|  |
| --- |
| **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.02.01 Разработка и администрирование баз данных .

Специальность 09.02.07\_Информационные системы и программирование .

Студентка Соломатина Екатерина Сергеевна.

*(фамилия, имя, отчество)*

Группа П50-3-20

Руководитель по практической подготовке от техникума

Пахомов Даниил Александрович.

*(фамилия, имя, отчество)*

«\_\_» \_\_\_\_\_\_ 2023 года

Оглавление

[Практическая работа №1 3](#_Toc144990886)

[Практическая работа №2 7](#_Toc144990887)

# Практическая работа №1

Тема «Создание калькулятора и конвертора валют».

Цель работы: Создать веб-приложение с тремя страницами (Главная, Калькулятор, Конвертер валют), где пользователь может переходить между страницами, выполнять базовые вычисления с переходом на страницу с ответом при использовании Калькулятора и конвертировать валюты на странице Конвертера с использованием аннотаций @GetMapping, @PostMapping и параметров @RequestParam.

Ход работы:

1. Для начала была создана разметка для страницы «Конвертер валют»:

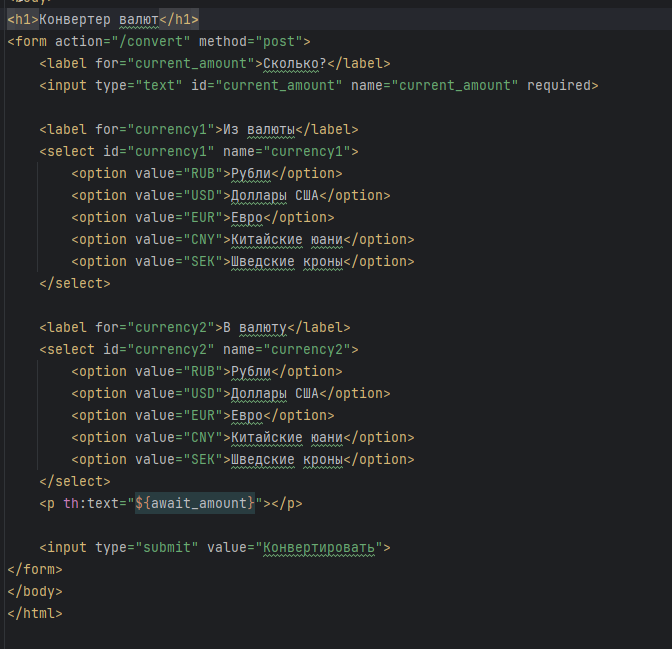


Рисунок 1 – разметка конвертера

1. Далее добавлены стили:

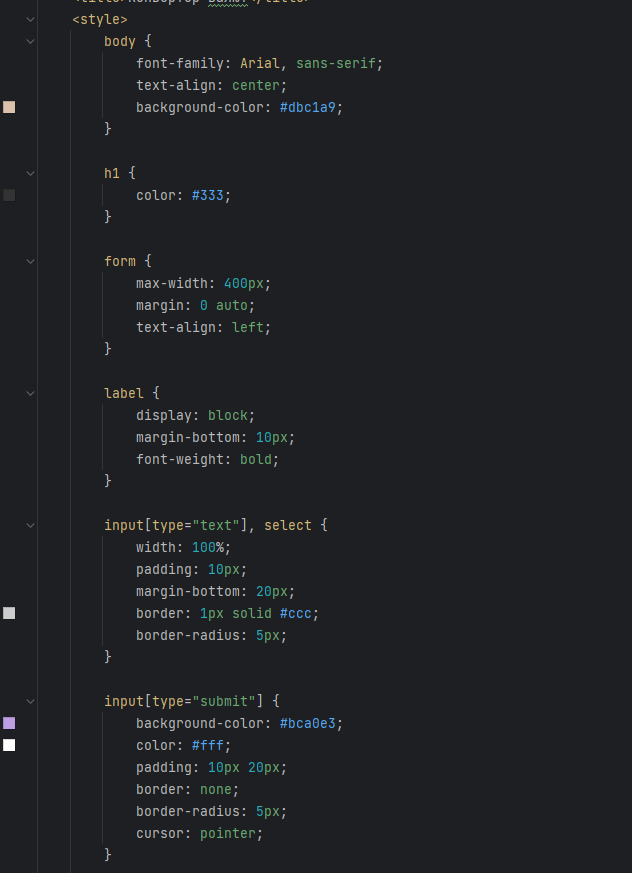


Рисунок 2 - стили для конвертера

1. Для реализации работы конвертера был создан класс контроллера с аннотацией @Controller, а в нём метод, который принимает выбранные значения в выпадающих списках и в зависимости от них считает валюту по курсу:



Рисунок 3 - метод конвертации валют

1. Для передачи значений в предыдущий метод была использована аннотация @RequestParam. Для получения значения конвертации использована аннотация @PostMapping:

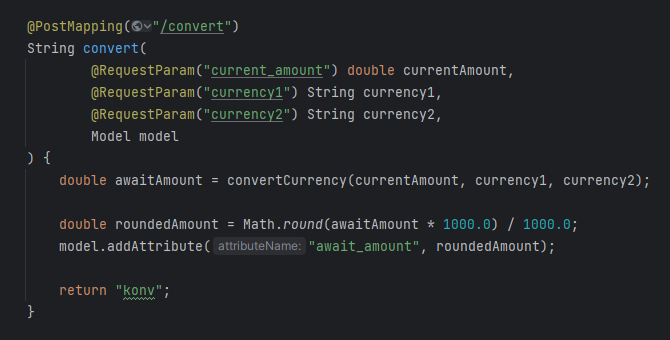


Рисунок 4 – POST-метод конвертации

1. И чтобы отобразить страницу, используем метод с аннотацией @GetMapping:

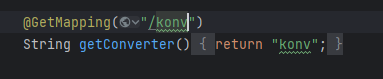


Рисунок 5 - получение страницы конвертера

# Практическая работа №2

Тема «Создание моделей и DAO»

Цель работы: разработать несколько моделей данных, создать соответствующие объекты доступа к данным (DAO), реализовать методы для выполнения операций добавления, изменения и удаления (CRUD) данных в этих моделях, а также разработать контроллеры для взаимодействия с пользовательскими интерфейсами.

Ход работы:

* Начнем с создания нескольких моделей, оборудованных атрибутами, конструкторами, методами для получения и установки значений полей.

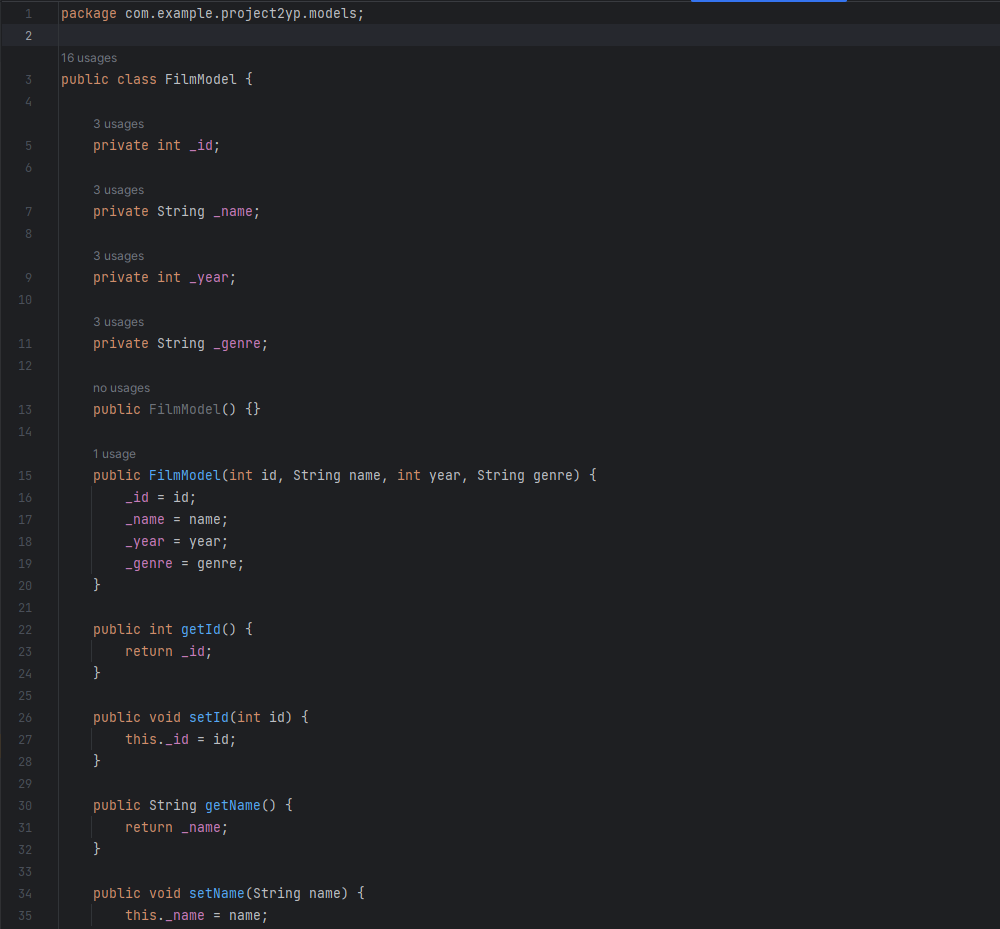


Рисунок 6 - пример модели

* Для оптимизации процесса кодирования DAO, пропишем базовый интерфейс, что позволит существенно ускорить написание однотипного кода.

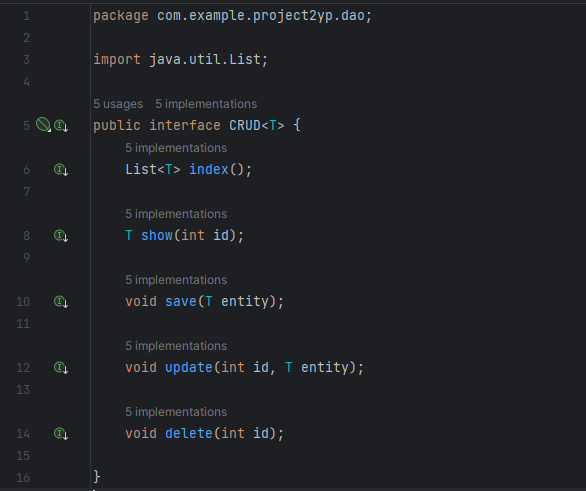


Рисунок 7 - CRUD интерфейс

* Следующим этапом будет создание объекта доступа к данным DAO. Этот компонент будет содержать методы для взаимодействия с базой данных.

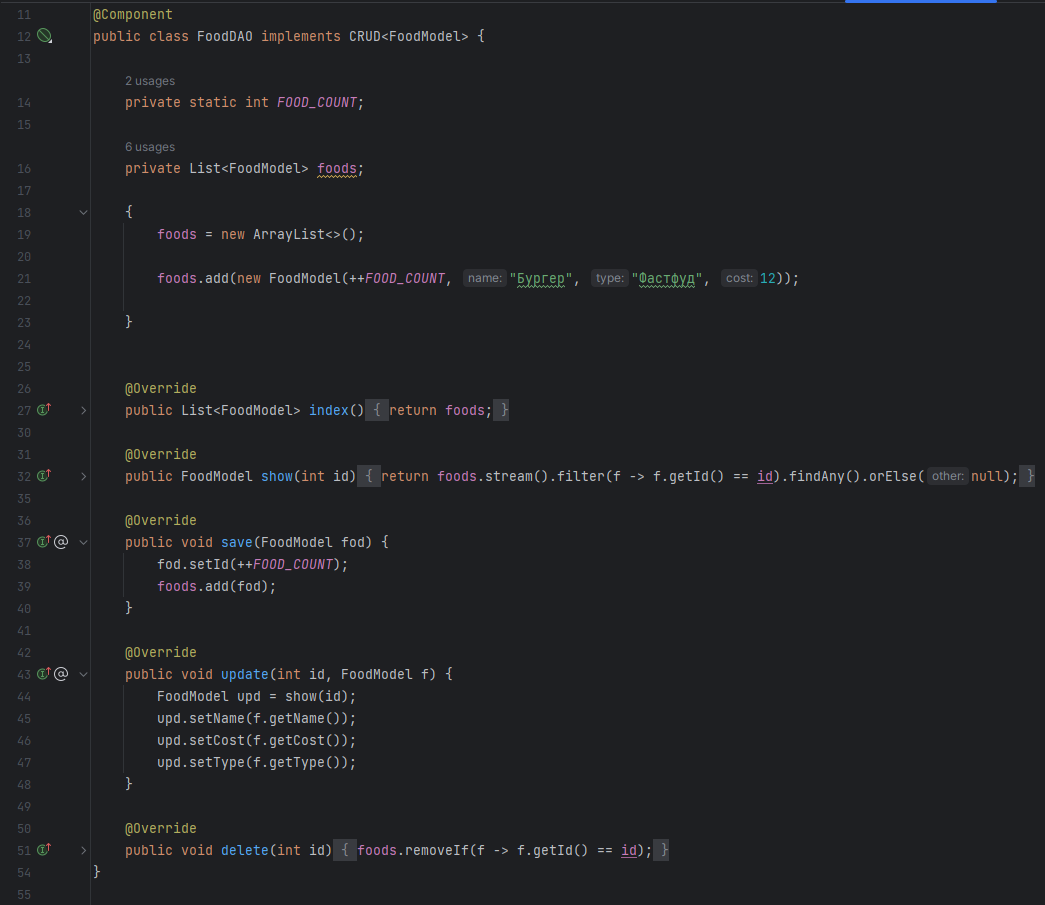


Рисунок 8 - пример DAO

* Создаем класс контроллера, помечая его аннотацией из предыдущей практической. Также воспользуемся следующими аннотациями:

@RequestMapping - аннотация, которую используют в Spring для указания, какой метод контроллера должен обрабатывать HTTP-запросы. Она позволяет определить путь и тип запроса (GET, POST и т. д.).

@PatchMapping - аннотация, которая специализирована для обработки HTTP-запросов типа PATCH. Этот тип запроса используется для частичного обновления данных на сервере, когда нужно изменить только часть информации.

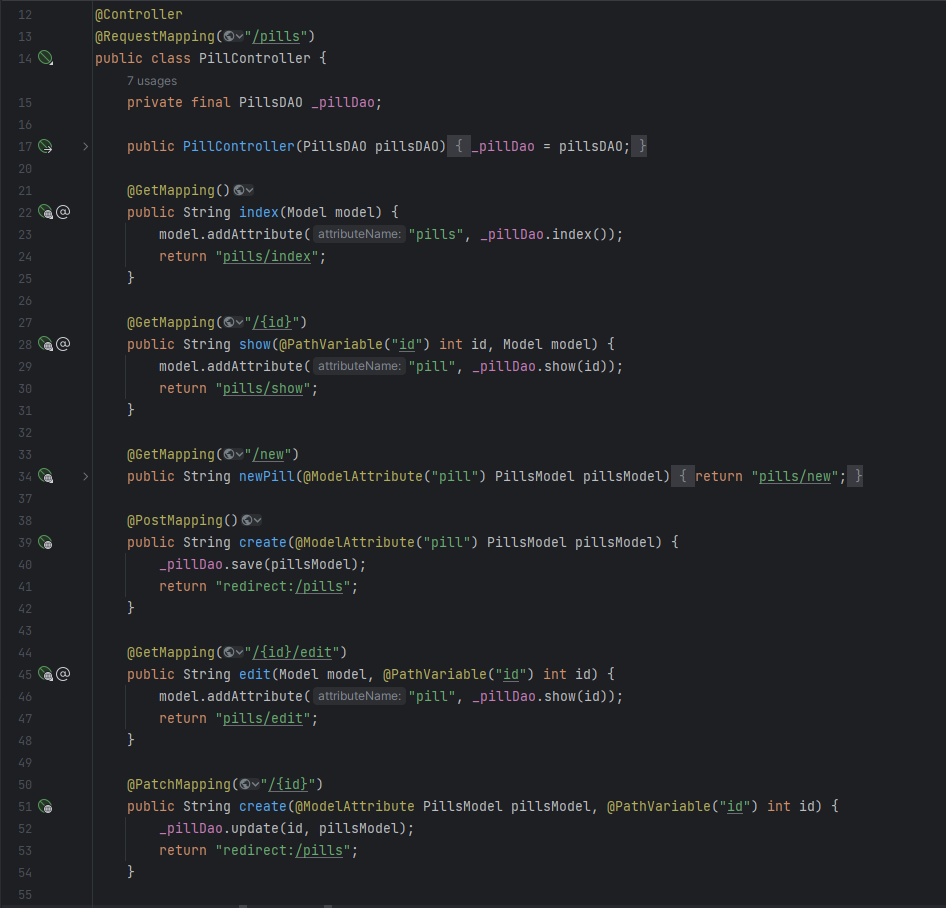


Рисунок 9 - пример контроллера

* Также в этом контроллере присутствует аннотация @DeleteMapping - аннотация в Spring, которая используется для обработки HTTP-запросов методом DELETE. Она позволяет указать, что метод контроллера должен выполнять операцию удаления, то есть удалять данные или ресурсы на сервере. Еще мы используем @PathVariable - аннотацию, которая используется в Spring для извлечения значения из переменной части URI. То есть, мы можем извлечь id с помощью пути и использовать его в методе контроллера.

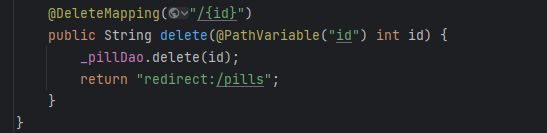


Рисунок 10 - аннотация @DeleteMapping и @PathVariable

* Теперь можем прописать верстку нашего сайта. На скриншоте показано использование атрибута href ( используется для определения URL-адреса, на который будет перенаправлен пользователь при щелчке на элемент), также атрибутов th:each (обычно применяется для создания повторяющихся элементов на странице для каждого элемента в коллекции), th:href (в отличие от обычного href, th:href позволяет использовать выражения Thymeleaf для динамической генерации URL-адреса), th:text (используется для определения текстового содержания элемента HTML. Он позволяет вставлять динамические значения или переменные в текст элемента.)

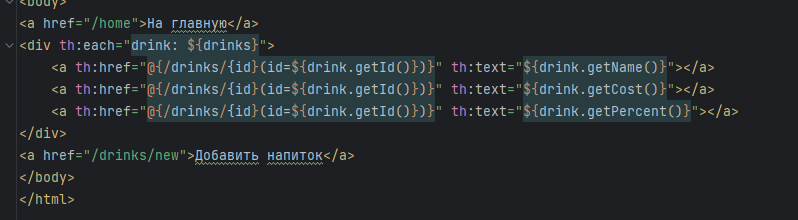


Рисунок 11 - основная часть верстки

* Далее разберем атрибут th:action, который прописан в show.html. Данный атрибут определяет целевой URL-адрес, на который будет отправлена форма при её отправке. Этот атрибут обычно применяется к элементам <form> в HTML.

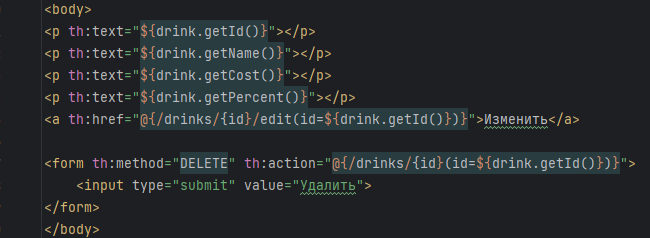


Рисунок 12 - пример атрибута th:action

* Затем, разберём атрибуты th:object и th:field. Первый позволяет обратиться к конкретной модели, чтобы впоследствии привязать поля этой модели к input на нашей страницы при помощи th:field:

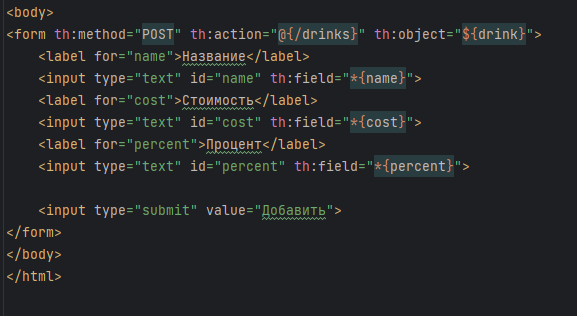


Рисунок 13 - пример атрибутов th:object и th:field

* На странице редактирования используем всё те же атрибуты:

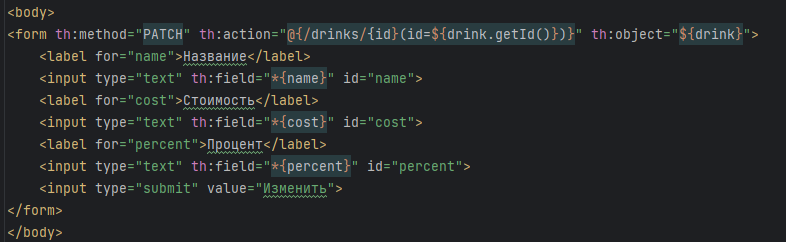


Рисунок 14 - пример использования атрибутов с методом Patch

* Для навигации по сайту был создан HomeController:

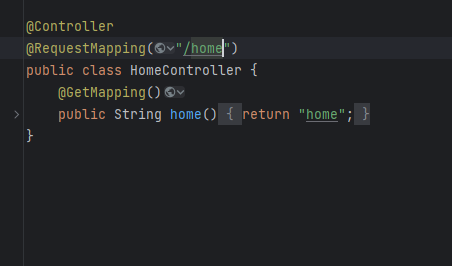


Рисунок 15 - логика HomeController

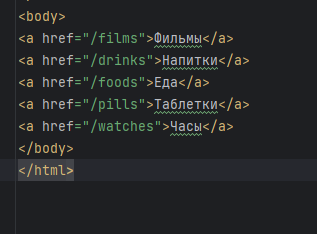


Рисунок 16 - вёрстка для HomeController

Результат:

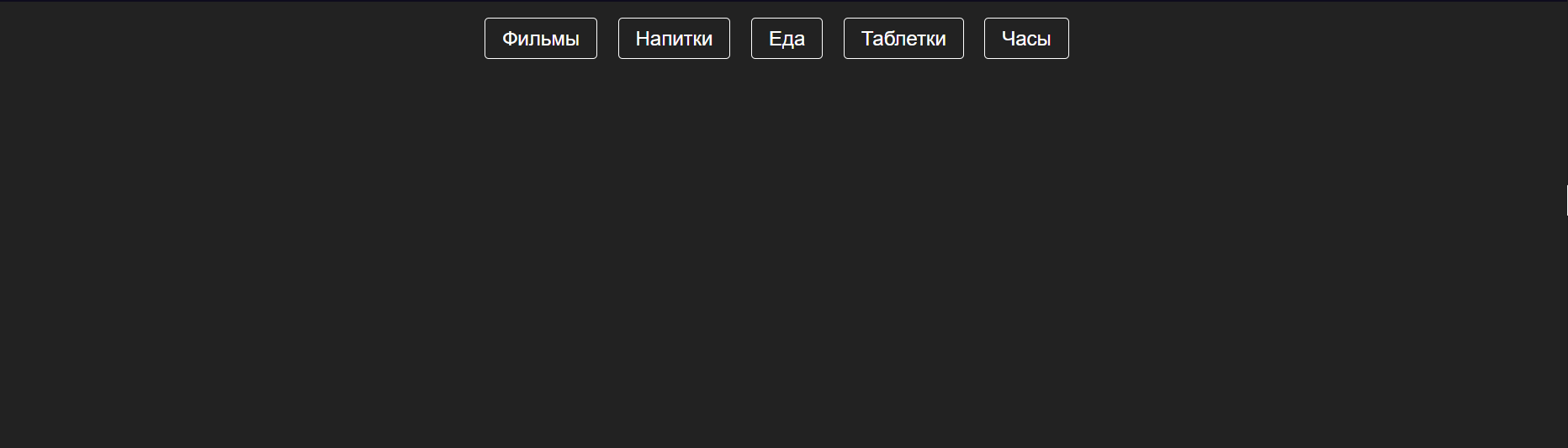


Рисунок 17 - home

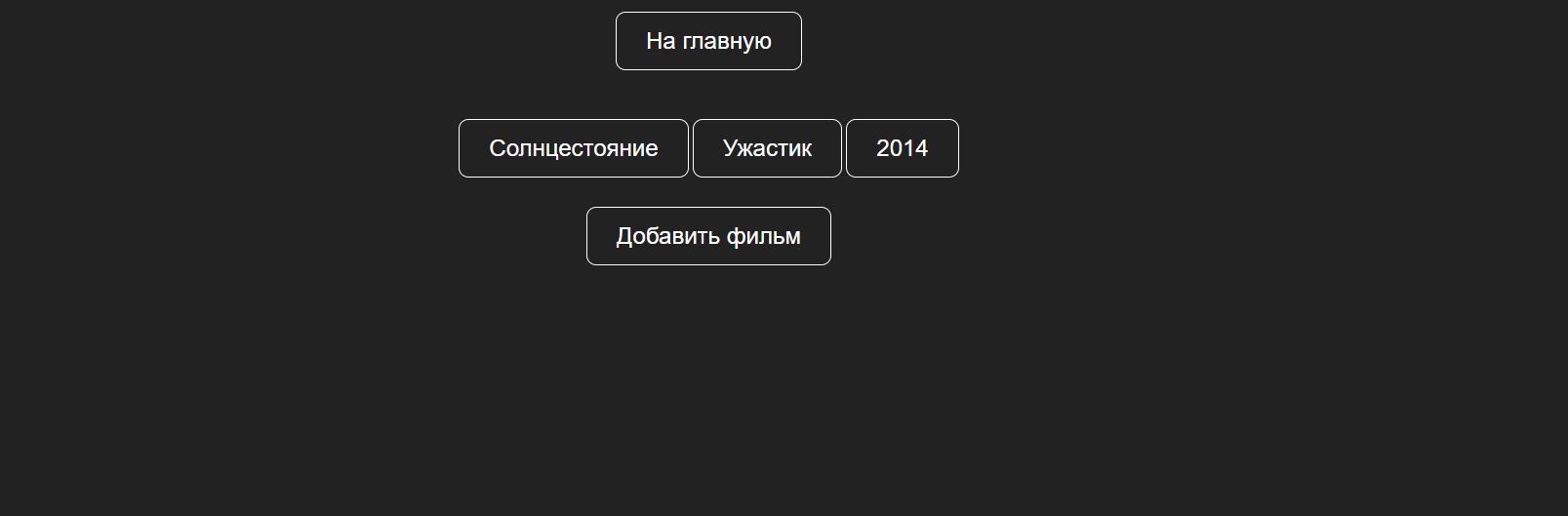


Рисунок 18 - films

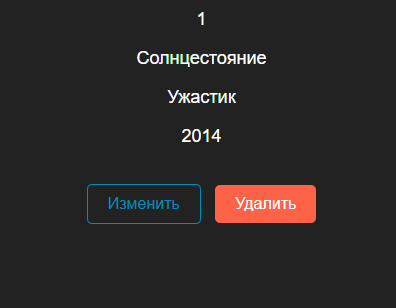


Рисунок 19 - show

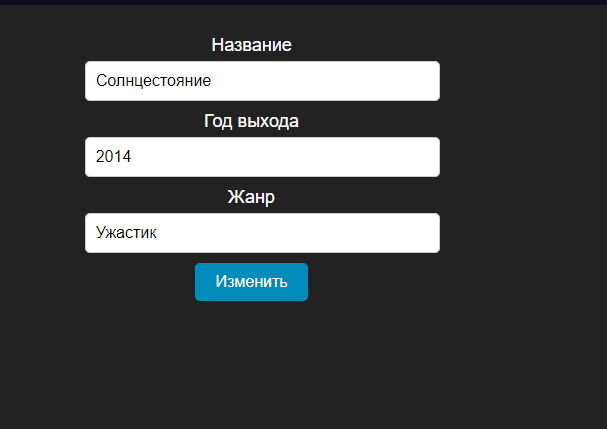


Рисунок 20 – edit

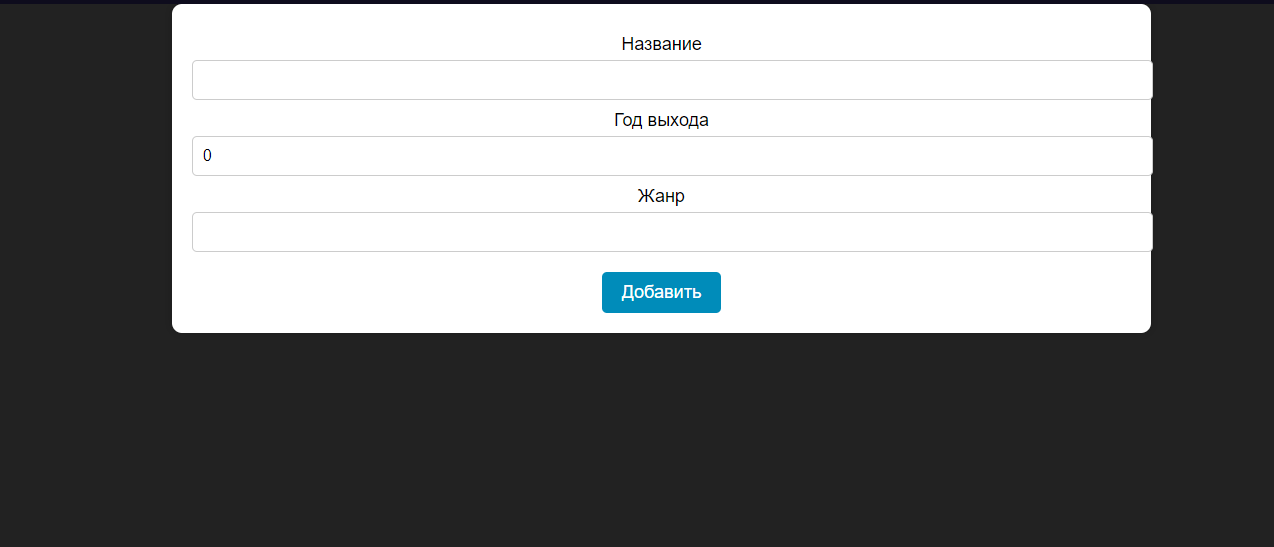


Рисунок 21 – new

Вывод: разработаны несколько моделей данных, созданы соответствующие объекты доступа к данным (DAO), реализованы методы для выполнения операций добавления, изменения и удаления (CRUD) данных в этих моделях, а также разработаны контроллеры для взаимодействия с пользовательскими интерфейсами.