

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные системы управления базами
данных»
Тема: Каталог фантастики

Студенты гр. 2303

Ибрагимов Э.М.

Степанов Д.А.

Чецкий Я.А.

Репкин В.С.

Косолапов П.Е.

Заславский М.М.

Преподаватель

Санкт-Петербург

2025

ЗАДАНИЕ

Студенты: Ибрагимов Э.М., Степанов Д.А., Чецкий Я.А., Репкин В.С.,
Косолапов П.Е.

Группа 2303

Тема работы: Каталог фантастики

Исходные данные:

Задача - сделать сервис, где будет собрана информация про книги, фильмы и сериалы в жанре фантастики и фентези. Нужно построить базу, где все произведения будут тщательно разбиты по жанрам, темам, тегам, взаимосвязям. Организовать интерфейсы для рекомендации произведений по вкусам пользователя.

Содержание пояснительной записи:

“Содержание”, “Введение”, “Качественные требования к решению”, “Сценарий использования”, “Модель данных”, “Разработанное приложение”, “Заключение”, “Список использованных источников”, “Приложение”.

Предполагаемый объем пояснительной записи:

Не менее 10 страниц.

Дата выдачи задания: 11.02.2025

Дата сдачи реферата: 21.05.2025

Дата защиты реферата: 21.05.2025

Студенты гр. 2303

Ибрагимов Э.М.

Степанов Д.А.

Чецкий Я.А.

Репкин В.С.

Косолапов П.Е.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

Индивидуальное домашнее задание было посвящено разработке информационного сервиса для любителей фантастики и фэнтези. Основная цель – создание платформы с обширной базой данных книг, фильмов и сериалов, тщательно каталогизированных по жанрам, темам, тегам и взаимосвязям. Методы разработки включают создание бэкенда на Java Spring Boot для управления данными и API, а также фронтенда на TypeScript Angular для пользовательского интерфейса. Важной частью проекта является реализация гибкой системы фильтров, которая позволяет пользователям точно настраивать поиск и отбор произведений по множеству критериев. В результате получен веб-сервис, демонстрирующий основные функции каталогизации и рекомендаций, готовый к дальнейшему развитию и наполнению контентом. Проект решает задачу удобного поиска и открытия новых произведений в указанных жанрах. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql1h25-fantasy>

SUMMARY

The individual home assignment was dedicated to the development of an information service for science fiction and fantasy enthusiasts. The main goal is to create a platform with an extensive database of books, films, and series, meticulously categorized by genres, themes, tags, and interconnections. Development methods include creating a backend using Java Spring Boot for data management and API, as well as a frontend using TypeScript Angular for the user interface. An important part of the project is the implementation of a flexible filter system that allows users to precisely customize their search and selection of works based on multiple criteria. The result is a web service demonstrating the core cataloging and recommendation functionalities, ready for further development and content population. The project addresses the challenge of conveniently searching for and discovering new works in the specified genres. The source code and all additional information can be found at the following link: <https://github.com/moevm/nosql1h25-fantasy>

СОДЕРЖАНИЕ

Введение	4
1. Качественные требования к решению	8
2. Сценарий использования	9
2.1 Макет UI	9
2.2 Описание сценариев использования	9
3. Модель данных	17
3.1. Нереляционная модель	17
3.2. Реляционная модель	22
3.3. Сравнение моделей	32
4. Разработанное приложение	35
Заключение	41
Список использованных источников	43
Приложение	44

ВВЕДЕНИЕ

Жанры фантастики и фэнтези пользуются неизменной популярностью, предлагая читателям и зрителям богатые и разнообразные миры. Однако ориентироваться в огромном количестве книг, фильмов и сериалов, находить произведения, соответствующие индивидуальным вкусам, и отслеживать новинки может быть сложной задачей. Существующие платформы не всегда предлагают достаточно детализированную каталогизацию или гибкие инструменты для точного поиска, что затрудняет открытие нового контента, особенно для искушенных ценителей.

Данный курсовой проект направлен на решение этой проблемы путем разработки специализированного информационного сервиса. Основная цель проекта – создать платформу, которая соберет обширную базу данных произведений в жанрах фантастики и фэнтези. Ключевой особенностью сервиса является тщательная каталогизация контента по множеству параметров, включая жанры, поджанры, темы, теги и взаимосвязи между произведениями.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Разрабатываемый информационный сервис должен обеспечивать эффективное управление и представление каталога произведений фантастики и фэнтези. Ключевым требованием является использование MongoDB для хранения данных, что позволит гибко структурировать информацию о книгах, фильмах и сериалах с их разнообразными атрибутами (жанры, теги, связи) и обеспечит производительность при выполнении сложных запросов. Бэкенд, реализованный на Java Spring Boot, должен предоставлять четкий API для фронтенда на Angular, позволяющий оперативно получать и фильтровать данные. Фронтенд, в свою очередь, должен обеспечивать отзывчивый пользовательский интерфейс для удобного просмотра каталога и применения многоокритериальных фильтров. Система должна быть спроектирована таким образом, чтобы фильтрация по различным параметрам в MongoDB выполнялась быстро, предоставляя пользователю релевантные результаты без значительных задержек.

2. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

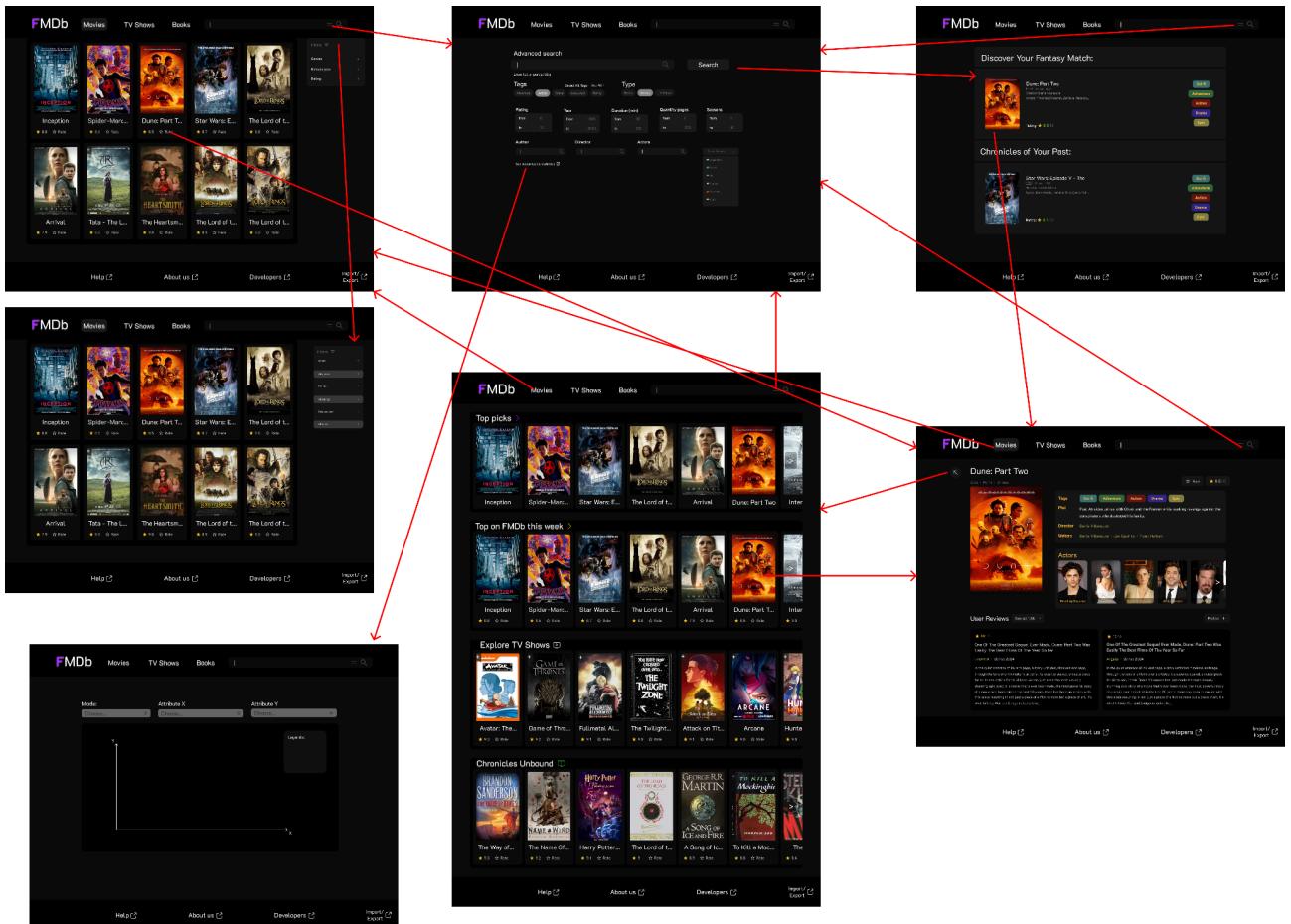


Рисунок 1 – Дизайн интерфейса.

2.2. Описание сценариев использования

- Действующее лицо: Пользователь
- Основной сценарий: Пользователь заходит на сайт с целью поиска информации о книге, фильме или сериале, относящемся к жанру фантастики или фэнтези. Пользователь взаимодействует с различными элементами интерфейса для навигации, поиска, фильтрации и получения детальной информации о произведениях.

Взаимодействие с интерфейсом на различных страницах

1) Главная страница

На главной странице по умолчанию отображается список рекомендуемых произведений в жанре фантастики.

- Действия пользователя и реакции системы:
 - Переход на вкладку "Movies": Система отображает страницу со списком всех фильмов.
 - Переход на вкладку "TV Shows": Система отображает страницу со списком всех сериалов.
 - Переход на вкладку "Books": Система отображает страницу со списком всех книг.
 - Использование поиска: Пользователь вводит текст в поле поиска. Система динамически фильтрует отображаемые на текущей странице произведения, оставляя те, в названии которых присутствует введенная подстрока.
 - Выбор произведения: Пользователь нажимает на изображение (иконку) произведения. Система открывает страницу (меню) с детальной информацией о выбранном произведении.
 - Открытие фильтров: Пользователь нажимает кнопку "Фильтры". Система отображает окно для настройки параметров фильтрации.
 - Просмотр рекомендаций: Пользователь прокручивает главную страницу для ознакомления с предложенными рекомендациями.
 - Взаимодействие с рекомендованными категориями: Пользователь прокручивает горизонтальную ленту с тематическими подборками произведений.
 - Нажатие кнопки "Help": Система отображает модальное окно с инструкцией по использованию сайта.

- Нажатие кнопки "About us": Система отображает модальное окно с описанием проекта.
- Нажатие кнопки "Developers": Система отображает модальное окно с информацией о разработчиках.
- Нажатие кнопки "Rate" у произведения: Пользователь отмечает выбранное произведение. При повторном нажатии отметка снимается.
- Нажатие кнопки "Import/Export": Система отображает модальное окно для выполнения операций импорта или экспорта данных.

2) Вкладка "Movies" (Фильмы)

На данной странице отображается список всех фильмов, доступных в системе.

- Действия пользователя и реакции системы:
 - Нажатие на заголовок сайта (FMDd): Система перенаправляет пользователя на главную страницу.
 - Переход на вкладку "TV Shows": Система отображает страницу со списком всех сериалов.
 - Переход на вкладку "Books": Система отображает страницу со списком всех книг.
 - Использование поиска: Пользователь вводит текст. Система фильтрует список фильмов по наличию введенной подстроки в названии.
 - Выбор фильма: Пользователь нажимает на изображение фильма. Система открывает страницу с детальной информацией о выбранном фильме.
 - Открытие фильтров: Пользователь нажимает кнопку "Фильтры". Система отображает окно для настройки параметров фильтрации фильмов.

- Просмотр списка фильмов: Пользователь прокручивает страницу для ознакомления со списком фильмов.
- Нажатие кнопки "Rate" у фильма: Пользователь отмечает выбранный фильм. При повторном нажатии отметка снимается.
- Использование дополнительных фильтров (слева): Пользователь выбирает заранее подготовленные фильтры (например, по жанру, году). Система обновляет список фильмов в соответствии с выбранными фильтрами без перезагрузки страницы.
- Нажатие кнопок "Help", "About us", "Developers", "Import/Export": Аналогично поведению на главной странице.

3) Вкладка "TV Shows" (Сериалы)

На данной странице отображается список всех сериалов, доступных в системе.

- Действия пользователя и реакции системы:
 - Нажатие на заголовок сайта (FMDd): Система перенаправляет пользователя на главную страницу.
 - Переход на вкладку "Movies": Система отображает страницу со списком всех фильмов.
 - Переход на вкладку "Books": Система отображает страницу со списком всех книг.
 - Использование поиска: Пользователь вводит текст. Система фильтрует список сериалов по наличию введенной подстроки в названии.
 - Выбор сериала: Пользователь нажимает на изображение сериала. Система открывает страницу с детальной информацией о выбранном сериале.

- Открытие фильтров: Пользователь нажимает кнопку "Фильтры". Система отображает окно для настройки параметров фильтрации сериалов.
- Просмотр списка сериалов: Пользователь прокручивает страницу для ознакомления со списком сериалов.
- Нажатие кнопки "Rate" у сериала: Пользователь отмечает выбранный сериал. При повторном нажатии отметка снимается.
- Использование дополнительных фильтров (слева): Пользователь выбирает заранее подготовленные фильтры. Система обновляет список сериалов без перезагрузки страницы.
- Нажатие кнопок "Help", "About us", "Developers", "Import/Export": Аналогично поведению на главной странице.

4) Вкладка "Books" (Книги)

На данной странице отображается список всех книг, доступных в системе.

- Действия пользователя и реакции системы:
 - Нажатие на заголовок сайта (FMDd): Система перенаправляет пользователя на главную страницу.
 - Переход на вкладку "Movies": Система отображает страницу со списком всех фильмов.
 - Переход на вкладку "TV Shows": Система отображает страницу со списком всех сериалов.
 - Использование поиска: Пользователь вводит текст. Система фильтрует список книг по наличию введенной подстроки в названии.
 - Выбор книги: Пользователь нажимает на изображение книги. Система открывает страницу с детальной информацией о выбранной книге.

- Открытие фильтров: Пользователь нажимает кнопку "Фильтры". Система отображает окно для настройки параметров фильтрации книг.
- Просмотр списка книг: Пользователь прокручивает страницу для ознакомления со списком книг.
- Нажатие кнопки "Rate" у книги: Пользователь отмечает выбранную книгу. При повторном нажатии отметка снимается.
- Использование дополнительных фильтров (слева): Пользователь выбирает заранее подготовленные фильтры. Система обновляет список книг.
- Нажатие кнопок "Help", "About us", "Developers", "Import/Export": Аналогично поведению на главной странице.

5) Страница (меню) детальной информации о произведении (фильм/сериал/книга)

На данной странице отображается подробная информация о выбранном произведении.

- Действия пользователя и реакции системы:
 - Навигация (заголовок сайта, вкладки "Movies", "TV Shows", "Books"): Аналогично поведению на других страницах.
 - Нажатие стрелки "Назад": Система возвращает пользователя на предыдущую страницу, с которой был совершен переход.
 - Нажатие кнопки "Rate": Пользователь отмечает текущее произведение. При повторном нажатии отметка снимается.
 - Нажатие кнопки "See all" (для комментариев): Система открывает страницу или секцию со всеми доступными комментариями к произведению в виде прокручиваемой ленты.
 - Горизонтальная прокрутка списка актеров (если применимо): Пользователь пролистывает список актеров по горизонтали.

- Использование поиска, открытие фильтров, кнопки "Help", "About us", "Developers", "Import/Export": Аналогично поведению на главной странице. (Примечание: функционал "Import/Export" здесь также может включать статистику по данным).

6) Страница настройки фильтров

Данная страница/окно позволяет пользователю задать критерии для отбора произведений.

- Действия пользователя и реакции системы:
 - Навигация (заголовок сайта, вкладки "Movies", "TV Shows", "Books"): Аналогично поведению на других страницах.
 - Настройка фильтров: Пользователь выбирает или вводит значения для различных предложенных фильтров (например, жанр, год, рейтинг, теги).
 - Применение фильтров (например, нажатие кнопки "Поиск" или "Применить"): Система обрабатывает выбранные фильтры и отображает страницу с результатами, соответствующими заданным критериям.
 - Выбор произведения из отфильтрованного списка: Пользователь нажимает на изображение произведения. Система открывает страницу с детальной информацией о нем.
 - Нажатие кнопки "Rate" у произведения, кнопки "Help", "About us", "Developers", "Import/Export", "See customizable statistics": Аналогично поведению на других страницах. Кнопка "See customizable statistics" перенаправляет на страницу статистики.

7) Страница результатов поиска/фильтрации

На данной странице отображаются произведения, соответствующие запросу пользователя.

- Действия пользователя и реакции системы:

- Навигация (заголовок сайта, вкладки "Movies", "TV Shows", "Books"): Аналогично поведению на других страницах.
- Повторное использование поиска или фильтров: Пользователь может уточнить или изменить свой запрос.
- Выбор произведения: Пользователь нажимает на изображение произведения. Система открывает страницу с детальной информацией о нем.
- Нажатие кнопки "Rate" у произведения, кнопки "Help", "About us", "Developers", "Import/Export": Аналогично поведению на главной странице.

8) Вкладка/Страница статистики

Данная страница позволяет пользователю просматривать и настраивать отображение статистических данных.

- Действия пользователя и реакции системы:
 - Навигация (заголовок сайта, вкладки "Movies", "TV Shows", "Books"): Аналогично поведению на других страницах.
 - Использование элементов управления статистикой (кнопки "mode", "attribute Y", "attribute X"): Пользователь выбирает тип отображаемой статистики (mode), атрибуты для оси X и оси Y, задавая многоокритериальный фильтр для выбора подмножества данных для анализа и визуализации.
 - Нажатие кнопок "Help", "About us", "Developers", "Import/Export", "Фильтры", "Поиск": Аналогично поведению на других страницах.

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель.

Графическое представление модели

```
{  
    "id": "ObjectId",  
    "type": "String", // 'Book', 'Film' или 'Series'  
    "title": "String",  
    "description": "String",  
    "tags": ["String"],  
    "rating": "Number",  
    "year": "String", // либо год (Film, Book) либо диапазон (Series)  
    "duration": "Number", // для фильмов  
    "quantityPages": "Number", // для книг  
    "seasons": "Number", // для сериалов  
    "author": "String",  
    "director": "String",  
    "actors": ["String"],  
    "country": "String",  
    "reviews": [  
        {  
            "content": "String",  
            "rating": "Number",  
            "reviewer": "String",  
            "reviewDate": "Date"  
        }  
    ]  
}
```

Рисунок 2 – JSON-схема нереляционной модели.

Описание назначений коллекций, типов данных и сущностей.

Для нашей модели данных определены следующие сущности:

Основная сущность (медиаконтент), объединяющая книги, фильмы и сериалы. Каждый объект имеет:

- id: уникальный идентификатор (24 байта - ObjectId в MongoDB);
- type: тип объекта (book, film, series) (8 байт - String);
- title: название (в среднем 50 байт - String);
- description: описание (в среднем 500 байт - String);
- Набор тегов (tags): ключевые слова, характеризующие объект (в среднем 10 тегов * 15 байт = 150 байт - Array of Strings);
- rating: рейтинг объекта (8 байт - Double);
- year: год выхода (для сериалов – интервал) (8 байт - String);
- duration: продолжительность (применимо к фильмам) (4 байта - Int32);
- quantity-pages: количество страниц (для книг) (4 байта - Int32);
- seasons: количество сезонов (для сериалов) (4 байта - Int32);
- authors: авторы (массив, так как может быть несколько авторов) (в среднем 3 автора * 40 байт = 120 байт - Array of Embedded Documents);
- directors: режиссеры (массив, так как может быть несколько режиссеров) (в среднем 2 режиссера * 40 байт = 80 байт - Array of Embedded Documents);
- screenwriters: сценаристы (массив) (в среднем 2 сценариста * 40 байт = 80 байт - Array of Embedded Documents);
- producers: продюсеры (массив) (в среднем 2 продюсера * 40 байт = 80 байт - Array of Embedded Documents);
- actors: актеры (массив) (в среднем 10 актеров * 40 байт = 400 байт - Array of Embedded Documents);
- country: страна издания (30 байт - String).

Участники создания (авторы, режиссеры, актеры и т.д.):

- id: уникальный идентификатор (24 байта - ObjectId);
- name: имя участника (40 байт - String);
- role: роль в создании произведения (20 байт - String).

Отзывы: Каждый объект может иметь отзывы, которые содержат:

- content: текст отзыва (в среднем 300 байт - String);
- rating: рейтинг отзыва (8 байт - Double);

- reviewer: имя автора отзыва (30 байт - String);
- reviewDate: дата публикации отзыва (8 байт - Date).

Оценка объема информации, хранимой в модели

- N – количество произведений (документов в catalog_items).
- B – средний объём данных для одного произведения без отзывов (в байтах).

$B = 24(\text{id}) + 8(\text{type}) + 50(\text{title}) + 500(\text{description}) + 150(\text{tags}) + 8(\text{rating}) + 8(\text{year}) + 4(\text{duration}) + 4(\text{quantity-pages}) + 4(\text{seasons}) + 120(\text{authors}) + 80(\text{directors}) + 80(\text{screenwriters}) + 80(\text{producers}) + 400(\text{actors}) + 30(\text{country}) = 1550$ байт

- R – среднее число отзывов на произведение. Примем R = 5 (константа).
- A – средний объём одного отзыва (в байтах).

$A = 300(\text{content}) + 8(\text{rating}) + 30(\text{reviewer}) + 8(\text{reviewDate}) = 346$ байт.

Тогда общий объём хранения для коллекции можно приблизительно оценить по формуле:

$V(N) = N * (B + R * A) = N * (1550 + 3465) = N(1550 + 1730) = 3280N$ байт

Избыточность данных

В нереляционной БД сохраняется дополнительная служебная информация (метаданные, индексы). Пусть «чистый» объём данных равен $V_{\text{clean}} = N * (B + R * A) = 3280N$, а фактический объём хранения с учетом метаданных составляет $V_{\text{actual}} = N * (B + R * A + O)$, где O – средний служебный overhead на документ (примем O = 200 байт).

Фактор избыточности можно выразить так:

$F(O) = (B + R * A + O) / (B + R * A) = (1550 + 3465 + 200) / (1550 + 3465) = 3480 / 3280 \approx 1.06$

Направление роста модели при увеличении количества объектов каждой сущности

При линейном росте количества произведений (N) и/или отзывов (R) общий объём данных растёт линейно. При увеличении количества отзывов для

каждого произведения при фиксированном N объём данных увеличивается пропорционально R .

Примеры данных

Фильм:

```
{  
    "id": "1",  
    "type": "Film",  
    "title": "Интерстеллар",  
    "description": "Эпическая научно-фантастическая драма.",  
    "tags": ["Научная фантастика", "Драма", "Космос"],  
    "rating": 8.7,  
    "year": "2014",  
    "duration": 169,  
    "authors": [{ "id": "101", "name": "Нил Бломкамп", "role":  
    "Сценарист" }],  
    "directors": [{ "id": "201", "name": "Кристофер Нолан", "role":  
    "Режиссер" }],  
    "screenwriters": [  
        { "id": "301", "name": "Джонатан Нолан", "role": "Сценарист" }  
    ],  
    "producers": [{ "id": "401", "name": "Эмма Томас", "role":  
    "Продюсер" }],  
    "actors": [  
        { "id": "501", "name": "Мэттью МакКонахи", "role": "Актер" },  
        { "id": "502", "name": "Энн Хэтэуэй", "role": "Актер" }  
    ],  
    "country": "США",  
    "reviews": [  
        {  
            "content": "Фильм заставляет задуматься о будущем  
человечества.",  
            "rating": 9,  
            "reviewer": "Reviewer1",  
            "reviewDate": "2015-01-15"  
        }  
    ]  
}
```

Книга:

```
{  
    "id": "2",  
    "type": "Book",  
    "title": "1984",  
    "description": "Антиутопический роман Джорджа Оруэлла.",  
    "tags": ["Антиутопия", "Политическая фантастика"],  
    "rating": 9.2,  
    "year": "1949",  
    "quantityPages": 328,  
    "authors": [{ "id": "601", "name": "Джордж Оруэлл", "role": "Автор" }],  
}
```

```
"country": "Великобритания",
"reviews": [
  {
    "content": "Потрясающее видение тоталитарного будущего.",
    "rating": 10,
    "reviewer": "Reviewer1",
    "reviewDate": "1950-03-10"
  }
]
```

Примеры запросов

Найти произведение по названию:

```
db.catalog_items.find({ "title": "Интерстеллар" })
```

Агрегация – средний рейтинг для конкретного произведения

```
db.catalog_items.aggregate([
  { $match: { "title": "Интерстеллар" } },
  { $unwind: "$reviews" },
  { $group: { _id: "$_id", avgRating: { $avg: "$reviews.rating" } } }
])
```

Получить все произведения с определённым тегом:

```
db.catalog_items.find({ "tags": "Космос" })
```

Найти самые популярные произведения (по рейтингу):

```
db.catalog_items.find().sort({ "rating": -1 }).limit(10)
```

Найти самые непопулярные произведения (по рейтингу):

```
db.catalog_items.find().sort({ "rating": 1 }).limit(10)
```

Найти пользователей, оставивших больше всего отзывов:

```
db.catalog_items.aggregate([
  { $unwind: "$reviews" },
  { $group: { _id: "$reviews.reviewer", count: { $sum: 1 } } },
  { $sort: { count: -1 } },
  { $limit: 10 }
])
```

Вычислить статистику по жанрам (количество отзывов):

```
db.catalog_items.aggregate([
  { $unwind: "$tags" },
  { $unwind: "$reviews" },
  { $group: { _id: "$tags", count: { $sum: 1 } } },
  { $sort: { count: -1 } }
])
```

- Количество запросов:

При простом поиске одного элемента требуется один запрос. При агрегировании с разворачиванием отзывов – один агрегатный запрос. Если же необходимо выполнить сложный поиск с фильтрами, может потребоваться сначала запрос на выборку фильтруемых данных, затем агрегация – общее число запросов зависит от выбранной реализации (обычно 1–2 запроса).

- Задействованные коллекции:

В нереляционной модели – одна коллекция catalog_items.

3.2. Реляционная модель.

Графическое представление модели

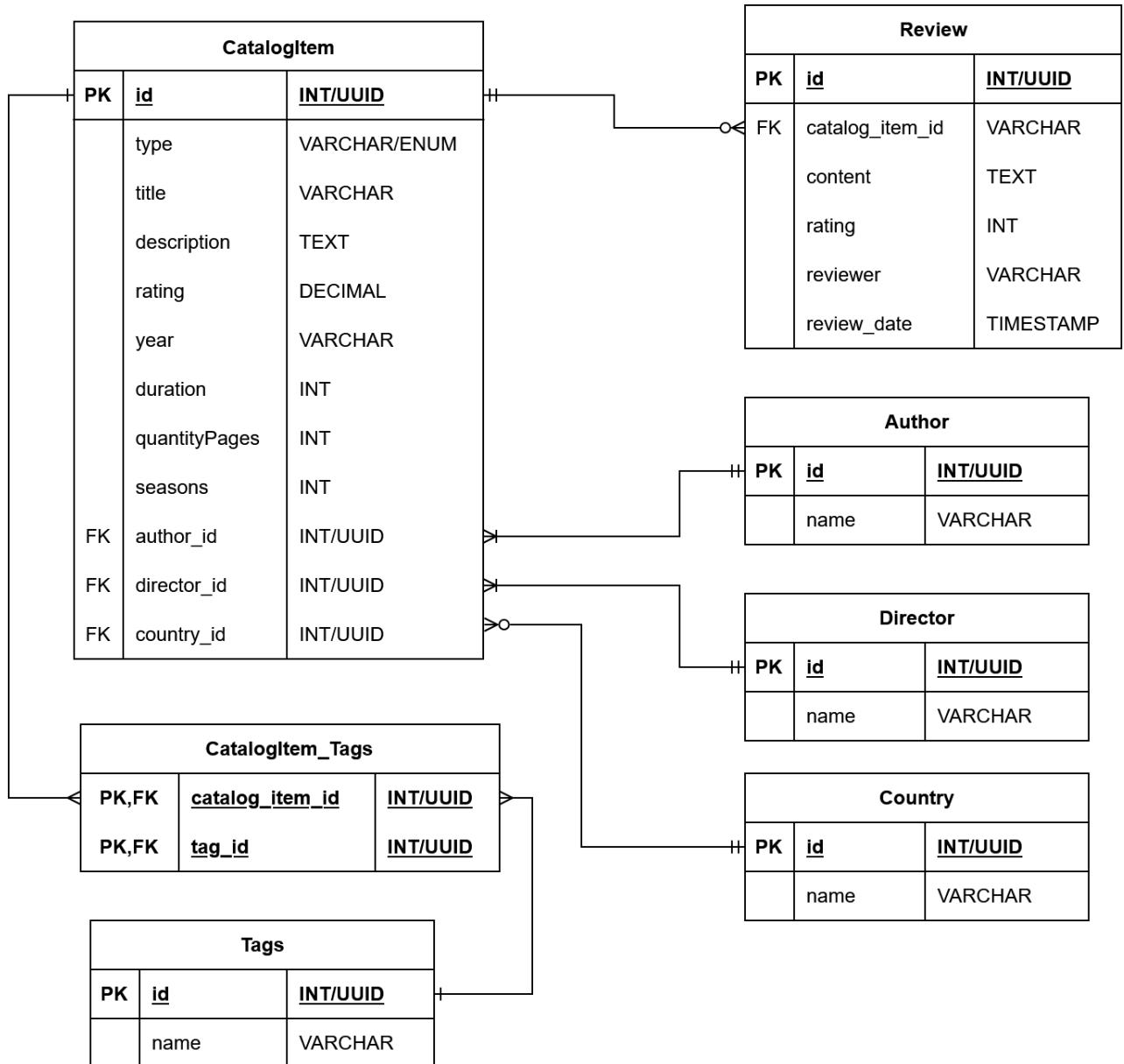


Рисунок 3 – ER-модель.

Описание назначений коллекций, типов данных и сущностей.

Таблица CatalogItem:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор произведения

type	VARCHAR/EN UM	8	Тип произведения (например, 'Book', 'Film', 'Series')
title	VARCHAR	50	Заголовок произведения
description	TEXT	500	Описание или аннотация
rating	DECIMAL	8	Средняя оценка произведения
year	VARCHAR	8	Год выпуска или интервал (например, "2014" или "2008-2018")
duration	INT	4	Продолжительность (в минутах, для фильмов)
quantityPages	INT	4	Количество страниц (для книг)
seasons	INT	4	Количество сезонов (для сериалов)
country_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на С

Таблица Person:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор персоны
name	VARCHAR	40	Имя персоны

Таблица Role:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор роли
name	VARCHAR	20	Название роли (автор, режиссер, актер и т.д.)

Таблица CatalogItem_Person:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор записи
catalog_item_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на CatalogItem.id
person_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на Person.id
role_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на Role.id

Таблица Country:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор страны
name	VARCHAR	30	Название страны

Таблица Review:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор отзыва
catalog_item_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на CatalogItem.id
content	TEXT	300	Текст отзыва (может быть пустым, если отзыв представляет собой только оценку)
rating	INT	4	Оценка, выставленная в отзыве
reviewer	VARCHAR	30	Имя ревьюера
review_date	TIMESTAMP	8	Дата и время создания отзыва

Таблица Tags:

Поле	Тип данных	Размер (байт)	Описание
id	INT/UUID (PK)	4	Уникальный идентификатор тега
name	VARCHAR	15	Название тега

Таблица CatalogItem_Tags:

Поле	Тип данных	Размер (байт)	Описание
catalog_item_id	INT/UUID (FK)	4	Внешний ключ,

			ссылающийся на CatalogItem.id
tag_id	INT/UUID (FK)	4	Внешний ключ, ссылающийся на Tags.id

- CatalogItem: Хранит основную информацию о произведении
- Person: Содержит данные о всех участниках создания произведений (авторах, режиссерах, актерах и т.д.)
- Role: Хранит различные роли, которые могут выполнять персоны
- CatalogItem_Person: Связующая таблица для реализации отношения "многие ко многим" между произведениями и персонами с указанием роли
- Review: Каждый отзыв связан с произведением через внешний ключ. Отзывы хранят оценку, текст и дату
- Country: Содержит информацию о странах издания или производства произведений
- Tags: Хранит уникальные теги, которые используются для классификации и поиска произведений
- CatalogItem_Tags: Является связующей таблицей для реализации отношения «многие ко многим» между таблицами CatalogItem и Tags

Оценка объема информации, хранимой в модели

- N – число записей в таблице CatalogItem,
- B' – средний объём хранения одной записи CatalogItem, $B' = 4 (\text{id}) + 8 (\text{type}) + 50 (\text{title}) + 500 (\text{description}) + 8 (\text{rating}) + 8 (\text{year}) + 4 (\text{duration}) + 4 (\text{quantityPages}) + 4 (\text{seasons}) + 4 (\text{country_id}) = 594$ байт
- R – среднее число отзывов на произведение. Примем R = 5 (константа).

- A' – средний объём хранения одной записи Review, $A' = 4(id) + 4(catalog_item_id) + 300(content) + 4(rating) + 30(reviewer) + 8(review_date) = 350$ байт
- P – среднее число персон, связанных с одним произведением. Примем $P = 17$ (константа).
- L' – средний объём хранения одной записи CatalogItem_Person, $L' = 4(id) + 4(catalog_item_id) + 4(person_id) + 4(role_id) = 16$ байт
- T – среднее число тегов для одного произведения. Примем $T = 10$ (константа).
- G' – средний объём хранения одной записи CatalogItem_Tags, $G' = 4(catalog_item_id) + 4(tag_id) = 8$ байт

Общий объём хранения:

$$V(N) = N \cdot B' + N \cdot R \cdot A' + N \cdot P \cdot L' + N \cdot T \cdot G' = 594N + 350N \cdot 5 + 16N \cdot 17 + 8N \cdot 10 = 594N + 1750N + 272N + 80N = 2696N \text{ байт}$$

Избыточность данных

Коэффициент избыточности:

$$F(O') = (B' + R \cdot A' + P \cdot L' + T \cdot G' + O') / (B' + R \cdot A' + P \cdot L' + T \cdot G')$$

где O' – overhead на запись (индексы, метаданные), примем $O' = 200$ байт.

$$F(O') = (594 + 350 \cdot 5 + 16 \cdot 17 + 8 \cdot 10 + 200) / (594 + 350 \cdot 5 + 16 \cdot 17 + 8 \cdot 10) = 2896 / 2696 \approx 1.074$$

Направление роста модели при увеличении количества объектов каждой сущности

Объём данных растет линейно от числа записей в CatalogItem (N), числа отзывов ($N \cdot R$), числа персон, связанных с произведениями ($N \cdot P$), и числа тегов ($N \cdot T$). При нормализации избыточность ниже, чем в нереляционной модели.

Примеры данных

Пример – Фильм

Таблица CatalogItem:

id	type	title	desc ripti on	rati ng	year	dur atio n	qua ntity Pag es	seas ons	cou ntry _id
1	Film	Интерстеллар	Эпическая научно-фантастическая драма	8.7	2014	169	NULL	NULL	1

Таблица Person:

id	name
1	Нил Бломкамп
2	Кристофер Нолан
3	Мэттью МакКонахи
4	Энн Хэтэуэй
5	Джонатан Нолан
6	Эмма Томас

Таблица Role:

id	name
1	Автор

2	Режиссер
3	Актёр
4	Сценарист
5	Продюсер

Таблица CatalogItem_Person:

id	catalog_item_id	person_id	role_id
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	3
5	1	5	4
6	1	6	5

Таблица Country:

id	name
1	США

Таблица Review:

id	catalog_item_id	content	rating	reviewer	review_date
1	1	Фильм заставляет задуматься о будущем человечества.	9	Reviewer1	2015-01-15 12:00:00

Таблица Tags:

id	name
1	Научная фантастика
2	Драма
3	Космос

Таблица CatalogItem_Tags:

catalog_item_id	tag_id
1	1
1	2
1	3

Примеры запросов

Найти произведение по названию:

```
SELECT * FROM CatalogItem WHERE title = 'Интерстеллар';
```

Получить отзывы для произведения:

```
SELECT * FROM Review WHERE catalog_item_id = 1;
```

Агрегация – вычислить средний рейтинг для произведения:

```
SELECT catalog_item_id, AVG(rating) AS avgRating FROM Review WHERE catalog_item_id = 1 GROUP BY catalog_item_id;
```

Найти самые популярные произведения (по рейтингу):

```
SELECT * FROM CatalogItem ORDER BY rating DESC LIMIT 10;
```

Найти самые непопулярные произведения (по рейтингу):

```
SELECT * FROM CatalogItem ORDER BY rating ASC LIMIT 10;
```

Найти пользователей, оставивших больше всего отзывов:

```
SELECT reviewer, COUNT(*) as review_count
FROM Review
GROUP BY reviewer
ORDER BY review_count DESC
LIMIT 10;
```

Вычислить статистику по жанрам (количество отзывов):

```
SELECT t.name as tag_name, COUNT(r.id) as review_count
FROM Tags t
JOIN CatalogItem_Tags ct ON t.id = ct.tag_id
JOIN CatalogItem ci ON ct.catalog_item_id = ci.id
JOIN Review r ON ci.id = r.catalog_item_id
GROUP BY t.id, t.name
ORDER BY review_count DESC;
```

- Количество запросов:

Для одного юзкейса (например, получение страницы произведения с отзывами) потребуется минимум 2 запроса: один к CatalogItem и один к Review (или JOIN-запрос). При поиске с фильтрами может потребоваться дополнительное количество запросов в зависимости от сложности.

- Задействованные таблицы:

Основные таблицы: CatalogItem, Person, Role, CatalogItem_Person, Country, Review, Tags, CatalogItem_Tags.

3.3. Сравнение моделей.

Удельный объем информации

Нереляционная модель (NoSQL): Объем хранения оценивается по формуле $V(N) = 3280N$ байт. При этом в документе могут храниться избыточные данные, что увеличивает коэффициент избыточности $F \approx 1.06$.

Реляционная модель (SQL): Объем хранения оценивается по формуле $V(N) = 2696N$ байт. При нормализации данных избыточность снижается, коэффициент избыточности $F \approx 1.074$.

Запросы и их сложность

NoSQL:

Количество запросов для реализации юзкейсов может быть меньше за счёт того, что данные о произведении и отзывы хранятся в одном документе (отсутствуют JOIN-ы). При сложных агрегирований могут потребоваться агрегатные запросы, но количество задействованных коллекций – одна.

SQL:

Для большинства юзкейсов требуется как минимум два запроса (например, сначала запрос к CatalogItem, затем JOIN или дополнительный запрос к таблице Review). Количество задействованных таблиц – минимум две (CatalogItem и Review), а для сложных запросов могут потребоваться JOIN-ы нескольких таблиц.

Итоговое сравнение по объему и запросам

При равных объемах «чистых» данных модель NoSQL может иметь больший фактический объем хранения из-за хранения документов со встроенными метаданными, тогда как нормализация в SQL позволяет избежать избыточности. Запросы в NoSQL часто проще (отсутствие JOIN-ов), но сложные агрегации могут быть менее производительны при огромном объеме данных. При росте количества объектов обе модели демонстрируют линейное увеличение объема данных, однако реляционная модель за счёт нормализации может быть более оптимизирована с точки зрения хранения, а NoSQL – с точки зрения гибкости и масштабируемости.

Вывод

NoSQL подходит для проектов, где важна гибкость схемы, возможность быстрого прототипирования и масштабируемость при горизонтальном распределении данных. Для каталога, объединяющего разнородные данные (произведения и отзывы), модель с вложенными документами упрощает

реализацию основных сценариев (поиск, отображение отзывов) и позволяет избежать затрат на JOIN-операции.

SQL обладает преимуществом нормализации данных, что снижает избыточность и может быть эффективным при сложных запросах, требующих строгой целостности данных. Однако для нашей системы схема SQL может оказаться менее гибкой.

Для нашего проекта, где набор данных может постоянно расширяться и меняться, а отзывы являются важной составляющей, нереляционная модель предоставляет больше гибкости и упрощает разработку.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Наш проект представляет собой веб-приложение "FMDd" (Fantasy Media Database) – информационный сервис, предназначенный для каталогизации и поиска информации о книгах, фильмах и сериалах в жанрах фантастики и фэнтези.

Архитектура и технологии:

Приложение построено по клиент-серверной архитектуре:

1. Фронтенд (Пользовательский интерфейс):

- Разработан на Angular с использованием TypeScript.
- Отвечает за отображение информации пользователю, взаимодействие с ним (прием поисковых запросов, выбор фильтров, навигация) и отправку запросов на бэкенд.

2. Бэкенд (Серверная часть):

- Разработан на Java с использованием фреймворка Spring Boot.
- Предоставляет API (интерфейс программирования приложений), через который фронтенд запрашивает и отправляет данные.
- Отвечает за бизнес-логику: обработку запросов, фильтрацию, поиск, взаимодействие с базой данных.

3. База данных:

- Используется MongoDB – нереляционная (NoSQL) документо-ориентированная база данных.
- Основная информация о произведениях (книги, фильмы, сериалы), их атрибуты (название, описание, теги, рейтинг, год, создатели, актеры, страна).

Функциональные возможности

Пользователь, взаимодействуя с интерфейсом, может выполнять следующие действия:

1. Просмотр и навигация:

- Главная страница: Отображает список рекомендаций.
- Разделы по типам контента: Переходить на отдельные страницы для просмотра списков "Movies" (Фильмы), "TV Shows" (Сериалы) и "Books" (Книги).
- Просмотр деталей произведения: Нажав на карточку (иконку) произведения, пользователь попадает на страницу с подробной информацией о нем (описание, год, рейтинг, актеры, режиссеры, авторы и т.д., в зависимости от типа контента).

2. Поиск и фильтрация:

- Поиск по названию: Вводить текст в строку поиска для фильтрации произведений по наличию введенной подстроки в названии.
- Расширенная фильтрация: Открывать специальное окно/страницу для настройки фильтров по различным критериям (например, жанры, теги, год выпуска).

3. Взаимодействие с контентом:

- Оценка произведений: Пользователь может поставить "лайк" или оценку произведению. Повторное нажатие убирает отметку.

4. Дополнительные функции:

- Импорт/Экспорт данных: Присутствует функционал для импорта и экспорта данных.
- Просмотр статистики: Возможность перейти на страницу статистики, где данные могут быть представлены в виде графиков или диаграмм на основе выбранных атрибутов (mode, attribute X, attribute Y).

Основная цель приложения — предоставить удобный инструмент для исследования обширного мира фантастики и фэнтези, позволяя пользователям легко находить информацию о произведениях, открывать для себя новое и, возможно, делиться своим мнением. Использование MongoDB позволяет гибко управлять разнообразными данными, характерными для медиаконтента.

Далее представлены экраны реализованного приложения:

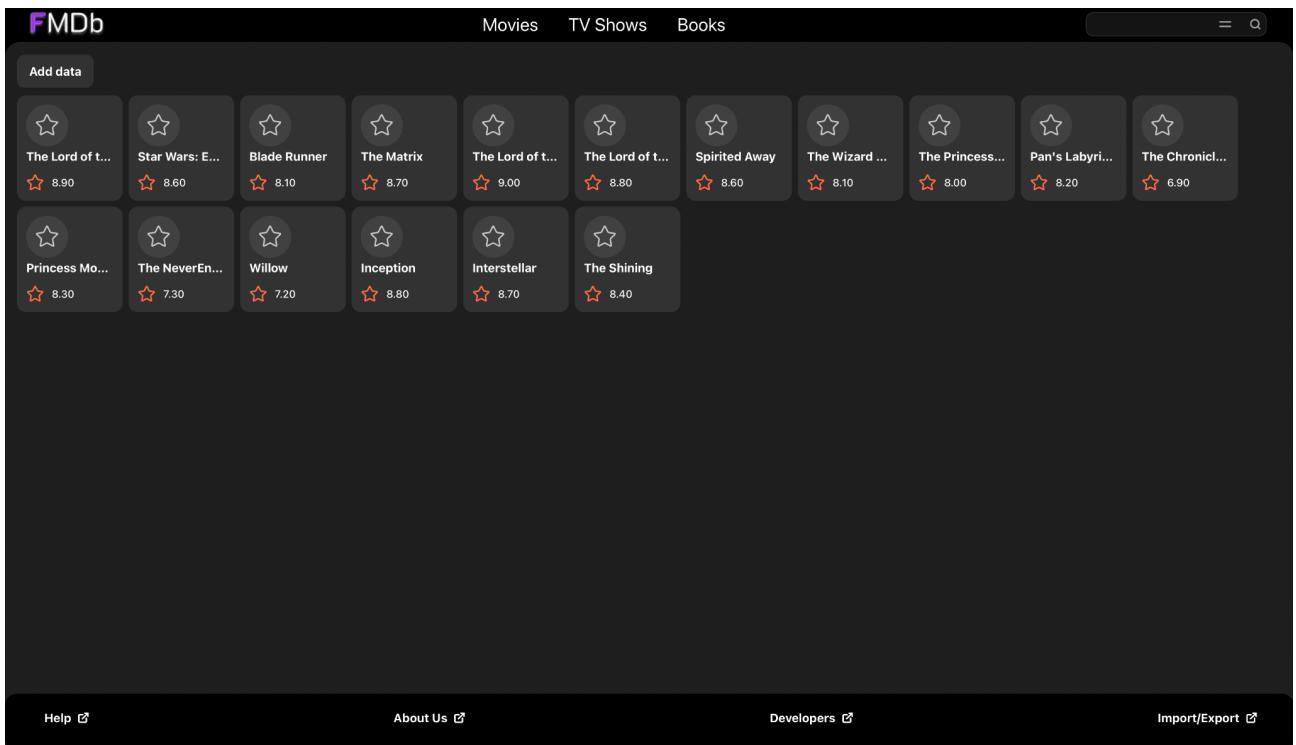


Рисунок 4 – Страница фильмов.

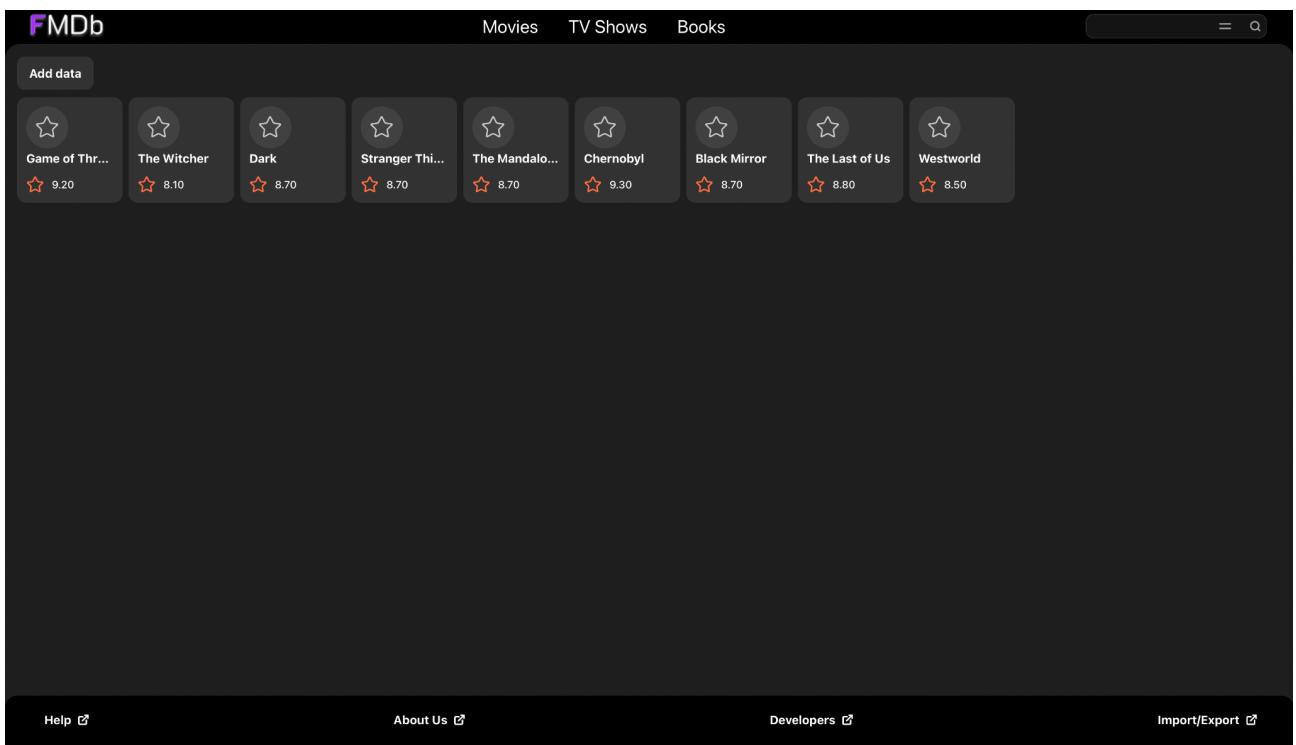


Рисунок 5 – Страница шоу и сериалов.

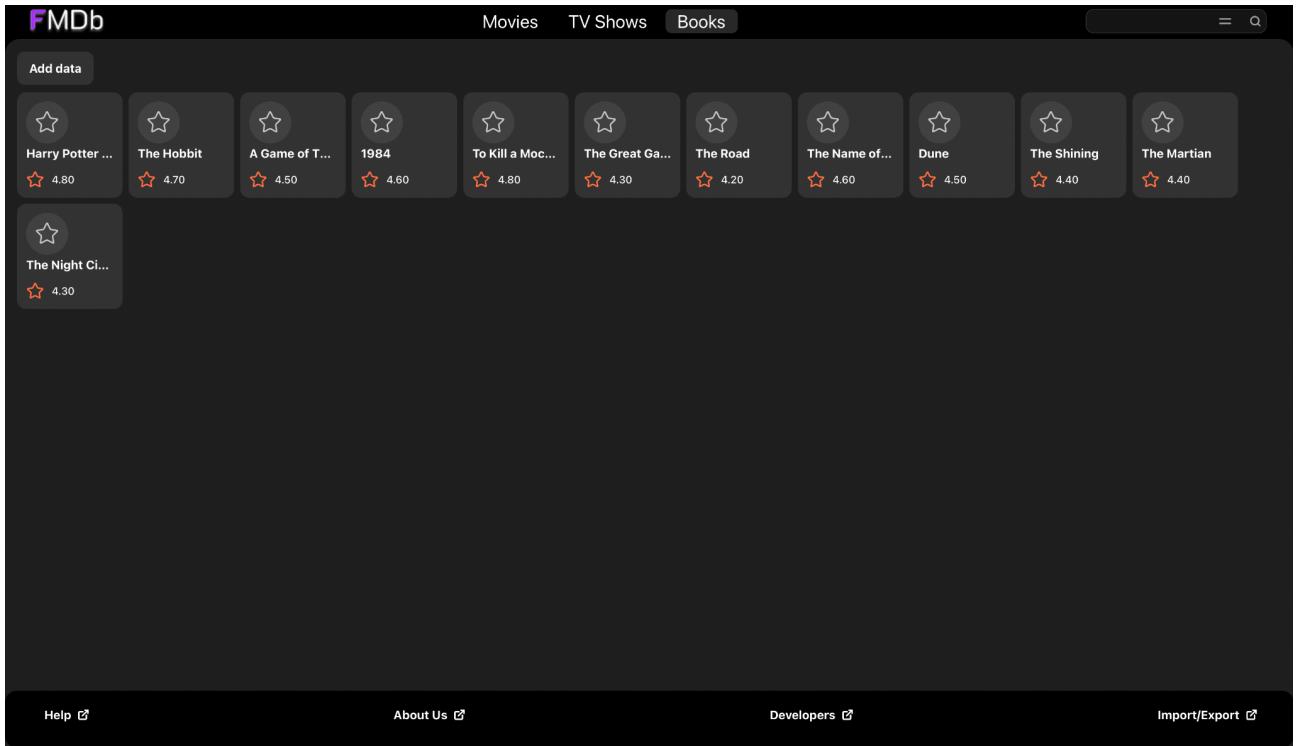


Рисунок 6 – Страница книг.

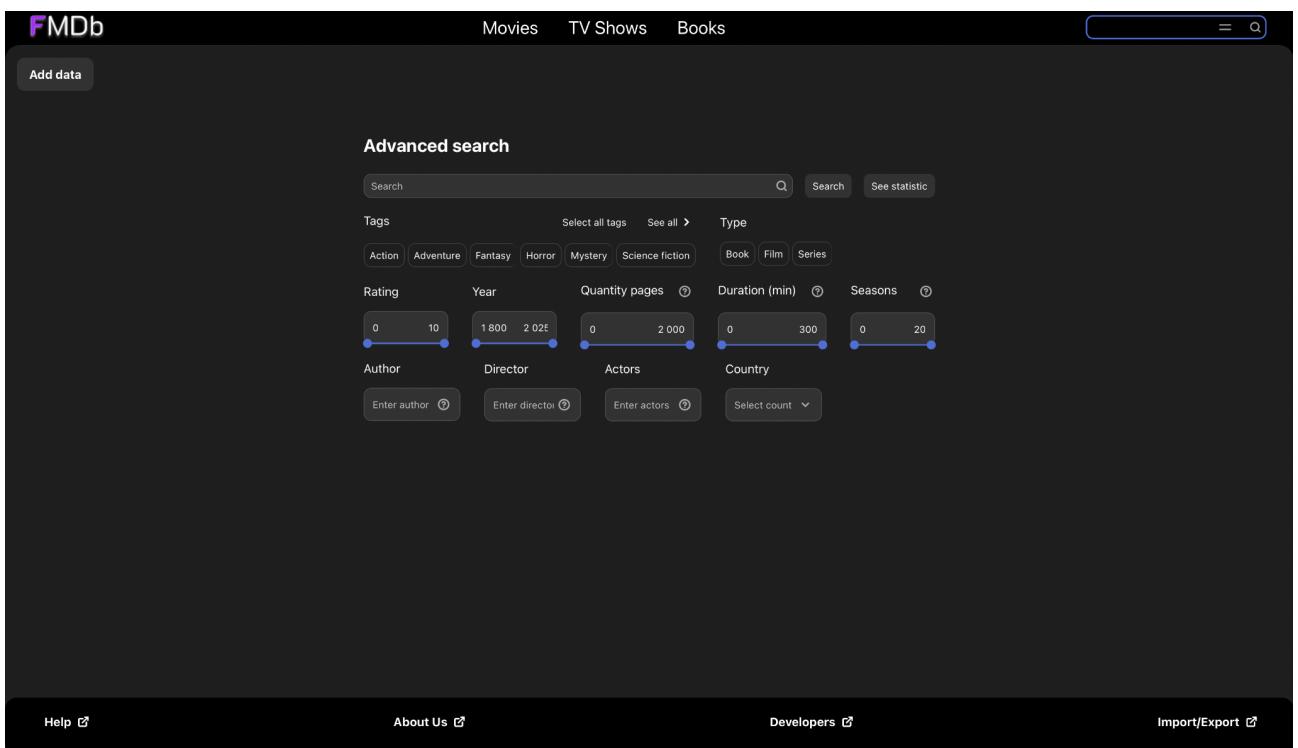


Рисунок 7 – Страница глобального поиска и фильтра.

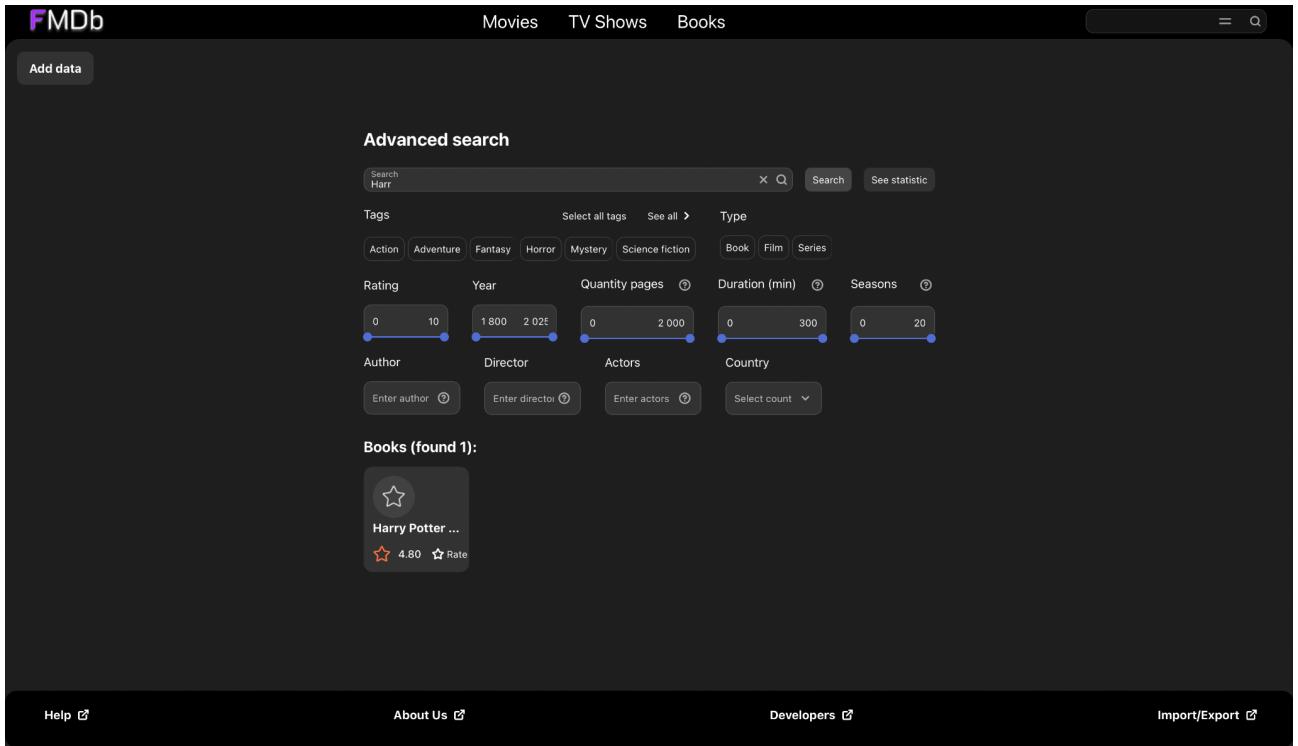


Рисунок 8 – Результат поиска произведения.

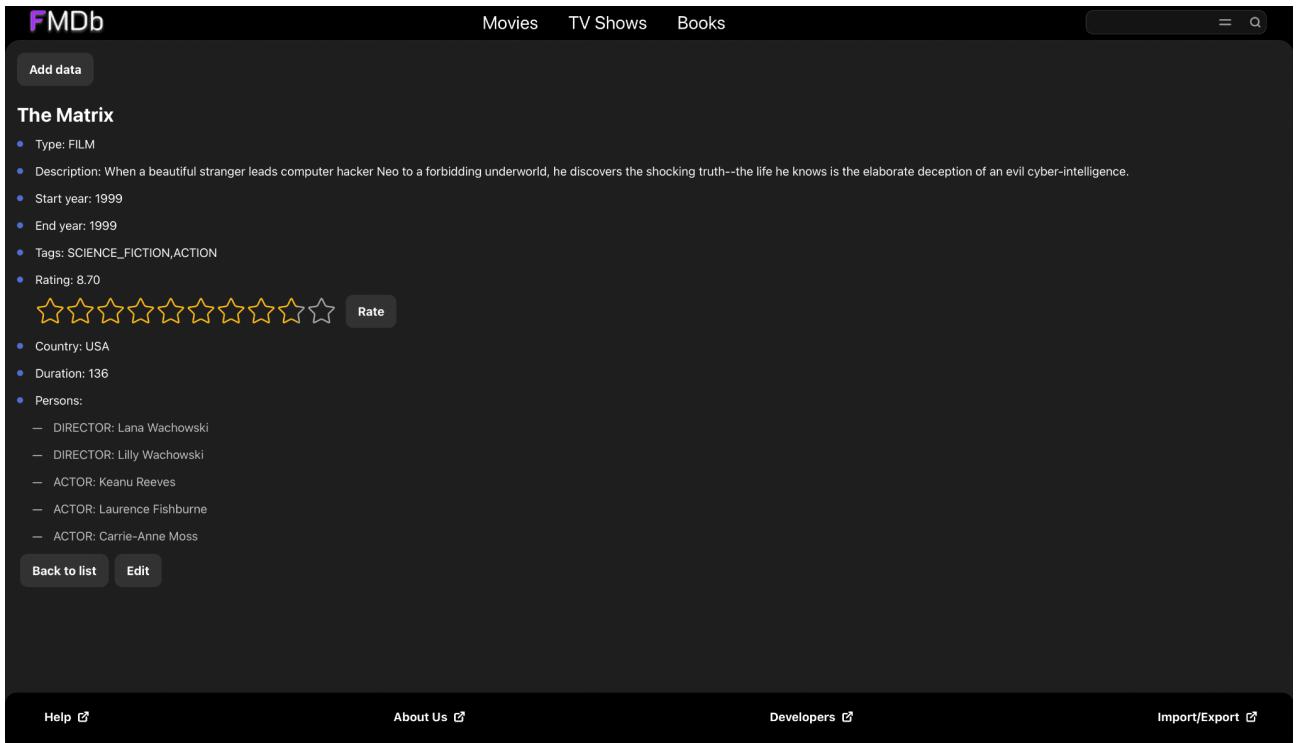


Рисунок 9 – Пример страницы произведения.

The screenshot shows the FMDb website interface. At the top, there are navigation links for Movies, TV Shows, and Books. Below the navigation, a modal window titled "Add data" is open. The modal contains fields for "Type" (set to "BOOK"), "Title" (with placeholder "Enter title"), "Description" (with placeholder "Enter description"), "Rating" (set to "10"), "Year" (range slider from 1800 to 2025), "Country" (dropdown menu with placeholder "Select country"), and "Duration (min/seasons/pages)" (set to "0"). There is also a "Try it" button and a "ExampleAuthor" input field. At the bottom of the modal is a large blue "Отправить" (Send) button.

Рисунок 10 – Форма добавления нового произведения.



Рисунок 11 – Страница представления статистики.

ЗАКЛЮЧЕНИЕ

Результат

Разработанное веб-приложение "FMDd" представляет собой информационный сервис для каталогизации и поиска книг, фильмов и сериалов в жанрах фантастики и фэнтези. В ходе проекта была успешно реализована клиент-серверная архитектура с использованием Java Spring Boot для бэкенда, Angular для фронтенда и MongoDB в качестве базы данных. Такой стек технологий позволил создать отзывчивую и масштабируемую систему.

Возможные направления для улучшения и развития:

1. Пользовательские аккаунты и персонализация:

- Реализация регистрации и авторизации пользователей.
- Создание личных списков (например, "просмотрено", "хочу посмотреть", "любимое").
- Более продвинутая система рекомендаций, основанная на истории просмотров и оценок пользователя.

2. Расширение функционала взаимодействия:

- Возможность для пользователей оставлять полноценные рецензии и комментарии, а не только ставить оценки.
- Система рейтинга пользователей и их отзывов.
- Уведомления о выходе новых серий или фильмов по подпискам.

3. Административная панель:

- Создание полноценного интерфейса для администраторов для удобного добавления, редактирования и удаления контента, управления тегами, пользователями и просмотра статистики.

4. Оптимизация и UI/UX:

- Дальнейшая оптимизация запросов к базе данных, особенно для сложных фильтров и агрегаций.

- Улучшение дизайна и пользовательского опыта (UI/UX),
повышение адаптивности интерфейса для мобильных устройств.

Инструкция по развертыванию проекта

Для развертывания проекта с использованием Docker и Docker Compose выполните следующие шаги:

1. Предварительные требования:

- Установленный Docker Engine.
- Установленный Docker Compose.

2. Подготовка:

- Клонируйте репозиторий проекта (если он еще не скопирован):

```
git clone https://github.com/moevm/nosql1h25-fantasy.git
cd nosql1h25-fantasy
```

- Создайте файл .env в корневой директории проекта (там же, где находится файл docker-compose.yml). Этот файл будет содержать переменные окружения, необходимые для конфигурации сервисов.

```
MONGODB_USER=your_mongo_user
MONGODB_PASSWORD=your_mongo_password
MONGODB_DATABASE=your_fantasy_db_name
MONGODB_PORT=27017
```

3. Сборка и запуск контейнеров:

- Откройте терминал или командную строку в корневой директории проекта.
- Соберите образы Docker для бэкенда и фронтенда:

```
docker-compose up -b -d
```

4. Доступ к приложению:

- После успешного запуска контейнеров:

Фронтенд-приложение будет доступно по: <http://127.0.0.1:4200>

Бэкенд API будет доступен по адресу: <http://127.0.0.1:8080>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: <https://docs.mongodb.com/manual/>
2. Документация Angular: <https://v17.angular.io/docs>
3. Документация Spring Boot: <https://docs.spring.io/spring-boot/index.html>
4. Ссылка на репозиторий с кодом проекта: https://github.com/moevm/nosql1h_25-fantasy

ПРИЛОЖЕНИЕ
ССЫЛКА НА РЕПОЗИТОРИЙ С КОДОМ

Github: <https://github.com/moevm/nosql1h25-fantasy>