|  |
| --- |
| **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 .

Студент Некрасов Константин Александрович.

*(фамилия, имя, отчество)*

Группа П50-3-20

Руководитель по практической подготовке от техникума

Пахомов Даниил Александрович.

*(фамилия, имя, отчество)*

«\_\_» \_\_\_\_\_\_ 2023 года

# Практическая работа №2

Тема «Создание моделей и DAO».

Цель работы: создать несколько моделей данных и для них Dao, а также добавить контроллеры для взаимодействия с представлениями.

Ход работы:

1. Для начала создадим несколько моделей с полями, конструктором, геттерами и сеттерами:

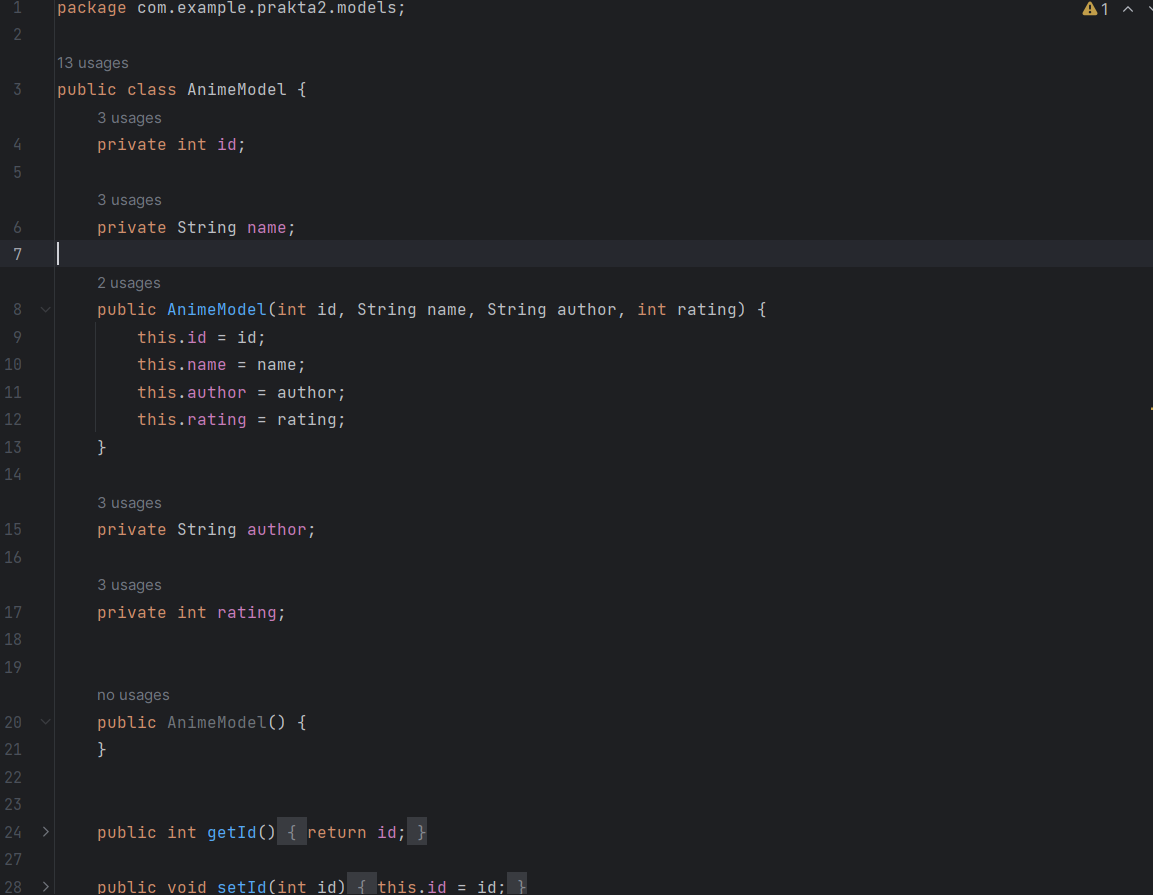


Рисунок 16. Модель

1. Далее, создадим DAO (Data Access Object). Он будет хранить в себе методы для взаимодействия с базой данных (в нашем случае с mock-базой данных). Выделив такие методы в отдельные dao, мы повысим читаемость и качество нашего кода:

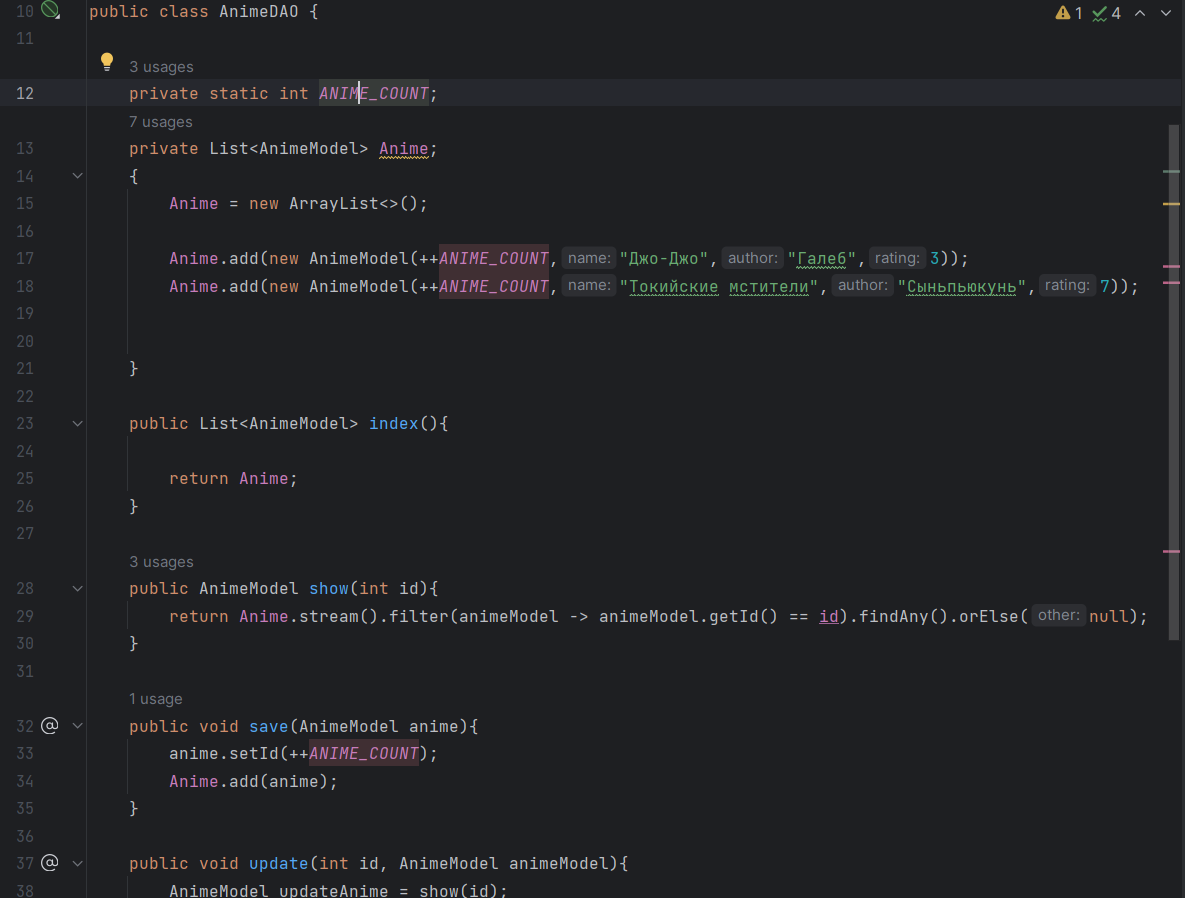


Рисунок 17. Dao

1. Затем, для каждой модели был создан контроллер. Используем аннотацию @RequestMapping для определения end-point’а нашего контроллера. Инициализируем наш DAO, и далее обращаемся к нему, вызывая нужные методы. Аннотация @PostMapping используется для выполнения POST-запросов, redirect возвращает нас на начальную страницу:



Рисунок 19. GET и POST методы в контроллере

1. Также прописываем методы для изменения и удаления, помечая их аннотациями @PatchMapping @DeleteMapping соответственно. @PathVariable используем для считывания id в строку:



Рисунок 20. PATCH и DELETE методы

1. Повторяем вышеуказанные действия для других моделей, и пишем вёрстку, для вывода списка моделей используем th:each. Он работает как цикл foreach, th:href укажет ссылку на нужный объект, th:text выведет нужный текст:

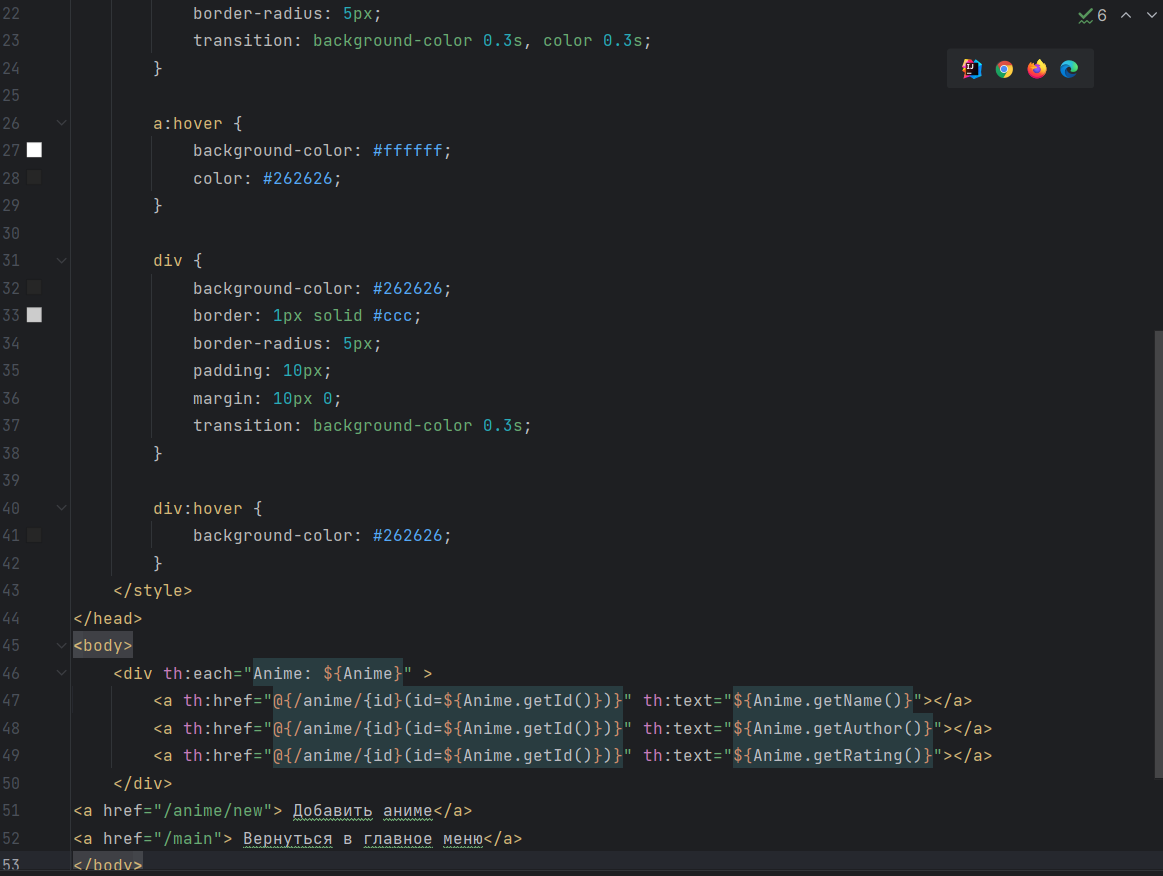


Рисунок 21. Вёрстка index.html

1. Для перехода на URL с указанием id сущности используем th:action с указанием пути до модели с нужным идентификатором, который берём прямо из сущности:

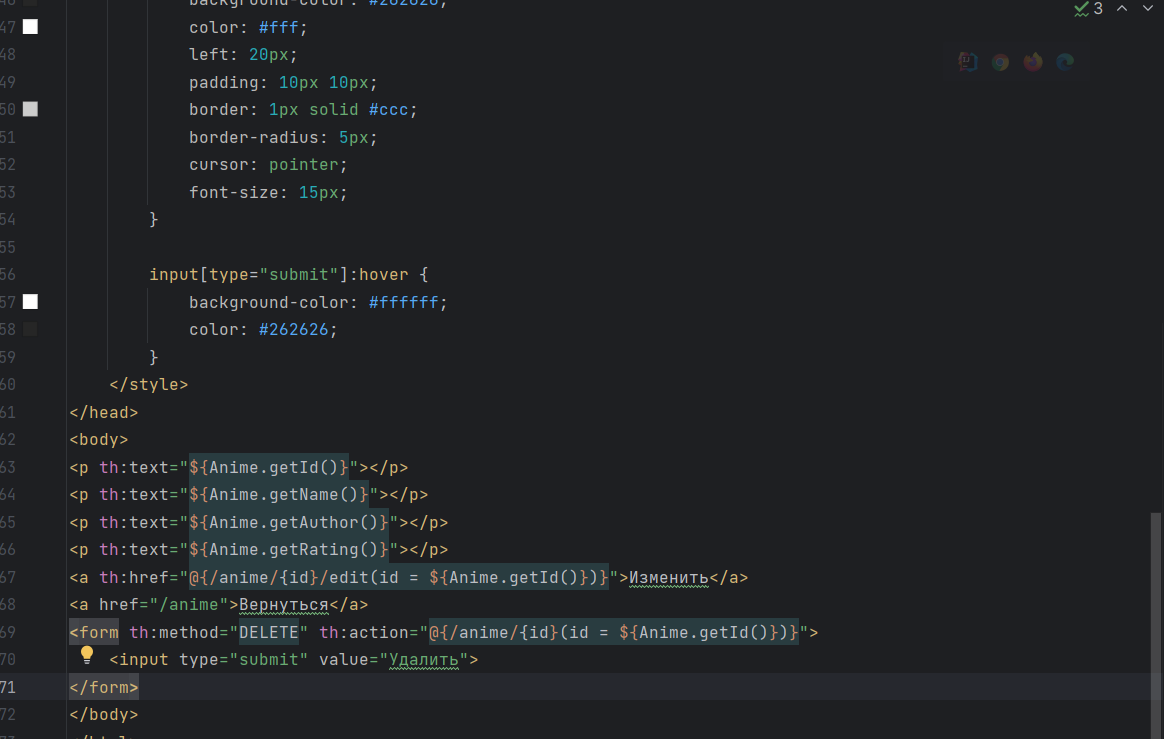


Рисунок 22. Вёрстка show.html

1. Для создания новой сущности, указываем ссылку на объект, который будем добавлять (th:object), и при помощи th:field привязываем поля для ввода к полям сущности, и добавим обязательность заполнения полей и максимальное допустимое значение в поле рейтинг:

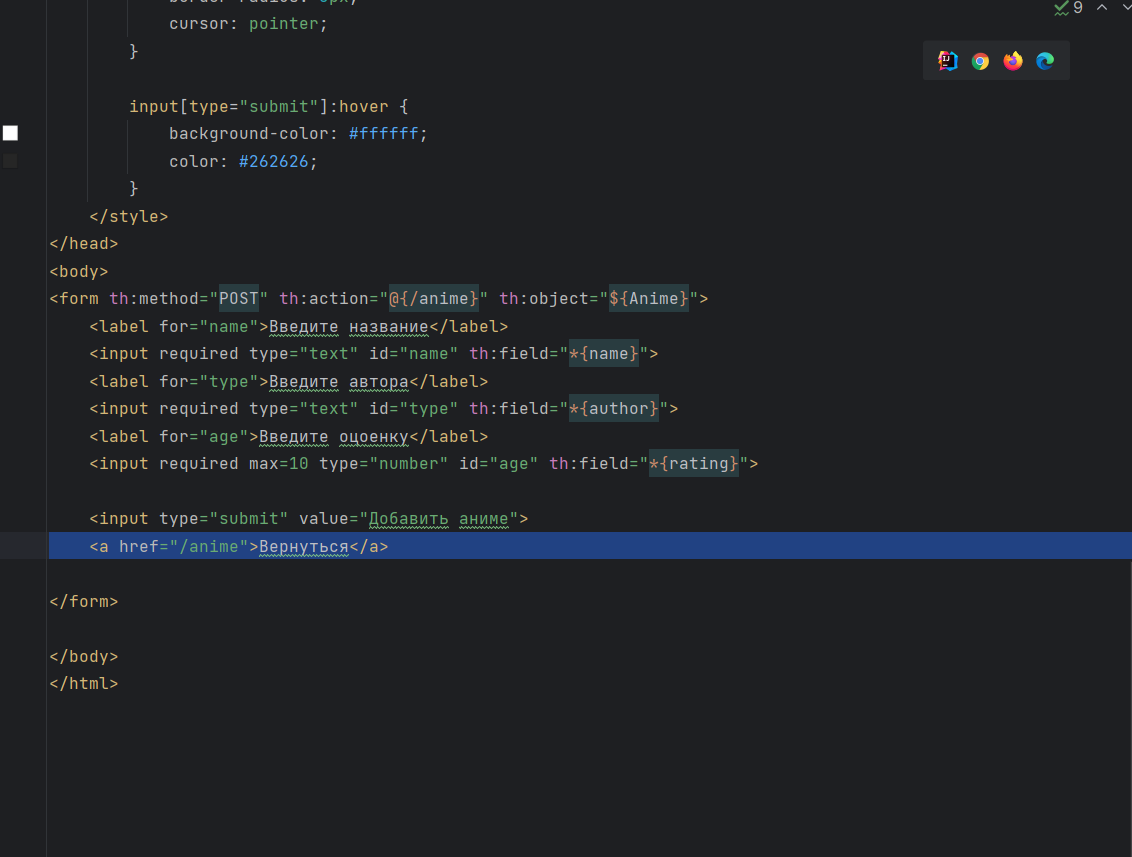


Рисунок 23. Разметка new.html

1. Для изменения сущности используем те же функции, только с методом PATCH, также добавим обязательность заполнения полей и максимальное значение для поля рейтинг:

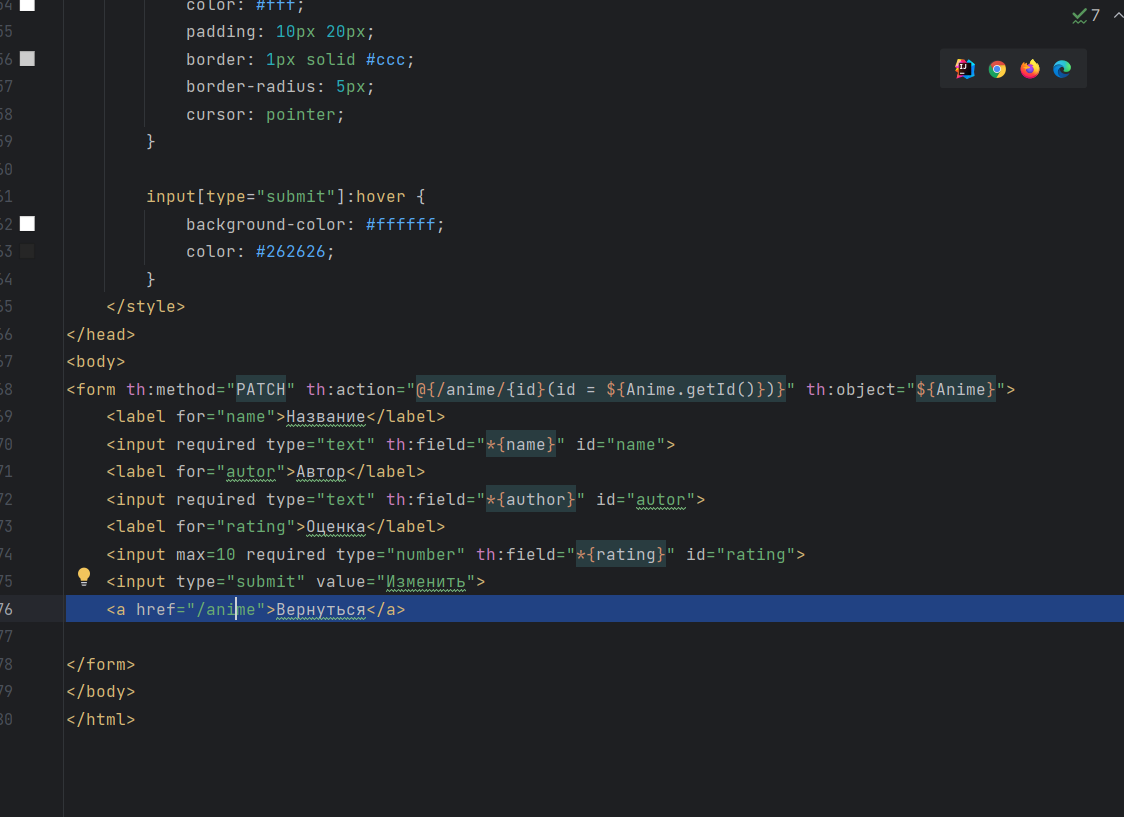


Рисунок 24. Разметка edit.html

1. Выполняем те же действия для других моделей, и создаём страницу для перехода на все списки:

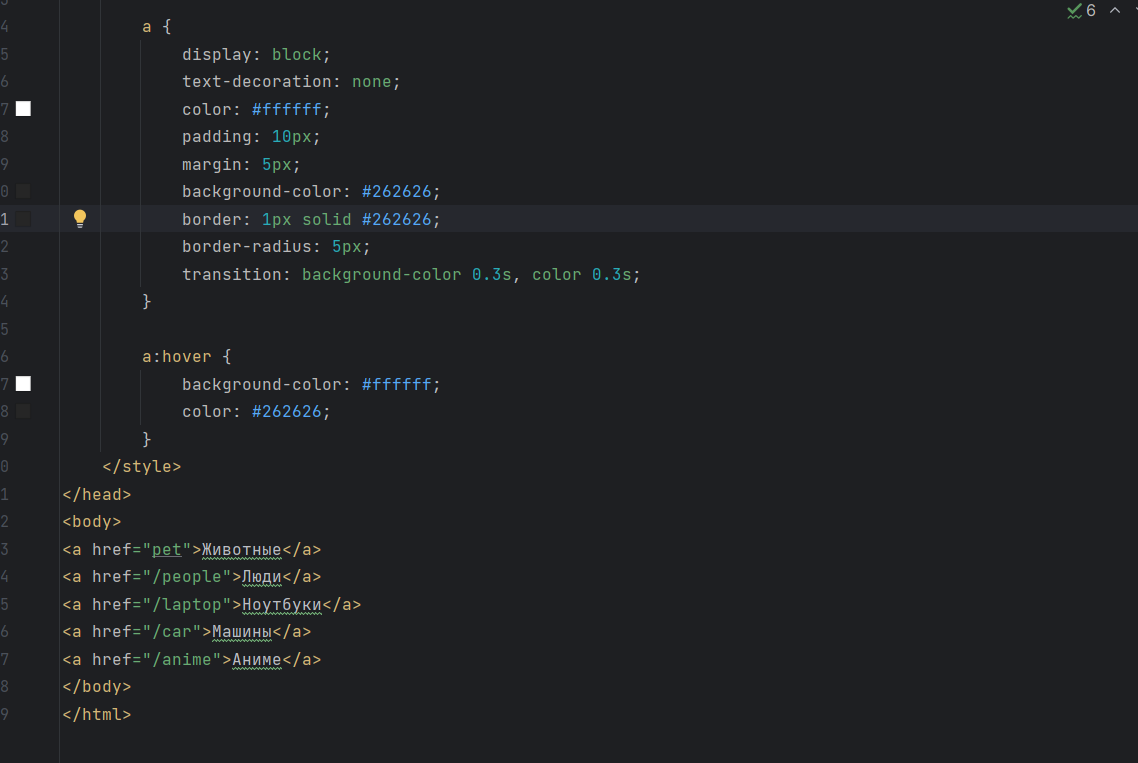


Рисунок 25. Разметка main.html

Результат:

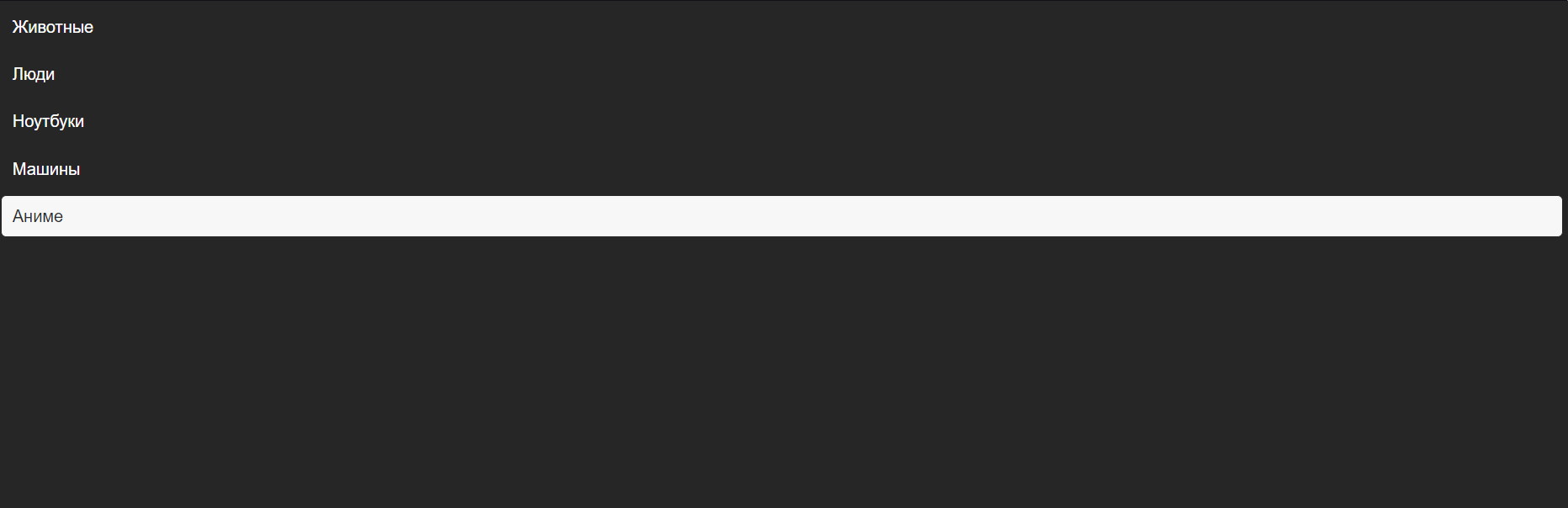


Рисунок 26. Главная страница

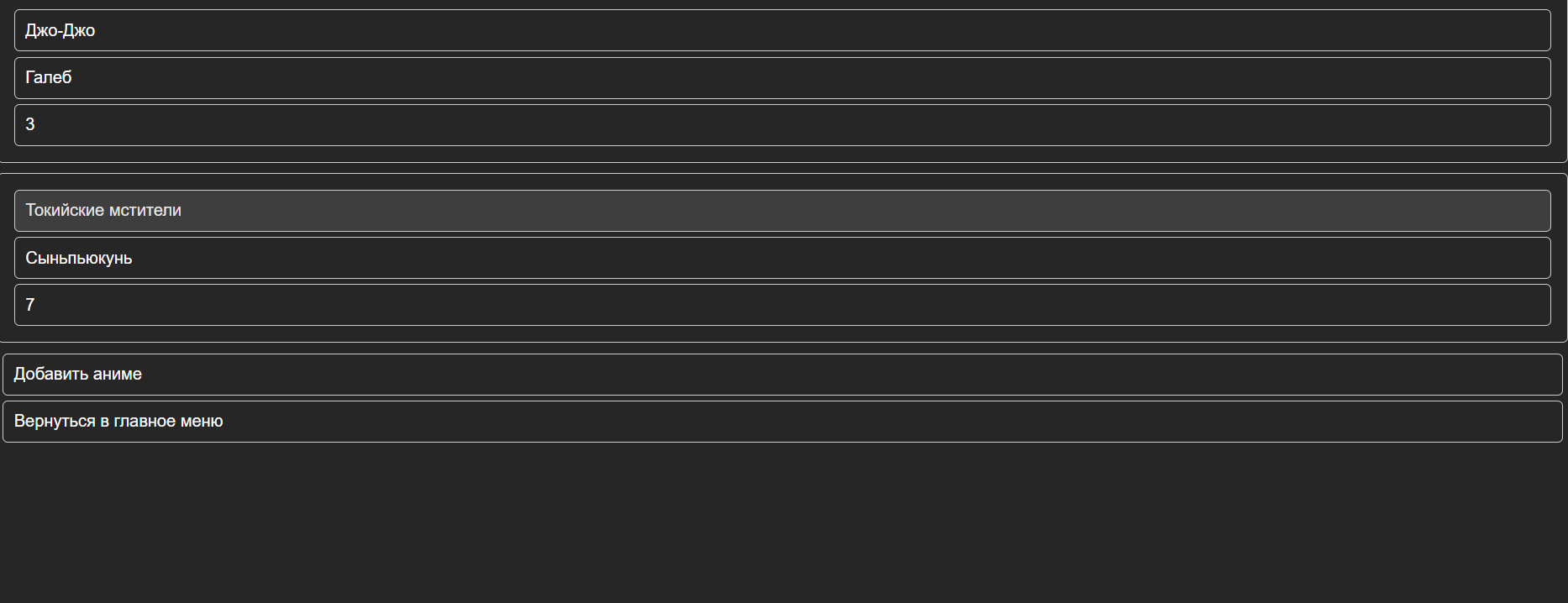


Рисунок 27. Страница списка

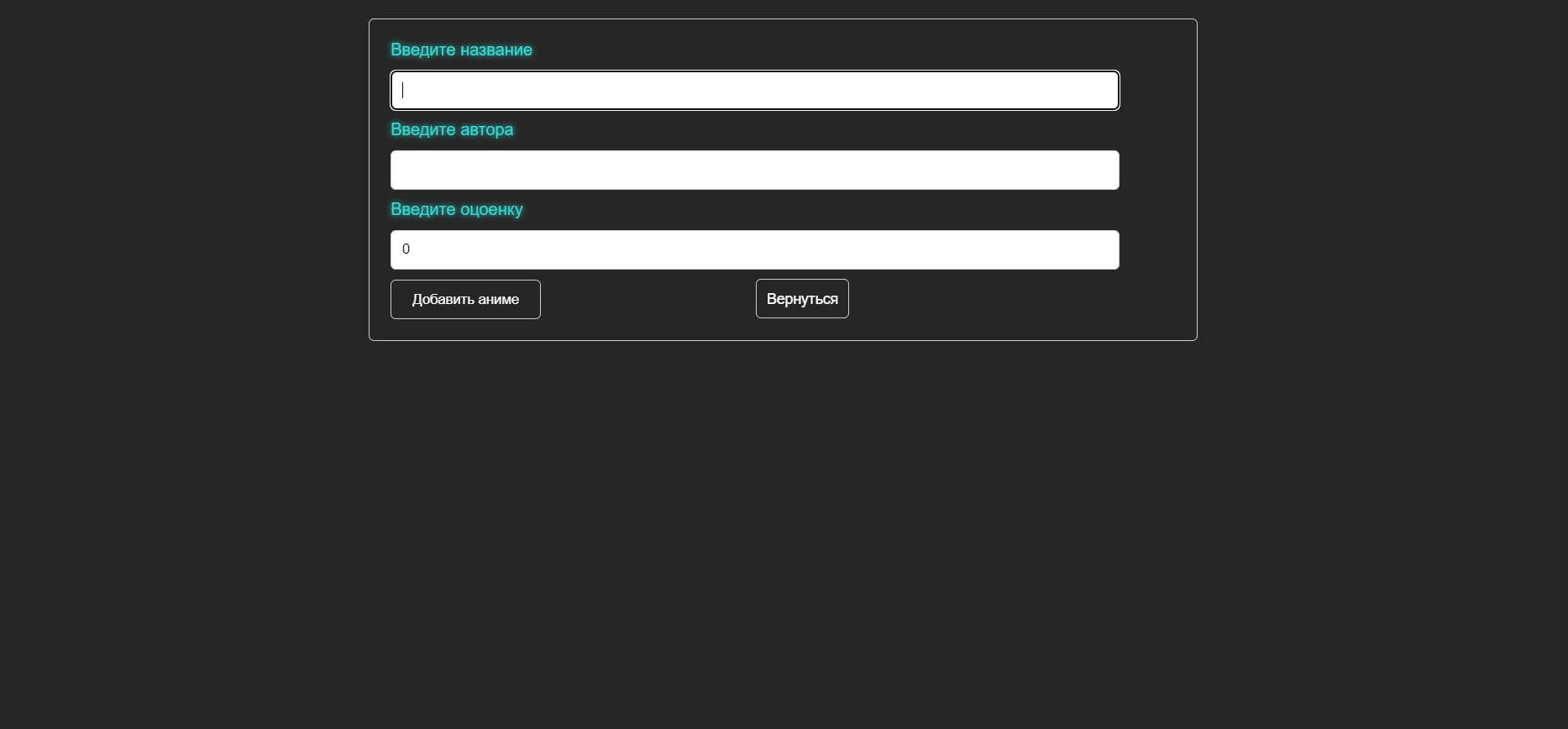


Рисунок 28. Страница создания



Рисунок 29. Страница информации

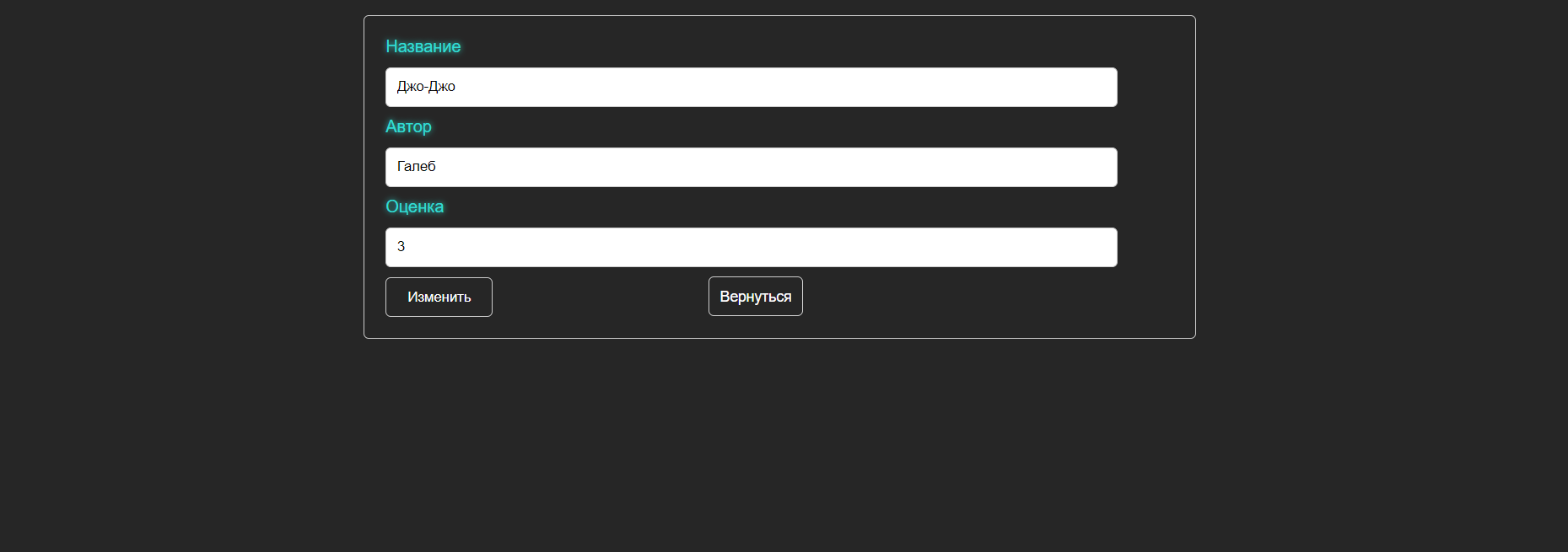


Рисунок 30. Страница изменения

Вывод: созданы несколько моделей данных и для них Dao а также добавлены контроллеры для взаимодействия с представлениями.