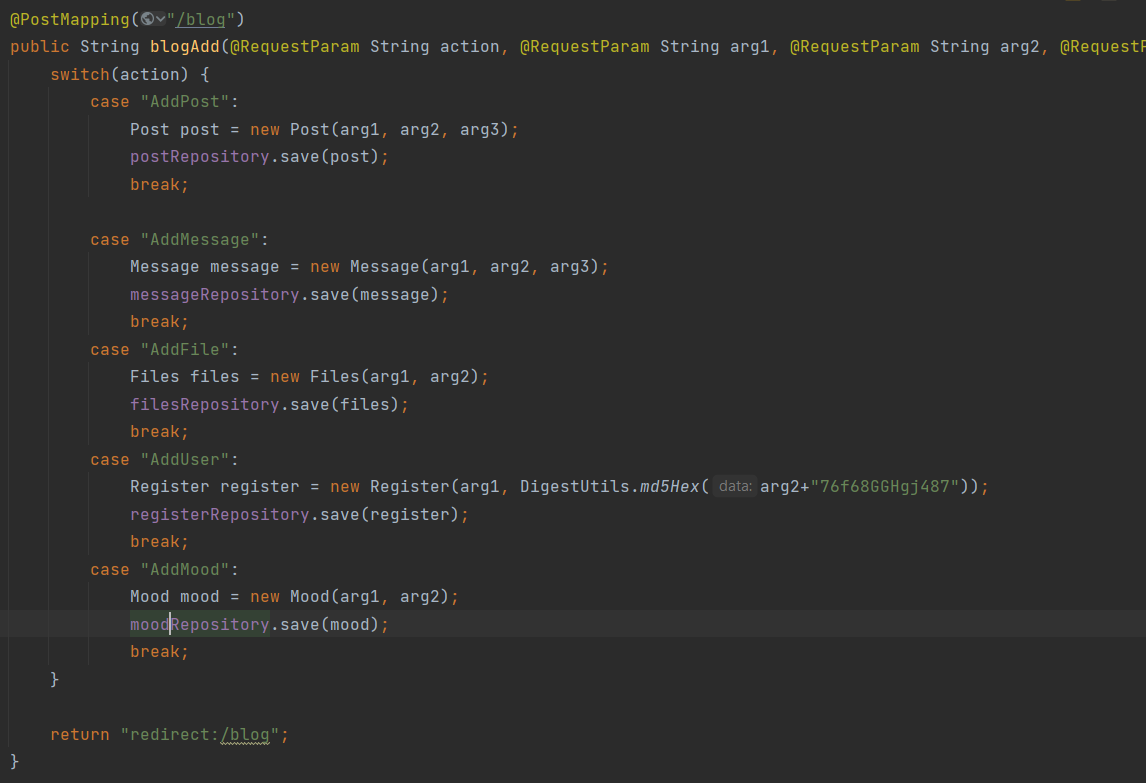


Рисунок 12 - @PostMapping

1. Создаем Html страницу с элементами ввода и управления, откуда будут браться значения и куда будет выводится результат. Для оформления используем CSS и Bootstrap.

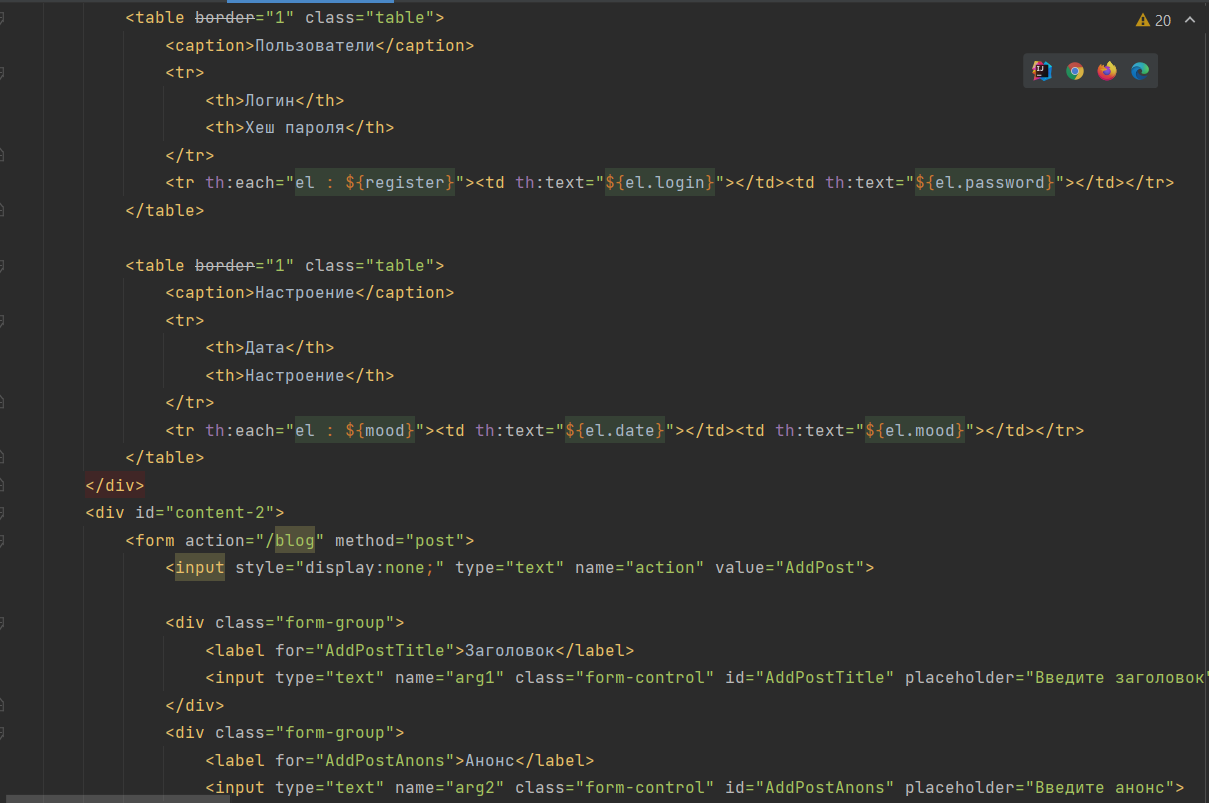


Рисунок 13 - Фрагмент файла html

1. Открываем в браузере «localhost:8081» и проверяем работу.

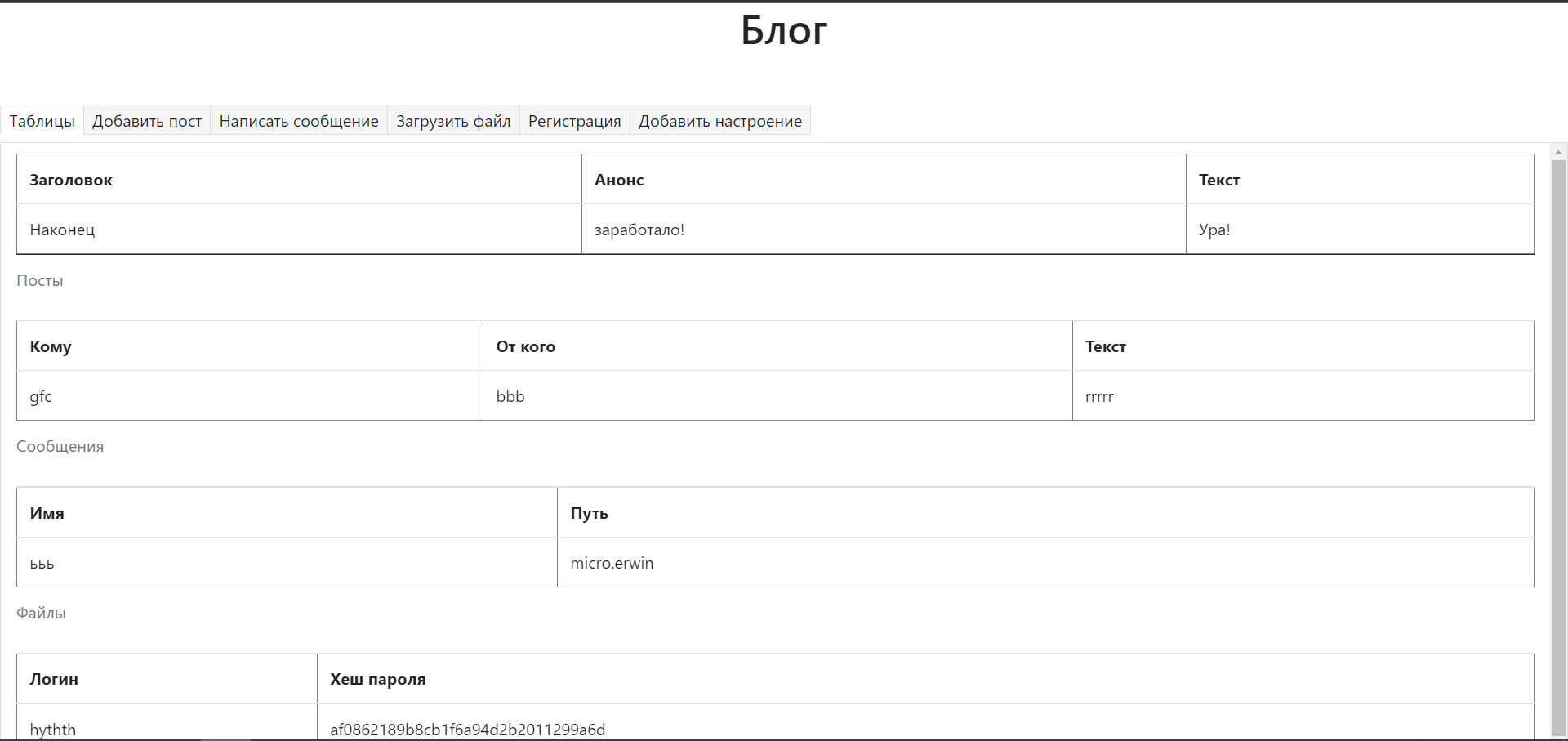


Рисунок 14 - Демонстрация таблиц

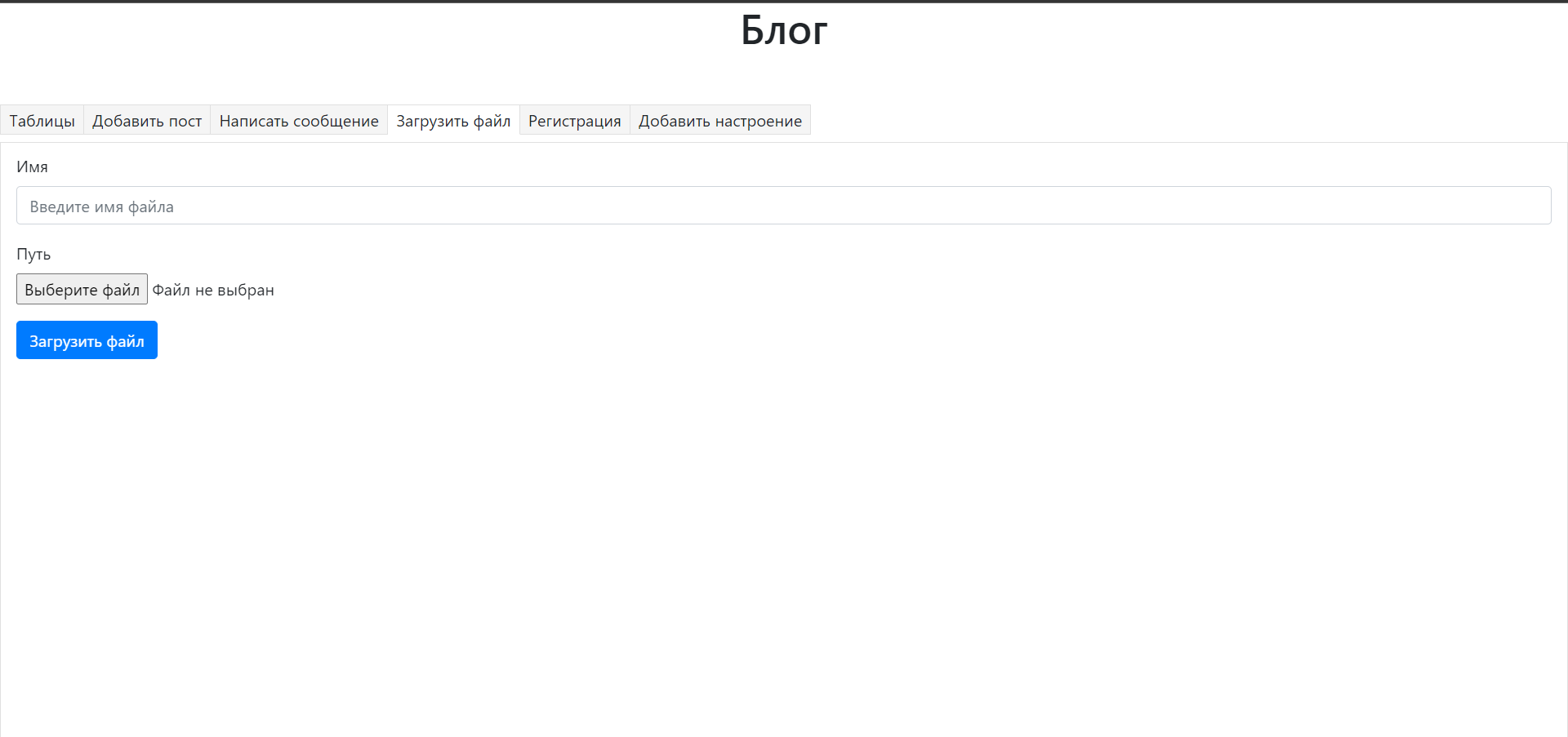


Рисунок 15 - Демонстрация формы добавления значений

Вывод: Cделали добавление и вывод данных для 5 таблиц из MySQL используя Spring Data.

# Практическая работа №3

## Работа с БД. Поиск и вывод данных из таблицы

Цель работы: Cделать поиск и подробный вывод данных из таблиц MySQL используя Spring Data.

Результат работы:

1. Начнем с поиска. В каждый репозиторий добавим findBy(элемент модели) для поиска по полному совпадению и findBy(элемент модели)Containing частичного совпадения запроса и искомого значения.

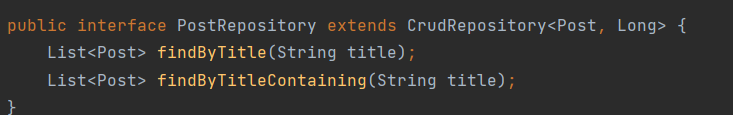


Рисунок 86 - Обновленный репозиторий

1. Создаем новую Html страницу, куда будем вводить запрос и выводить полученный результат. В контроллере создаем новый метод-обработчик запроса с аннотацией @PostMapping, которая получает значения Post запроса. При помощи Switch определяем, в какой таблице вести поиск, находим значение по запросу и передаем его модели.

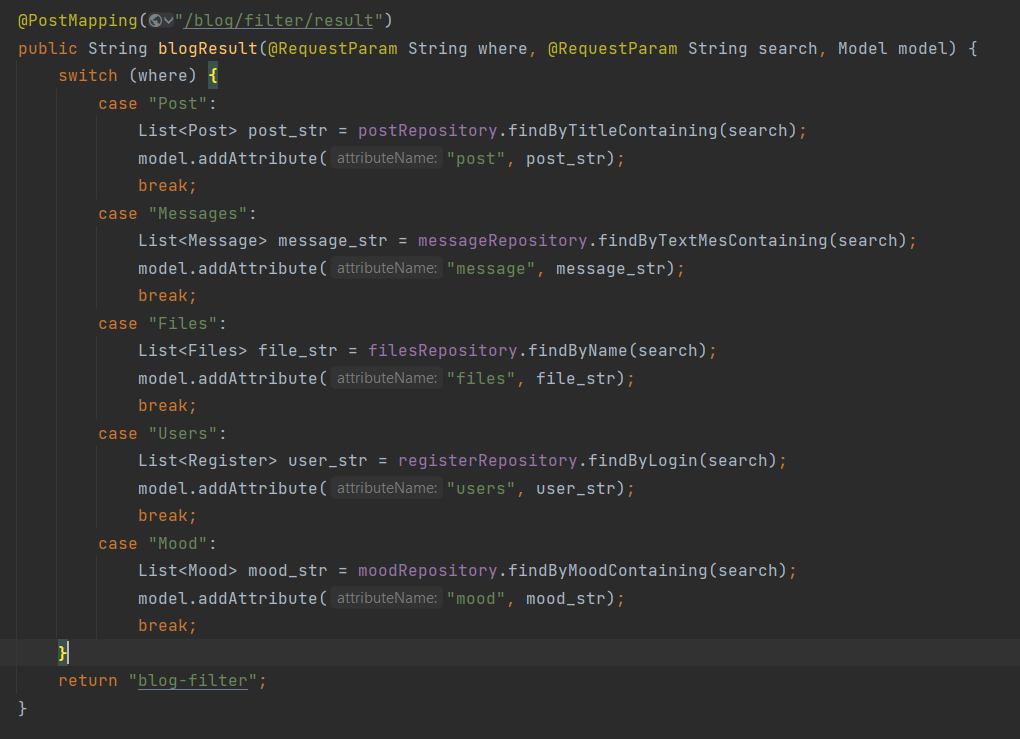


Рисунок 17 – Обработчик вывода

1. Далее сделаем подробный вывод значений. Для этого в последний столбец таблицы вставляем ссылку, ведущую на страницу с подробным результатом. Обрабатываем её в котроллере аннотацией @GetMapping и получаем динамическое значение при помощи аннотации @PathVariable. Ищем значение по репозиторию, переводим его в массив и выводим на страницу пользователю.



Рисунок 18 – Фрагмент кода

1. Создаем HTML страницу поиска

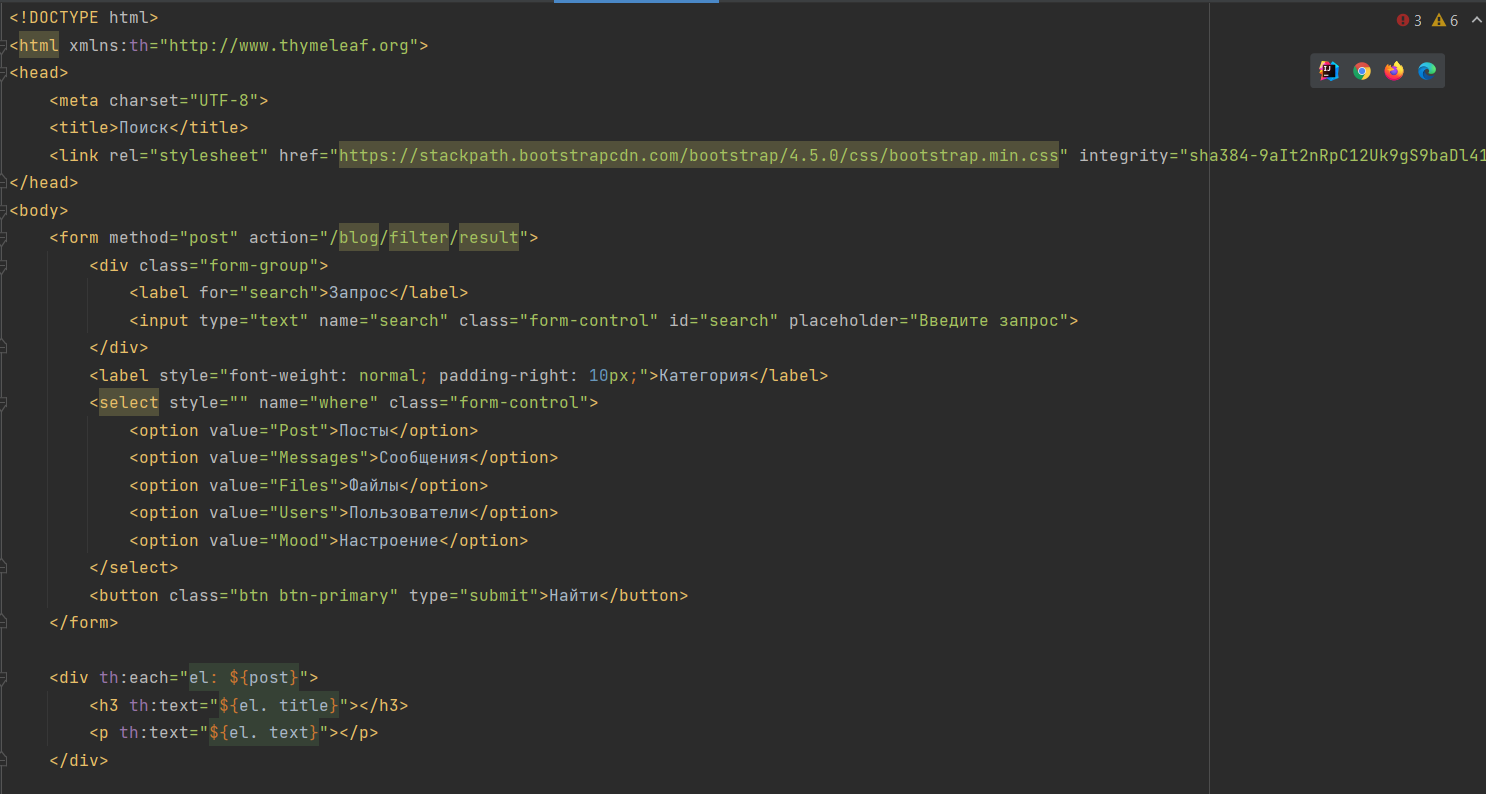


Рисунок 19 - Фрагмент кода

1. Создаем HTML страницу подробного вывода

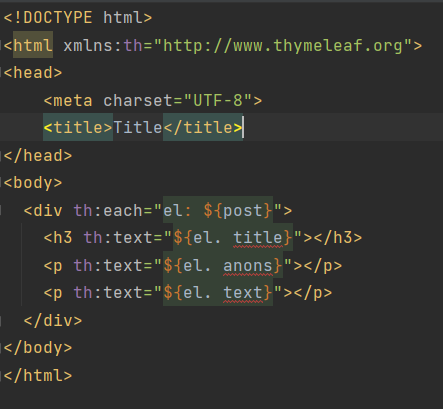


Рисунок 20 - Фрагмент кода

1. Открываем в браузере «localhost:8081» и проверяем работу.

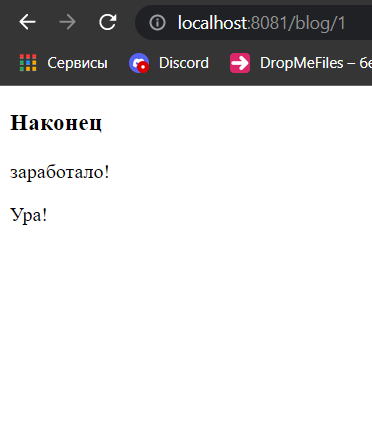


Рисунок 21 – Подробная информация

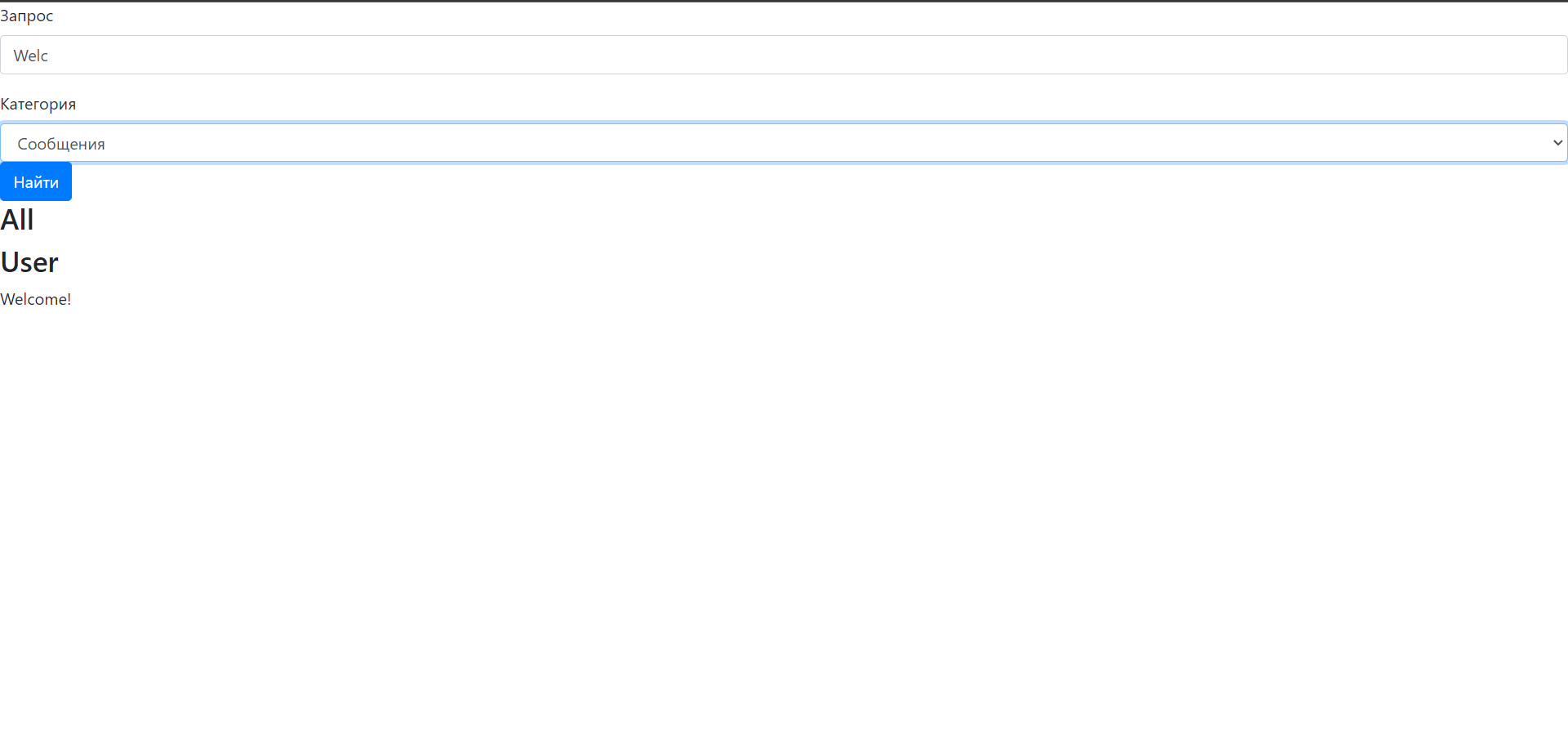


Рисунок 22 - Демонстрация поиска

Вывод: Cделали поиск и подробный вывод данных из таблиц MySQL используя Spring Data.

# Практическая работа №4

## Работа с БД. Редактирование и удаление данных из таблицы

Цель работы: Cделать удаление и изменение значений из таблиц MySQL используя Spring Data.

Результат работы:

1. Удаление. Создаем метод-обработчик, который при помощи аннотации @PathVariable получает id записи и id таблицы. При помощи конструкции switch-case выбирается нужная таблица, создается объект репозитория из которого удаляется запись по ранее переданному id.



Рисунок 23 - Удаление записи

1. Изменение. Создаем метод-обработчик, который при помощи аннотации @PathVariable получает id записи, ищет по этому id записи в репозитории, переводит их в массив и выводит в модель. Далее создаем обработчик Post, который также приминает id и параметры, которые необходимо перезаписать. Ищем запись в репозитории по id или выбрасываем исключение, задаем полю измененное значение и сохраняем репозиторий.



Рисунок 24 - Изменение данных

1. Создаем HTML файл. В нем расположены поля для ввода значений, которые необходимо изменить.



Рисунок 25 - html страница

1. Открываем в браузере «localhost:8081» и проверяем работу

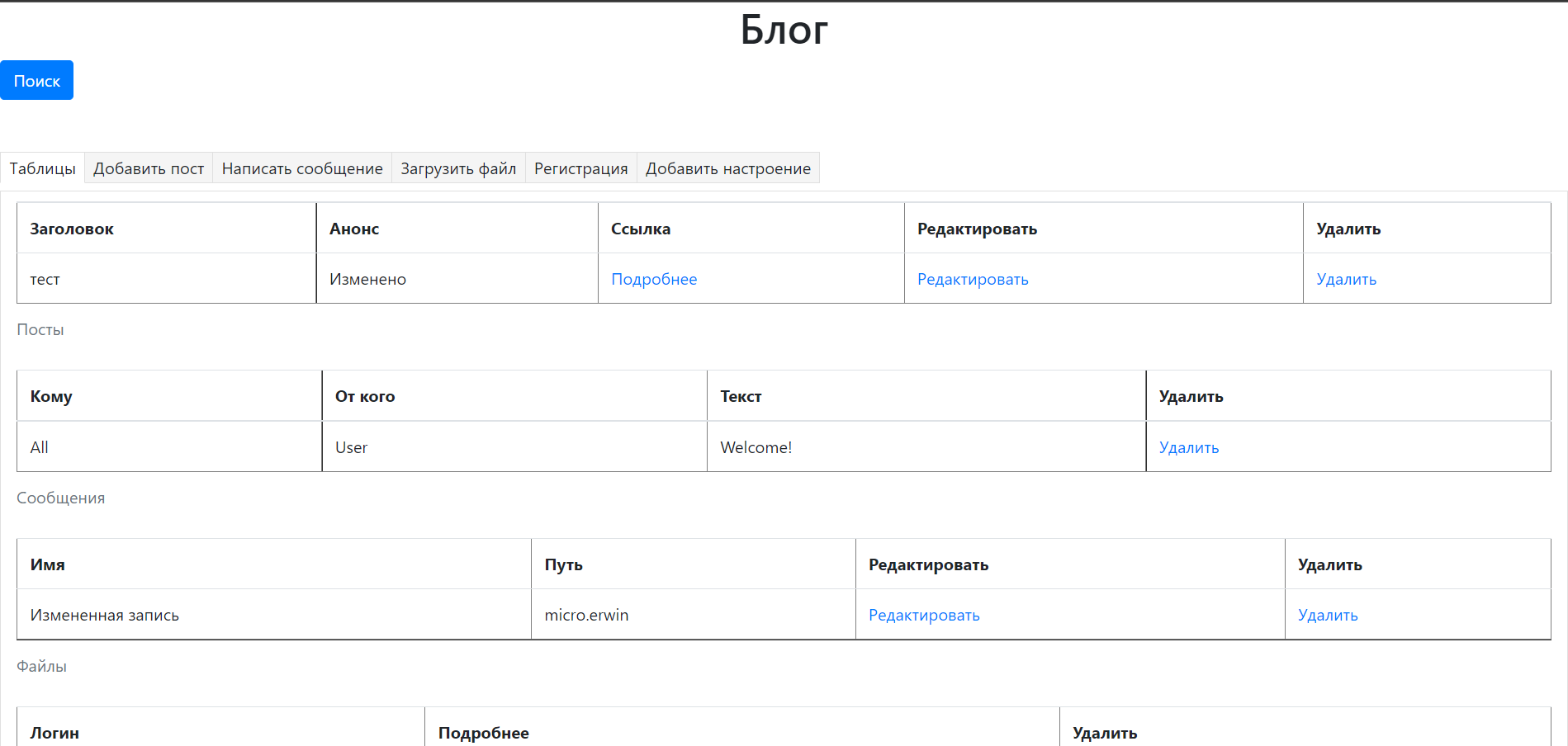


Рисунок 26 – Главная страница

1. Страница изменения записи

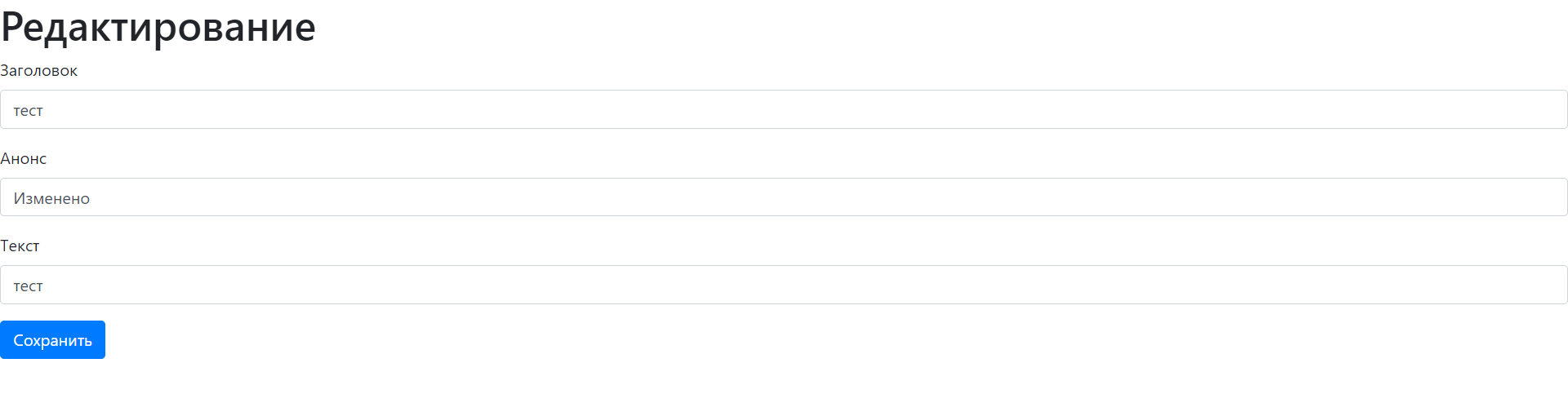


Рисунок 27 - Форма изменения записи

1. Удаление записи. Первый пост был удален.

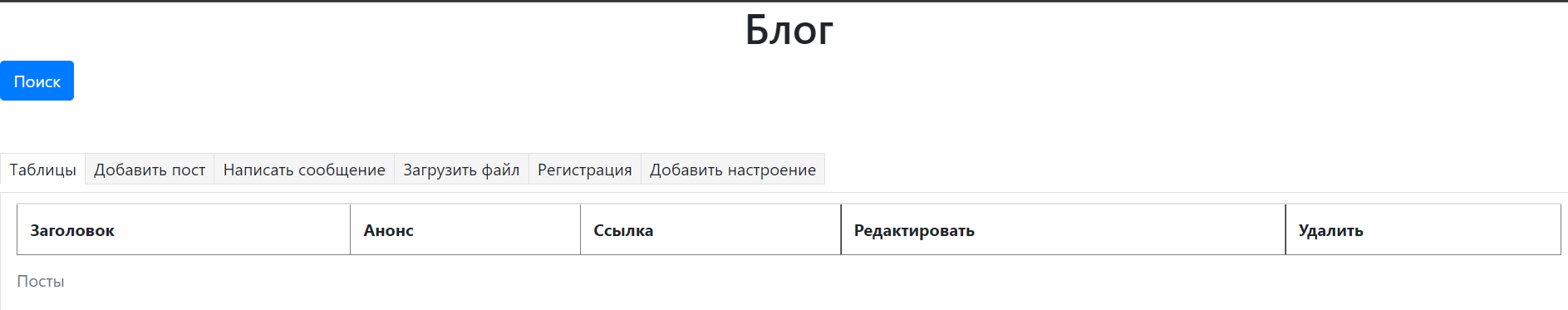


Рисунок 28 - Форма удаления записи

Вывод: Cделали удаление и изменение значений из таблиц MySQL используя Spring Data.

# Практическая работа №5

## Работа с БД. Валидация

Цель работы: Cделать верификацию полей форм при помощи верификатора.

Результат работы:

1. Добавляем зависимости. Hibernate validator – Валидатор, является эталонной реализацией Beam Validator.

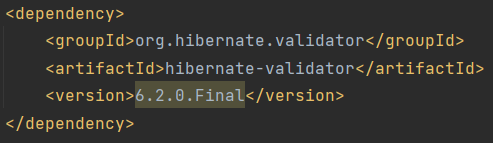


Рисунок 29 - Новая зависимость

1. Изменяем модели. Добавляем к каждому полю аннотации в зависимости от требований к этим полям:
   1. @DecimalMax - Применима к переменным типов BigDecimal, BigInteger, CharSequence, byte, short, int, long и их классов-обёрток. Значение должно быть меньше, либо равно указанному значению, либо быть null для непримитивов.
   2. @DecimalMin - Аналогична @DecimalMax, значение переменной должно быть числом и быть больше, либо равной указанной значению, либо быть null для непримитивов.
   3. @Digits - Количество символов слева от запятой должно быть меньше, либо равным integer, а справа — меньше, либо равным fraction, null является валидным значением. Применима к BigDecimal, BigInteger, CharSequence, byte, short, int, long и их классам-обёртам.
   4. @Max - Значение должно быть меньше, либо равно указанному значению, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   5. @Min - Значение должно быть больше, либо равно указанному значению, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   6. @Negative - Значение должно быть отрицательным, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   7. @NegativeOrZero - Значение должно быть отрицательным, равняться 0, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   8. @Positive - Значение должно быть положительным, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   9. @PositiveOrZero - Значение должно быть отрицательным, равняться 0, либо быть null. Применима к переменным типов BigDecimal, BigInteger, byte, short, int, long и их классов-обёрток.
   10. @Future - Значение переменной должно быть будущим временем. Применима к Date, Calendar и многим типам из пакета java.time.
   11. @FutureOrPresent - Значение переменной должно быть будущим либо настоящим временем. Применима к Date, Calendar и многим типам из пакета java.time
   12. @Past - Значение переменной должно быть прошедшим временем. Применима к Date, Calendar и многим типам из пакета java.time.
   13. @PastOrPresent - Значение переменной должно быть прошедшим либо настоящим временем. Применима к Date, Calendar и многим типам из пакета java.time.
   14. @Email - Значение должно быть адресом электронной почты; применима к CharSequence. Поведение зависит от конкретной реализации.
   15. @NotBlank - Значение типа CharSequence не должно быть null, пустым или состоять из одних лишь пробельных символов.
   16. @Pattern - Значение типа CharSequence должно соответствовать указанному регулярному выражению.
   17. @AssertFalse - Аннотация применима к переменным типов boolean и Boolean, значение которых должно быть false, либо null.
   18. @AssertTrue - Противоположность @AssertFalse, значение должно быть true или null.
   19. @NotEmpty - Значение типов CharSequence, Collection, Map или массив не должно быть null и должно содержать хотя бы 1 элемент.
   20. @NotNull - Значение не должно быть null.
   21. @Null - Значение должно быть null.
   22. @Size - Размер значения типов CharSequence, Collection, Map или массива должен быть в указанном диапазоне — между min и max.

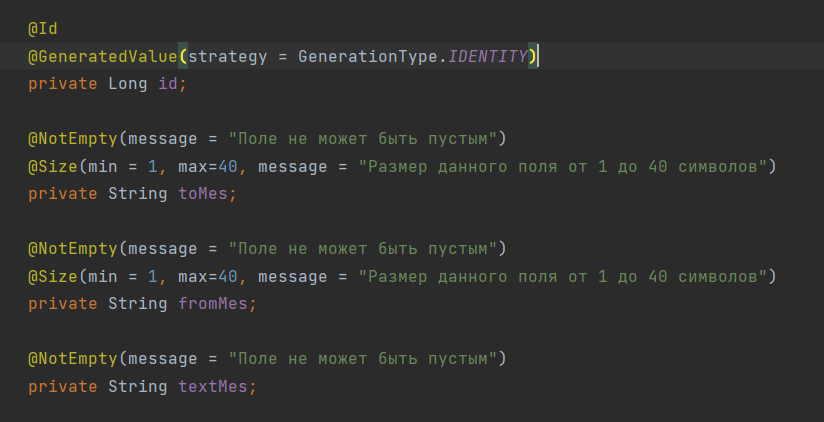


Рисунок 30 - Измененная модель

1. В аннотации @GetMapping вместо стандартной модели передаем конкретную модель, с которой мы будем работать в форме. Далее в @PostMapping передаем аннотацию @Valid и аргументы, с которыми будут производиться действия. Ищем вхождения подстроки в одно из полей репозитория для определения уникальности введенной строки. Если строка уникальна, получены все данные и ошибок нет, то сохраняем репозиторий методом save().



Рисунок 31 – Обработка ошибок

1. Изменяем шаблон так, чтобы полученные ошибки выводились. В частности используем th:object чтобы использовать объект, th:field чтобы использовать поле, th:errors чтобы получить сообщение ошибки и fields.hasErrors для получения ошибок глобального уровня.



Рисунок 32 – Измененный шаблон

1. Переходим на сайт и смотрим результат.

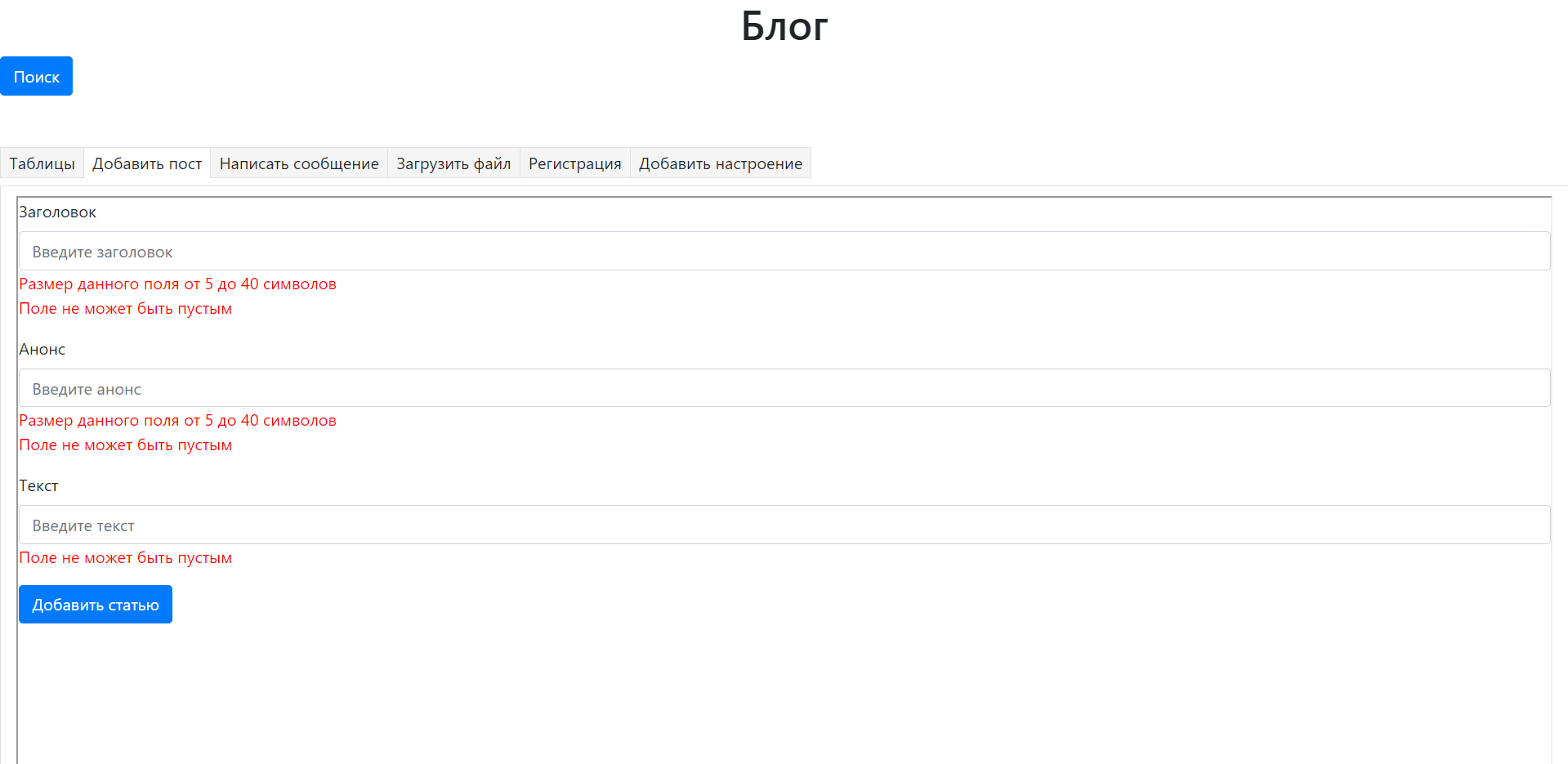


Рисунок 33 - Результат

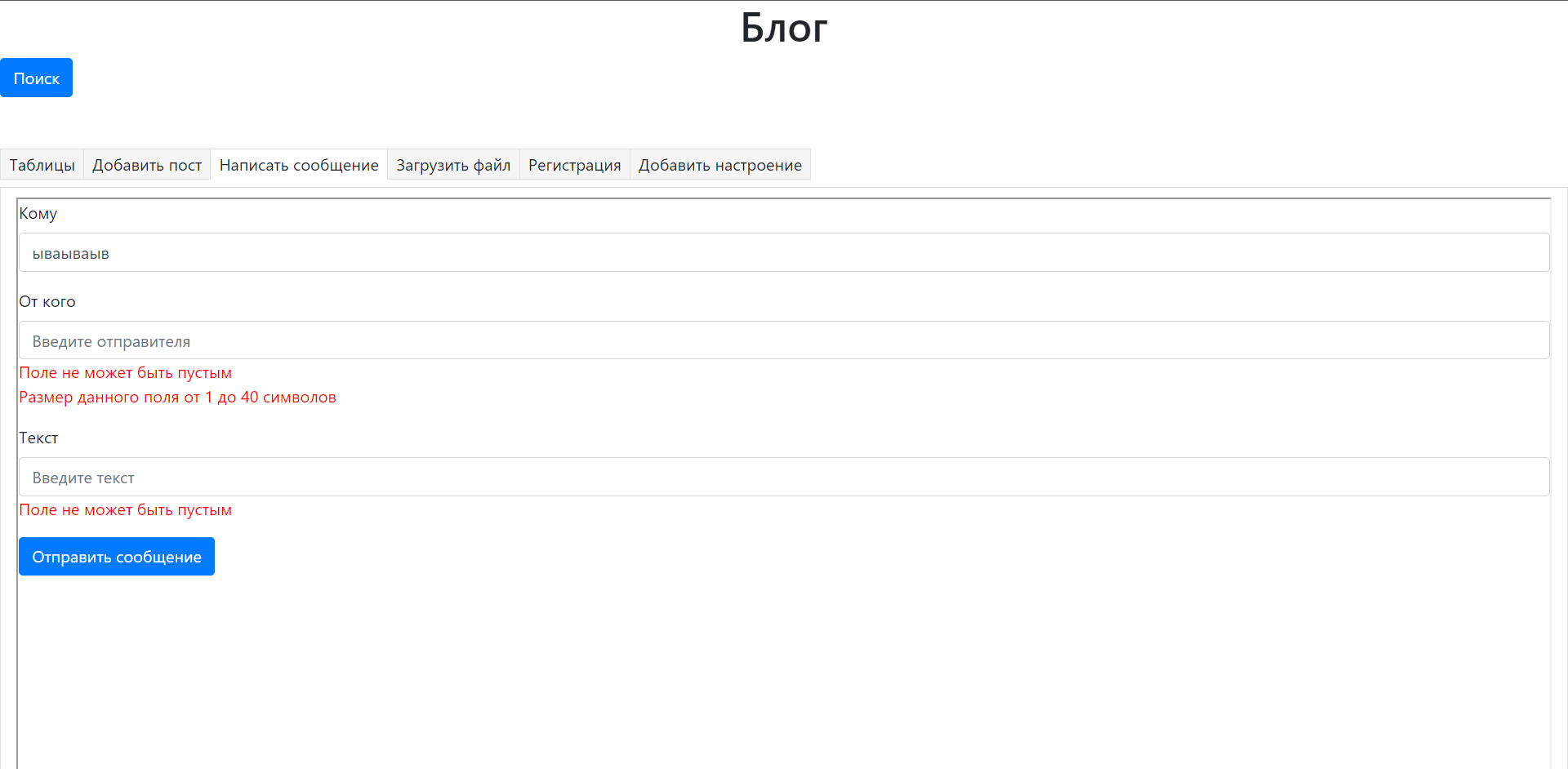


Рисунок 34 - Результат

Вывод: Cделали верификацию полей форм при помощи верификатора.

# Практическая работа №6

## Работа с БД. Связи

Цель работы: реализовать три типа соединений таблиц: один к одному, один ко многим, много ко многим.

Результат работы:

1. Для определения связей между сущностями Hibernate использует аннотации:
   1. @Table – добавляет имя таблице в БД.
   2. @OneToOne – обозначает связь один к одному;
   3. @ManyToOne – обозначает связь много к одному;
   4. @OneToMany – один ко многим;
   5. @ManyToMany – обозначает связь многие ко многим;
   6. @JoinTable – определяет таблицу и поля для связи;
   7. @JoinColumn – используется для связывания полей.
2. Создаем модели и для таблиц.

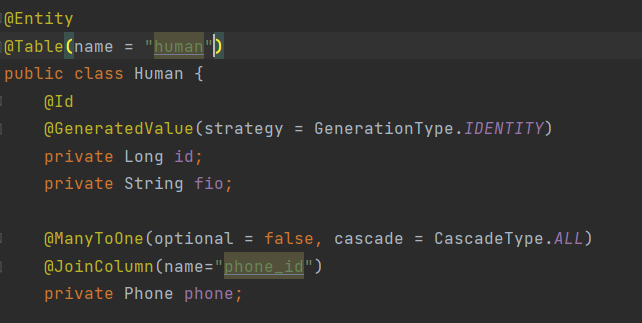


Рисунок 35 – Пример аннотации ManyToOne

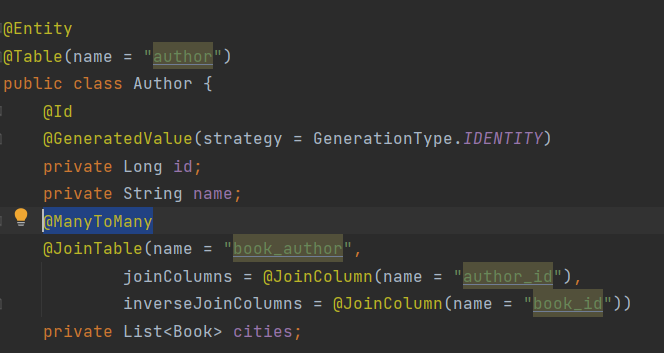


Рисунок 36 - Пример аннотации ManyToMany

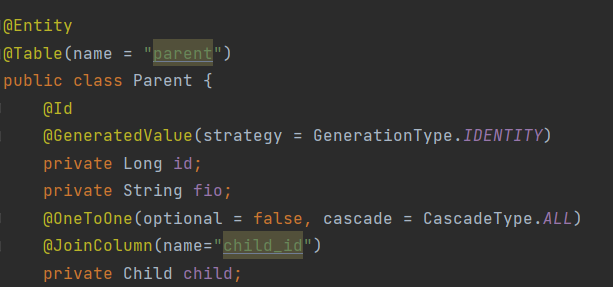


Рисунок 37 - Пример аннотации OneToOne

1. Создаем репозитории для моделей

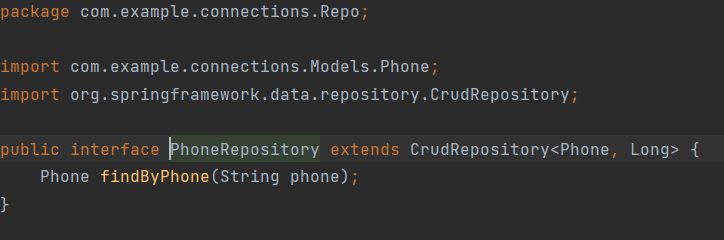


Рисунок 38 - Репозиторий

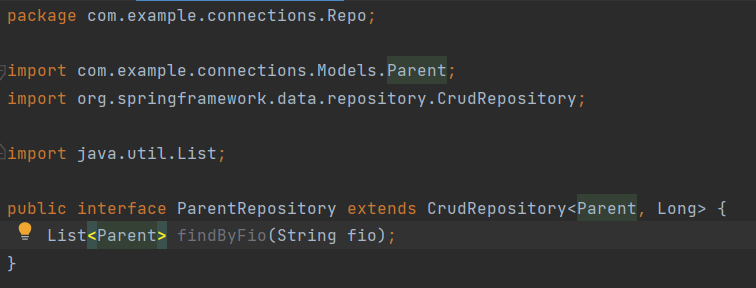


Рисунок 39 - Репозиторий

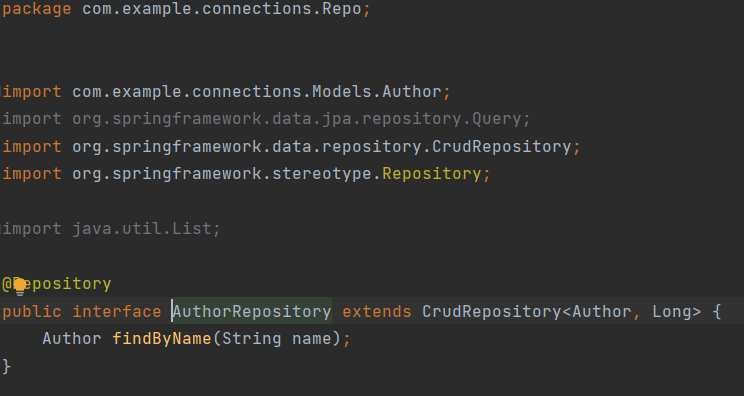


Рисунок 40 - Репозиторий

1. Создаем контроллер и обработчики запросов. Для связывания контроллера и репозитория используем аннотацию @Autowired.

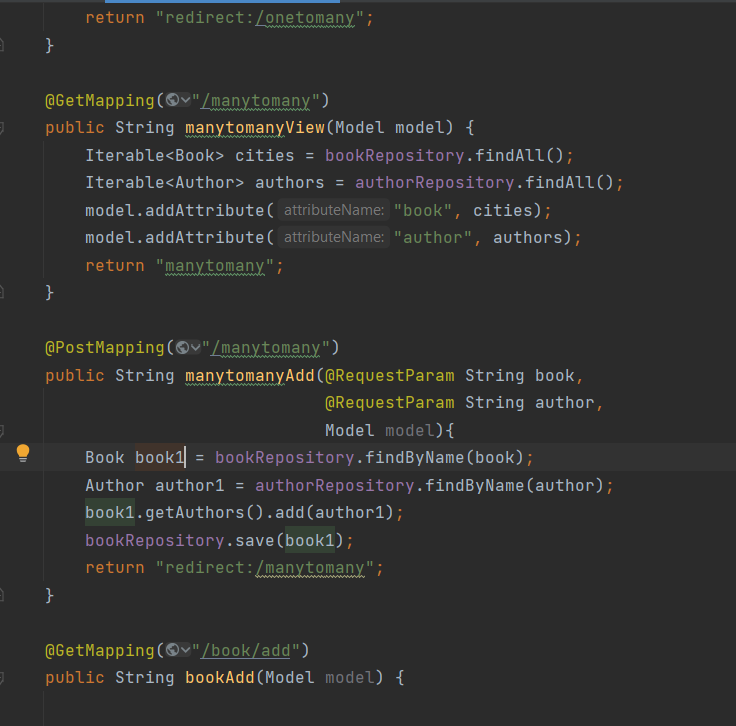


Рисунок 41 - Контроллер



Рисунок 42 - Контроллер

1. Создаем шаблоны для страниц.

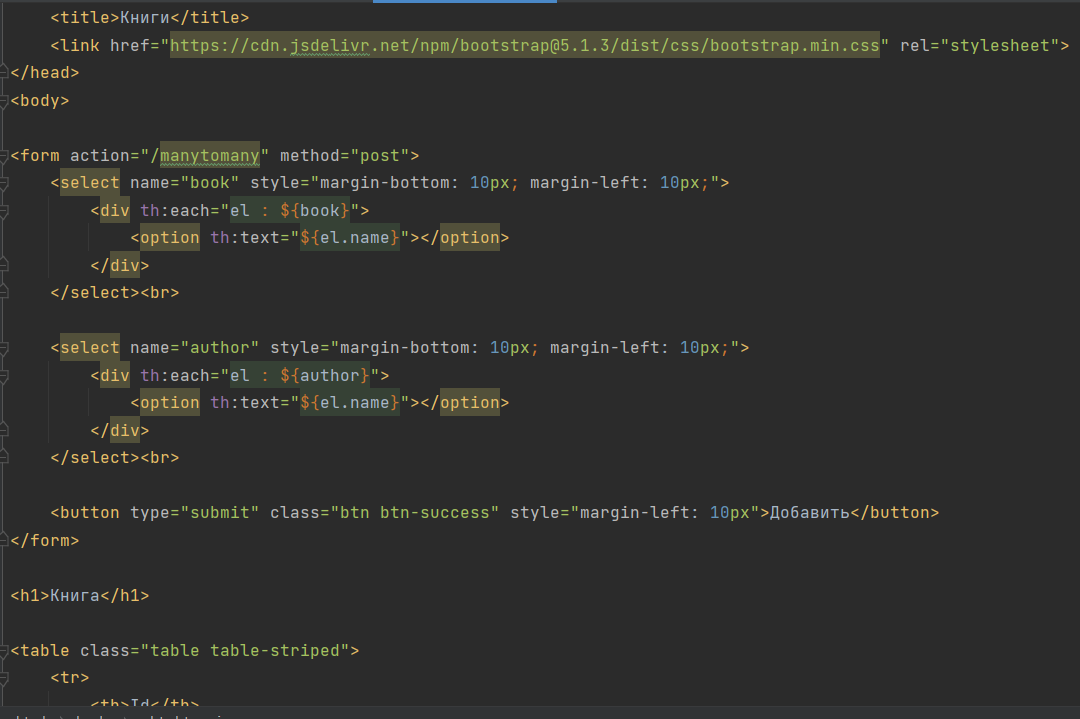


Рисунок 43 - Фрагмент кода

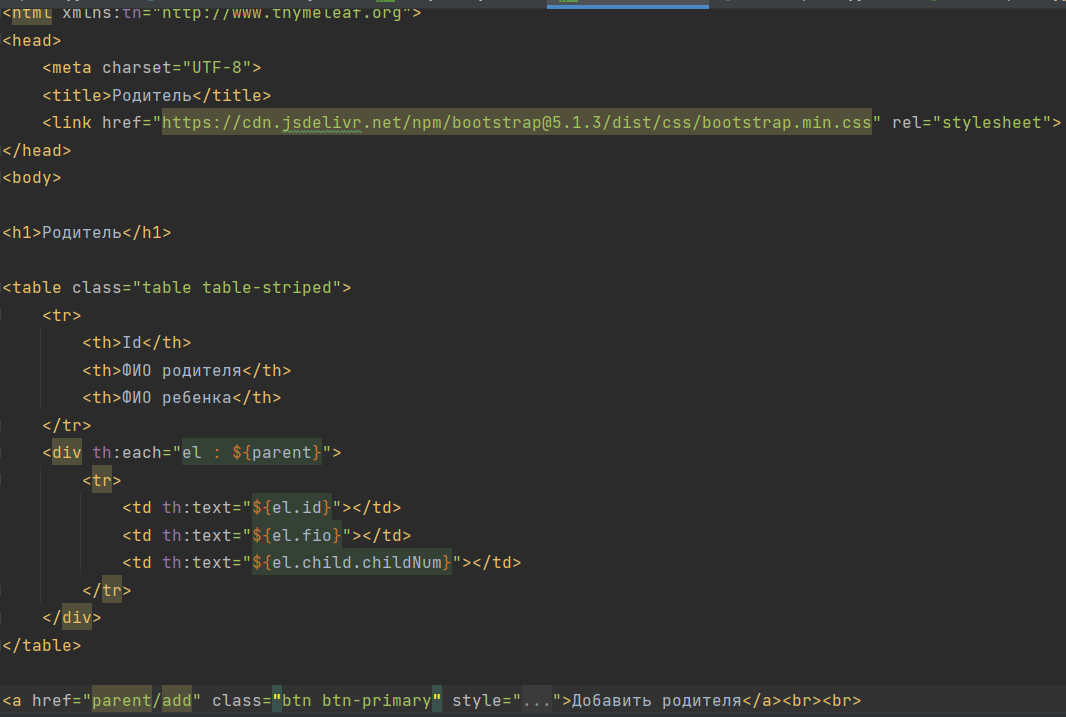


Рисунок 44 - Фрагмент кода

1. Результат работы.

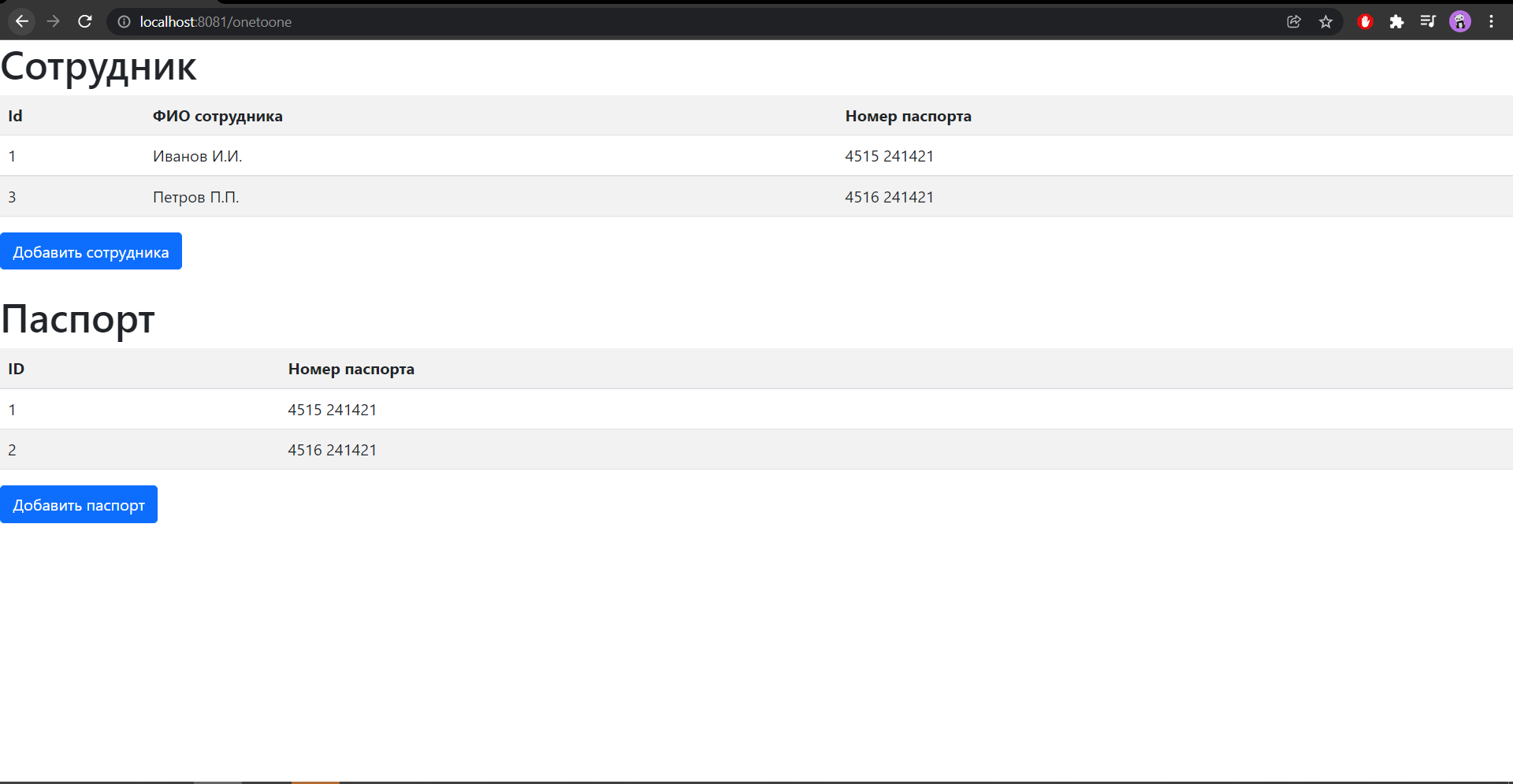


Рисунок 45 - Таблицы связи OneToOne

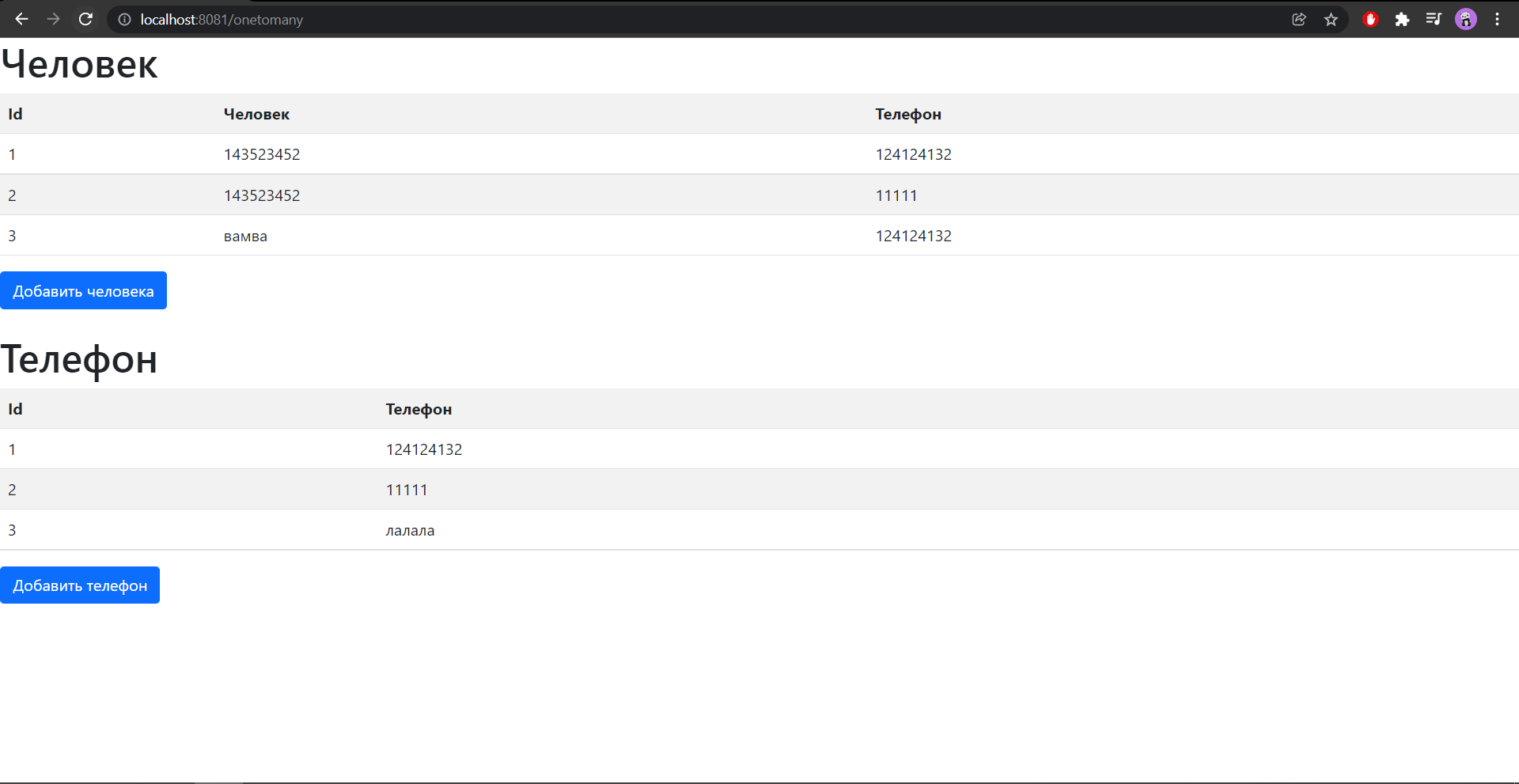


Рисунок 46 - Таблицы связи OneToMany

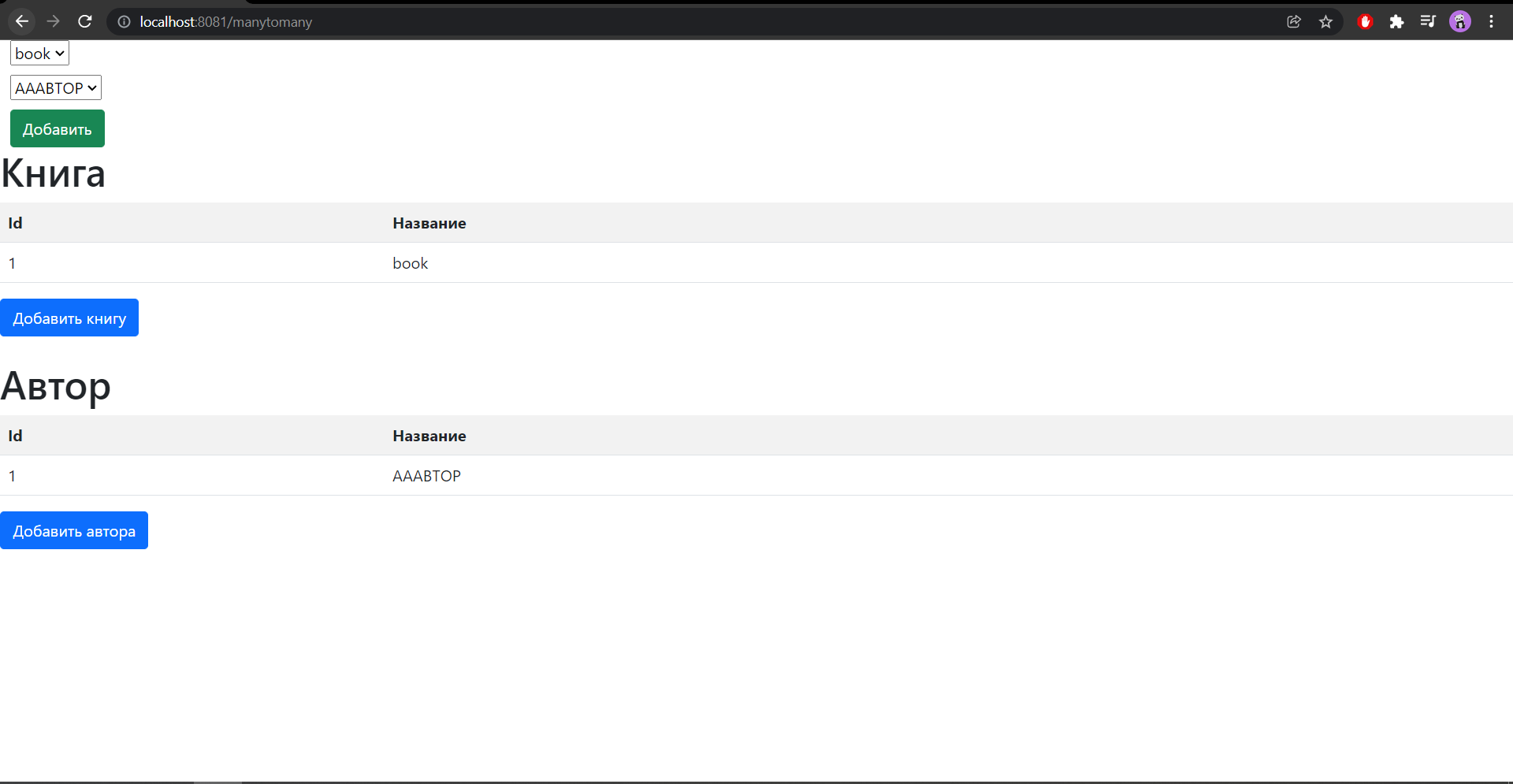


Рисунок 47 - таблицы связи ManyToMany

Вывод: реализовали три типа соединений таблиц.

# Индивидуальный проект

## Склад радиокомпонентов

1. Описание предметной области

Название предметной области – Склад радиокомпонентов.

База данных имеет иерархическую структуру, с тремя видами вложений: Класс, категория, группа, компонент. Присутствует таблица транзакций, где ведется учет прихода/расхода компонентов.

В компонентах есть внешние связи на таблицу «Единица» - наименования единиц (штуки, упаковки, пр.), «поставщик» - наименования поставщиков и их расчетные счета, а также «Местоположение» - наименование места на складе. В компонентах есть внешние связи на таблицу «Паспорт» - в ней хранятся паспортные данные сотрудников.

1. Схема базы данных

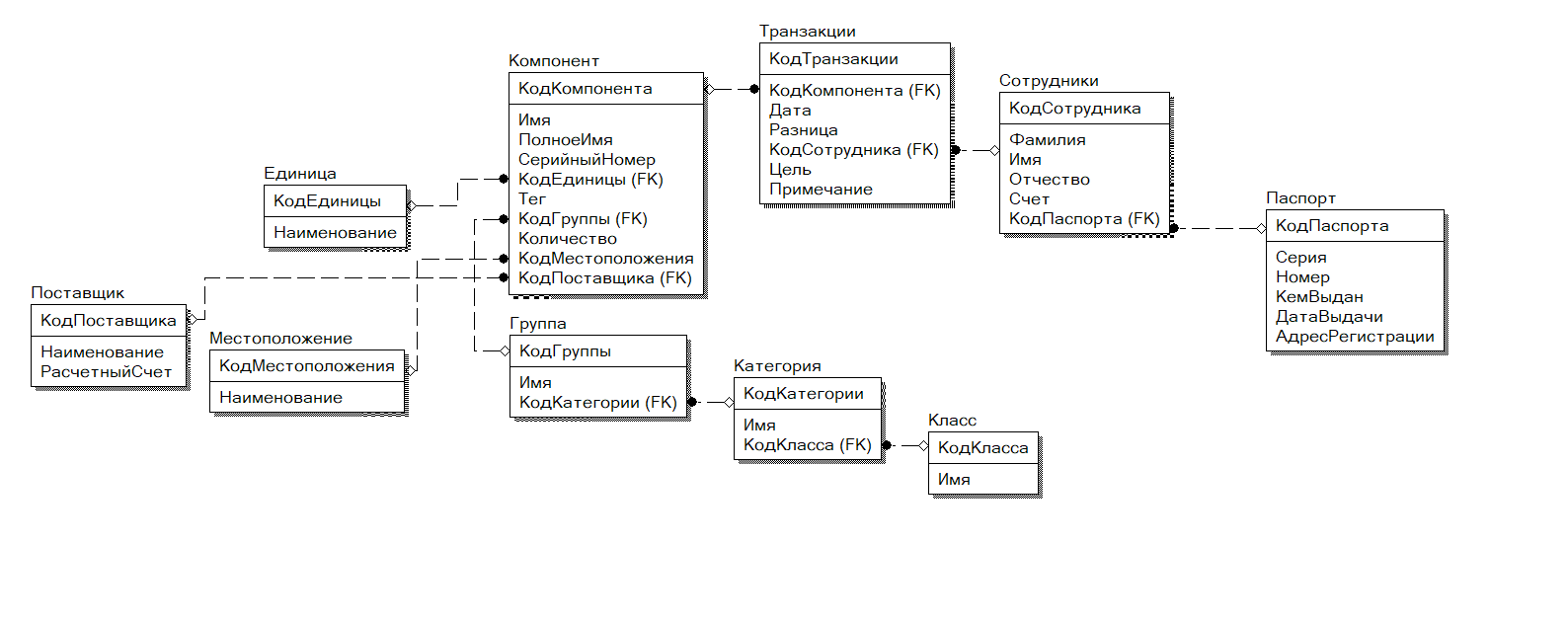


Рисунок 48 - Логическая схема

1. Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Обязательность заполнения | Назначение |
| 1 | 2 | 3 | 4 | 5 |
| Таблица Component | | | | |
| PK | id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | component\_amount | varchar(255) | Not null | Количество |
|  | component\_full\_name | varchar(255) | Not null | Полное имя |
|  | component\_group\_id | varchar(255) | Not null | Id группы |
|  | component\_location | varchar(255) | Not null | Местоположение |
|  | component\_name | varchar(255) | Not null | Имя |
|  | component\_serial | varchar(255) | Allow null | Серийный номер |
|  | component\_tag | varchar(255) | Allow null | Тег |
|  | component\_unit | varchar(255) | Not null | Единица измерения |
| Таблица component\_category | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | Component\_category\_name | varchar(255) | Not null | Имя категории |
|  | Component\_category\_class\_id | varchar(255) | Not null | Id класса |
| Таблица component\_group | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | Component\_group\_name | varchar(255) | Not null | Имя группы |
|  | Component\_group\_category\_id | varchar(255) | Not null | Id категории |
| Таблица component\_class | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | Class\_name | varchar(255) | Not null | Имя класса |
| Таблица staff | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | name | varchar(255) | Not null | Имя |
|  | patronymic | varchar(255) | Allow null | Отчество |
|  | surname | varchar(255) | Not null | Фамилия |
| Таблица transaction | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | Component\_id | Long | Not null | Id компонента |
|  | Date | Date | Not null | Дата |
|  | different | Long | Not null | Разница |
|  | note | varchar(255) | Allow null | Заметка |
|  | place | varchar(255) | Not null | Цель |
|  | Staff\_id | Long | Not null | Id сотрудника |
| Таблица user | | | | |
| PK | Id | Long | Not null | Уникальный идентификатор с атрибутом Auto increment |
|  | active | varchar(255) | Not null | Активность пользователя |
|  | username | varchar(255) | Not null | Логин |
|  | password | varchar(255) | Not null | Пароль |
| Таблица user\_role | | | | |
| PK | User\_id | Long | Not null | Идентификатор пользователя |
|  | roles | varchar(255) | Not null | Имя роли |

1. Листинг скрипта базы данных

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- База данных: `warehouse`

--

-- --------------------------------------------------------

--

-- Структура таблицы `component`

--

CREATE TABLE `component` (

`id` bigint(20) NOT NULL,

`component\_amount` varchar(255) DEFAULT NULL,

`component\_full\_name` varchar(255) DEFAULT NULL,

`component\_group\_id` varchar(255) DEFAULT NULL,

`component\_location` varchar(255) DEFAULT NULL,

`component\_name` varchar(255) DEFAULT NULL,

`component\_serial` varchar(255) DEFAULT NULL,

`component\_tag` varchar(255) DEFAULT NULL,

`component\_unit` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `component`

--

-- --------------------------------------------------------

--

-- Структура таблицы `component\_category`

--

CREATE TABLE `component\_category` (

`id` bigint(20) NOT NULL,

`component\_category\_name` varchar(255) DEFAULT NULL,

`component\_category\_class\_id` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `component\_category`

--

-- --------------------------------------------------------

--

-- Структура таблицы `component\_class`

--

CREATE TABLE `component\_class` (

`id` bigint(20) NOT NULL,

`component\_class\_name` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `component\_class`

--

-- --------------------------------------------------------

--

-- Структура таблицы `component\_group`

--

CREATE TABLE `component\_group` (

`id` bigint(20) NOT NULL,

`component\_group\_category\_id` varchar(255) DEFAULT NULL,

`component\_group\_name` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `component\_group`

--

-- --------------------------------------------------------

--

-- Структура таблицы `hibernate\_sequence`

--

CREATE TABLE `hibernate\_sequence` (

`next\_val` bigint(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `hibernate\_sequence`

--

-- -------------------------------------------------------

--

-- Структура таблицы `staff`

--

CREATE TABLE `staff` (

`id` bigint(20) NOT NULL,

`name` varchar(255) DEFAULT NULL,

`patronymic` varchar(255) DEFAULT NULL,

`surname` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `staff`

--

-- --------------------------------------------------------

--

-- Структура таблицы `transaction`

--

CREATE TABLE `transaction` (

`id` bigint(20) NOT NULL,

`component\_id` varchar(255) DEFAULT NULL,

`date` varchar(255) DEFAULT NULL,

`different` varchar(255) DEFAULT NULL,

`note` varchar(255) DEFAULT NULL,

`place` varchar(255) DEFAULT NULL,

`staff\_id` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `transaction`

--

-- --------------------------------------------------------

--

-- Структура таблицы `user`

--

CREATE TABLE `user` (

`id` bigint(20) NOT NULL,

`active` bit(1) NOT NULL,

`password` varchar(255) DEFAULT NULL,

`username` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `user`

--

-- --------------------------------------------------------

--

-- Структура таблицы `user\_role`

--

CREATE TABLE `user\_role` (

`user\_id` bigint(20) NOT NULL,

`roles` varchar(255) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Дамп данных таблицы `user\_role`

--

--

-- Индексы сохранённых таблиц

--

--

-- Индексы таблицы `component`

--

ALTER TABLE `component`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `component\_category`

--

ALTER TABLE `component\_category`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `component\_class`

--

ALTER TABLE `component\_class`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `component\_group`

--

ALTER TABLE `component\_group`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `staff`

--

ALTER TABLE `staff`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `transaction`

--

ALTER TABLE `transaction`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `user`

--

ALTER TABLE `user`

ADD PRIMARY KEY (`id`);

--

-- Индексы таблицы `user\_role`

--

ALTER TABLE `user\_role`

ADD KEY `FK859n2jvi8ivhui0rl0esws6o` (`user\_id`);

--

-- AUTO\_INCREMENT для сохранённых таблиц

--

--

-- AUTO\_INCREMENT для таблицы `component`

--

ALTER TABLE `component`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- AUTO\_INCREMENT для таблицы `component\_category`

--

ALTER TABLE `component\_category`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `component\_class`

--

ALTER TABLE `component\_class`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=6;

--

-- AUTO\_INCREMENT для таблицы `component\_group`

--

ALTER TABLE `component\_group`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT для таблицы `staff`

--

ALTER TABLE `staff`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT для таблицы `transaction`

--

ALTER TABLE `transaction`

MODIFY `id` bigint(20) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=8;

--

-- Ограничения внешнего ключа сохраненных таблиц

-

--

-- Ограничения внешнего ключа таблицы `user\_role`

--

ALTER TABLE `user\_role`

ADD CONSTRAINT `FK859n2jvi8ivhui0rl0esws6o` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`);

COMMIT;

1. Код программы
   1. Листинг файла MvcConfig.java

package com.dryupin.warehouse.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");

}

}

* 1. Листинг файла WebSecurityConfig.java

package com.dryupin.warehouse.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.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import javax.sql.DataSource;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

private DataSource dataSource;

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

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

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.permitAll();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.jdbcAuthentication()

.dataSource(dataSource)

.passwordEncoder(NoOpPasswordEncoder.getInstance())

.usersByUsernameQuery("select username,password, active from user where username=?")

.authoritiesByUsernameQuery("select u.username, ur.roles from user u inner join user\_role ur on u.id = ur.user\_id where u.username=?");

}

}

* 1. Листинг файла MainController.java

package com.dryupin.warehouse.controllers;

import com.dryupin.warehouse.models.\*;

import com.dryupin.warehouse.repo.ComponentCategoryRepository;

import com.dryupin.warehouse.repo.ComponentClassRepository;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.validation.ObjectError;

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

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

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

import javax.validation.Valid;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

@Controller

public class MainController {

@Autowired

private ComponentClassRepository ComponentClassRepository;

@Autowired

private com.dryupin.warehouse.repo.ComponentCategoryRepository ComponentCategoryRepository;

@Autowired

private com.dryupin.warehouse.repo.ComponentGroupRepository ComponentGroupRepository;

@Autowired

private com.dryupin.warehouse.repo.ComponentRepository ComponentRepository;

@Autowired

private com.dryupin.warehouse.repo.TransactionRepository TransactionRepository;

@Autowired

private com.dryupin.warehouse.repo.StaffRepository StaffRepository;

@GetMapping("/")

public String warehouseMain(Model model) {

String Tree = "";

String SelectCategory = "";

String SelectGroup = "";

String SelectComponent = "";

String SelectStaff = "";

Iterable<Component\_Class> ComponentClass = ComponentClassRepository.findAll();

for(var CompClass : ComponentClass) {

SelectCategory += "<option value='" + CompClass.getId() + "'>" + CompClass.getComponentClassName() + "</option>";

Tree += "<li>" + CompClass.getComponentClassName();

Iterable<Component\_Category> ComponentCategory = ComponentCategoryRepository.GetComponentCategoryByClassId(CompClass.getId().toString());

for (var CompCategory : ComponentCategory) {

SelectGroup += "<option value='" + CompCategory.getId() + "'>" + CompClass.getComponentClassName() + "." + CompCategory.getComponentCategoryName() + "</option>";

Tree += "<ul>";

Tree += "<li>" + CompCategory.getComponentCategoryName();

Iterable<Component\_Group> ComponentGroup = ComponentGroupRepository.GetComponentGroupByCategoryId(CompCategory.getId().toString());

for (var CompGroup : ComponentGroup) {

SelectComponent += "<option value='" + CompGroup.getId() + "'>" + CompClass.getComponentClassName() + "." + CompCategory.getComponentCategoryName() + "." + CompGroup.getComponentGroupName() + "</option>";

Tree += "<ul>";

Tree += "<li>" + CompGroup.getComponentGroupName();

Iterable<Component> Component = ComponentRepository.GetComponentByGroupId(CompGroup.getId().toString());

for (var CompComponent : Component) {

Tree += "<ul>";

Tree += "<li><a data-toggle='modal' data-target='#ComponentInfo\_" + CompComponent.getId() + "'>" + CompComponent.getComponentName() + "</a></li>";

Tree += "<div class='modal fade component-info' id='ComponentInfo\_" + CompComponent.getId() + "' tabindex='-1' role='dialog' aria-labelledby='exampleModalLabel' aria-hidden='true'>";

Tree += " <div class='modal-dialog modal-dialog-centered' role='document'>";

Tree += " <div class='modal-content'>";

Tree += " <div class='modal-header'>";

Tree += " <h5 class='modal-title' id='exampleModalLabel'>" + CompComponent.getComponentName() + "</h5>";

Tree += " </div>";

Tree += " <div class='modal-body'>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Id</label><label>" + CompComponent.getId() + "</label><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Наименование</label><input class='modal-parameter-input' id='ComponentFieldName\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentName() + "'><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Полное наименование</label><input class='modal-parameter-input' id='ComponentFieldFullName\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentFullName() + "'><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Серийный номер</label><input class='modal-parameter-input' id='ComponentFieldSerial\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentSerial() + "'><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Количество</label><input class='modal-parameter-input' id='ComponentFieldAmount\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentAmount() + "'><input class='modal-parameter-input' style='width: 125px;' id='ComponentFieldUnit\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentUnit() + "'><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Расположение</label><input class='modal-parameter-input' id='ComponentFieldLocation\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentLocation() + "'><br>";

Tree += " <label style='font-weight: normal; padding-right: 10px;'>Тег</label><input class='modal-parameter-input' id='ComponentFieldTag\_" + CompComponent.getId() + "' value='" + CompComponent.getComponentTag() + "'><br>";

Tree += " </div>";

Tree += " <div class='modal-footer'>";

Tree += " <button type='button' style='left:15px; position:absolute;' id='ComponentInfoDelete\_" + CompComponent.getId() + "' class='btn btn-danger component-info-delete'>Удалить</button>";

Tree += " <button type='button' id='ComponentInfoSave\_" + CompComponent.getId() + "' class='btn btn-primary component-info-save'>Сохранить</button>";

Tree += " <button type='button' id='TransactionAdd\_" + CompComponent.getId() + "' class='btn btn-info transaction-add' data-toggle='modal' data-target='#TransactionAdd'>Транзакция</button>";

Tree += " <button type='button' class='btn btn-secondary' data-dismiss='modal'>Закрыть</button>";

Tree += " </div>";

Tree += " </div>";

Tree += "</div>";

Tree += "</ul>";

}

Tree += "</li>";

Tree += "</ul>";

}

Tree += "</li>";

Tree += "</ul>";

}

Tree += "</li>";

}

Iterable<Staff> Staff = StaffRepository.findAll();

for(var StaffIterables : Staff) {

SelectStaff+="<option value='" + StaffIterables.getId() + "'>"+StaffIterables.getName()+" "+StaffIterables.getPatronymic()+" "+StaffIterables.getSurname()+"</option>";

}

System.out.println(Tree);

Iterable<Component\_Class> Comclass = ComponentClassRepository.findAll();

model.addAttribute("ComponentsTree", Tree);

model.addAttribute("SelectCategory", SelectCategory);

model.addAttribute("SelectGroup", SelectGroup);

model.addAttribute("SelectComponent", SelectComponent);

model.addAttribute("SelectStaff", SelectStaff);

model.addAttribute("component\_class", Comclass);

return "Warehouse";

}

@GetMapping("/AddComponent")

public String AddComponent(@RequestParam String ComponentGroup, @RequestParam String ComponentName, @RequestParam String ComponentFullName, @RequestParam String ComponentSerial, @RequestParam String ComponentAmount, @RequestParam String ComponentUnit, @RequestParam String ComponentLocation, @RequestParam String ComponentTag) {

Component component = new Component(ComponentName, ComponentFullName, ComponentSerial, ComponentUnit, ComponentTag, ComponentGroup, ComponentAmount, ComponentLocation);

ComponentRepository.save(component);

return "Warehouse";

}

@GetMapping("/UpdateComponent")

public String UpdateComponent(@RequestParam Long ComponentId, @RequestParam String ComponentName, @RequestParam String ComponentFullName, @RequestParam String ComponentSerial, @RequestParam String ComponentAmount, @RequestParam String ComponentUnit, @RequestParam String ComponentLocation, @RequestParam String ComponentTag) {

Component component = ComponentRepository.findById(ComponentId).orElseThrow();

component.setComponentName(ComponentName);

component.setComponentFullName(ComponentFullName);

component.setComponentSerial(ComponentSerial);

component.setComponentAmount(ComponentAmount);

component.setComponentUnit(ComponentUnit);

component.setComponentLocation(ComponentLocation);

component.setComponentTag(ComponentTag);

ComponentRepository.save(component);

return "Warehouse";

}

@GetMapping("/DeleteComponent")

public String DeleteComponent(@RequestParam Long ComponentId) {

Component component = ComponentRepository.findById(ComponentId).orElseThrow();

ComponentRepository.delete(component);

return "Warehouse";

}

@GetMapping("/AddClass")

public String AddClass(@Valid Component\_Class Component\_Class, BindingResult bindingResult, @RequestParam String ClassName) {

List<Component\_Class> res = ComponentClassRepository.findBycomponentClassName(ClassName);

if(res.size()>0) {

ObjectError error = new ObjectError("title", "Такой класс уже есть");

bindingResult.addError(error);

return "Warehouse";

}

Component\_Class componentClass = new Component\_Class(ClassName);

ComponentClassRepository.save(componentClass);

return "Warehouse";

}

@GetMapping("/AddCategory")

public String AddCategory(@RequestParam String CategoryName, @RequestParam String CategoryClass) {

Component\_Category componentCategory = new Component\_Category(CategoryName, CategoryClass);

ComponentCategoryRepository.save(componentCategory);

return "Warehouse";

}

@GetMapping("/AddGroup")

public String AddGroup(@RequestParam String GroupName, @RequestParam String GroupCategory) {

Component\_Group componentGroup = new Component\_Group(GroupCategory, GroupName);

ComponentGroupRepository.save(componentGroup);

return "Warehouse";

}

@GetMapping("/AddTransaction")

public String AddTransaction(@RequestParam String TransactionComponentId, @RequestParam String TransactionDate, @RequestParam String TransactionDifferent, @RequestParam String TransactionStaffId, @RequestParam String TransactionPlace, @RequestParam String TransactionNote) {

Transaction transaction = new Transaction(TransactionComponentId, TransactionDate, TransactionDifferent, TransactionStaffId, TransactionPlace, TransactionNote);

TransactionRepository.save(transaction);

Component component = ComponentRepository.findById(Long.parseLong(TransactionComponentId)).orElseThrow();

int NewAmount = Integer.parseInt(component.getComponentAmount()) + Integer.parseInt(TransactionDifferent);

component.setComponentAmount(String.valueOf(NewAmount));

ComponentRepository.save(component);

return "Warehouse";

}

@GetMapping("/GetTransactions")

@ResponseBody

public String GetTransactions() {

String JSON = "[";

int i=0;

Iterable<Transaction> Transactions = TransactionRepository.GetAll();

for(var TransactionsList : Transactions) {

JSON+="{\"id\":\""+TransactionsList.getId()+"\",";

JSON+="\"ComponentName\":\""+TransactionsList.getComponentId()+"\",";

JSON+="\"StaffName\":\""+TransactionsList.getStaffId()+"\",";

JSON+="\"TransactionDate\":\""+TransactionsList.getDate()+"\",";

JSON+="\"TransactionDifferent\":\""+TransactionsList.getDifferent()+"\",";

JSON+="\"TransactionPlace\":\""+TransactionsList.getPlace()+"\",";

JSON+="\"TransactionNote\":\""+TransactionsList.getNote()+"\"}";

if(i++ < Transactions.spliterator().getExactSizeIfKnown()-1){

JSON+=",";

}

}

JSON+="]";

System.out.println(JSON);

return JSON;

}

/\*TODO

\*/

}

* 1. Листинг файла RegistrationController.java

package com.dryupin.warehouse.controllers;

import com.dryupin.warehouse.models.Role;

import com.dryupin.warehouse.models.User;

import com.dryupin.warehouse.repo.UserRepository;

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 RegistrationController {

@Autowired

private UserRepository UserRepository;

@GetMapping("/registration")

public String registration() {

return "registration";

}

@PostMapping("/registration")

public String addUser(User user, Model model) {

User userFromDb = UserRepository.findByUsername(user.getUsername());

if (userFromDb != null) {

model.addAttribute("message", "Такой пользователь уже существует");

return "registration";

}

user.setActive(true);

user.setRoles(Collections.singleton(Role.USER));

user.setPassword(user.getPassword());

UserRepository.save(user);

return "redirect:/login";

}

}

* 1. Листинг файла UserController.java

package com.dryupin.warehouse.controllers;

import com.dryupin.warehouse.models.Role;

import com.dryupin.warehouse.models.User;

import com.dryupin.warehouse.repo.UserRepository;

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.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Optional;

@Controller

@RequestMapping("/admin")

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

public class UserController {

@Autowired

private UserRepository userRepository;

@GetMapping

public String userList(Model model) {

model.addAttribute("users", userRepository.findAll());

return "userList";

}

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

public String adminEdit(@PathVariable(value = "id") long id, Model model) {

Optional<User> user = userRepository.findById(id);

ArrayList<User> res = new ArrayList<>();

user.ifPresent(res::add);

model.addAttribute("user", res);

model.addAttribute("roles", Role.values());

return "userEdit";

}

@PostMapping

public String userSave(@RequestParam String username, @RequestParam(name = "roles[]", required = false) String[] roles, @RequestParam("userId") User user) {

user.setUsername(username);

user.getRoles().clear();

if(roles!=null) {

Arrays.stream(roles).forEach(r->user.getRoles().add(Role.valueOf(r)));

}

userRepository.save(user);

return "redirect:/admin";

}

}

* 1. Листинг файла Component.java

package com.dryupin.warehouse.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotEmpty;

@Entity

public class Component {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotEmpty(message = "Поле не может быть пустым")

private String ComponentName;

private String ComponentFullName;

private String ComponentSerial;

private String ComponentUnit;

private String ComponentTag;

private String ComponentGroupId;

private String ComponentAmount;

private String ComponentLocation;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getComponentName() {

return ComponentName;

}

public void setComponentName(String componentName) {

ComponentName = componentName;

}

public String getComponentFullName() {

return ComponentFullName;

}

public void setComponentFullName(String componentFullName) {

ComponentFullName = componentFullName;

}

public String getComponentSerial() {

return ComponentSerial;

}

public void setComponentSerial(String componentSerial) {

ComponentSerial = componentSerial;

}

public String getComponentUnit() {

return ComponentUnit;

}

public void setComponentUnit(String componentUnit) {

ComponentUnit = componentUnit;

}

public String getComponentTag() {

return ComponentTag;

}

public void setComponentTag(String componentTag) {

ComponentTag = componentTag;

}

public String getComponentGroupId() {

return ComponentGroupId;

}

public void setComponentGroupId(String componentGroupId) {

ComponentGroupId = componentGroupId;

}

public String getComponentAmount() {

return ComponentAmount;

}

public void setComponentAmount(String componentAmount) {

ComponentAmount = componentAmount;

}

public String getComponentLocation() {

return ComponentLocation;

}

public void setComponentLocation(String componentLocation) {

ComponentLocation = componentLocation;

}

public Component(String ComponentName, String ComponentFullName, String ComponentSerial, String ComponentUnit, String ComponentTag, String ComponentGroupId, String ComponentAmount, String ComponentLocation) {

this.ComponentName = ComponentName;

this.ComponentFullName = ComponentFullName;

this.ComponentSerial = ComponentSerial;

this.ComponentUnit = ComponentUnit;

this.ComponentTag = ComponentTag;

this.ComponentGroupId = ComponentGroupId;

this.ComponentAmount = ComponentAmount;

this.ComponentLocation = ComponentLocation;

}

public Component() {

}

}

* 1. Листинг файла Component\_Category.java

package com.dryupin.warehouse.models;

import javax.persistence.\*;

import java.util.Collection;

@Entity

public class Component\_Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String componentCategoryName;

private String componentCategoryClassId;

public Component\_Category(String ComponentCategoryName, String ComponentCategoryClassId) {

this.componentCategoryName = ComponentCategoryName;

this.componentCategoryClassId = ComponentCategoryClassId;

}

public Component\_Category() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getComponentCategoryName() {

return componentCategoryName;

}

public void setComponentCategoryName(String componentCategoryName) {

this.componentCategoryName = componentCategoryName;

}

public String getComponentCategoryClassId() {

return componentCategoryClassId;

}

public void setComponentCategoryClassId(String componentCategoryClassId) {

this.componentCategoryClassId = componentCategoryClassId;

}

}

* 1. Листинг файла Component\_Class.java

package com.dryupin.warehouse.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Component\_Class {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String componentClassName;

public Component\_Class(String ComponentClassName) {

this.componentClassName = ComponentClassName;

}

public Component\_Class() {

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getComponentClassName() {

return componentClassName;

}

public void setComponentClassName(String ComponentClassName) {

ComponentClassName = ComponentClassName;

}

}

* 1. Листинг файла Component\_Group.java

package com.dryupin.warehouse.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Component\_Group {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String componentGroupName;

private String componentGroupCategoryId;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getComponentGroupName() {

return componentGroupName;

}

public void setComponentGroupName(String componentGroupName) {

this.componentGroupName = componentGroupName;

}

public String getComponentCategoryCategoryId() {

return componentGroupCategoryId;

}

public void setComponentCategoryCategoryId(String componentCategoryCategoryId) {

this.componentGroupCategoryId = componentCategoryCategoryId;

}

public Component\_Group(String ComponentGroupName, String ComponentGroupCategoryId) {

this.componentGroupName = ComponentGroupName;

this.componentGroupName = ComponentGroupCategoryId;

}

public Component\_Group() {

}

}

* 1. Листинг файла Role.java

package com.dryupin.warehouse.models;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {

USER, ADMIN;

@Override

public String getAuthority() { return name();}

}

* 1. Листинг файла Staff.java

package com.dryupin.warehouse.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Staff {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String Name;

private String Patronymic;

private String Surname;

public Staff(String Name, String Patronymic, String Surname) {

this.Name = Name;

this.Patronymic = Patronymic;

this.Surname = Surname;

}

public Staff() {

}

public String getPatronymic() {

return Patronymic;

}

public void setPatronymic(String patronymic) {

Patronymic = patronymic;

}

public String getSurname() {

return Surname;

}

public void setSurname(String surname) {

Surname = surname;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return Name;

}

public void setName(String name) {

Name = name;

}

}

* 1. Листинг файла Transaction.java

package com.dryupin.warehouse.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotEmpty;

@Entity

public class Transaction {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String ComponentId;

private String Date;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getComponentId() {

return ComponentId;

}

public void setComponentId(String componentId) {

ComponentId = componentId;

}

public String getDate() {

return Date;

}

public void setDate(String date) {

Date = date;

}

public String getDifferent() {

return Different;

}

public void setDifferent(String different) {

Different = different;

}

public String getStaffId() {

return StaffId;

}

public void setStaffId(String staffId) {

StaffId = staffId;

}

public String getPlace() {

return Place;

}

public void setPlace(String place) {

Place = place;

}

public String getNote() {

return Note;

}

public void setNote(String note) {

Note = note;

}

private String Different;

private String StaffId;

private String Place;

private String Note;

public Transaction(String ComponentId, String Date, String Different, String StaffId, String Place, String Note) {

this.ComponentId = ComponentId;

this.Date = Date;

this.Different = Different;

this.StaffId = StaffId;

this.Place = Place;

this.Note = Note;

}

public Transaction() {

}

}

* 1. Листинг файла User.java

package com.dryupin.warehouse.models;

import javax.persistence.\*;

import java.util.Set;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

private boolean active;

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

@CollectionTable(name = "user\_role", joinColumns = @JoinColumn(name="user\_id"))

@Enumerated(EnumType.STRING)

private Set<Role> roles;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

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<Role> getRoles() {

return roles;

}

public void setRoles(Set<Role> roles) {

this.roles = roles;

}

}

* 1. Листинг файла ComponentCategoryRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component\_Category;

import com.dryupin.warehouse.models.Component\_Class;

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

import org.springframework.data.repository.CrudRepository;

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

public interface ComponentCategoryRepository extends CrudRepository<Component\_Category, Long> {

@Query("select c from Component\_Category c where c.componentCategoryClassId = :#{#ClassId}")

Iterable<Component\_Category> GetComponentCategoryByClassId(@Param("ClassId") String ClassId);

}

* 1. Листинг файла ComponentClassRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component\_Class;

import org.springframework.data.repository.CrudRepository;

import java.util.List;

public interface ComponentClassRepository extends CrudRepository<Component\_Class, Long> {

List<Component\_Class> findBycomponentClassName(String name);

}

* 1. Листинг файла ComponentGroupRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component\_Category;

import com.dryupin.warehouse.models.Component\_Group;

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

import org.springframework.data.repository.CrudRepository;

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

public interface ComponentGroupRepository extends CrudRepository<Component\_Group, Long> {

@Query("select g from Component\_Group g where g.componentGroupCategoryId = :#{#CategoryId}")

Iterable<Component\_Group> GetComponentGroupByCategoryId(@Param("CategoryId") String CategoryId);

}

* 1. Листинг файла ComponentRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component;

import com.dryupin.warehouse.models.Component\_Category;

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

import org.springframework.data.repository.CrudRepository;

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

public interface ComponentRepository extends CrudRepository<Component, Long> {

@Query("select c from Component c where c.ComponentGroupId = :#{#GroupId}")

Iterable<Component> GetComponentByGroupId(@Param("GroupId") String GroupId);

}

* 1. Листинг файла StaffRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component\_Class;

import com.dryupin.warehouse.models.Staff;

import org.springframework.data.repository.CrudRepository;

public interface StaffRepository extends CrudRepository<Staff, Long> {

}

* 1. Листинг файла TransactionRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.Component;

import com.dryupin.warehouse.models.Transaction;

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

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

import org.springframework.data.repository.CrudRepository;

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

public interface TransactionRepository extends CrudRepository<Transaction, Long> {

@Query(value = "SELECT \* FROM transaction, component, staff WHERE component\_id = component.id AND staff\_id = staff.id",

nativeQuery = true)

Iterable<Transaction> GetAll();

}

* 1. Листинг файла UserRepository.java

package com.dryupin.warehouse.repo;

import com.dryupin.warehouse.models.User;

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

public interface UserRepository extends JpaRepository<User, Long> {

User findByUsername(String username);

}

* 1. Листинг файла common.css

\*,

\*::before,

\*::after {

box-sizing: border-box;

}

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol", "Noto Color Emoji";

font-size: 16px;

font-weight: 400;

line-height: 1.5;

color: #212529;

text-align: left;

background-color: #fff;

}

/\* Tabs \*/

.tabs {

font-size: 0;

max-width: 350px;

margin-left: 0;

margin-right: auto;

margin-top: 50px;

}

.tabs>input[type="radio"] {

display: none;

}

.tabs>div {

/\* скрыть контент по умолчанию \*/

display: none;

border: 1px solid #e0e0e0;

padding: 10px 15px;

font-size: 16px;

height: 100%;

width: 100%;

position: fixed;

left: 0;

}

/\* отобразить контент, связанный с вабранной радиокнопкой (input type="radio") \*/

#tab-btn-1:checked~#content-1,

#tab-btn-2:checked~#content-2{

display: block;

}

.tabs>label {

display: inline-block;

text-align: center;

vertical-align: middle;

user-select: none;

background-color: #f5f5f5;

border: 1px solid #e0e0e0;

padding: 2px 8px;

font-size: 16px;

line-height: 1.5;

transition: color 0.15s ease-in-out, background-color 0.15s ease-in-out;

cursor: pointer;

position: relative;

top: 1px;

}

.tabs>label:not(:first-of-type) {

border-left: none;

}

.tabs>input[type="radio"]:checked+label {

background-color: #fff;

border-bottom: 1px solid #fff;

}

/\* Menu's and submenu's \*/

.abix-tree-list li {

list-style: none;

}

.abix-tree-list .collapsed span.icon,

.abix-tree-list .expanded span.icon {

margin-right: 6px;

cursor: pointer;

}

/\* Switch off modal background and z-index \*/

.modal-backdrop, .modal-backdrop.fade.in{

position: relative;

}

.modal-parameter-input {

width: 250px;

right: 30px;

position: absolute;

}

* 1. Листинг файла common.js

(function(e, t, n, r) {

function o(t, n) {

this.element = t;

this.$element = e(t);

this.options = e.extend({}, s, n);

this.\_defaults = s;

this.\_name = i, this.$topLevelBranches, this.$allBranches, this.init()

}

var i = "abixTreeList",

s = {

collapsedIconClass: "glyphicon glyphicon-plus",

expandedIconClass: "glyphicon glyphicon-minus"

};

o.prototype.init = function() {

var t = this;

t.$topLevelBranches = t.$element.children("li");

t.$allBranches = t.$element.find("li");

t.$element.addClass("abix-tree-list");

t.$allBranches.not(t.$topLevelBranches).hide();

t.$allBranches.each(function() {

var n = e(this).children("ul,ol");

if (n.size() > 0) {

e(this).addClass("collapsed");

e('<span class="icon ' + t.options.collapsedIconClass + '"><\/span>').prependTo(e(this))

}

});

t.$allBranches.children("span.icon").on("click", function(n) {

if (e(this).parent().hasClass("collapsed")) {

t.expand(e(this).parent());

n.stopPropagation()

}

if (e(this).parent().hasClass("expanded")) {

t.collapse(e(this).parent());

n.stopPropagation()

}

});

e("#tree-expand-all").on("click", function(e) {

e.preventDefault();

t.expandAll();

e.stopPropagation()

});

e("#tree-collapse-all").on("click", function(e) {

e.preventDefault();

t.collapseAll();

e.stopPropagation()

})

};

o.prototype.expand = function(e) {

var t = this;

e.children("ul,ol").children("li").show(500, function() {

e.removeClass("collapsed").addClass("expanded");

e.children("span.icon").removeClass(t.options.collapsedIconClass).addClass(t.options.expandedIconClass)

})

};

o.prototype.collapse = function(e) {

var t = this;

e.children("ul,ol").children("li").hide(500, function() {

e.removeClass("expanded").addClass("collapsed");

e.children("span.icon").removeClass(t.options.expandedIconClass).addClass(t.options.collapsedIconClass)

})

};

o.prototype.collapseAll = function() {

var e = this;

e.$allBranches.not(e.$topLevelBranches).hide(1e3, function() {

e.$allBranches.removeClass("expanded").addClass("collapsed");

e.$allBranches.children("span.icon").removeClass(e.options.expandedIconClass).addClass(e.options.collapsedIconClass)

})

};

o.prototype.expandAll = function() {

var e = this;

e.$allBranches.show(1e3, function() {

e.$allBranches.removeClass("collapsed").addClass("expanded");

e.$allBranches.children("span.icon").removeClass(e.options.collapsedIconClass).addClass(e.options.expandedIconClass)

})

};

e.fn[i] = function(t) {

return this.each(function() {

if (!e.data(this, "plugin\_" + i)) {

e.data(this, "plugin\_" + i, new o(this, t))

}

})

}

})(jQuery, window, document)

$(document).ready(function() {

$('#tree').abixTreeList();

});

//actions

$(document).ready(function() {

$(".component-info-save").click(function(event){ //Sending request to index.php to update data about a component

var ClickedComponentButtonSaveId = $(event.target)[0].id.replace('ComponentInfoSave\_','');

//alert(ClickedComponentButtonSaveId);

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

$.get(

"UpdateComponent",

{"ComponentId" : ClickedComponentButtonSaveId, "ComponentName" : $("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val(), "ComponentFullName" : $("#ComponentFieldFullName\_"+ClickedComponentButtonSaveId).val(), "ComponentSerial" : $("#ComponentFieldSerial\_"+ClickedComponentButtonSaveId).val(), "ComponentAmount" : $("#ComponentFieldAmount\_"+ClickedComponentButtonSaveId).val(), "ComponentUnit" : $("#ComponentFieldUnit\_"+ClickedComponentButtonSaveId).val(), "ComponentLocation" : $("#ComponentFieldLocation\_"+ClickedComponentButtonSaveId).val(), "ComponentTag" : $("#ComponentFieldTag\_"+ClickedComponentButtonSaveId).val()}, //UCV - Update Component Value

function(data) {

//alert('page content: ' + data);

alert("Сохранено");

}

);

});

$(".component-info-delete").click(function(event){ //Sending request to index.php to delete a component

var ClickedComponentButtonDeleteId = $(event.target)[0].id.replace('ComponentInfoDelete\_','');

//alert(ClickedComponentButtonSaveId);

if(confirm("Вы уверены, что хотите удалить компонент "+ClickedComponentButtonDeleteId+"?")) {

$.get(

"DeleteComponent",

{"ComponentId" : ClickedComponentButtonDeleteId}, //DTC - Delete This Component

function(data) {

//alert('page content: ' + data);

document.location.reload();

}

);

}

});

$(".component-info-add").click(function(event){ //Sending request to index.php to add a new component

//alert(ClickedComponentButtonSaveId);

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

if($("#ComponentFieldAddName").val().length > 0 && $("#ComponentFieldAddFullName").val().length > 0 && $("#ComponentFieldAddAmount").val().length > 0 && $("#ComponentFieldAddUnit").val().length > 0 && $("#ComponentFieldAddLocation").val().length > 0 && $("#ComponentFieldAddGroup").val().length > 0) {

$.get(

"AddComponent",

{"ComponentName" : $("#ComponentFieldAddName").val(), "ComponentFullName" : $("#ComponentFieldAddFullName").val(), "ComponentSerial" : $("#ComponentFieldAddSerial").val(), "ComponentAmount" : $("#ComponentFieldAddAmount").val(), "ComponentUnit" : $("#ComponentFieldAddUnit").val(), "ComponentLocation" : $("#ComponentFieldAddLocation").val(), "ComponentTag" : $("#ComponentFieldAddTag").val(), "ComponentGroup" : $("#ComponentFieldAddGroup").val()}, //ANC - Add New Component

function(data) {

//alert('page content: ' + data);

document.location.reload();

}

);

} else alert("Проверьте правильность заполнения полей");

});

$(".tree-class-add").click(function(event){ //Sending request to index.php to add a new class

//alert("asd");

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

if($("#ClassFieldAddName").val().length > 0) {

$.get(

"AddClass",

{"ClassName" : $("#ClassFieldAddName").val()}, //ATC - Add Tree Class

function(data) {

//alert('page content: ' + data);

document.location.reload();

}

);

}

else alert("Проверьте правильность заполнения полей");

});

$(".tree-category-add").click(function(event){ //Sending request to index.php to add a new category

//alert($("#CategoryFieldAddClass").val());

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

if($("#CategoryFieldAddName").val().length > 0 && $("#CategoryFieldAddClass").val().length > 0) {

$.get(

"AddCategory",

{"CategoryName": $("#CategoryFieldAddName").val(), "CategoryClass": $("#CategoryFieldAddClass").val()}, //ATT - Add Tree caTegory

function (data) {

//alert('page content: ' + data);

document.location.reload();

}

);

} else alert("Проверьте правильность заполнения полей");

});

$(".tree-group-add").click(function(event) { //Sending request to index.php to add a new group

//alert("asd");

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

//alert($("#GroupFieldAddName").val());

//alert($("#GroupFieldAddCategory").val());

if($("#GroupFieldAddName").val().length > 0 && $("#GroupFieldAddCategory").val().length > 0) {

$.get(

"AddGroup",

{"GroupName": $("#GroupFieldAddName").val(), "GroupCategory": $("#GroupFieldAddCategory").val()}, //ATG - Add Tree Group

function (data) {

//alert('page content: ' + data);

document.location.reload();

}

);

} else alert("Проверьте правильность заполнения полей");

});

setInterval(function(){ //Updating transactions table every 5 seconds if table is visible

if($("#content-2").css('display') != "none") {

//alert("asd");

$.ajax({

type: "GET",

url: "GetTransactions",

data: {}

}).done(function( data ) {

//alert(data)

//alert( data.split('<')[0] );

let obj = JSON.parse( data );

$('#TransactionsTable tbody').empty();

$.each(obj, function(i, item) {

$('#TransactionsTable > tbody:last-child').append("<tr><td>" + item.TransactionDate + "</td><td>" + item.ComponentName + "</td><td>" + item.TransactionDifferent + "</td><td>" + item.StaffName + "</td><td>" + item.TransactionPlace + "</td><td>" + item.TransactionNote + "</td></tr>");

});

});

}

}, 5000);

$(".transaction-add-confirm").click(function(event){ //Sending request to index.php for add a new transaction

//alert("Добавления нет, но вы держитесь. Оно появится, но не очень, а если и да, то может быть и нет.");

//alert($("#ComponentFieldName\_"+ClickedComponentButtonSaveId).val());

if($("#TransactionFieldAddDate").val().length > 0 && $("#TransactionFieldAddDifferent").val().length > 0 && $("#TransactionFieldAddStaff").val().length > 0) {

$.get(

"AddTransaction",

{"TransactionComponentId" : $("#TransactionFieldAddId").val(), "TransactionDate" : $("#TransactionFieldAddDate").val(), "TransactionDifferent" : $("#TransactionFieldAddDifferent").val(), "TransactionStaffId" : $("#TransactionFieldAddStaff").val(), "TransactionPlace" : $("#TransactionFieldAddPlace").val(), "TransactionNote" : $("#TransactionFieldAddNote").val()}, //ANT - Add New Transaction

function(data) {

//alert('page content: ' + data);

document.location.reload();

}

);

} else alert("Проверьте правильность заполнения полей");

});

$(".transaction-add").click(function(event){ //Opening transaction modal window handler

var ClickedTransactionButtonAddId = $(event.target)[0].id.replace('TransactionAdd\_','');

//alert(ClickedTransactionButtonAddId);

$('.component-info').modal('hide');

$('#TransactionFieldAddId').val(ClickedTransactionButtonAddId);

});

});

* 1. Листинг файла 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">

<title>Вход</title>

<style>

@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {

width: 360px;

padding: 8% 0 0;

margin: auto;

}

.form {

position: relative;

z-index: 1;

background: #FFFFFF;

max-width: 360px;

margin: 0 auto 100px;

padding: 45px;

text-align: center;

box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form input {

font-family: "Roboto", sans-serif;

outline: 0;

background: #f2f2f2;

width: 100%;

border: 0;

margin: 0 0 15px;

padding: 15px;

box-sizing: border-box;

font-size: 14px;

}

.form button {

font-family: "Roboto", sans-serif;

text-transform: uppercase;

outline: 0;

background: #2783f5;

width: 100%;

border: 0;

padding: 15px;

color: #FFFFFF;

font-size: 14px;

-webkit-transition: all 0.3 ease;

transition: all 0.3 ease;

cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

background: #3090ff;

}

.form .message {

margin: 15px 0 0;

color: #b3b3b3;

font-size: 12px;

}

.form .message a {

color: #2783f5;

text-decoration: none;

}

.container .info {

margin: 50px auto;

text-align: center;

}

.container .info h1 {

margin: 0 0 15px;

padding: 0;

font-size: 36px;

font-weight: 300;

color: #1a1a1a;

}

.container .info span {

color: #2783f5;

font-size: 12px;

}

.container .info span a {

color: #000000;

text-decoration: none;

}

.container .info span .fa {

color: #2783f5;

}

body {

font-family: "Roboto", sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

</style>

</head>

<body>

<div th:if="${param.error}">

Неправильный логин или пароль.

</div>

<div th:if="${param.logout}">

Вы вышли из системы.

</div>

<div class="login-page">

<div class="form">

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

<input type="text" name="username" placeholder="Логин"/>

<input type="password" name="password" placeholder="Пароль"/>

<button type="submit">Войти</button>

<p class="message">Нет аккаунта? <a href="/registration">Зарегистрируйтесь</a></p>

</form>

</div>

</div>

</body>

</html>

* 1. Листинг файла registration.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">

<title>Регистрация</title>

<style>

@import url(https://fonts.googleapis.com/css?family=Roboto:300);

.login-page {

width: 360px;

padding: 8% 0 0;

margin: auto;

}

.form {

position: relative;

z-index: 1;

background: #FFFFFF;

max-width: 360px;

margin: 0 auto 100px;

padding: 45px;

text-align: center;

box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);

}

.form input {

font-family: "Roboto", sans-serif;

outline: 0;

background: #f2f2f2;

width: 100%;

border: 0;

margin: 0 0 15px;

padding: 15px;

box-sizing: border-box;

font-size: 14px;

}

.form button {

font-family: "Roboto", sans-serif;

text-transform: uppercase;

outline: 0;

background: #2783f5;

width: 100%;

border: 0;

padding: 15px;

color: #FFFFFF;

font-size: 14px;

-webkit-transition: all 0.3 ease;

transition: all 0.3 ease;

cursor: pointer;

}

.form button:hover,.form button:active,.form button:focus {

background: #3090ff;

}

.form .message {

margin: 15px 0 0;

color: #b3b3b3;

font-size: 12px;

}

.form .message a {

color: #2783f5;

text-decoration: none;

}

.container .info {

margin: 50px auto;

text-align: center;

}

.container .info h1 {

margin: 0 0 15px;

padding: 0;

font-size: 36px;

font-weight: 300;

color: #1a1a1a;

}

.container .info span {

color: #2783f5;

font-size: 12px;

}

.container .info span a {

color: #000000;

text-decoration: none;

}

.container .info span .fa {

color: #2783f5;

}

body {

font-family: "Roboto", sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

</style>

</head>

<body>

<div th:if="${#messages}">

<p th:text="${message}"></p>

</div>

<div class="login-page">

<div class="form">

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

<input type="text" name="username" placeholder="Логин"/>

<input type="password" name="password" placeholder="Пароль"/>

<button type="submit">Регистрация</button>

<p class="message">Есть аккаунт? <a href="/login">Войдите</a></p>

</form>

</div>

</div>

</body>

</html>

* 1. Листинг файла userEdit.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>

<title>Редактирование пользователей</title>

</head>

<body>

<form th:action="${'/admin'}" method="post">

<div th:each="el: ${user}">

<input type="hidden" th:value="${el.id}" name="userId">

<input type="text" th:value="${el.username}" name="username">

</div>

<div th:each="role: ${roles}">

<div th:each="el: ${user}">

<input type="checkbox" th:id="${role}" th:name="${'roles[]'}" th:value="${role}" th:checked="${#lists.contains(el.roles, role)}"/>

<label th:text="${role}"></label>

</div>

</div>

</form>

</body>

</html>

* 1. Листинг файла userList.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>

<title> </title>

</head>

<body>

<div th:each="el : ${users}">

<p th:text="${el.username}"></p>

<p th:text="${el.roles}"></p>

<a th:href="'/admin/'+${el.id}+'/edit'">Редактировать</a>

</div>

</body>

</html>

* 1. Листинг файла Warehouse.html

<!DOCTYPE html>

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

<head>

<meta charset="UTF-8">

<title>Склад</title>

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

<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" rel="stylesheet" type="text/css">

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js" type="text/javascript"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>

<link href="../static/css/common.css" th:href="@{/css/common.css}" rel="stylesheet" />

<script type="text/javascript" th:src="@{/js/common.js}"></script>

</head>

<body>

<!-- Modals section -->

<!-- Add component -->

<div class="modal fade" id="ComponentAdd" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Новый компонент</h5>

</div>

<div class="modal-body">

<label style="font-weight: normal; padding-right: 10px;">Наименование</label><input class="modal-parameter-input" id="ComponentFieldAddName"><br>

<label style="font-weight: normal; padding-right: 10px;">Полное наименование</label><input class="modal-parameter-input" id="ComponentFieldAddFullName"><br>

<label style="font-weight: normal; padding-right: 10px;">Серийный номер</label><input class="modal-parameter-input" id="ComponentFieldAddSerial"><br>

<label style="font-weight: normal; padding-right: 10px;">Количество</label><input class="modal-parameter-input" id="ComponentFieldAddAmount"><input class="modal-parameter-input" style="width: 125px;" id="ComponentFieldAddUnit"><br>

<label style="font-weight: normal; padding-right: 10px;">Расположение</label><input class="modal-parameter-input" id="ComponentFieldAddLocation"><br>

<label style="font-weight: normal; padding-right: 10px;">Тег</label><input class="modal-parameter-input" id="ComponentFieldAddTag"><br>

<label style="font-weight: normal; padding-right: 10px;">Группа</label>

<select class="form-control" id="ComponentFieldAddGroup">

[(${SelectComponent})]

</select><br>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" class="btn btn-primary component-info-add">Добавить</button>

</div>

</div>

</div>

</div>

<!-- Add class -->

<div class="modal fade" id="GeneralAddClass" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Новый класс</h5>

</div>

<div class="modal-body">

<label style="font-weight: normal; padding-right: 10px;">Наименование</label><input type="text" required class="modal-parameter-input" id="ClassFieldAddName"><br>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="submit" id="TreeClassAdd" class="btn btn-primary tree-class-add">Добавить</button>

</div>

</div>

</div>

</div>

<!-- Add category -->

<div class="modal fade" id="GeneralAddCategory" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Новая категория</h5>

</div>

<div class="modal-body">

<label style="font-weight: normal; padding-right: 10px;">Наименование</label><input class="modal-parameter-input" id="CategoryFieldAddName"><br>

<label style="font-weight: normal; padding-right: 10px;">Класс</label><select id="CategoryFieldAddClass" class="form-control">

[(${SelectCategory})]

</select>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" id="TreeCategoryAdd" class="btn btn-primary tree-category-add">Добавить</button>

</div>

</div>

</div>

</div>

<!-- Add group -->

<div class="modal fade" id="GeneralAddGroup" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Новая группа</h5>

</div>

<div class="modal-body">

<label style="font-weight: normal; padding-right: 10px;">Наименование</label><input class="modal-parameter-input" id="GroupFieldAddName"><br>

<label style="font-weight: normal; padding-right: 10px;">Категория</label><select id="GroupFieldAddCategory" class="form-control">

[(${SelectGroup})]

</select>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" id="TreeGroupAdd" class="btn btn-primary tree-group-add">Добавить</button>

</div>

</div>

</div>

</div>

<!-- Add transaction -->

<div class="modal fade" id="TransactionAdd" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Новая транзакция</h5>

</div>

<div class="modal-body">

<label style="font-weight: normal; padding-right: 10px;">Id компонента</label><input class="modal-parameter-input" id="TransactionFieldAddId" readonly><br>

<label style="font-weight: normal; padding-right: 10px;">Дата</label><input type="date" class="modal-parameter-input" id="TransactionFieldAddDate"><br>

<label style="font-weight: normal; padding-right: 10px;">Разница</label><input class="modal-parameter-input" id="TransactionFieldAddDifferent"><br>

<label class="control-label" style="font-weight: normal; padding-right: 10px;">Сотрудник</label>

<select id="TransactionFieldAddStaff" class="form-control">

[(${SelectStaff})]

</select>

<label style="font-weight: normal; padding-right: 10px;">Цель</label><input class="modal-parameter-input" id="TransactionFieldAddPlace"><br>

<label style="font-weight: normal; padding-right: 10px;">Примечание</label><input class="modal-parameter-input" id="TransactionFieldAddNote"><br>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Закрыть</button>

<button type="button" class="btn btn-primary transaction-add-confirm">Добавить</button>

</div>

</div>

</div>

</div>

<!-- Modals section -->

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

<input type="submit" class="btn btn-danger" value="Выйти" style="position: absolute; right: 10px;">

</form>

<input style="position: absolute; top: 10px; width: 100%;" type="text" name="" id="Search" placeholder="Поиск...">

<div class="tabs">

<input checked id="tab-btn-1" name="tab-btn" type="radio" value=""> <label for="tab-btn-1">Склад</label> <input id="tab-btn-2" name="tab-btn" type="radio" value=""> <label for="tab-btn-2">Приход/Расход</label>

<div id="content-1" style="position: absolute;">

<p>

<button id="tree-expand-all" class="btn btn-primary">Развернуть все</button>

<button style="left: 10px; position: relative;" id="tree-collapse-all" class="btn btn-primary">Свернуть все</button>

<div class="dropdown show">

<button class="btn btn-primary dropdown-toggle"id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" style="">Добавить</button>

<div class="dropdown-menu" aria-labelledby="dropdownMenuLink">

<button style="position: relative; width: 100%;" class="btn btn-primary dropdown-item" data-toggle='modal' data-target="#ComponentAdd">Компонент</button><br>

<button style="position: relative; width: 100%; top: 3px;" class="btn btn-primary dropdown-item" data-toggle='modal' data-target="#GeneralAddGroup">Группу</button><br>

<button style="position: relative; width: 100%; top: 5px;" class="btn btn-primary dropdown-item" data-toggle='modal' data-target="#GeneralAddCategory">Категорию</button><br>

<button style="position: relative; width: 100%; top: 7px;" class="btn btn-primary dropdown-item" data-toggle='modal' data-target="#GeneralAddClass">Класс</button><br>

</div>

</div>

</p>

<ul id="tree">

[(${ComponentsTree})]

</ul>

</div>

<div id="content-2" style="position: absolute;">

<table class="table" id="TransactionsTable">

<thead>

<tr>

<th scope="col">Дата</th>

<th scope="col">Id компонента</th>

<th scope="col">Разница</th>

<th scope="col">Id Сотрудника</th>

<th scope="col">Цель</th>

<th scope="col">Примечание</th>

</tr>

</thead>

<tbody>

<!-- Asynchronous data filling -->

</tbody>

</table>

</div>

</div>

<script type="text/javascript"> //Поиск

var TreeElem = document.getElementById("tree").getElementsByTagName("li");

$("#Search").keypress(function(event){

console.log(String.fromCharCode(event.which));

//$('#tree-expand-all').trigger('click');

for (i = 0; i < TreeElem.length; i++) {

if (TreeElem[i].innerText.toLowerCase().indexOf($("#Search").val().toLowerCase()) > -1 && $("#Search").val().length > 0) {

//TreeElem[i].style.border = "1px solid red";

TreeElem[i].style.display = "block";

}

else {

//TreeElem[i].style.border = "none";

TreeElem[i].style.display = "none";

}

if ($("#Search").val() == '') {

TreeElem[i].style.display = "block";

}

//console.log(TreeElem[i].innerText);

}

//alert($("#Search").val());

}).on('keydown', function(e) {

if (e.keyCode==8) $('#Search').trigger('keypress');

});

</script>

</body>

</html>

1. Авторизация и регистрация. Создаем конфиг MVC



Рисунок 50 – Конфигурационный файл

1. Создаем конфиг, где прописываем, куда переходить при старте приложения. Указываем страницы, на которые можно перейти без авторизации и где нужна обязательная авторизация. Прописываем запросы для получения списка пользователей.

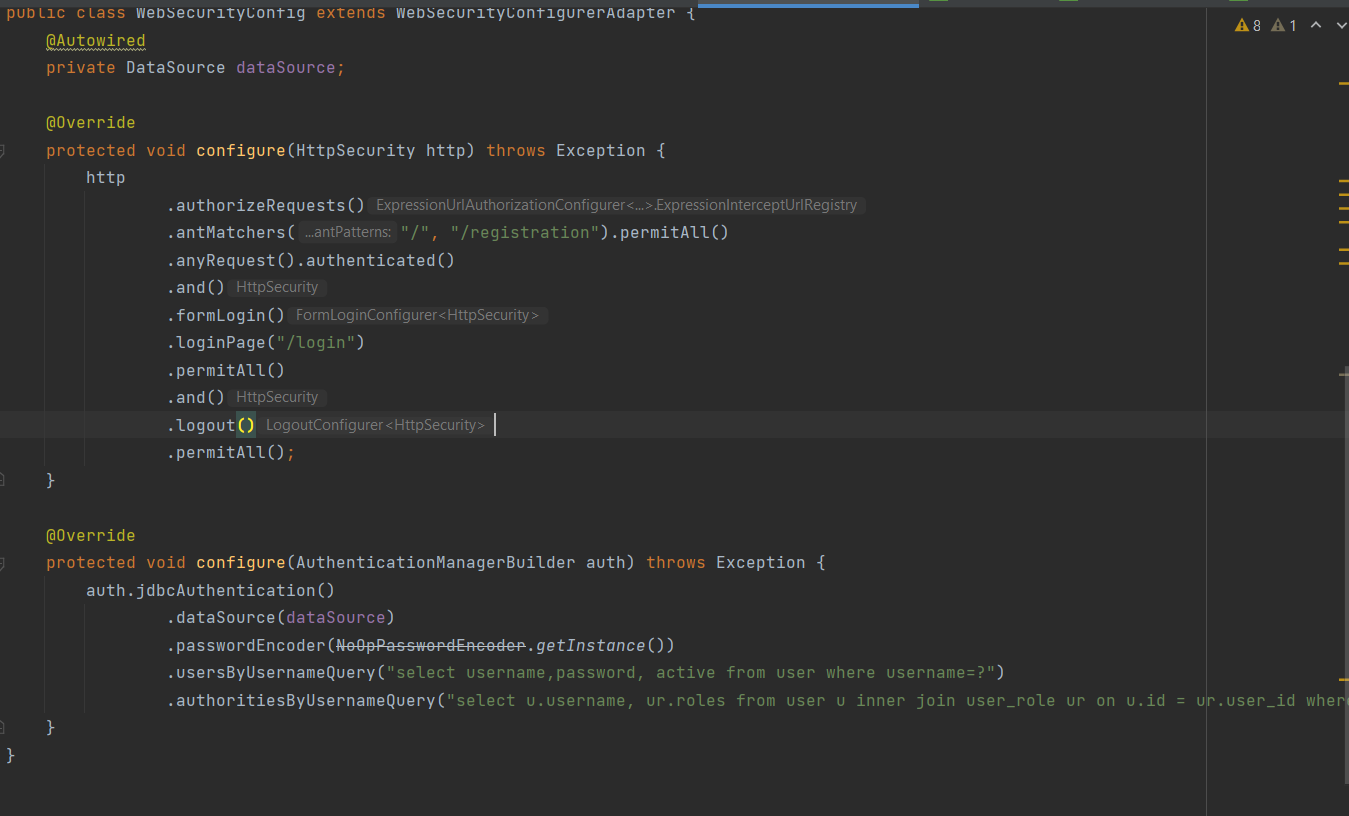


Рисунок 51 – Конфигурационный файл

1. В шаблоне главной страницы добавляем кнопку выхода

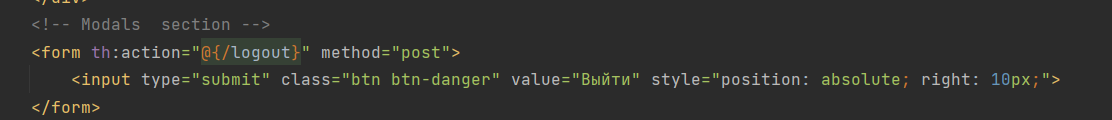


Рисунок 52 – Кнопка выхода

1. Аннотация @CollectionTable описывает что данное поле будет хранится в отдельной таблице. Пишем название таблицы user\_role. Таким образом это нам позволяет создать таблицу user\_role для набора ролей, которая будет соединяться с текущей таблицей через user\_id.



Рисунок 53 – Модель пользователя

1. Создадим шаблоны авторизации и регистрации

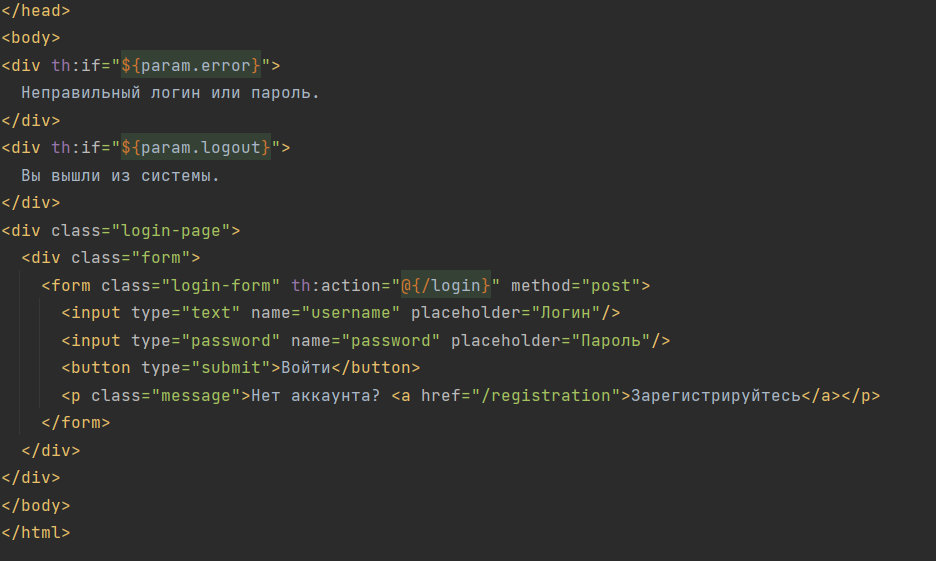


Рисунок 54 -Шаблон авторизации



Рисунок 55 – Шаблон регистрации

1. Результат работы. При открытии страницы попадаем на форму авторизации. При нажатии на кнопку регистрации переходим на форму регистрации.

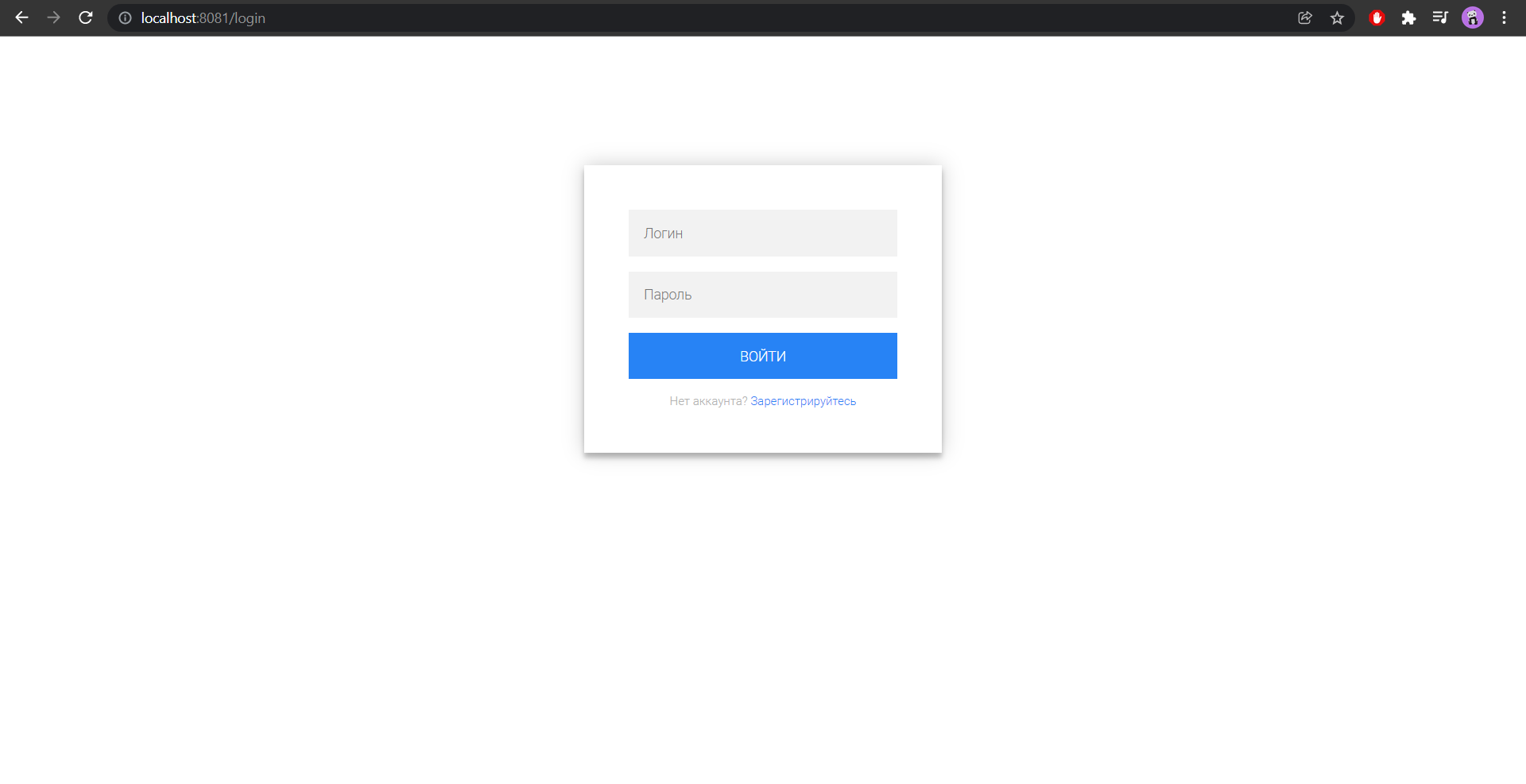


Рисунок 56 – Форма авторизации

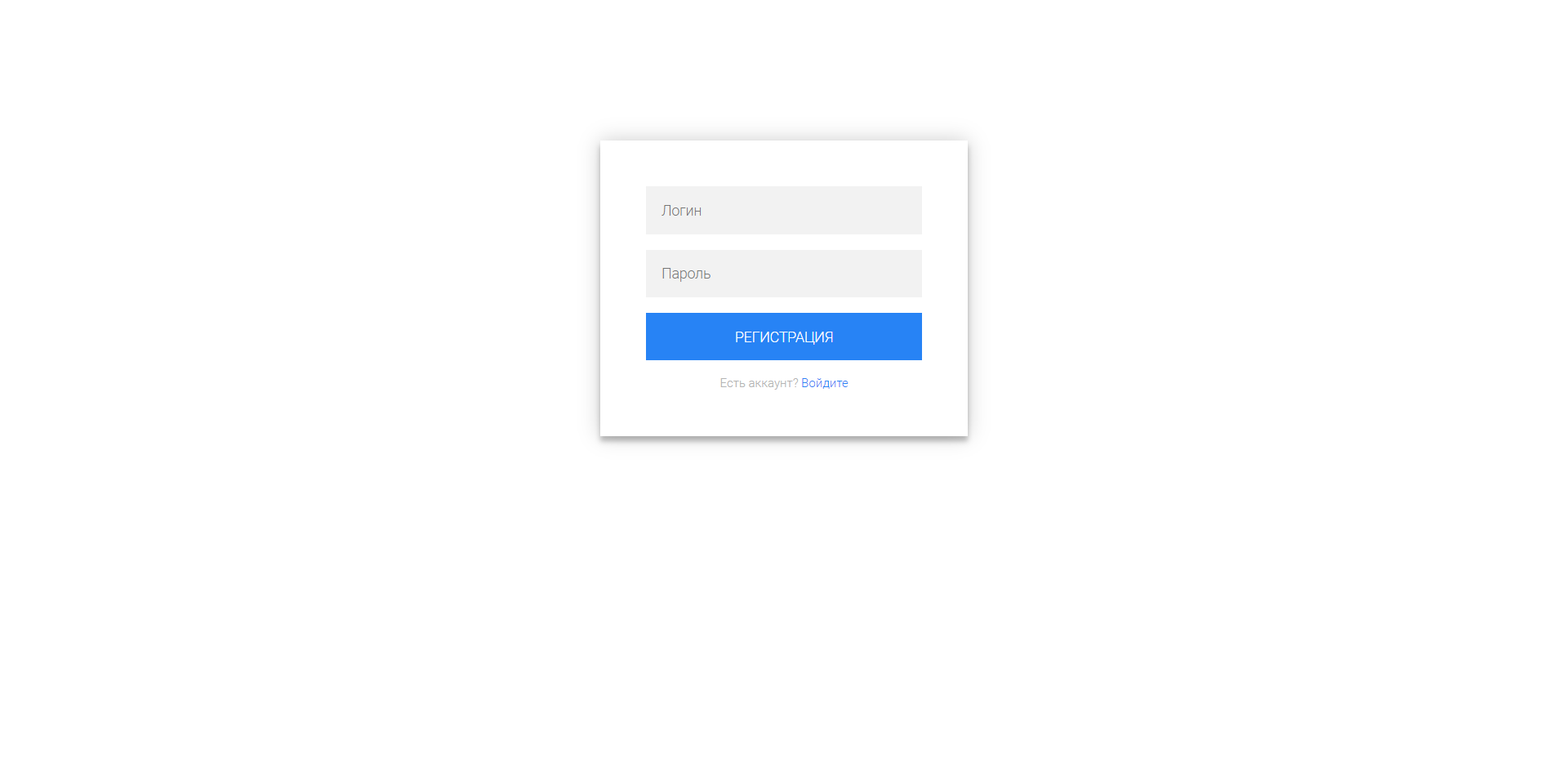


Рисунок 57 – Форма регистрации

1. После успешной авторизации попадаем на главную страницу с деревом компонентов, транзакциями, поиском и кнопками действий.

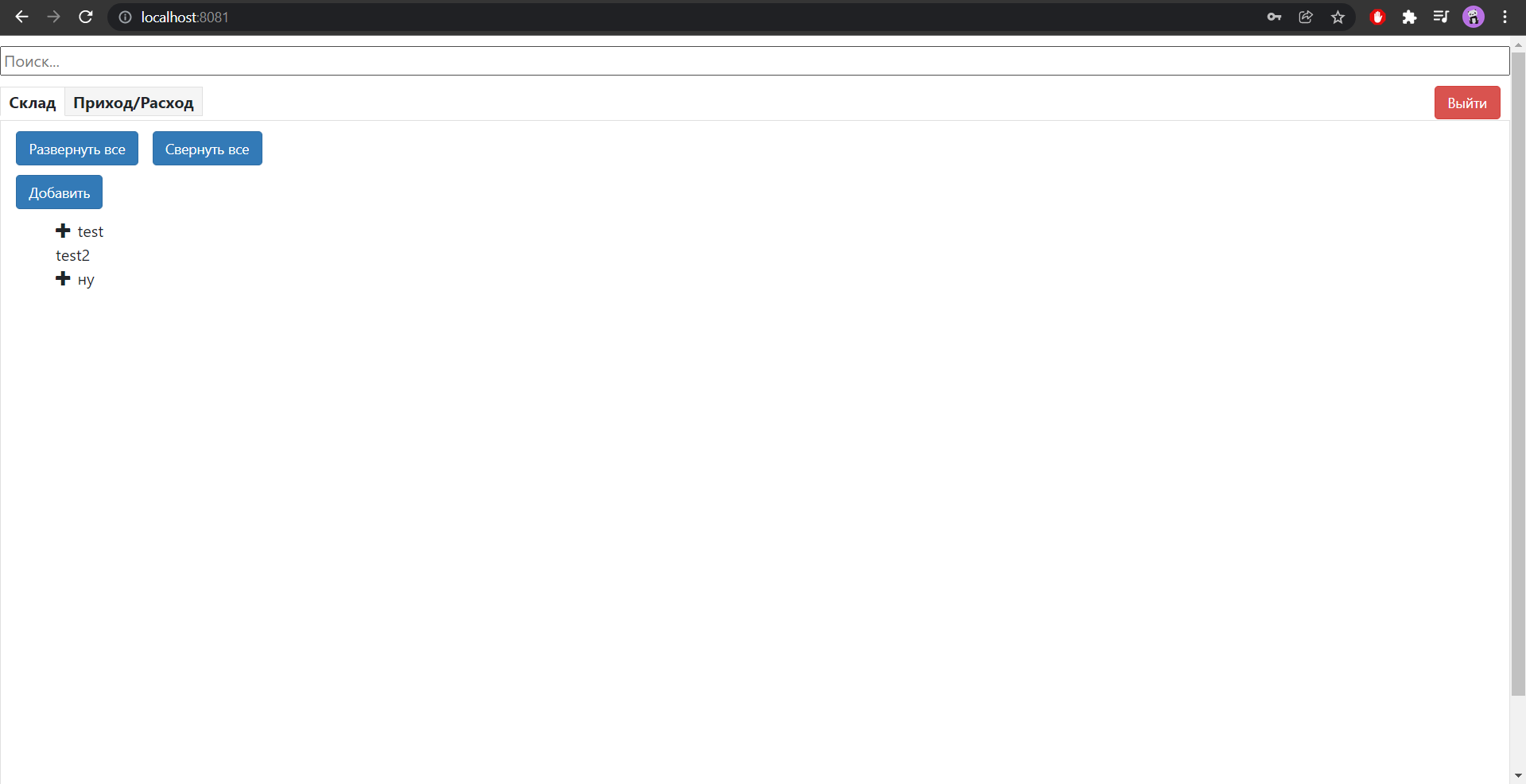


Рисунок 58 – Главная страница

1. Дерево компонентов можно развернуть, открыть нужный компонент или провести динамический поиск по дереву.



Рисунок 59 – Развернутое дерево

1. Также можно открыть карточку компонента, чтобы изменить данные, удалить компонент или совершить с ним транзакцию.

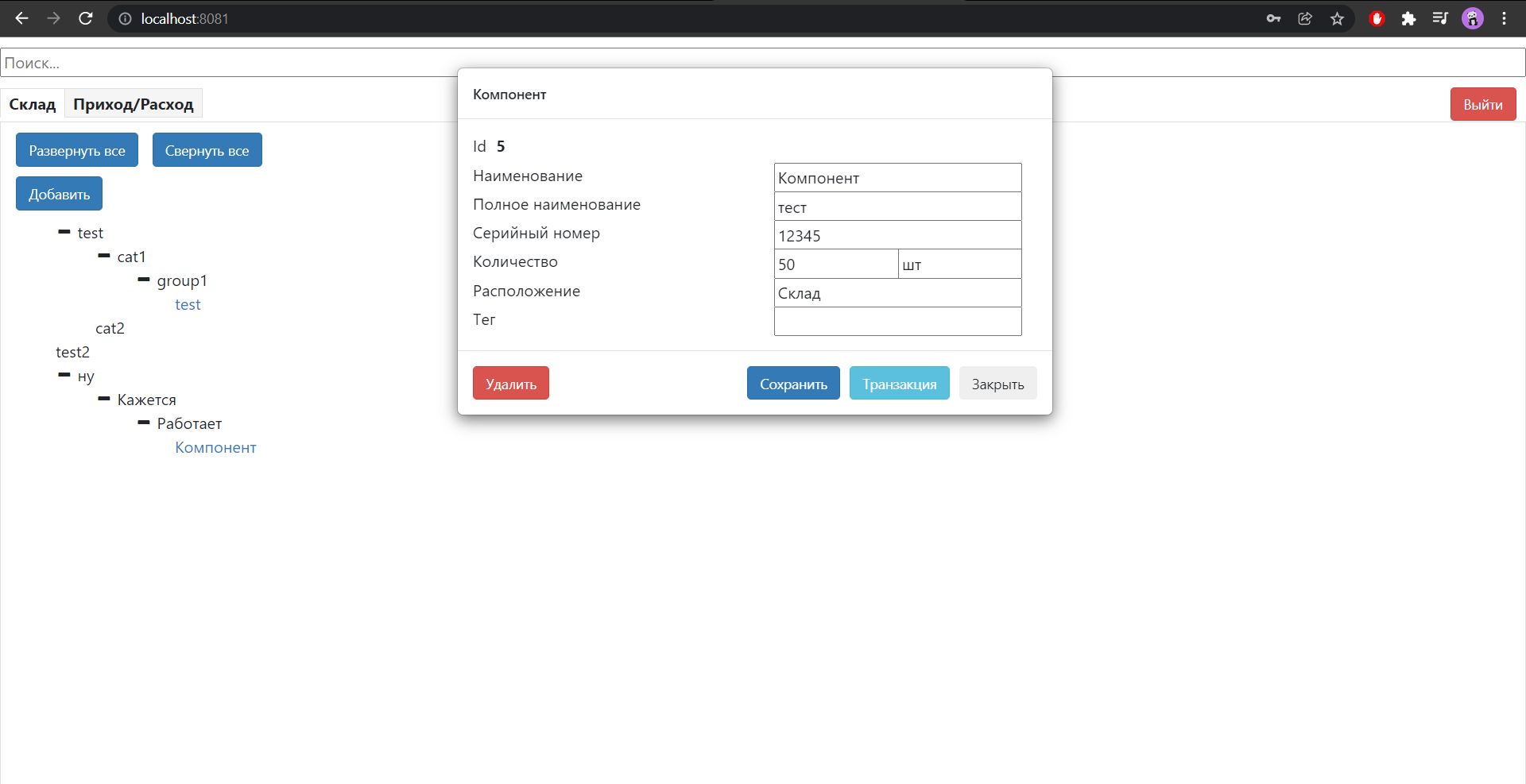


Рисунок 60 – Карточка компонента

1. Таблица транзакций. Сюда асинхронно добавляются новые транзакции для большего удобства сотрудников склада.

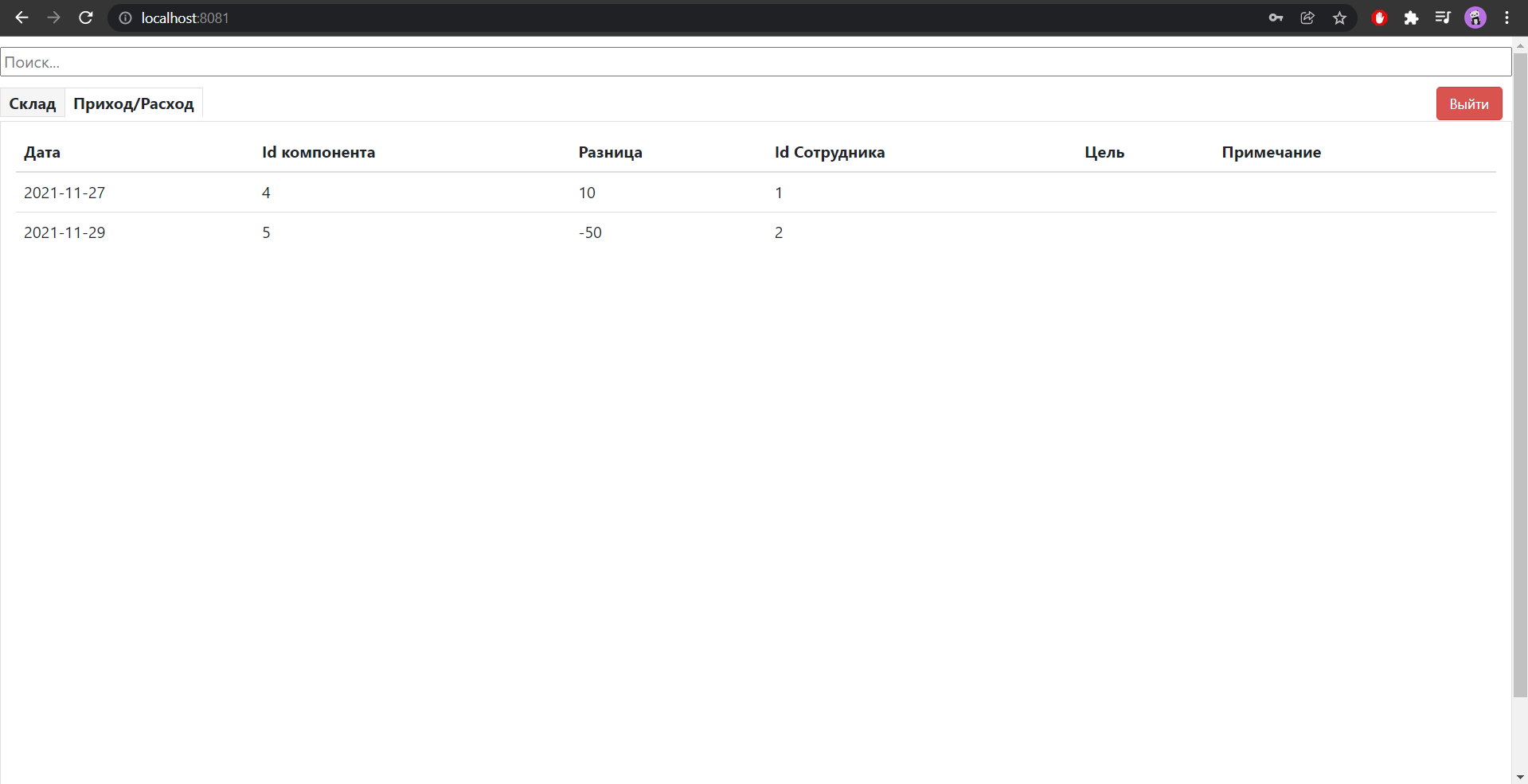


Рисунок 55 – Таблица транзакций

1. В дерево можно добавлять новые компоненты при помощи вспомогательных кнопок

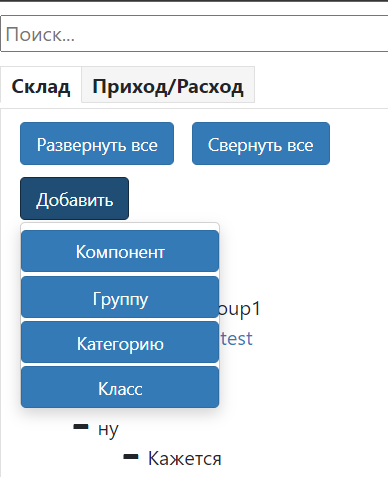


Рисунок 56 – Вспомогательные кнопки

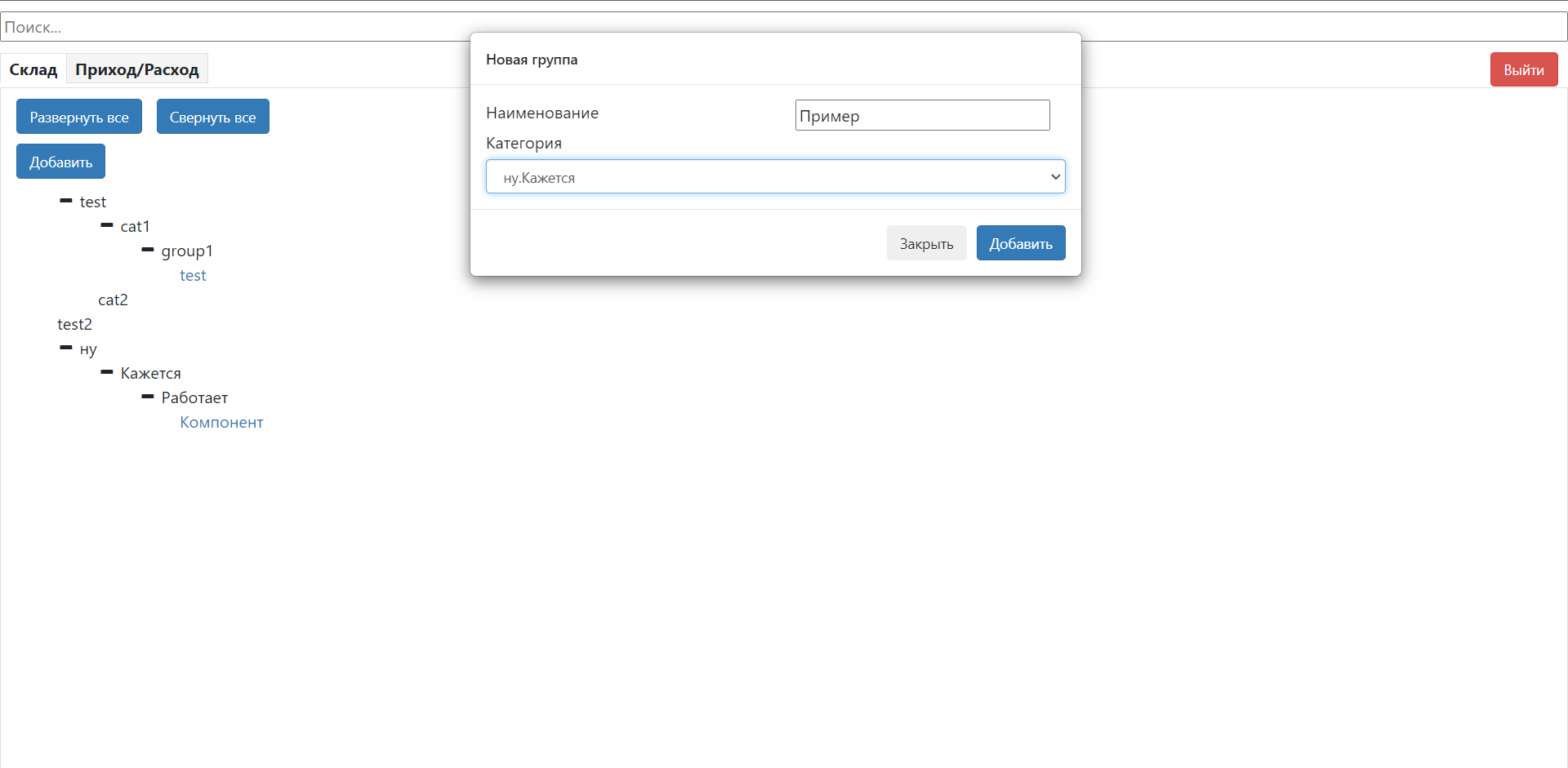


Рисунок 57 – Добавление новой группы

Вывод: Изучили основы работы с фреймворком Spring, Bootstrap, Thymeleaf и разработали проект по индивидуальной теме.