Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКИЕ РАБОТЫ

УП 04.01 ВППО

|  |  |
| --- | --- |
| Выполнил студент  группы П50-6-20  Баранов Андрей Викторович | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ Д.В. Серяк  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2023 года |

Москва 2023

ОГЛАВЛЕНИЕ

[ПРАКТИЧЕСКАЯ РАБОТА №1 3](#_Toc154545059)

[Калькулятор и конвертер валют 3](#_Toc154545060)

[ПРАКТИЧЕСКАЯ РАБОТА №2 10](#_Toc154545061)

[Работа с паттерном DAO 10](#_Toc154545062)

[ПРАКТИЧЕСКАЯ РАБОТА №3 14](#_Toc154545063)

[Работа с JPA и Validator 14](#_Toc154545064)

[ПРАКТИЧЕСКАЯ РАБОТА №4 18](#_Toc154545065)

[Связи 18](#_Toc154545066)

[ПРАКТИЧЕСКАЯ РАБОТА №5 21](#_Toc154545067)

[Авторизация и регистрация 21](#_Toc154545068)

[ПРАКТИЧЕСКАЯ РАБОТА №6 26](#_Toc154545069)

[Разделение прав доступа 26](#_Toc154545070)

# ПРАКТИЧЕСКАЯ РАБОТА №1

## Калькулятор и конвертер валют

Цель работы: создать веб-приложение с калькулятором и конвертером валют на Spring Boot.

1. Создание проекта

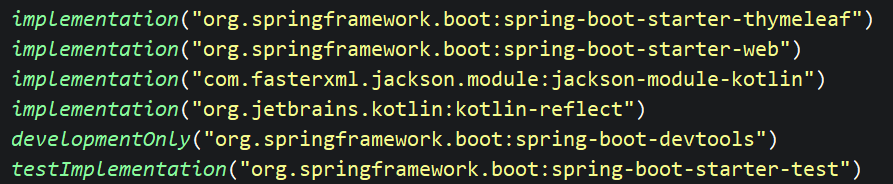


Рисунок 1 – Подключение зависимостей

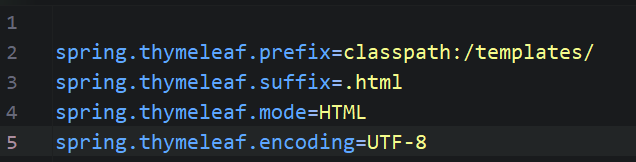


Рисунок 2 – Конфигурация Thymeleaf

1. Главный сайт

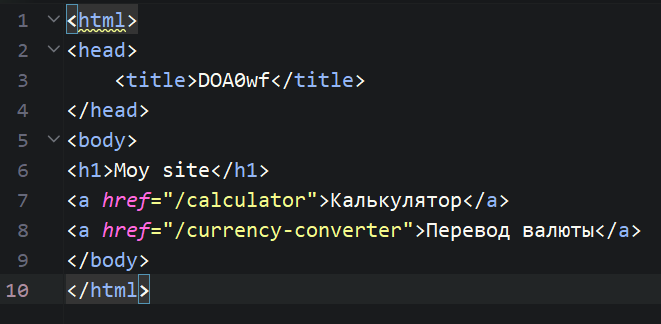


Рисунок 3 – Index.html

1. Конвертер, через form идёт запрос http в Spring.

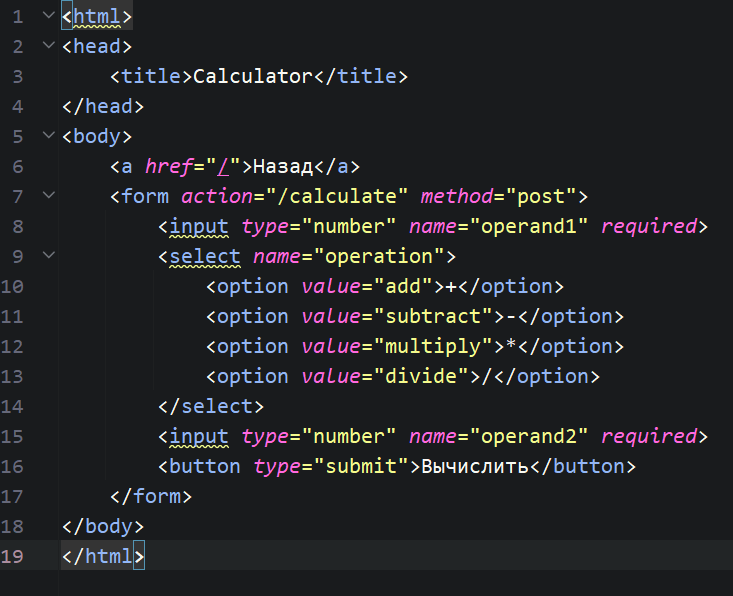


Рисунок 4 – Конвертер валюты

1. Конвертер контроллер, получение страницы, и вычисление конвертации.

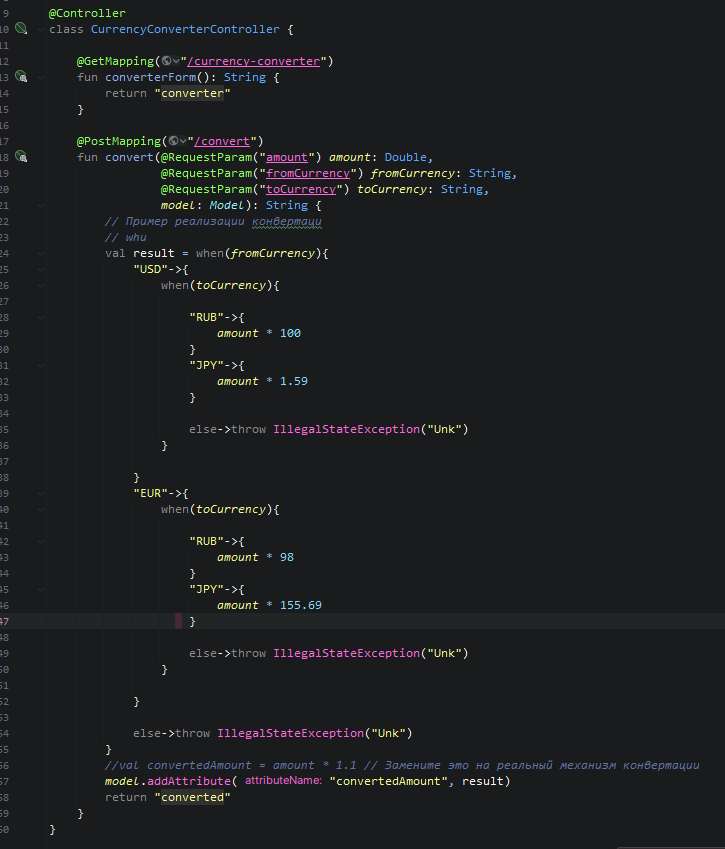


Рисунок 5 – Контроллер для валюты

1. Калькулятор он прибавляет складывает, минусует, значения и так далее.



Рисунок 6 – Страница калькулятора



Рисунок 7 – Контроллер калькулятора

1. Результат работы

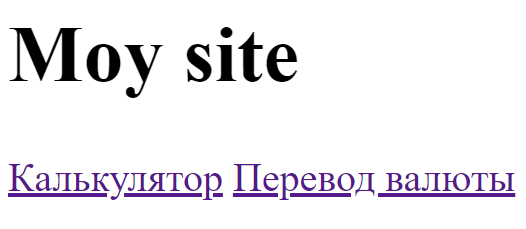


Рисунок 8 – Главная страница

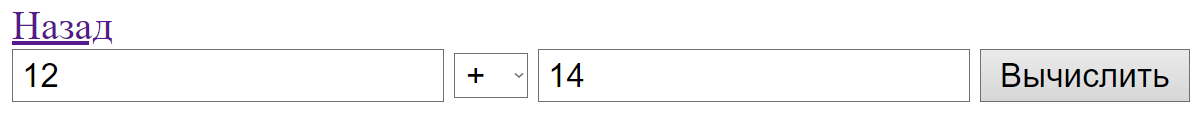


Рисунок 9 – Калькулятор

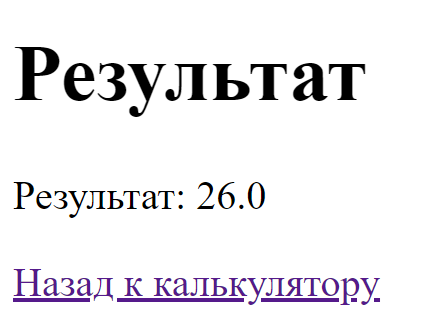


Рисунок 10 – Результат

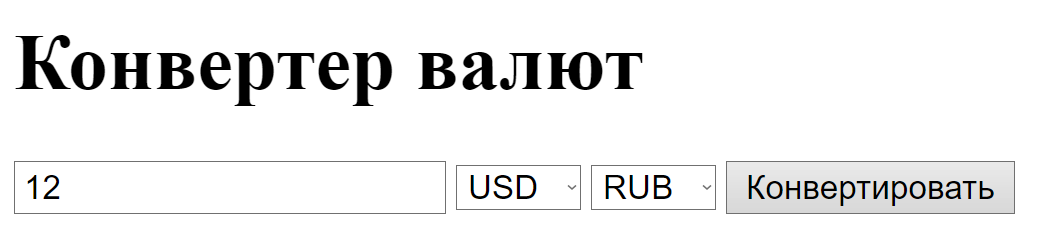


Рисунок 11 – Конвертер валют

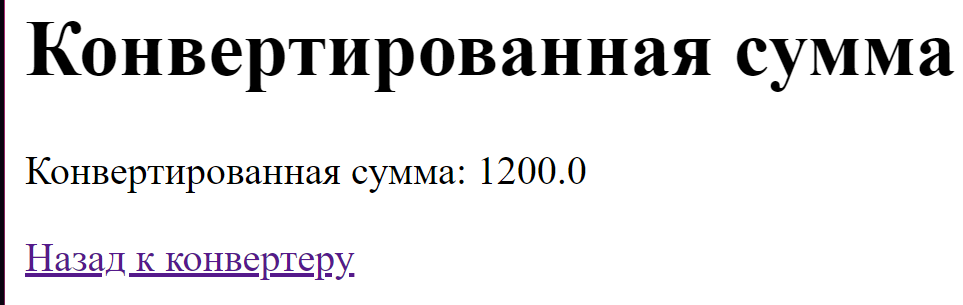


Рисунок 12 – Конвертированная сумма

Вывод: Приложение демонстрирует использование Spring MVC для обработки запросов и динамического отображения результатов.

# ПРАКТИЧЕСКАЯ РАБОТА №2

## Работа с паттерном DAO

Цель работы: разработка паттерна Data Access Object (DAO) для управления данными, создание моделей и контроллеров для взаимодействия с этими данными, а также разработка шаблонов страниц, которые будут загружаться через Endpoints

1. Все модели будут имплементировать этот интерфейс.

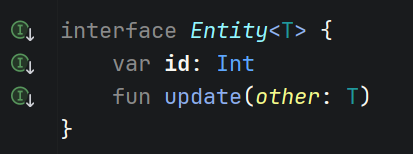


Рисунок 13 - Интерфейс

1. Импровизация бд с дженериком по интерфейсу.

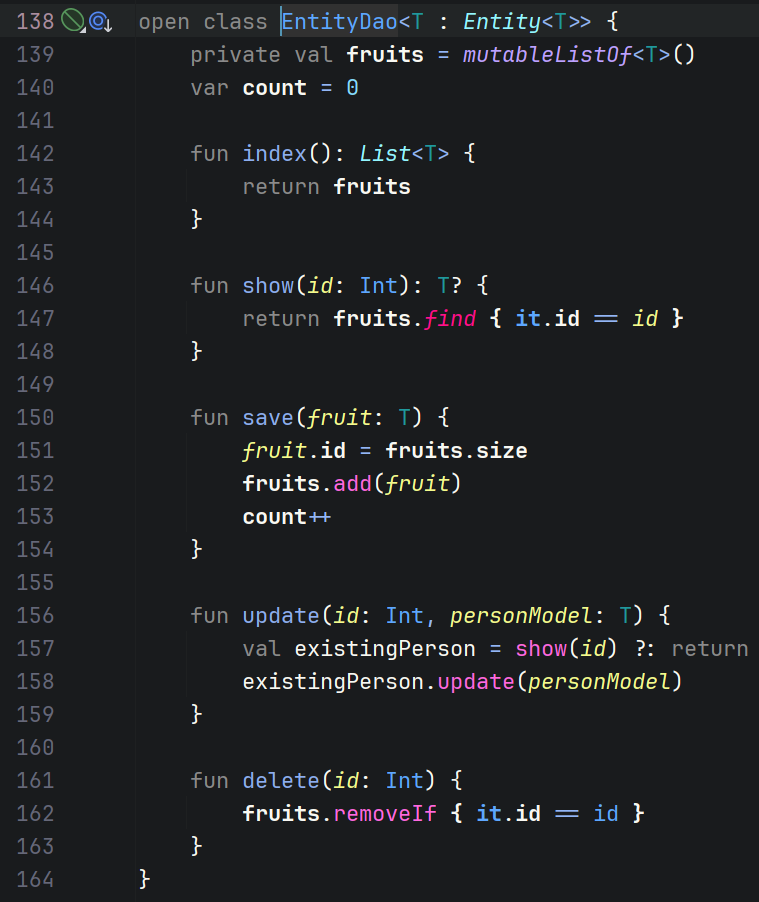


Рисунок 14 – GenericDAO

1. Создание моделей



Рисунок 15 - Модели

1. Заполняем данными, и @Component нужен для создания bean, для Dependency Injection.



Рисунок 16 – BookDAO

1. Создание контроллера, который подключает к себе компонент DaoBook, индекс для загрузки страницы, отображение, добавление, изменение, удаление.



Рисунок 17 – Контроллер Book

1. Пишем много HTML, подключаем туда Bootstrap для стилей. Проделываем так для каждого Entity.

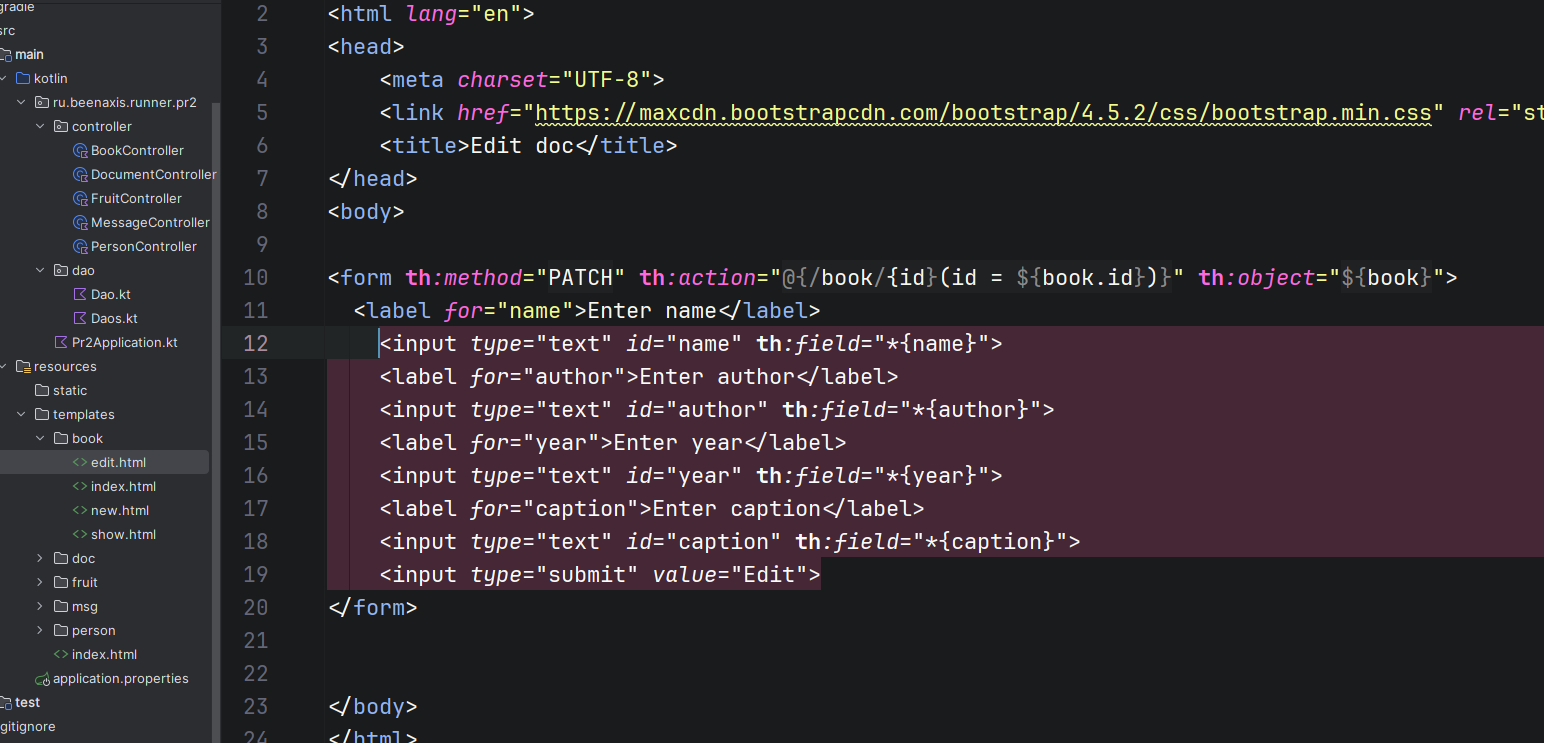


Рисунок 18 – Html страница Edit

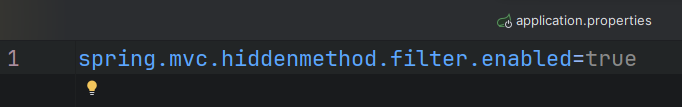


Рисунок 19 – Если это не добавить, не будет ничего работать в MVC.

Вывод: В ходе практической был разработан паттерн DAO, созданы модели, контроллеры, и шаблоны страниц для загрузки из Endpoints.

# ПРАКТИЧЕСКАЯ РАБОТА №3

## Работа с JPA и Validator

Цель работы: Освоение принципов создания и управления сущностями в рамках Java Persistence API (JPA) и Validator, а также разработка контроллеров для взаимодействия с базой данных и реализация функциональности простого поиска в приложении.

1. Конфигурация подключения к БД Postgres.

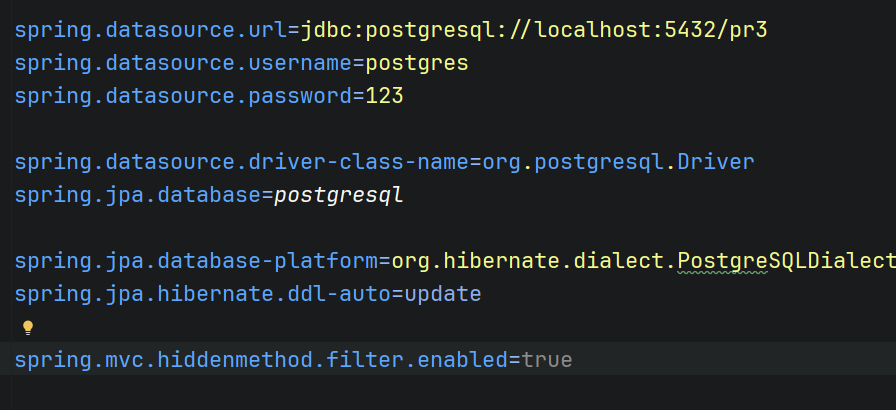


Рисунок 20 – Конфиг спринга

1. Подключение зависимостей и драйвера постгрес.

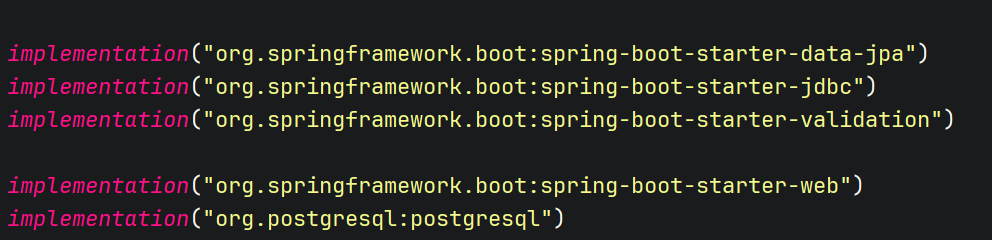


Рисунок 21 – Подключение зависимостей

1. Использование аннотаций Id – поле с айди, @Entity – обозначает что этот класс является сущностью, @Table – определяет название таблицы. @Column – параметры столбца. @Size – размеры строки.

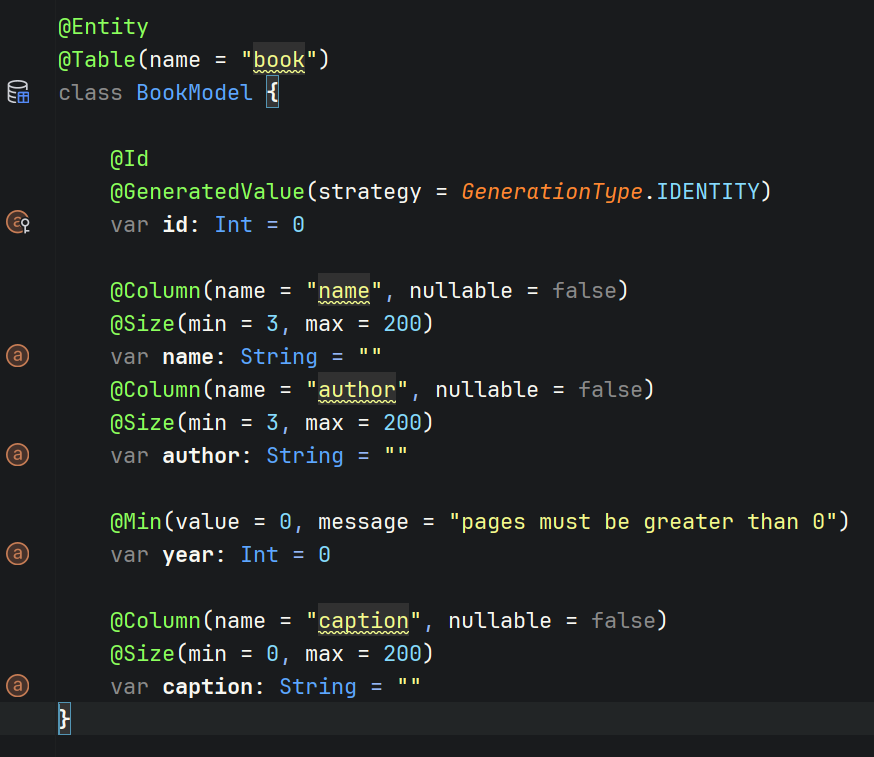


Рисунок 22 – Аннотации JPA и Validator

1. MinMax, означает минимальное и максимально возможное значение поля.

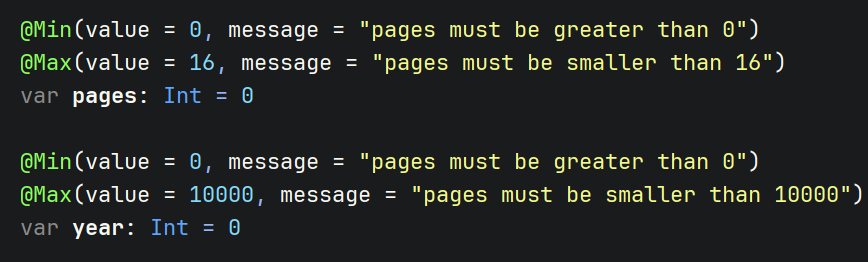


Рисунок 23 – MinMax Аннотации

1. Аннотации Positive означает что в этом поле всегда будет положительное число

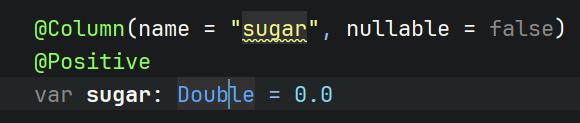


Рисунок 24 - @Positive

1. Создание прокси репозитория, добавление прокси функции, которую сгенерирует Spring.

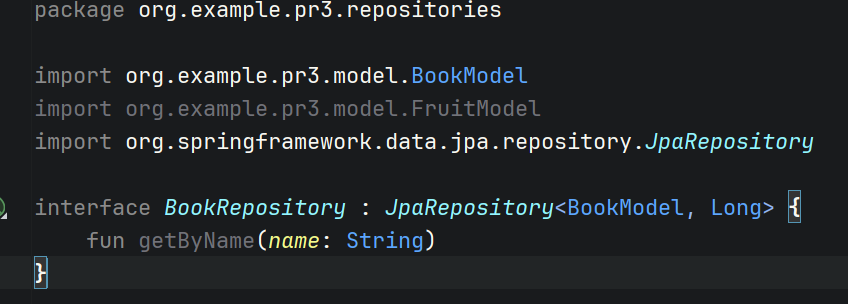


Рисунок 25 – BookRepository

1. Страница для поиска, через форму отправляется запрос.

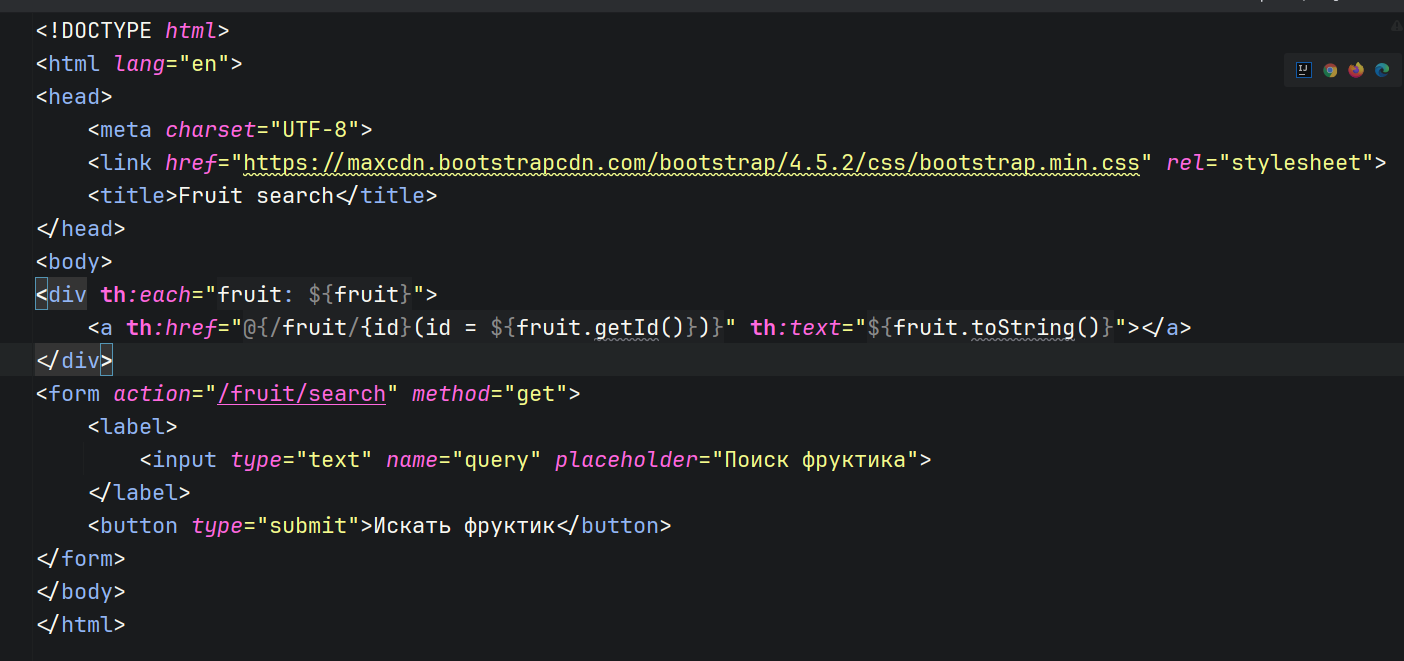


Рисунок 26 – Поиск

1. Поиск с вызовом getByName, который вернёт по имени сущность.



Рисунок 27 – Реализация поиска

Вывод: В ходе практической работы был разработаны сущности через JPA & Validator, созданы контроллеры для модификации их в БД, подключена база данных, и разработан простейший поиск.

# ПРАКТИЧЕСКАЯ РАБОТА №4

## Связи

Цель работы: добавление и настройка различных типов связей между сущностями в модели данных, с целью определения и организации полной структуры базы данных.

1. OneToOne – один к одному, @ManyToOne – многие к одному, @ManyToMany – МКМ. @JoinColumn значит что по айди сущности будет происходить заполнени этого поля. Каскад означает какие поля будут добавляться, изменятся удаляться.



Рисунок 28 – Аннотации связей

1. Для связи МКМ, joinColumns – Указывает колонку в промежуточной таблице, которая будет хранить идентификаторы сущности. inverseJoinColumns - Указывает колонку в промежуточной таблице, которая будет хранить идентификаторы сущности

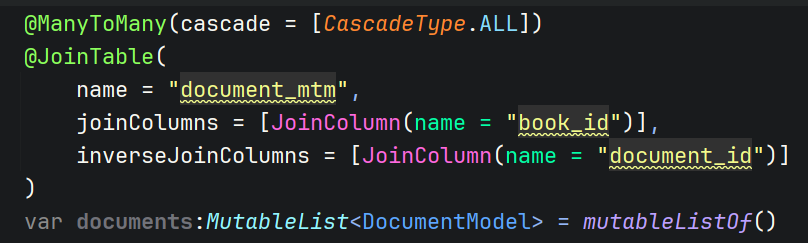


Рисунок 29 – МКМ

1. В книгах ManyToOne writer будет иметь множество книг в сущности PersonModel, где это определяется в виде списка, а в книге в виде одной сущности.

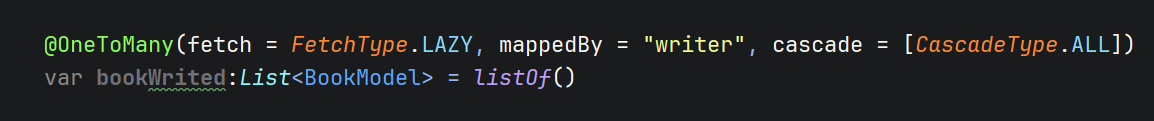


Рисунок 30 – OneToMany и ManyToOne

1. В DataGrip построим диаграмму для визуализации связей.

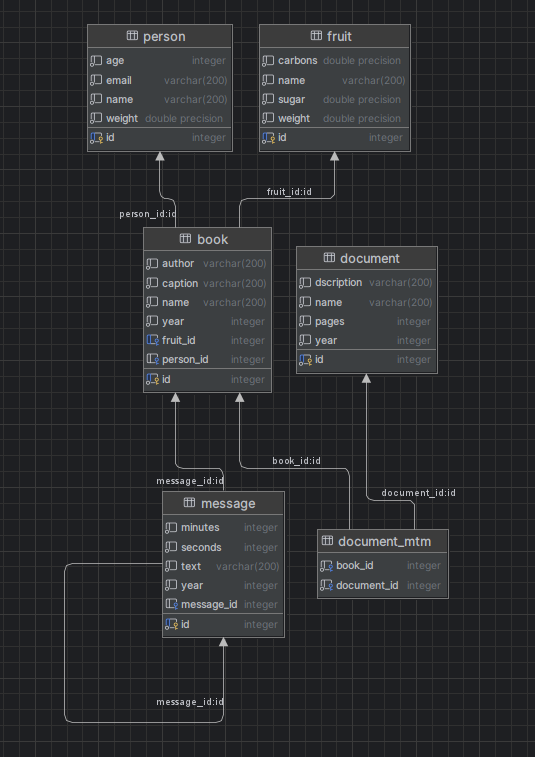


Рисунок 31 – Результат работы

Вывод: В ходе практической работы в модели сущностей были добавлены различные связи, для определения структуры всей БД.

# ПРАКТИЧЕСКАЯ РАБОТА №5

## Авторизация и регистрация

Цель работы: Добавить регистрацию и авторизацию в свой проект.

1. Подключение спринг security, по умолчанию он заблокирует все пути к эндпоинтам.



Рисунок 32 – Подключение Spring Security

1. Создаём модель юзера, и наследуем от UserDetails интерфейса чтобы указать что это авторизируемый класс.



Рисунок 33 – Модель юзера

1. Конфигурация доступов ко всем путям



Рисунок 34 Пути

1. Добавляем репозиторий User

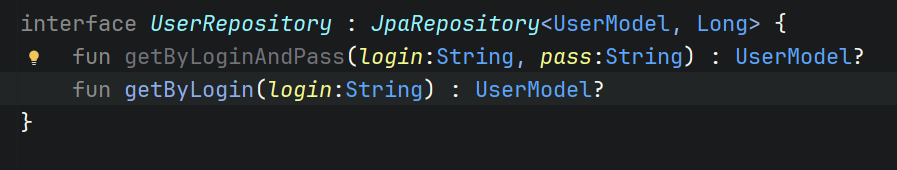


Рисунок 35 – UserRepository

1. Создание контроллера для регистрации

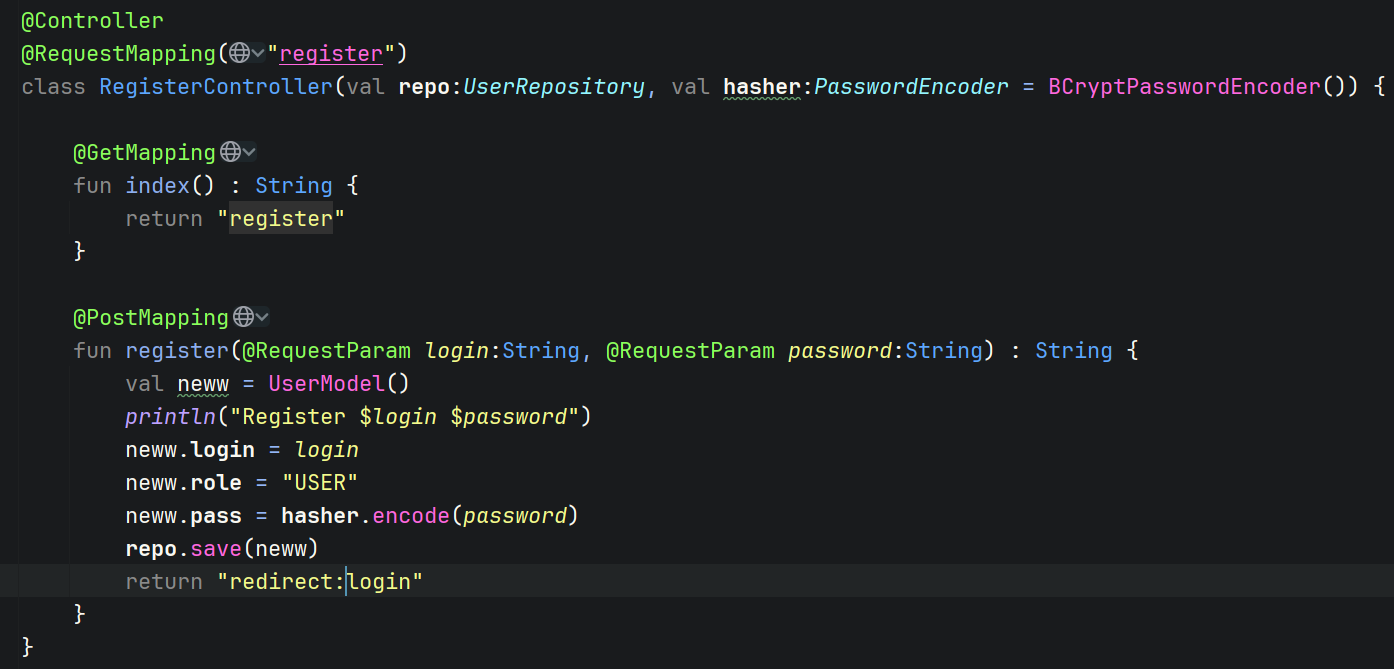


Рисунок 36 – Регистрация

1. Определение шаблона страницы регистрации

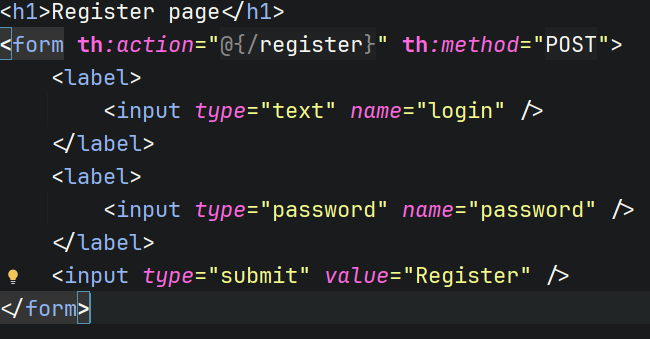


Рисунок 37 – Шаблон регистрации

1. Добавление кнопки logout



Рисунок 38 – Logout

1. Регистрируемся через /register



Рисунок 39 – Register Page

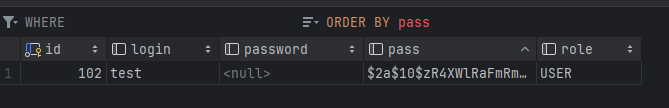


Рисунок 40 – User в БД

1. После успешной регистрации, мы попадаем уже на авторизованную страницу. Если заходить сюда без авторизации то перекинет на страницу логина.

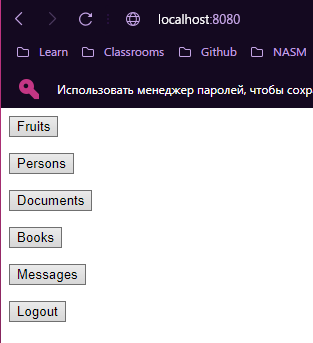


Рисунок 41 – Главная

1. Тем самым у нас теперь есть доступ ко всему почти.

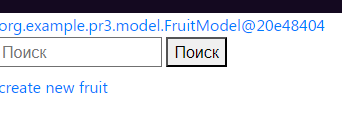


Рисунок 42 – Авторизованный

1. Если зайти в msg, доступ запрещён

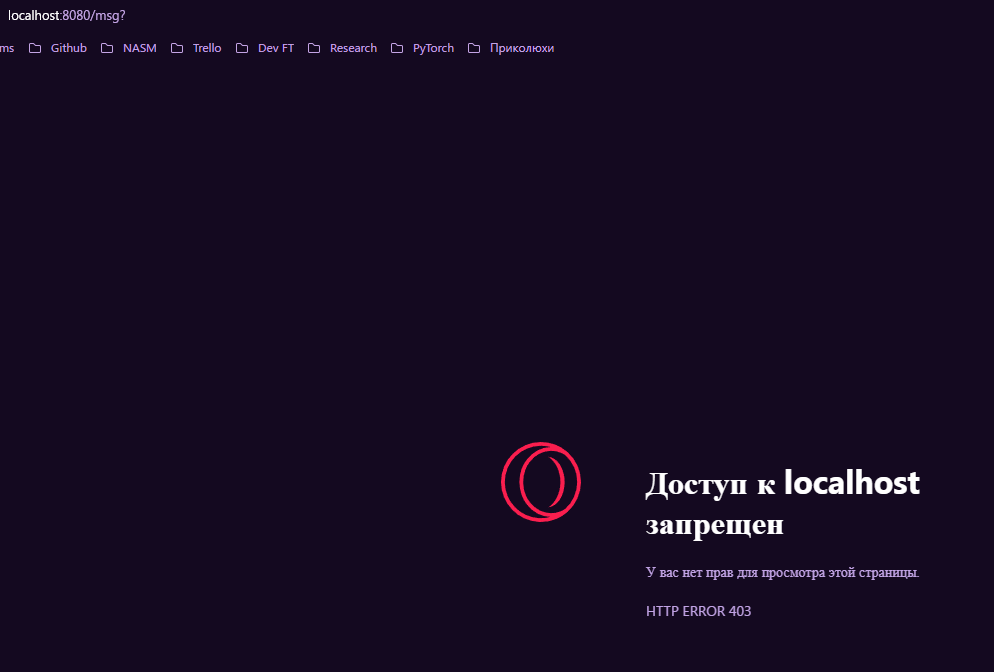


Рисунок 43 - /msg

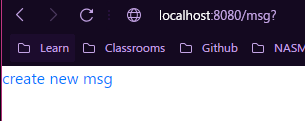


Рисунок 44 - /msg с ролью ADMIN

1. Изменяем роль на юзера, и у нас снова нет доступа к ADMIN.



Вывод: В ходе практической работы была разработана авторизация и регистрация, а так же вход выход, и роли.

# ПРАКТИЧЕСКАЯ РАБОТА №6

## Разделение прав доступа

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

1. Механизм шифрования, везде где будет @Autowired с PasswordEncoder – будет браться этот объект в методе. BCryptPasswordEncoder – хэширует пароли.

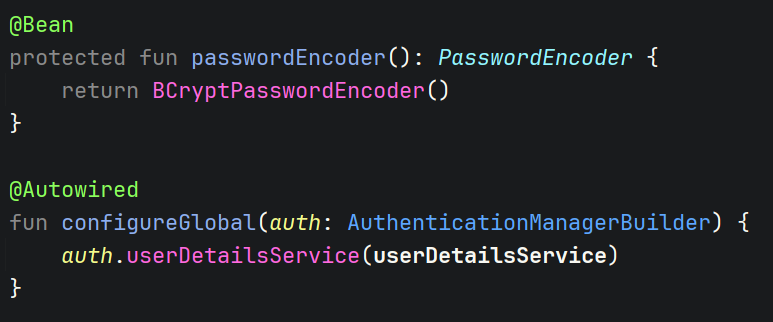


Рисунок 45 – Bean Шифрования

1. Реализация юзер дитейлс, для того чтобы определить юзеров в авторизации. Добавляем новую роль ADMIN



Рисунок 46 – Авторизация

1. Шифрование на пароль в юзере.

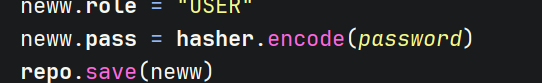
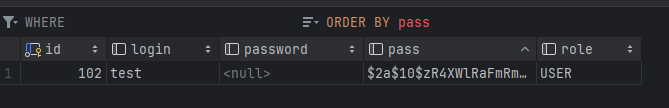


Рисунок 47 – Регистрация с шифрованием

1. В БД строка с паролём зашифрована.



1. Задействуем роль в путях.

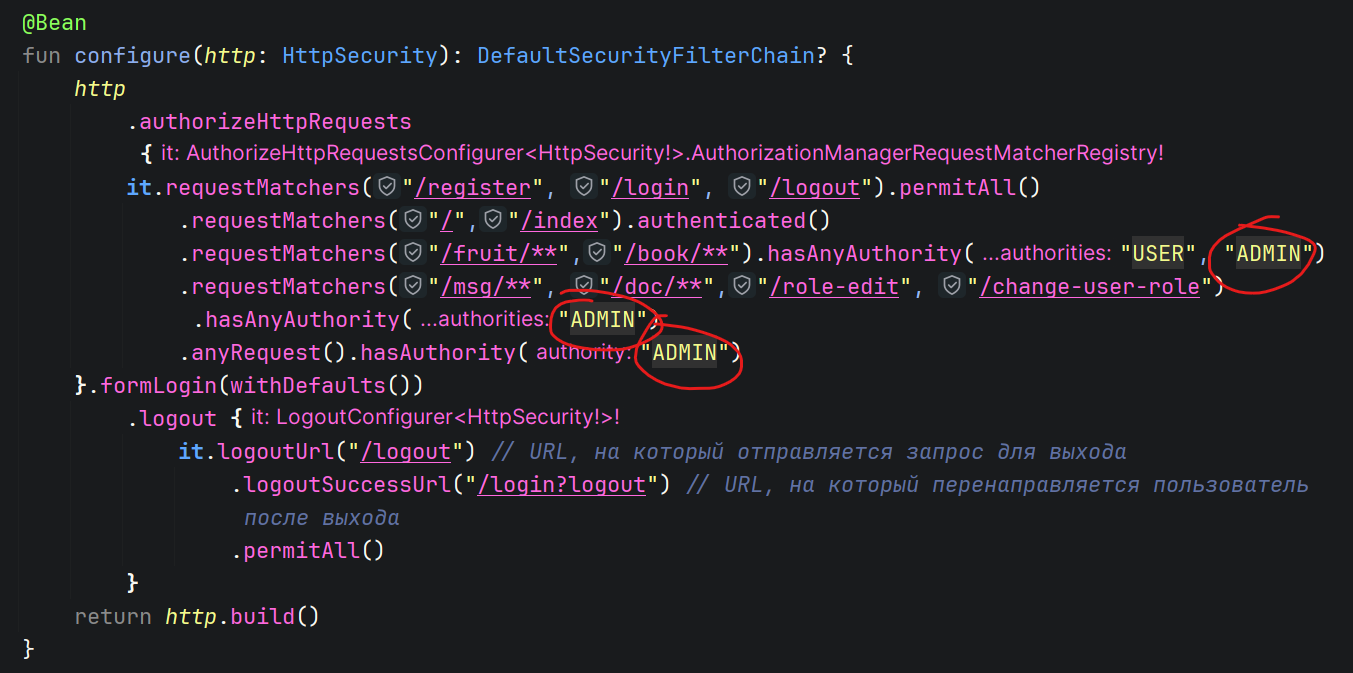


Рисунок 48 – Новая роль и права

1. Создаём контроллер, где будет гет и пост для изменений.



Рисунок 49 – Get

1. Смена роли, получаем юзер из БД, меняем роль, и сохраняем.

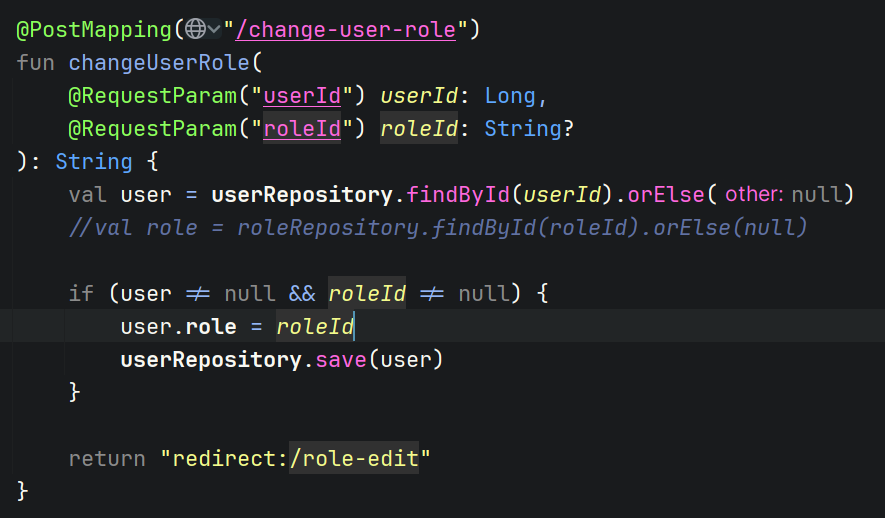


Рисунок 50 – Смена роли

1. При входе с USER на /msg, выдает запрет доступа.

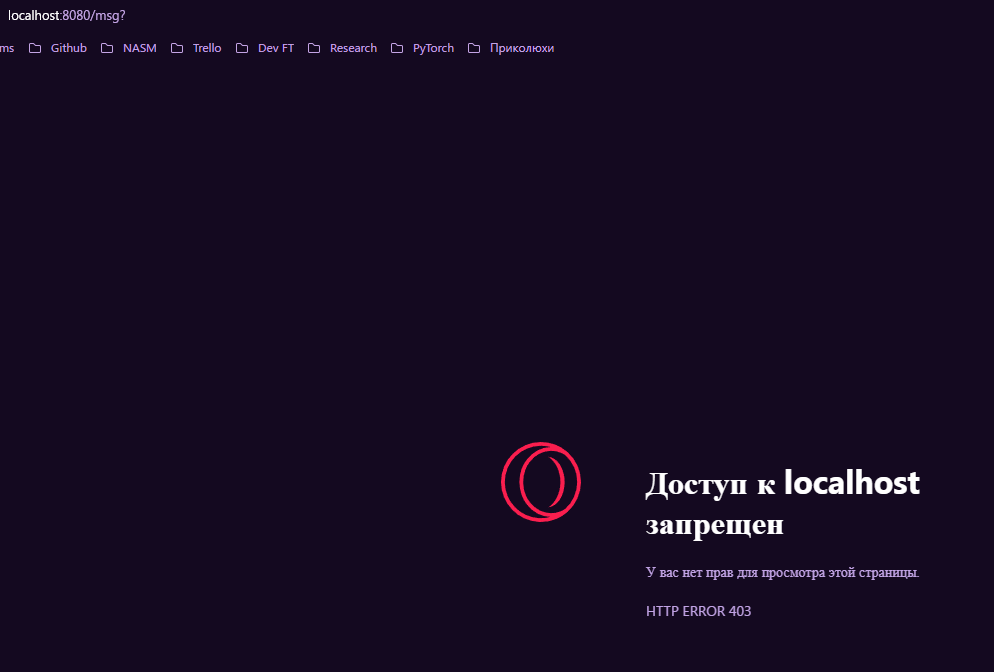


Рисунок 51 – Доступ запрещён (USER)

1. При входе с ADMIN (изменения из БД), доступ есть.

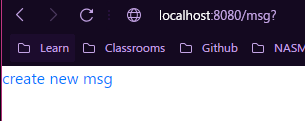


Рисунок 52 – Доступ разрешён (ADMIN)

1. При изменении роли снова на USER, доступ снова запретиться.



Рисунок 53 – Страница изменения роли

Вывод: В ходе практической работы был разработана новая роль, шифрование пароля, и разделение доступа по ролям.