**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

**Колледж инновационных технологий и предпринимательства**

КАФЕДРА ФИЗИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

**ПО ДИСЦИПЛИНЕ «Технология разработки программного обеспечения»**

**на тему «Веб-приложение видеокаталогизации управления домашней коллекции»**

Студента Годовикова Вадим Артёмовича

Специальность: 09.02.03 "Программирование в компьютерных системах"

Группы ПКспк-220

*Руководитель*: доцент каф. ФиПМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лексин А.Ю.

Владимир 2023

ВЛАДИМИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ

Зав. кафедрой

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

по дисциплине «Технология разработки программного обеспечения»

Выдано студенту Годовикову Вадиму Артёмовичу группы ПКсп-120

Специальность подготовки СПО 09.02.03 «Программирование в компьютерных системах»

1. Тема работы «Веб-приложение видеокаталогизации управления домашней коллекции»

2. Срок сдачи законченной работы 21.12.2023

3. Исходные данные к курсовой работе постановка задачи.

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Введение.

1. Анализ предметной области.
2. Постановка задачи
3. Проектирование приложения.
4. Разработка приложения.
5. Тестирование.

Заключение.

5. Постановка задачи

ЧАСТЬ 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1. Выполнить описание предметной области.
2. На основании описания предметной области провести обзор существующих аналогов.

ЧАСТЬ 2. ПОСТАНОВКА ЗАДАЧИ

1. Определить требования и задачи для достижения поставленной цели.
2. Определить минимальные требования для работы программы.

ЧАСТЬ 3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

1. На основании описания предметной области выполнить разработку ER-модели, реляционной модели данных.
2. Составить словарь данных, в котором представлены все поля таблиц базы данных с указанием ключа, наименования, типа, обязательности заполнения и заметок.
3. Представить диаграмму прецедентов, диаграммы активностей.
4. Расписать основные методы, используемые в проектировании интерфейса.

ЧАСТЬ 4. РАЗРАБОТКА ПРИЛОЖЕНИЯ

1. Описать выбор инструментальных средств разработки приложения.
2. На основании ER-диаграммы базы данных, описать физическую реализацию базы данных. Если в работе с базой данных используются представления, то описать подробно каждое из них.
3. Описать все классы приложения с пояснением для каких целей они предназначены и методы с описанием: назначения, входных значений и выходных значений каждого метода.
4. Кратко описать структуру и компоненты, используемые при разработке интерфейса.

ЧАСТЬ 5. ТЕСТИРОВАНИЕ

1. Составить тест-требования, определяющие, что должно быть протестировано, но не определяющие, как это должно быть сделано.
2. По тест-требованиям разработать тест-план, в каждом тестовом примере которого обязательно перечислены все входные значения и ожидаемые выходные значения, а также сценарий, описывающий последовательность действий, которые необходимо выполнить для выполнения тестового примера.

Дата выдачи задания 05.10.2023

Руководитель Лексин А.Ю.

Задание принял к исполнению Годовиков В.А.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc153495400)

[1. Анализ предметной области 4](#_Toc153495401)

[1.1 Описание предметной области 4](#_Toc153495402)

[1.2. Анализ существующих решений задач 5](#_Toc153495403)

[2.ПОСТАНОВКА ЗАДАЧИ 14](#_Toc153495404)

[3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 15](#_Toc153495405)

[3.1 ER-диаграмма 15](#_Toc153495406)

[3.2 Словарь данных 16](#_Toc153495407)

[3.3 Диаграмма вариантов использования 20](#_Toc153495408)

[3.4 Диаграммы деятельности 21](#_Toc153495409)

[3.5 Проектирование макетов пользовательского интерфейса 24](#_Toc153495410)

[4. РАЗРАБОТКА ПРИЛОЖЕНИЯ 31](#_Toc153495411)

[4.1 Инструментальные средства разработки 31](#_Toc153495412)

[4.2 База данных 33](#_Toc153495413)

[4.3 Разработка приложения 34](#_Toc153495414)

[5. ТЕСТИРОВАНИЕ 42](#_Toc153495415)

[5.1 Разработка тест-требований 42](#_Toc153495416)

[5.2 Разработка тест-плана 44](#_Toc153495417)

[Заключение 55](#_Toc153495418)

[Список использованных источников 56](#_Toc153495419)

[Приложение А 57](#_Toc153495420)

[Приложение Б 58](#_Toc153495421)

[Приложение В 64](#_Toc153495422)

ВВЕДЕНИЕ

В современном мире всё больше людей заинтересованы в управлении своей домашней коллекцией видео материалов. С появлением интернета и развитием технологий, видеокаталогизация стала гораздо проще и доступнее. Веб-приложения предоставляют возможность удобно организовать и хранить свою видеоколлекцию, добавлять новые фильмы, а также проводить поиск и сортировку информации.

Целью данной работы является разработка и реализация веб-приложения для упрощения каталогизации фильмов, их сортировки и получения информации о фильмах. Данное приложение будет предоставлять пользователям удобный и понятный интерфейс для добавления новых фильмов, сортировки фильмов по различным критериям, таким как жанр, год выпуска, актёры, страна и рейтинг. Поиск фильмов по популярным жанрам.

Таким образом, разработка данного веб-приложения поможет людям более эффективно управлять своими домашними видеоколлекциями, сэкономив время на поиск и сортировку фильмов, а также предоставив доступ к обширной и актуальной информации о каждом фильме.

1. Анализ предметной области

## 1.1 Описание предметной области

Для анализа предметной области была взята тема каталогизации фильмов.

Каталогизация фильмов – это процесс систематизации и классификации информации о фильмах с целью обеспечения удобного доступа к этой информации со стороны пользователей. Каталогизация фильмов играет важную роль в современной индустрии развлечений, поскольку помогает организовать и управлять огромным количеством фильмов, а также облегчает поиск и отбор фильмов по определенным критериям [1].

Основной задачей каталогизации фильмов является создание единой структуры базы данных, содержащей информацию о каждом фильме. Информация о фильмах включает в себя: название фильма, описание фильма, год выпуска, страну выпуска, жанр, режиссера, актеров, рейтинг и возрастное ограничение.

В настоящее время почти каждый человек любит смотреть фильмы. Скорее всего, часть каких-то фильмов хранится на устройствах: на компьютере, на флеш-накопителе, на телефоне и на других накопителях. А если фильмов очень большое количество, то скорее всего встаёт вопрос о том, чтобы фильмы искать не только по названию, но ещё и сортировать по актёрам, жанрам, стране выпуска, по рейтингу и возрастному ограничению.

Работа каталогизатора начинается с авторизации в приложении. Дальше пользователь должен загрузить фильм в библиотеку в приложении, и ему сразу же выдаётся информация о фильме, который он загрузил. Информация о фильмах берётся из интернета, а точнее из «КиноПоиск API». На этой же странице имеются предложенные фильмы, по типу загруженного фильма.

API – это интерфейс программирования приложений. API сервис предоставляет набор готовых процедур, функций и структур, с помощью которых будет разрабатываться веб-приложение. Данный API сервис нужен для того, чтобы получать данные о фильме, которые в последствие будет записываться в базу данных.

После того, как пользователь загрузит большое количество фильмов, ему в его библиотеке станет намного проще искать нужные фильмы путём сортировки по жанрам, актёрам, по году выпуска и по стране фильма.

Сортировка происходит путём поиска по определённому атрибуту из базы данных. К примеру, если человек выбирает сортировку по жанру, то ему предоставляется список жанров, которые имеются в его библиотеке.

На главной странице каталогизатора будут представлены новинки фильмов в виде слайдера, а также фильмы по популярным жанрам. Кроме того, на главной странице будет отображаться список фильмов, которые пользователь загрузил за всё время.

В целом каталогизация фильмов играет важную роль в организации и управлении фильмами, облегчая поиск и выбор фильмов пользователей.

## 1.2. Анализ существующих решений задач

По выбранной теме было найдено шесть программ аналогов, три из которых являются условно-бесплатными и три платных.

Условно-бесплатные программное обеспечение – программное обеспечение с безвозмездным (или возмездным при определённых условиях) использованием.

Найденные условно-бесплатные ПО:

* Plex.
* Kodi Media Player.
* Stremio.

Найденные платные ПО:

* All My Movies.
* MyFilms.
* Extreme Movie Manager.

**Plex TV** – ПО в котором хранятся свои файлы (обычно на компьютере или внешнем жёстком диске). Сканирует и упорядочивает файлы, автоматически красиво и интуитивно сортируя медиа файлы в библиотеке Plex TV [2].

Достоинства:

1. Кроссплатформенность.
2. Имеет возможность загружать музыку, фильмы и фото.
3. Простота настройки.
4. Бесплатная версия (хватит для повседневного использования).

Недостатки:

1. Платный доступ к расширенным настройкам.
2. Платный доступ к трансляции фильмов или музыки.
3. Рекламное сопровождение в бесплатной версии.

Внешний вид приложения представлен на рисунке 1.

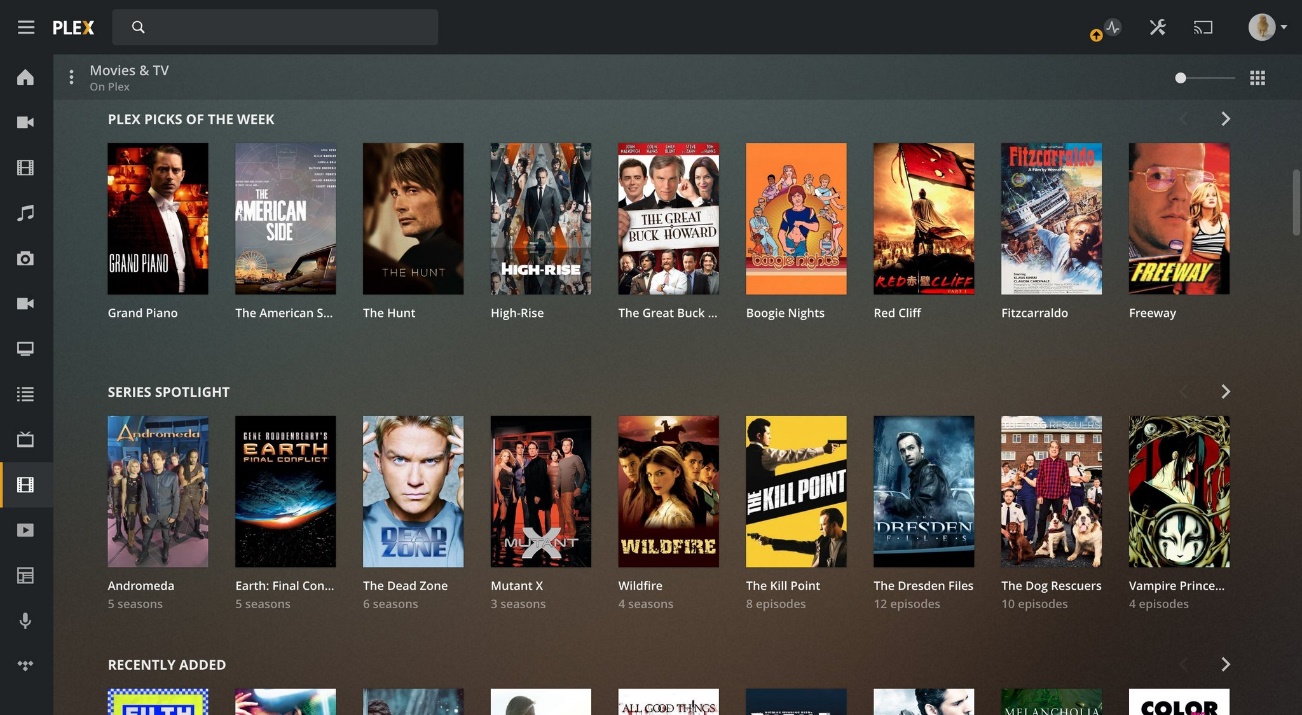


Рисунок 1 – Внешний вид приложения «Plex TV»

**Kodi Media Player** – медиаплеер, который доступен для различных операционных систем и аппаратных платформ. Данный медиаплеер позволяет воспроизводить и смотреть любые расширения файлов мультимедиа прямо из Интернета. Воспроизводит видео и музыку пользователей, которые хранятся на каком-либо накопителе [3].

Достоинства:

1. Кроссплатформенность.
2. Имеет возможность загружать музыку, видео и фото.
3. Абсолютно бесплатное приложение.
4. Удобен для использования.
5. Расширенный функционал настройки интерфейса.

Недостатки:

1. Отсутствие оконного режима – может работать только в полноэкранном режиме.
2. Сенсорное управление на мобильных устройствах – не понятный интерфейс для использования.
3. Платные расширения.

Внешний вид приложения представлен на рисунке 2.



Рисунок 2 – Внешний вид приложения «Kodi Media Player»

**Stremio** – приложение для медиацентра для настольных ПК, смартфонов и планшетов. Файлы медиа не загружаются на устройство пользователя, они запрашиваются с сервера [4].

Достоинства:

1. Хорошая безопасность.
2. Интуитивно понятный интерфейс.
3. Возможность установки дополнений.
4. Абсолютно бесплатное приложение.
5. Трансляций фильмов и телепередач из различных устройств.

Недостатки:

1. Нет возможности установить на большое количество телевизоров.
2. Не работают функции настройки на мобильных устройствах.
3. Наличие рекламы при работе с приложением.

Внешний вид приложения представлен на рисунке 3.

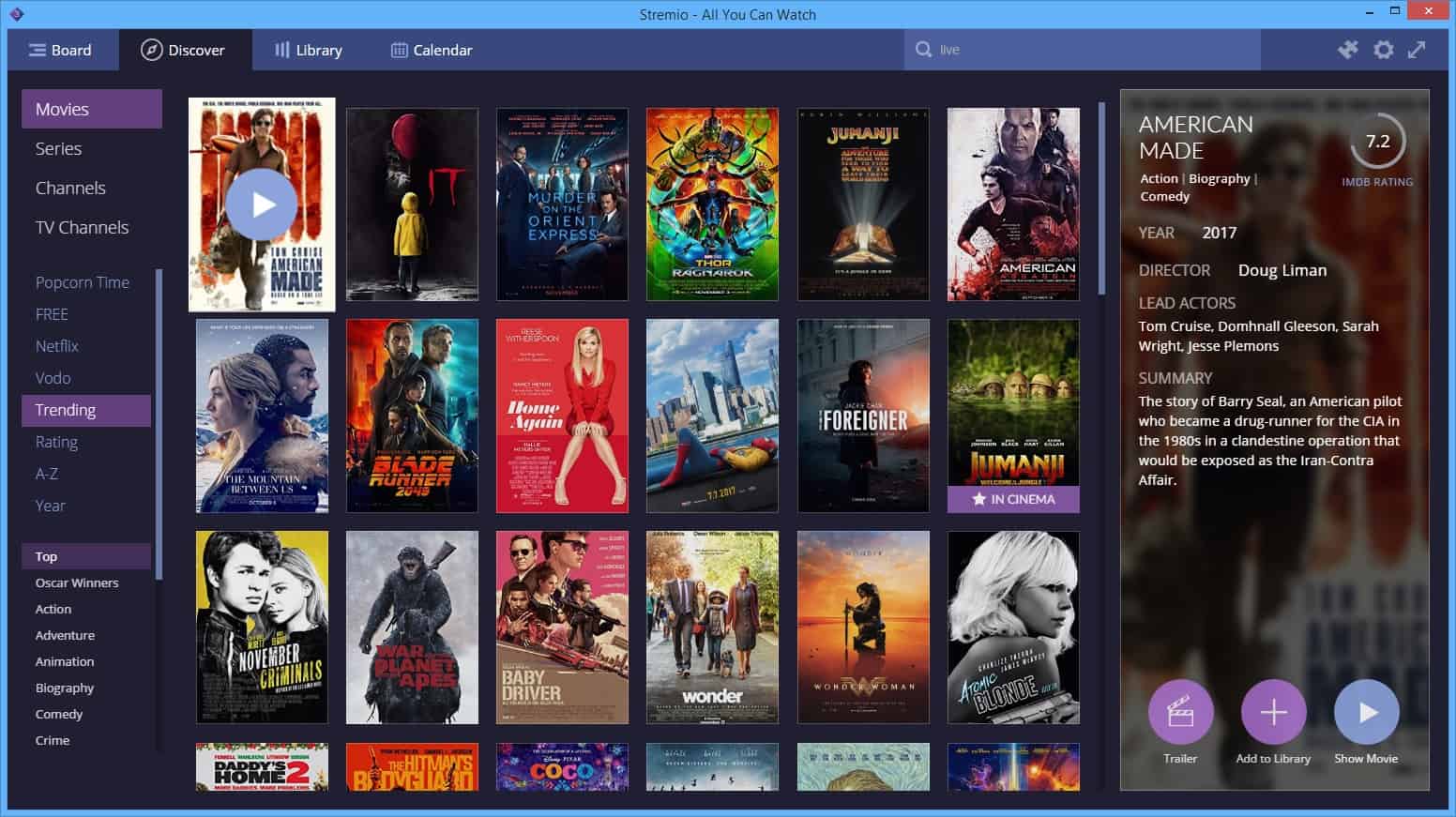


Рисунок 3 – Внешний вид приложения «Stremio»

**All My Movies** – программа для организации и управления вашей коллекцией фильмов. программа разработана для кинофилов, которые хотят подробно отслеживать информацию о своих фильмах, включая данные о фильмах, актерах, режиссерах, жанрах, годе выпуска и многом другом [5].

Достоинства:

1. Легкость использования – приложение обладает интуитивно понятным интерфейсом.
2. Автоматический импорт данных – приложение предоставляет возможность автоматически импортировать информацию о фильмах из различных источников.
3. Расширенные функции поиска и фильтрации – приложение предоставляет множество опций для поиска и фильтрации фильмов.

Недостатки:

1. Стоимость – чтобы пользоваться программой нужна лицензия.
2. Требуется поддержка онлайн-сервисов – для автоматического импорта данных программа зависит от доступности соответствующих онлайн-ресурсов.
3. Ограниченные дополнительные функции.

Внешний вид приложения представлен на рисунке 4.

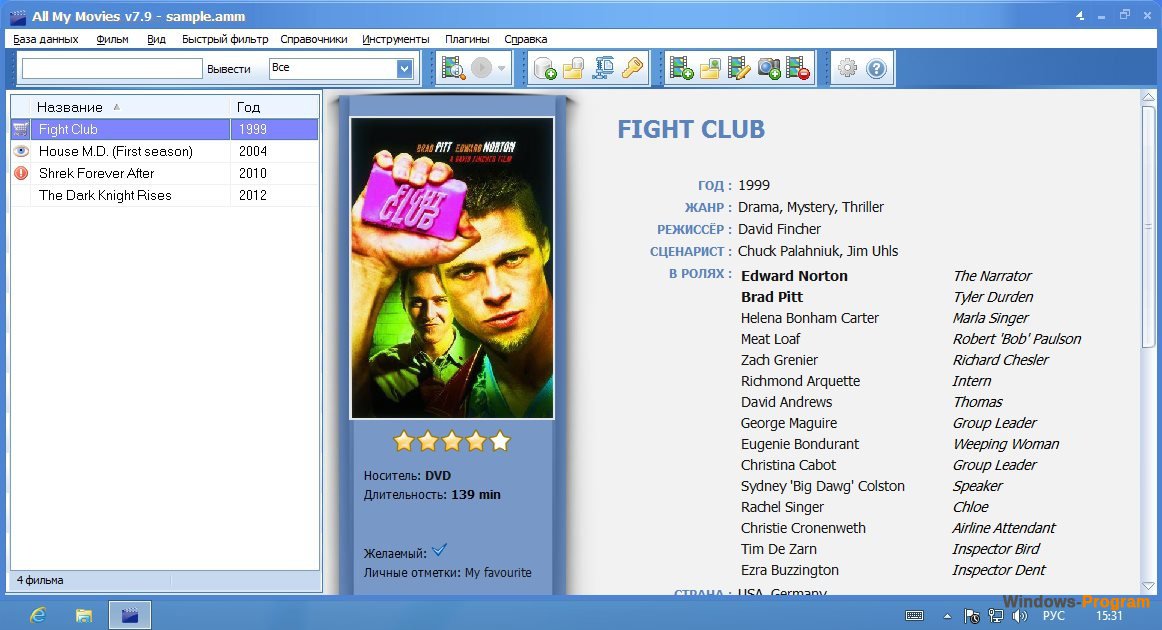


Рисунок 4 – Внешний вид приложения «All My Movies»

**MyFilms** – приложение для управления коллекцией фильмов. С помощью этой программы вы можете организовать и отслеживать любимые фильмы. Она предоставляет удобный интерфейс для добавления новых фильмов, редактирования существующих записей фильмов, а также для просмотра информации о фильмах [6].

Достоинства:

1. Удобство использования – имеет интуитивно понятный интерфейс.
2. Организация фильмов – помогает организовать фильмы в удобном и структурированном порядке.
3. Расширенная информация о фильмах – предоставляет дополнительную информацию о фильмах, такую как рейтинги, отзывы, трейлеры и другие детали.

Недостатки:

1. Ограниченные возможности – программа недостаточно функциональна для продвинутых пользователей, которые ищут расширенные функции управления коллекцией фильмов.
2. Конфликты с антивирусными программами – некоторые антивирусные программы могут распознать данное приложение, как потенциально вредоносное ПО или троян.
3. Ограниченная поддержка форматов – не поддерживает все возможные форматы (к примеру: «.ivi», «.avi»).

Внешний вид приложения представлен на рисунке 5.



Рисунок 5 – Внешний вид приложения «MyFilms»

**Extreme Movie Manager** – приложение для управления коллекцией фильмов и видеофайлов. Предоставляет множество возможностей для организации, классификации и отслеживания информации о фильмах. С помощью данного ПО вы можете добавлять новые фильмы, просматривать подробные данные о фильмах, а также импортировать информацию из онлайн баз данных [7].

Достоинства:

1. Расширенные функции управления – предлагает широкий спектр функций для управления коллекцией фильмов.
2. Поддержка различных форматов – программа поддерживает множество форматов видеофайлов.
3. Мощный поиск и фильтрация – предлагает множество инструментов для поиска и фильтрации фильмов, позволяя быстро находить нужные фильмы на основе различных категорий.

Недостатки:

1. Сложность использования – для новичка может потребоваться время и усилия, чтобы разобраться во всех функциях и возможностях;
2. Высокая стоимость.
3. Отсутствие мобильной версии.

Внешний вид приложения представлен на рисунке 6.



Рисунок 6 – Внешний вид приложения «Extreme Movie Manager»

После изучения аналогов и проведения анализа можно сказать, что все перечисленные приложения выполняют одну и ту же функцию – каталогизацию фильмов. Предоставляют удобный интерфейс для управления коллекцией фильмов и видеофайлов и для просмотра информации о фильмах.

Достоинства:

1. Кроссплатформенность – приложения предоставляют возможность управлять своими коллекциями на разных устройствах, таких как: компьютер, телевизор и мобильных устройствах.
2. Удобность в использовании приложений – у всех таких ПО интуитивно понятный интерфейс даже для рядового пользователя компьютера.
3. Удобность в поиске и фильтрации фильмов – предлагают множество инструментов, позволяя быстро находить нужные фильмы.

Недостатки:

1. У каждого ПО вся информация берётся из различных онлайн баз данных, то есть в режиме оффлайн не получится получить информацию о новом фильме.

Почти каждое приложение имеет платный доступ к расширенным настройкам.

2. ПОСТАНОВКА ЗАДАЧИ

Цель данной работы – разработать и реализовать веб-приложение для упрощения каталогизации фильмов, их сортировки и получения информации о фильмах. Для выполнения поставленной цели были реализованы следующие задачи:

* изучить существующие подходы и технологии для систематизации фильмов;
* проанализировать процессы управления домашней коллекцией фильмов, а также выявить недостатки существующих систем управления каталогизации;
* определить структуру базы данных и взаимодействие с внешними системами, необходимые для реализации веб-приложения;
* разработать и протестировать приложение;
* провести анализ результатов тестирования.

Создание веб-приложения видеокаталогизации поможет упростить процессы каталогизации фильмов и оптимизировать сортировку.

При разработке были выделены минимальные требования для работы программы:

* авторизация и регистрация пользователя;
* информация о фильмах;
* добавление нового фильма в библиотеку;
* сортировка фильмов;
* поиск фильмов по популярным жанрам.

Эти задачи помогут определить, что требуется для разработки веб-приложения и осуществления его успешного запуска.

3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

## 3.1 ER-диаграмма

Для отображения связей базы данных была разработана ER-диаграмма нотации Мартина. ER-диаграмма представлена на рисунке 7.

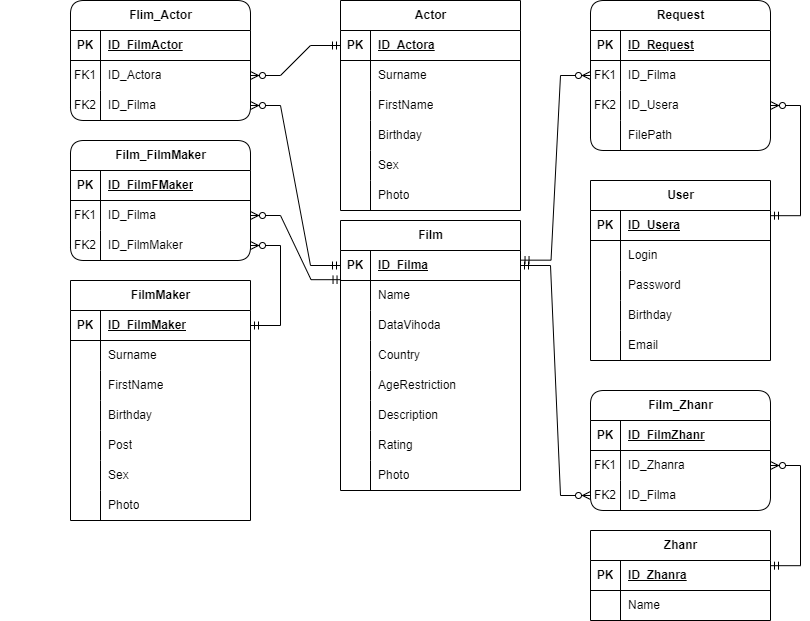


Рисунок 7 – ER-диаграмма

На диаграмме показано взаимодействие таблиц между собой и соответствующие атрибуты каждой таблицы. Между всеми таблицами организована связь «один-ко-многим».

## 3.2 Словарь данных

Таблица 1 – Словарь данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Film (Фильм) | | | | | |
| Ключ | Поле | Перевод | Тип | Обязательное поле | Примечание |
| Первичный ключ | ID\_Filma |  | bigint | Да | Идентификационный номер фильма |
|  | Name | Название | varchar(255) | Да | Название фильма |
|  | DataVihoda | Год выпуска | int | Да | Год выпуска фильма |
|  | Country | Страна | varchar(255) | Да | Страна фильма |
|  | AgeRestriction | Ограничение по возрасту | varchar(3) | Да | Ограничение по возрасту, для просмотра фильма |
|  | Description | Описание | varchar() | Да | Описание фильма |
|  | Rating | Рейтинг | decimal(1,1) | Да | Рейтинг фильма, может быть (1.0-5.0) |
|  | Photo | Фотография | varchar(255) | Да | Ссылка на фотографию фильма |
| Actor (Актёр) | | | | | |
| Первичный ключ | ID\_Actora |  | bigint | Да | Идентификационный номер актёра |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Surname | Фамилия | varchar(255) | Да | Фамилия актёра |
|  | FirstName | Фамилия | varchar(255) | Да | Имя актёра |
|  | Birthday | Дата рождения | date | Нет | Дата рождения актёра |
|  | Sex | Пол | varchar(10) | Да | Пол актёра |
|  | Photo | Фотография | varchar(255) | Да | Ссылка на фотографию актёра |
| Film\_Actor (Ассоциативная таблица для связи между таблицами Film и Actor) | | | | | |
| Первичный ключ | ID\_FilmActor |  | bigint | Да | Идентификационный номер |
| Вторичный ключ | ID\_Actora |  | bigint | Да | Идентификационный номер актёра |
| Вторичный ключ | ID\_Filma |  | bigint | Да | Идентификационный номер фильма |
| FilmMaker (Создатель фильма) | | | | | |
| Первичный ключ | ID\_FilmMaker |  | bigint | Да | Идентификационный номер создателя фильма |
|  | Surname | Фамилия | varchar(255) | Да | Фамилия создателя фильма |
|  | FirstName | Фамилия | varchar(255) | Да | Имя создателя фильма |
|  | Birthday | Дата рождения | date | Нет | Дата рождения создателя фильма |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Post | Должность | varchar(255) | Да | Должность создателя фильма (к примеру: «режиссёр») |
|  | Sex | Пол | varchar(10) | Да | Пол создателя фильма |
|  | Photo | Фотография | varchar(255) | Да | Ссылка на фотографию создателя фильма |
| Film\_FilmMaker (Ассоциативная таблица для связи между таблицами Film и FilmMaker) | | | | | |
| Первичный ключ | ID\_FilmFMaker |  | bigint | Да | Идентификационный номер |
| Вторичный ключ | ID\_Filma |  | bigint | Да | Идентификационный номер фильма |
| Вторичный ключ | ID\_FilmMaker |  | bigint | Да | Идентификационный номер создателя фильма |
| Zhanr (Жанр) | | | | | |
| Первичный ключ | ID\_Zhanra |  | bigint | Да | Идентификационный номер жанра |
|  | Name | Название | varchar(255) | Да | Название жанра |
| Film\_Zhanr (Ассоциативная таблица для связи между таблицами Film и Zhanr) | | | | | |
| Первичный ключ | ID\_FilmZhanr |  | bigint | Да | Идентификационный номер |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вторичный ключ | ID\_Zhanra |  | bigint | Да | Идентификационный номер жанра |
| Вторичный ключ | ID\_Filma |  | bigint | Да | Идентификационный номер фильма |
| User (Пользователь) | | | | | |
| Первичный ключ | ID\_Usera |  | bigint | Да | Идентификационный номер пользователя |
|  | Login | Логин | varchar(255) | Да | Логин пользователя для входа в приложение |
|  | Password | Пароль | varchar(255) | Да | Пароль пользователя для входа в приложение |
|  | Birthday | Дата рождения | date | Да | Дата рождения пользователя |
|  | Email | Электронная почта | varchar(255) | Да | Электронная почта пользователя |
| Request (Запросы – ассоциативная таблица) | | | | | |
| Первичный ключ | ID\_Request |  | bigint | Да | Идентификационный номер запроса |
| Вторичный ключ | ID\_Filma |  | bigint | Да | Идентификационный номер фильма |
| Вторичный ключ | ID\_Usera |  | bigint | Да | Идентификационный номер пользователя |
|  | FliePath | Путь к файлу | varchar(255) | Да | Путь к файлу загруженного фильма |

В базе данных содержится следующий набор сущностей с присущими им атрибутами:

* Фильм (Film): идентификатор фильма, название, год выпуска, страна фильма, возрастное ограничение, описание, рейтинг, ссылка на фотографию.
* Актёр (Actor): идентификатор актёра, фамилия, имя, дата рождения, пол, ссылка на фотографию.
* Создатель фильма (FilmMaker): идентификатор создателя фильма, фамилия, имя, дата рождения, должность, пол, ссылка на фотографию.
* Жанр (Zhanr): идентификатор жанра, название.
* Пользователь (User): идентификатор пользователя, псевдоним, пароль, дата рождения, электронная почта.

Между этими сущностями организованы связи типа «один-ко-многим», «многие-ко-многим» через ассоциативные таблицы.

## 3.3 Диаграмма вариантов использования

Для описания функционала разрабатываемого программного обеспечения была создана диаграмма вариантов использования, показана на рисунке 8.

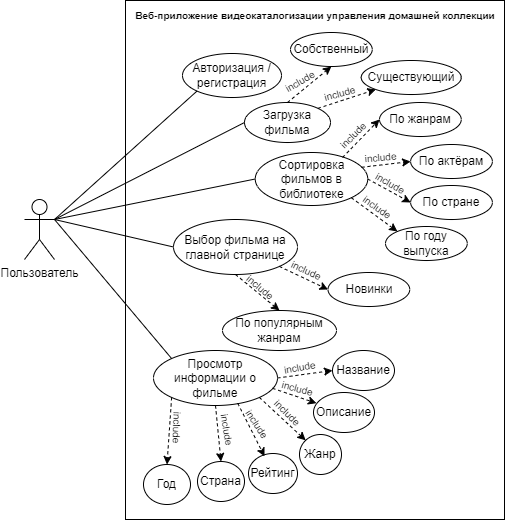


Рисунок 8 – Диаграмма вариантов использования.

На данном рисунке видно, что в программе будет существовать только одна роль, это роль пользователя.

Для начала пользователь входит в приложение, после чего может выполнять задачи, такие как: загрузка фильма, сортировка фильмов в библиотеке, выбор фильма на главной странице, просмотр информации о фильме.

Пользователь не имеет доступ к изменению\удалению фильмов из базы данных.

## 3.4 Диаграммы деятельности

Диаграмма вариантов деятельности, которая относится к варианту использования «Авторизация или регистрация» представлена на рисунке 9.



Рисунок 9 – Диаграмма вариантов деятельности «Авторизация или регистрация».

Диаграмма вариантов деятельности, которая относится к варианту использования «Загрузка фильмов» представлена на рисунке 10.

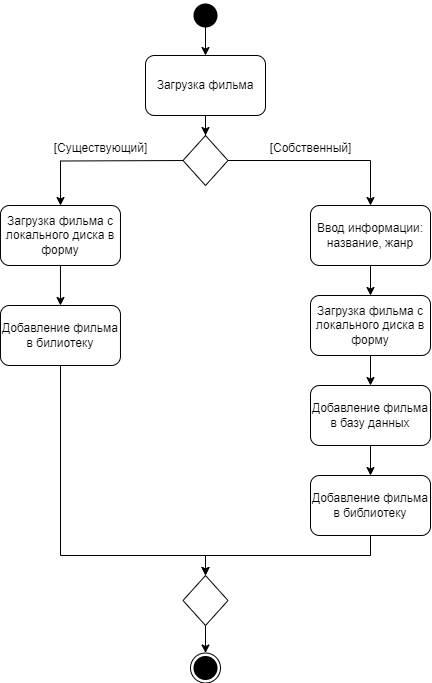


Рисунок 10 – Диаграмма вариантов деятельности «Загрузка фильмов».

Диаграмма вариантов деятельности, которая относится к варианту использования «Сортировка фильмов» представлена на рисунке 11.



Рисунок 11 – Диаграмма вариантов деятельности «Сортировка фильмов».

Диаграмма вариантов деятельности, которая относится к варианту использования «Выбор фильма и просмотр информации» представлена на рисунке 12.

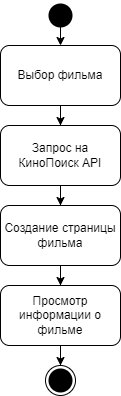


Рисунок 12 – Диаграмма вариантов деятельности «Выбор фильма и просмотр информации».

Данные диаграммы графически описывают функционал разрабатываемого программного обеспечения.

## 3.5 Проектирование макетов пользовательского интерфейса

Пользовательский интерфейс веб-приложения был спроектирован в веб-приложение Figma.

При входе в приложение открывается страница авторизации, которая предоставлена на рисунке 13.

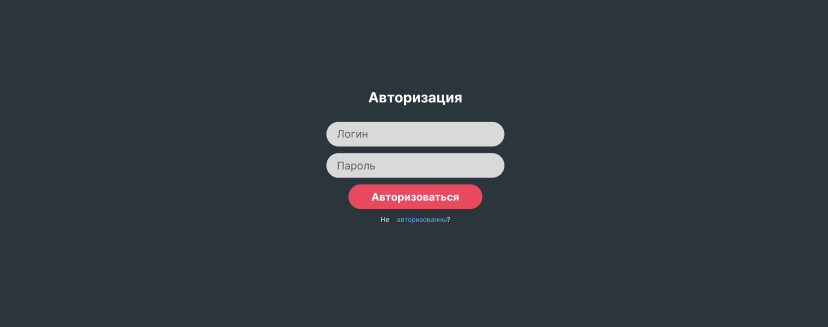


Рисунок 13 – Страница авторизации

На данной странице видно, что имеются поля для ввода «Логина» и «Пароля» пользовательского для входа в приложение. Также имеется кнопка «Авторизоваться» при нажатии которого происходит вход в приложение. Снизу кнопки имеется небольшой заголовок, который спрашивает пользователя, зарегистрирован он или нет, при нажатии на его открывается страница регистрации, которая предоставлена на рисунке 14.

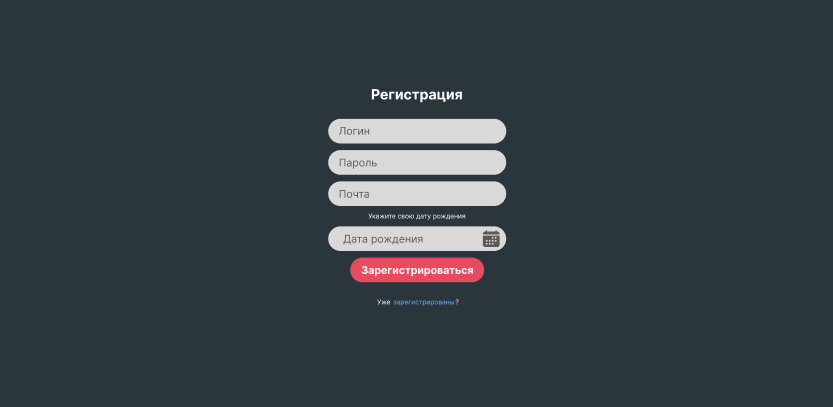


Рисунок 14 – Страница регистрации

На странице регистрации имеются поля для ввода «Логина», «Пароля», «Электронной почти» и «Даты рождения» пользователя. Имеется кнопка «Зарегистрироваться» после нажатия которой, пользователь регистрируется в веб-приложении. Также снизу от кнопки находится маленький заголовок, который спрашивает пользователя, зарегистрирован он или нет, при нажатии на его открывается страница авторизации.

Главная страница веб-приложения предоставлена на рисунке 15.

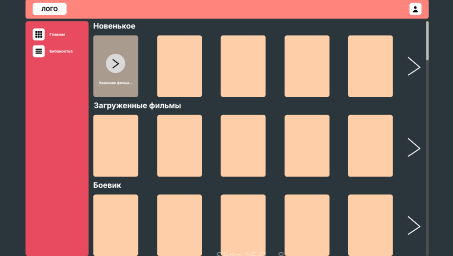


Рисунок 15 – Главная страница веб-приложения

На главной странице веб-приложения имеются слайдеры, в которых находится информация о фильмах, предложенные пользователю по новинкам, популярным жанрам, а также его загруженные фильмы.

При наведении на фильм, фон элемента становится затемнённым, что производит эффект того, что на данный элемент можно нажать. При нажатии на фильм будет открываться страница фильма.

Также на главной странице имеется шапка приложения, которая расположена сверху и сайдбар, который расположен слева. В шапке сверху находится изображение логотипа приложения, а справа иконка профиля, при нажатии на данную иконку, пользователя будет перебрасывать на страницу профиля пользователя, чтобы посмотреть информация о себе. В сайдбаре слева имеется две вкладки, такие как «Главная» и «Библиотека». При нажатии на вкладку «Главная» будет открываться главная страница приложения, а при нажатии на вкладку «Библиотека» будет открываться страница библиотеки фильмов пользователя, которая предоставлена на рисунке 16.

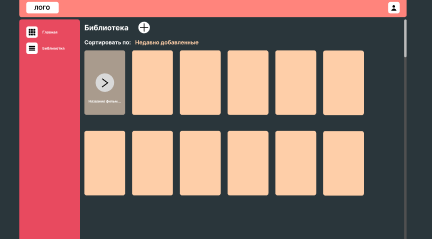


Рисунок 16 – Страница библиотеки веб-приложения

На странице библиотеки находится сетка фильмов, которые загрузил пользователь. Под заголовком странице находится подзаголовок для сортировки загруженных пользователем фильмов, при нажатии на который будет открываться список, по которому можно отсортировать фильм.

Также сверху находится кнопка в виде плюса, при нажатии на которую будет открываться главный поп-ап (всплывающее окно) для загрузки фильма, который предоставлен на рисунке 17.



Рисунок 17 – Главный поп-ап загрузки фильма

Данный поп-ап содержит в себе две кнопки «Существующий» и «Свой».

При нажатии на кнопку «Существующий» будет открываться поп-ап для загрузки существующего фильма.

Существующий фильм – фильм, который имеется в киноиндустрии.

При нажатии на кнопку «Свой» будет открываться поп-ап для загрузки собственного фильма.

Собственный фильм – фильм, который отсутствует в киноиндустрии, который записал сам пользователь.

Интерфейс поп-апа для загрузки существующего фильма предоставлен на рисунке 18.

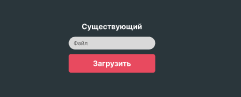


Рисунок 18 – Поп-ап загрузки существующего фильма

В данном поп-апе имеется поле «Файл», куда пользователь загружает фильм, и кнопка «Загрузить», после нажатия на эту кнопку фильм отправляется в базу данных и отображается на странице библиотеки пользователя.

Интерфейс поп-апа для загрузки собственного фильма представлен на рисунке 19.

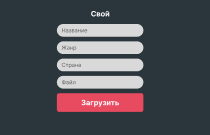


Рисунок 19 – Поп-ап загрузки собственного фильма

В данном поп-апе имеется поля «Название», «Жанр», «Страна» и «Файл» для ввода информации о фильме. Также имеется кнопка «Загрузить».

Страница профиля предоставлена на рисунке 20.

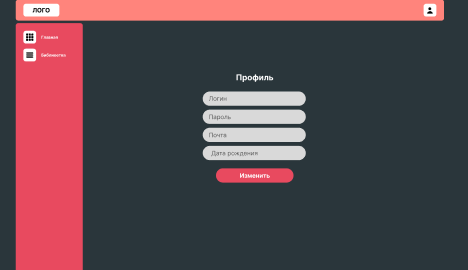


Рисунок 20 – Страница профиля

На странице профиля имеются поля «Логин», «Пароль» - поле, где пароль не отображается, чтобы обезопасить пользователя, «Почта» и «Дата рождения». Эти поля содержат информацию о пользователе, которую он вводил при регистрации.

Интерфейс страницы фильма предоставлен на рисунке 21.

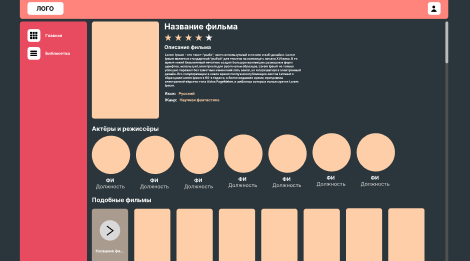


Рисунок 21 – Страница фильма

На странице фильма видна информация о фильме: название, рейтинг, описание фильма, фотография, язык и жанр. Под информацией о фильме имеется слайдер актёров и режиссёров, которые принимали участие в съёмке этого фильма. Под слайдером актёров и режиссёров находится слайдер подобных фильмов.

4. РАЗРАБОТКА ПРИЛОЖЕНИЯ

## 4.1 Инструментальные средства разработки

Figma - это инструмент для проектирования макета пользовательского интерфейса (UI) [8]. Его основными плюсами являются:

* Возможность работы в реальном времени с другими разработчиками, дизайнерами и заказчиками.
* Удобная система комментариев и обратной связи.
* Простота использования и интуитивный интерфейс.
* Возможность прототипирования и создания интерактивных макетов.
* Широкие возможности для создания и редактирования векторной графики.

Помимо этого, Figma предоставляет различные плагины и интеграции, что делает его универсальным инструментом для работы с дизайном интерфейса.

Выбор Figma вместо Photoshop обусловлен тем, что Figma является онлайн-инструментом, который позволяет сотрудничать с другими членами команды в режиме реального времени. Также, Figma предоставляет больше возможностей для прототипирования и интерактивности, что может быть важным при разработке пользовательского интерфейса.

React - это популярная библиотека JavaScript для разработки фронтенд-части приложений [9]. Основные преимущества React:

* Компонентный подход, который упрощает разделение интерфейса на независимые части.
* Виртуальная DOM, которая позволяет оптимизировать производительность и улучшить обновление компонентов.
* Широкая поддержка и большое сообщество разработчиков.
* Использование JSX - синтаксического расширения JavaScript, что упрощает создание компонентов.

Имеются аналоги React, такие как Angular и Vue.JS.

Однако, React был выбран вместо Angular и Vue.JS по следующим причинам:

* Имеет более активное сообщество разработчиков и большую пользовательскую базу, что обеспечивает большую поддержку и множество ресурсов для изучения.
* Является частью экосистемы инструментов от Facebook, включая React Native для разработки мобильных приложений. Это позволяет повторно использовать код для разработки приложений на разных платформах.
* Предоставляет мощный и эффективный подход к управлению состоянием приложения с помощью Redux или MobX.

Выбор React обусловлен его популярностью и широкой поддержкой в разработческом сообществе. Также, React имеет хорошую экосистему, включая библиотеки и инструменты для разработки, что делает его предпочтительным выбором для разработки фронтенд-части.

Node.js и фреймворк Express - это инструменты для разработки серверной части приложений [10]. Основные преимущества:

* Node.js позволяет использовать JavaScript для разработки серверной части, что облегчает переиспользование кода и ускоряет разработку.
* Express - это легковесный фреймворк, который предоставляет простой и интуитивно понятный интерфейс для разработки серверных приложений.
* Большое сообщество разработчиков и обширный выбор плагинов и модулей.
* Высокая производительность и масштабируемость.

Имеются аналоги Node.JS и его фреймворка Express, такие как Python (Django, Flask), Ruby. Но выбор Node.JS и Express основан на их простоте использования и распространенности в разработческом сообществе. Они также предлагают хорошую производительность и масштабируемость для разработки серверных приложений.

PostgreSQL - это реляционная база данных (СУБД), которая предлагает множество преимуществ:

* Открытый и бесплатный исходный код.
* Широкий набор функций и возможностей для работы с данными.
* Высокая производительность и надежность.
* Поддержка транзакций и целостности данных.
* Хорошая масштабируемость и поддержка больших объемов данных [11].

Выбор PostgreSQL обусловлен его богатыми возможностями и широкой поддержкой в разработческом сообществе. Он также имеет хорошую совместимость с другими инструментами разработки, такими как Node.js и Express.

## 4.2 База данных

Перед началом разработки приложения была создана физическая модель базы данных в СУБД PostgreSQL, показана на рисунке 22.

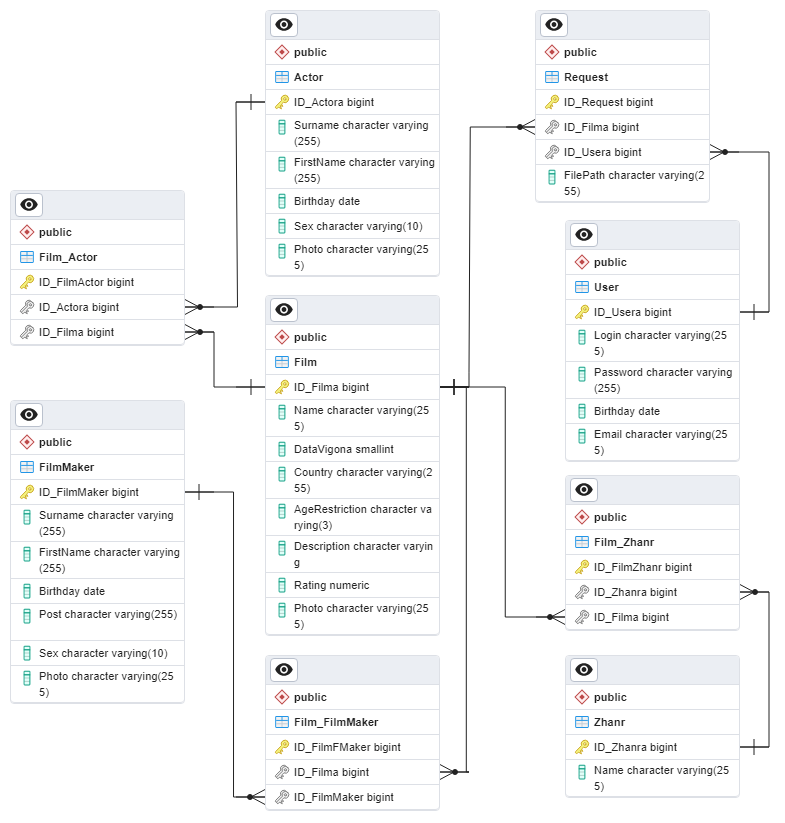


Рисунок 22 – Физическая модель базы данных

Физическая модель базы данных определяет способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Инициализация базы данных описано в приложении А.

## 4.3 Разработка приложения

При запуске приложения React открывает index.js файл, в котором запускается компонент App.js, в этом компоненте App, в котором прописаны маршруты для страниц сайта, создаётся компонент RouterProvider.

Маршруты сайта прописаны в компоненте router.js, которые создают массив данных, в котором хранится маршрут и соответсвенный компонент.

При роутинге /login пользователь переходит на страницу авторизации, которая показана на рисунке 23.

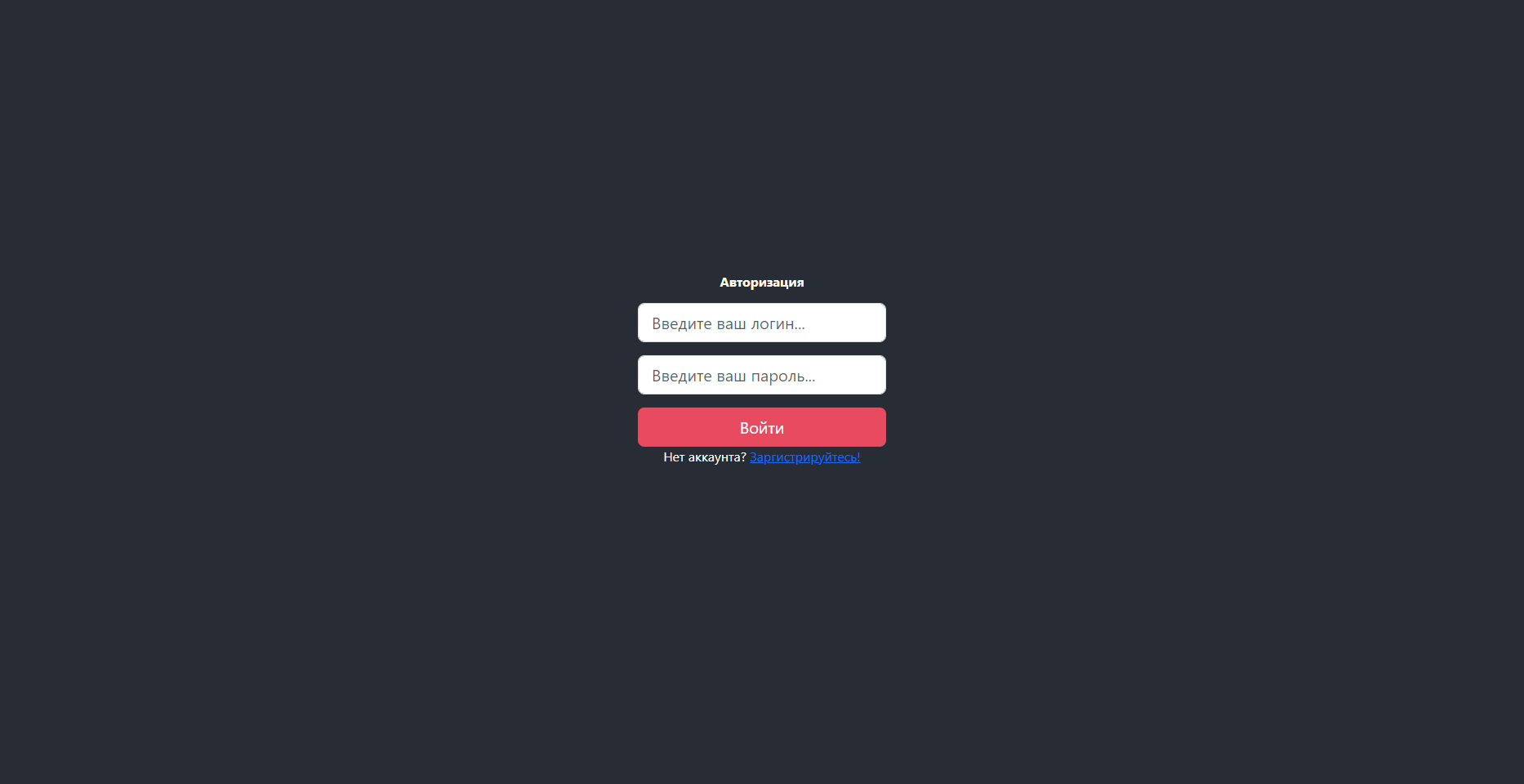


Рисунок 23 – Страница авторизации

На странице авторизации находятся поля «Логин» и «Пароль», и кнопка «Войти», а также кнопка «Зарегистрируйтесь!». К кнопке «Войти» привязана функция click, которая вызывается при нажатии на неё.

В функции click отправляются данные на сервер. На сервере вызывается контроллер userController с асинхронной функцией login.

В контроллере userController созданы асинхронные функции registration и login.

Функция login сверяет значения, поступившие с клиентской части, с данными из базы данных.

Модель базы данных описана в файле models.js со всеми таблицами и типами данных, а также со связями и первичными ключами. Подключение базы данных происходит в файле db.js.

Если же данные в функции login совпали, то авторизация была выполнена. Если же данные не совпали, то сервер отправляет ошибку, что пользователь не найден и на клиентской части выводится всплывающее окно с ошибкой о неправильности ввода логина или пароля.

Пример ошибки показан на рисунке 24.

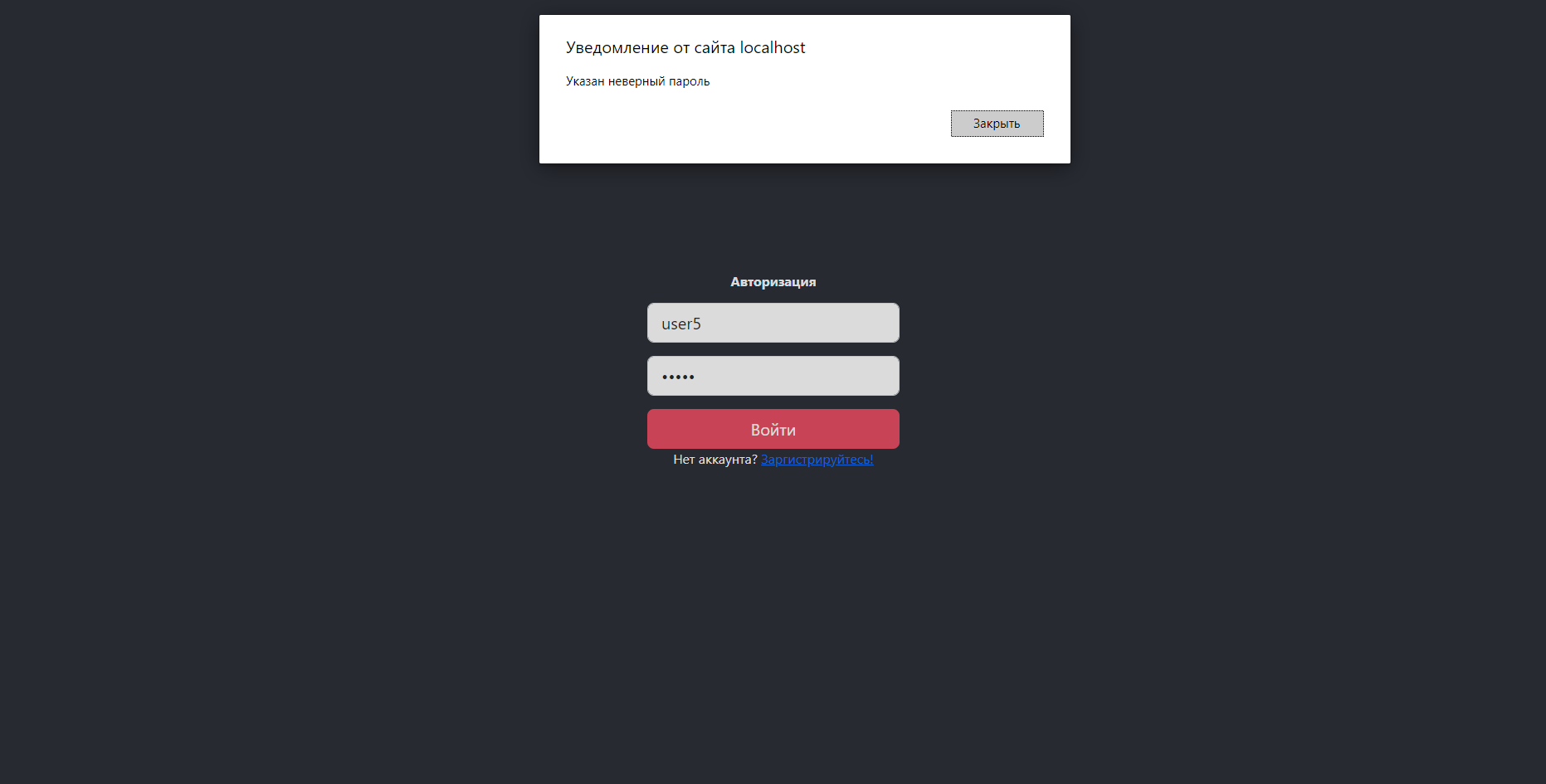


Рисунок 24 – Ошибка авторизации

Кнопка «Зарегистрируйтесь!», на страницу авторизации, генерирует переход на роутинг /registration через хук useNavigate. Страница регистрации показана на рисунке 25.

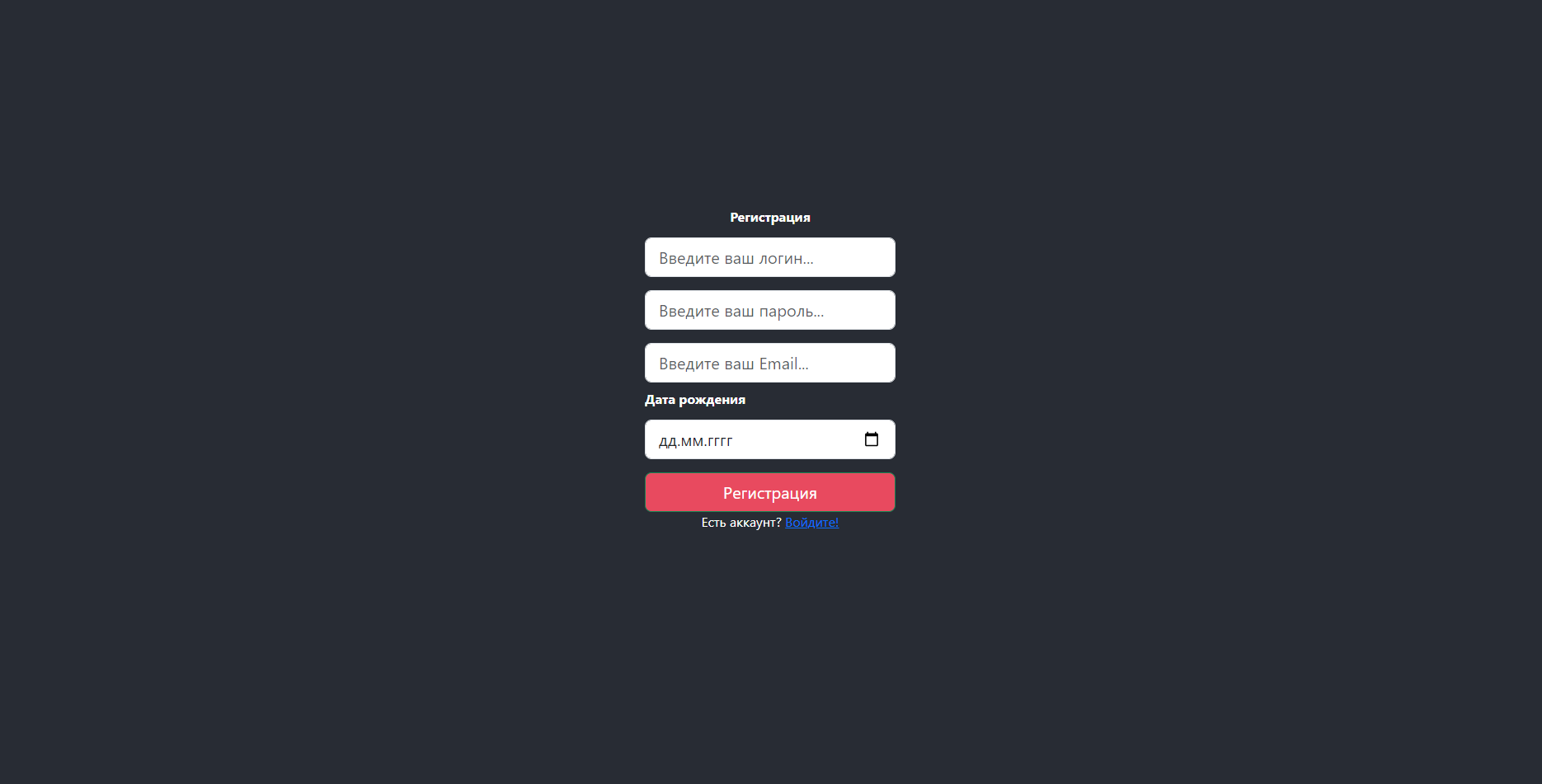


Рисунок 25 – Страница регистрации

На странице регистрации находятся поля «Логин», «Пароль», «Электронная почта» и «Дата рождения», а также кнопки «Регистрация» и «Войдите!». К кнопке «Регистрация» привязана функция click, которая вызывается при нажатии на неё.

В функции click отправляются данные на сервер. На сервере вызывается контроллер userController с описанной асинхронной функцией registration.

Функция registration создаёт нового пользователя в таблице User в базе данных.

Если же данные в функции registration совпали, с данными из базы данных, то на сервере отправляется ошибка, что такой пользователь уже зарегистрирован и на клиентской части выводится всплывающее окно с ошибкой. Если же данные не совпали, то сервер создаёт нового пользователя через User.create и данные этого пользователя записываются в базу данных.

На клиентской части на странице регистрации было реализовано поле с типом date, которая помогает пользователю выбрать нужную дату, а не вводить её руками. Пример формы показан на рисунке 26.

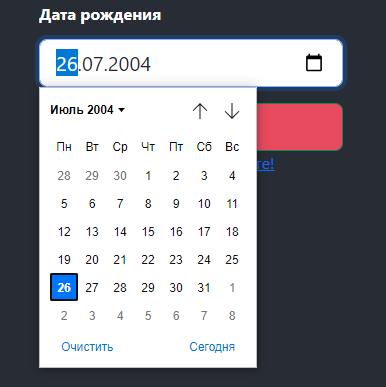


Рисунок 26 – Форма календаря в поле «Дата рождения»

На странице регистрации при нажатии на кнопку «Регистрация» происходит запрос на URL роутинга /login, то есть приложение с помощью useNavigate переходит на страницу авторизации.

На странице авторизации, после нажатия на кнопку «Войти», если же условия проверки данных на сервере выполнены, то с помощью useNavigate происходит запрос на URL роутинга /main, приложение переходит на главную страницу приложения. Главная страница приложения показана на рисунке 27.



Рисунок 28 – Главная страница приложения

На главной странице видно шапку и сайдбар. Шапка и сайдбар созданы в компоненте MainLayout, который является родительским роутингом для страниц, таких как «Библиотека» (с роутингом /library), «Главная» (/main), «Страница фильма» (/movie) и «Профиль» (/profile).

Шапка реализована вспомогательными компонентами из набора инструментов Bootstrap. Сайдбар реализован простой вёрсткой, состоящий из таких элементов, как «div» и «a», а также иконки «svg».

В компоненте MainLayout имеется хук useEffect, которая возникает при загрузке страницы. Данная функция проверяет, авторизован ли пользователь или нет. Если пользователь не авторизован, то есть не получили токен с сервера, то приложение переходит на страницу авторизации. Данная функция реализована для того, чтобы неавторизованный пользователь не мог находиться в приложении.

В шапке и сайдбаре реализованы кнопки в виде иконок, для каждой из них имеются функции.

При нажатии на кнопку профиля в правом верхнем углу шапки срабатывает функция profileClick, которая с помощью useNavigate делает URL запрос на роутинг /profile и переходит по данному роутингу на страницу профиля пользователя.

Страница профиля показана на рисунке 29.

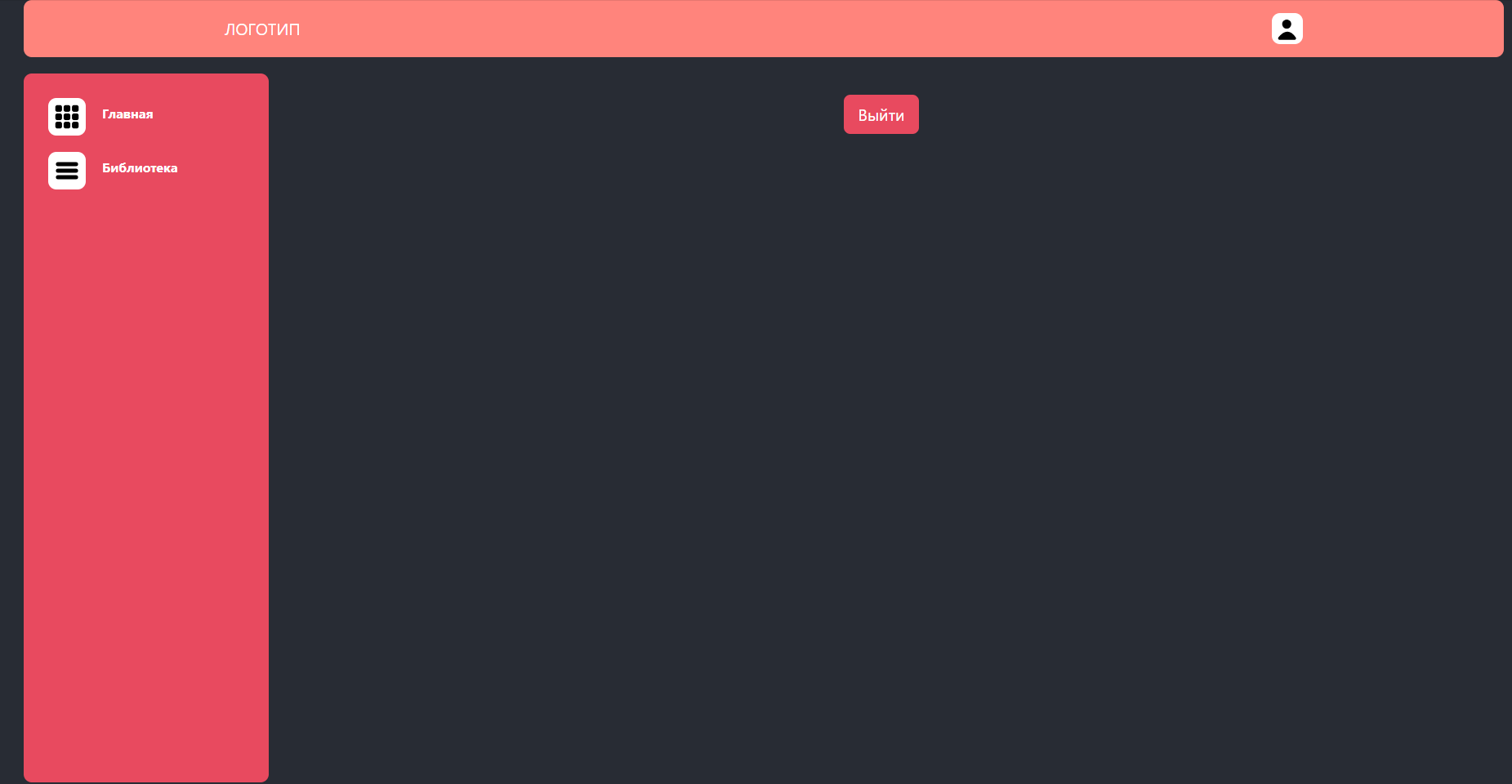


Рисунок 29 – Страница профиля

На странице профиля имеется кнопка «Выйти» на которую привязана функция logout.

Функция logout удаляет из локального хранилища токен пользователя и пользователь разлогинивается из приложения. Из-за того, что пользователь разлогинился, его переводит на страницу авторизации с помощью useNavigate, который делает URL запрос на роутинг /login.

В шапке на кнопку «Логотип» привязана функция mainClick.

Функция mainClick делает URL запрос на роутинг /main с помощью useNavigate и приложение переводит на главную страницу приложения.

В сайдбаре имеются две кнопки с изображениями «Главная» и «Библиотека», к которым привязаны функции mainClick для кнопки «Главная» и функция libraryClick для кнопки «Библиотека».

Функция libraryClick срабатывает после нажатия на кнопку, делает URL запрос на роутинг /library с помощью useNavigate и приложение переводит на страницу библиотеки пользователя. Страница библиотеки показана на рисунке 30.



Рисунок 30 – Страница библиотеки пользователя

На странице библиотеки пользователя будут фильмы, которые загрузит пользователь.

Клиентская часть приложения описана в приложении Б.

Серверная часть приложения описана в приложении В.

5. ТЕСТИРОВАНИЕ

## 5.1 Разработка тест-требований

Для создания приложения были разработаны следующие тест-требования:

1. Проверка входа в приложение:
2. проверить, что при неправильном вводе логина и пароля приложение отображает всплывающее окно с сообщением о том, что неправильно введены данные, а то есть логин или пароль;
3. проверить, что при правильном вводе логина и пароля происходит переход на главную страницу приложения.
4. Проверка регистрации в приложении:
5. проверить, что при некорректном вводе логина, пароля, электронной почты и даты рождения приложение отображает всплывающее окно с сообщением о том, что некорректно введены данные, а то есть логин, пароль, электронная почта или дата рождения;
6. проверить, что при корректном вводе логина, пароля, электронной почты и даты рождения происходит переход на страницу авторизации.
7. Проверка кнопок в шапке приложения:
8. проверить, что при нажатии на кнопку «Логотип» происходит переход на главную страницу приложения;
9. проверить, что при нажатии на кнопку «Профиль» происходит переход на страницу профиля пользователя.
10. Проверка кнопок в сайдбаре приложения:
11. проверить, что при нажатии на кнопку «Главная» происходит переход на главную страницу приложения;
12. проверить, что при нажатии на кнопку «Библиотека» происходит переход на страницу библиотеки пользователя.
13. Проверка страницы профиля:
14. проверить, имеется ли вывод информации о пользователе;
15. проверить, что при нажатии на кнопку «Выйти» происходит переход на страницу авторизации.
16. Проверка главной страницы:
17. проверить, что на главной странице имеются слайдеры фильмов по популярным жанрам и новинкам, а также загруженным фильмам пользователя;
18. проверить, что при нажатии на карточку фильма происходит переход на страницу фильма.
19. Проверка страницы фильма:
20. проверить, что на странице выводится информация о фильме, а то есть: название, описание, жанр, рейтинг, страна и год.
21. Проверка страницы библиотеки:
22. проверить, что на странице выводятся карточки фильмов, которые загрузил пользователь;
23. проверить, что при нажатии на сортировку и при выборе вида сортировки, а то есть: по жанрам, по актёрам, по стране, по году выпуска, происходит сортировка фильмов;
24. Проверка работы добавления фильма:
25. проверить, что при нажатии на кнопку «Добавить фильм» приложение открывает всплывающее окно с выбором, какой фильм загрузить, а то есть: «Существующий» и «Свой»;
26. Проверка работы добавления существующего фильма:
27. проверить, что при нажатии на кнопку «Существующий» приложение открывает всплывающее окно, где имеется поле «Файл» и кнопка «Загрузить»;
28. проверить, что при некорректной загрузке файла приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма;
29. проверить, что при корректной загрузке файла фильм добавляется на страницу библиотеки пользователя.
30. Проверка работы добавления своего фильма:
31. проверить, что при нажатии на кнопку «Свой» приложение открывает всплывающее окно, где имеется четыре поля, а то есть: название фильма, жанр, страна, файл и кнопка «Загрузить»;
32. проверить, что при некорректной загрузке файла приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма;
33. проверить, что при корректной загрузке файла фильм добавляется на страницу библиотеки пользователя.

## 5.2 Разработка тест-плана

После составленных тест-требований, необходимо описать тест-план. Он необходим для обеспечения чёткости ожидаемых результатов в процессе тестирования разработанного приложения и понимания, какие данные и куда необходимо вводить.

**Тестовый пример 1.**

Номер тест-требования 1a.

Описание требования: в данном тесте проверяется, что при вводе неправильного логина или пароля, приложение открывает всплывающее окно с сообщением о том, что неправильно введены данные, а то есть логин или пароль.

Входные данные:

* логин: user1;
* пароль: user2.

Ожидаемые выходные данные: при вводе некорректных данных, приложение должно открыть всплывающее окно с сообщением, о том, что данные были введены неправильно.

Сценарий:

* запустить приложение;
* заполнить поля «Логин» и «Пароль»;
* нажать на кнопку «Войти».

**Тестовый пример 2.**

Номер тест-требования 1b.

Описание теста: в данном тесте проверяется, что при вводе правильного логина и пароля, происходит переход на главную страницу.

Входные данные:

* логин: user1;
* пароль: user1.

Ожидаемые выходные данные: при вводе логина и пароля зарегистрированного пользователя, приложение должно перевести на главную страницу приложения.

Сценарий:

* запустить приложение;
* заполнить поля «Логин» и «Пароль»;
* нажать на кнопку «Войти».

**Тестовый пример 3.**

Номер тест-требования 2a.

Описание требования: в данном тесте проверяется, что при некорректном вводе логина, пароля, электронной почты и даты рождения, приложение отображает всплывающее окно с сообщением о некорректном вводе данных.

Входные данные:

* логин: user2;
* пароль: user2;
* электронная почта: user2@mail.ru;
* дата рождения: 26 июля.

Ожидаемые выходные данные: при некорректном вводе данных, приложение должно отобразить всплывающее окно с сообщением о некорректности введенных данных.

Сценарий:

* запустить приложение;
* заполнить поля «Логин», «Пароль», «Электронная почта» и «Дата рождения»;
* нажать на кнопку «Зарегистрироваться».

**Тестовый пример 4.**

Номер тест-требования 2b.

Описание требования: в данном тесте проверяется, что при корректном вводе логина, пароля, электронной почты и даты рождения на странице регистрации, происходит переход на страницу авторизации.

Входные данные:

* логин: user4;
* пароль: user4;
* электронная почта: user4@mail.ru;
* дата рождения: 26.07.2004.

Ожидаемые выходные данные: при корректном вводе данных, приложение должно перейти на страницу авторизации.

Сценарий:

* запустить приложение;
* нажать на кнопку «Зарегистрируйтесь!»;
* заполнить поля «Логин», «Пароль», «Электронная почта» и «Дата рождения»;
* нажать на кнопку «Зарегистрироваться».

**Тестовый пример 5.**

Номер тест-требования 3a.

Описание требования: в данном тесте проверяется, что при нажатии на кнопку "Логотип" в шапке, происходит переход на главную страницу приложения.

Ожидаемые выходные данные: при нажатии на кнопку "Логотип" в шапке, приложение должно перейти на главную страницу приложения.

Сценарий:

* запустить приложение;
* нажать на кнопку "Логотип" в шапке.

**Тестовый пример 6.**

Номер тест-требования 3b.

Описание требования: в данном тесте проверяется, что при нажатии на кнопку "Профиль" в шапке, происходит переход на страницу профиля пользователя.

Ожидаемые выходные данные: при нажатии на кнопку "Профиль" в шапке, приложение должно перейти на страницу профиля пользователя.

Сценарий:

* запустить приложение;
* нажать на кнопку "Профиль" в шапке.

**Тестовый пример 7.**

Номер тест-требования 4a.

Описание требования: в данном тесте проверяется, что при нажатии на кнопку "Главная" в сайдбаре, происходит переход на главную страницу приложения.

Ожидаемые выходные данные: при нажатии на кнопку "Главная" в сайдбаре, приложение должно перейти на главную страницу приложения.

Сценарий:

* запустить приложение;
* нажать на кнопку "Главная" в сайдбаре.

**Тестовый пример 8.**

Номер тест-требования 4b.

Описание требования: в данном тесте проверяется, что при нажатии на кнопку «Библиотека» в сайдбаре, происходит переход на страницу библиотеки пользователя.

Ожидаемые выходные данные: при нажатии на кнопку «Библиотека» в сайдбаре, приложение должно перейти на страницу библиотеки пользователя.

Сценарий:

* запустить приложение;
* нажать на кнопку "Библиотека" в сайдбаре.

**Тестовый пример 9.**

Номер тест-требования 5a.

Описание требования: в данном тесте проверяется, что при загруженной странице профиля имеется вывод информации о пользователе.

Ожидаемые выходные данные: на странице профиля отображается информация о пользователе.

Сценарий:

* запустить приложение;
* перейти на страницу профиля.

**Тестовый пример 10.**

Номер тест-требования 5b.

Описание требования: в данном тесте проверяется, что при нажатии на кнопку «Выйти» происходит переход на страницу авторизации.

Ожидаемые выходные данные: при нажатии на кнопку «Выйти» на странице профиля, происходит переход на страницу авторизации.

Сценарий:

* запустить приложение;
* перейти на страницу профиля;
* нажать на кнопку «Выйти».

**Тестовый пример 11.**

Номер тест-требования 6a.

Описание требования: в данном тесте проверяется, что на главной странице имеются слайдеры фильмов по популярным жанрам и новинкам, а также загруженным фильмам пользователя.

Ожидаемые выходные данные: на загруженной главной странице имеются слайдеры фильмов по популярным жанрам и новинкам, а также загруженным фильмам пользователя.

Сценарий:

* запустить приложение;
* перейти на главную страницу приложения.

**Тестовый пример 12.**

Номер тест-требования 6b.

Описание требования: в данном тесте проверяется, что при нажатии на карточку фильма происходит переход на страницу фильма.

Ожидаемые выходные данные: при нажатии на карточку фильма происходит переход на страницу фильма.

Сценарий:

* запустить приложение;
* перейти на главную страницу приложения;
* нажать на карточку любого фильма.

**Тестовый пример 13.**

Номер тест-требования 7a.

Описание требования: в данном тесте проверяется, что на странице фильма выводится информация о фильме, а то есть: название, описание, жанр, рейтинг, страна и год.

Ожидаемые выходные данные: на загруженной странице фильма выводится информация о фильме, а то есть: название, описание, жанр, рейтинг, страна и год.

Сценарий:

* запустить приложение;
* перейти на страницу фильма.

**Тестовый пример 14.**

Номер тест-требования 8a.

Описание требования: в данном тесте проверяется, что на странице библиотеки пользователя выводятся карточки фильмов, которые загрузил пользователь.

Ожидаемые выходные данные: на загруженной странице библиотеки выводятся карточки фильмов, которые загрузил пользователь.

Сценарий:

* запустить приложение;
* перейти на страницу библиотеки пользователя.

**Тестовый пример 15.**

Номер тест-требования 8b.

Описание требования: в данном тесте проверяется, что на странице библиотеки, при нажатии на кнопку сортировки и при выборе вида сортировки, а то есть: по жанрам, по актёрам, по стране, по году выпуска, происходит сортировка фильмов.

Ожидаемые выходные данные: на странице библиотеки, после нажатия на сортировку и при выборе вида сортировки, выводятся отсортированные фильмы.

Сценарий:

* запустить приложение;
* перейти на страницу библиотеки пользователя;
* нажать на кнопку сортировки;
* выбрать вид сортировки (по жанрам, по актёрам, по стране, по году выпуска).

**Тестовый пример 16.**

Номер тест-требования 9a.

Описание требования: в данном тесте проверяется, что на странице библиотеки, при нажатии на кнопку «Добавить фильм», приложение открывает всплывающее окно с выбором, какой фильм загрузить, а то есть: «Существующий» или «Свой».

Ожидаемые выходные данные: при нажатии на кнопку «Добавить фильм» приложение открывает всплывающее окно с выбором, какой фильм загрузить, а то есть: «Существующий» или «Свой».

Сценарий:

* запуск приложения;
* переход на страницу библиотеки;
* нажать на кнопку «Добавить фильм».

**Тестовые пример 17.**

Номер тест-требования 10a.

Описание требования: в данном тесте проверяется, что в всплывающем окне добавления фильма, при нажатии на кнопку «Существующий», приложение открывает ещё всплывающее окно, где имеется поле «Файл» и кнопка «Загрузить».

Ожидаемые выходные данные: при нажатии на кнопку «Существующий», в всплывающем окне добавления фильма, приложение открывает ещё всплывающее окно, где имеется поле «Файл» и кнопка «Загрузить».

Сценарий:

* запуск приложения;
* переход на страницу библиотеки;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Существующий».

**Тестовый пример 18.**

Номер тест-требования 10b.

Описание требования: в данном тесте проверяется, что в всплывающем окне при добавлении существующего фильма, при некорректной загрузке файла, приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма.

Входные данные:

* файл: пустое поле.

Ожидаемые выходные данные: в всплывающем окне при добавлении существующего фильма, при некорректной загрузке файла, приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма.

Сценарий:

* запуск приложения;
* переход на страницу библиотеки пользователя;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Существующий»;
* оставить пустое поле «Файл»;
* нажать на кнопку «Загрузить».

**Тестовый пример 19.**

Номер тест-требования 10c.

Описание требования: в данном тесте проверяется, что в всплывающем окне при добавлении существующего фильма, при корректной загрузке файла, фильм добавляется на страницу библиотеки пользователя.

Входные данные:

* файл: видеофайл с названием фильма (Трансформеры.avi).

Ожидаемые выходные данные: в сплывающем окне при добавлении существующего фильма, при корректной загрузке файла, фильм добавляется на страницу библиотеки пользователя.

Сценарий:

* запуск приложения;
* переход на страницу библиотеки пользователя;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Существующий»;
* заполнить поле «Файл»;
* нажать на кнопку «Загрузить»;
* переход на страницу библиотеки пользователя.

**Тестовые пример 20.**

Номер тест-требования 11a.

Описание требования: в данном тесте проверяется, что в всплывающем окне добавления фильма, при нажатии на кнопку «Свой», приложение открывает ещё всплывающее окно, где имеются поля «Название», «Жанр», «Страна», «Файл» и кнопка «Загрузить».

Ожидаемые выходные данные: при нажатии на кнопку «Свой», приложение открывает ещё всплывающее окно, где имеются поля «Название», «Жанр», «Страна», «Файл» и кнопка «Загрузить».

Сценарий:

* запуск приложения;
* переход на страницу библиотеки;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Свой».

**Тестовый пример 21.**

Номер тест-требования 11b.

Описание требования: в данном тесте проверяется, что в всплывающем окне при добавлении своего фильма, при некорректной загрузке файла, приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма.

Входные данные:

* название: Прогулка с собакой;
* жанр: боевик;
* страна: россия;
* файл: пустое поле.

Ожидаемые выходные данные: в всплывающем окне при добавлении своего фильма, при некорректной загрузке файла, приложение открывает всплывающее окно с сообщением о некорректной загрузке фильма.

Сценарий:

* запуск приложения;
* переход на страницу библиотеки пользователя;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Свой»;
* заполнить поля «Название», «Жанр», «Страна» и «Файл»;
* нажать на кнопку «Загрузить».

**Тестовый пример 22.**

Номер тест-требования 11c.

Описание требования: в данном тесте проверяется, что в всплывающем окне при добавлении своего фильма, при корректной загрузке файла, фильм добавляется на страницу библиотеки пользователя.

Входные данные:

* название: Прогулка с собакой;
* жанр: боевик;
* страна: россия;
* файл: видеофайл.

Ожидаемые выходные данные: в сплывающем окне при добавлении своего фильма, при корректной загрузке файла, фильм добавляется на страницу библиотеки пользователя.

Сценарий:

* запуск приложения;
* переход на страницу библиотеки пользователя;
* нажать на кнопку «Добавить фильм»;
* нажать на кнопку «Свой»;
* заполнить поля «Название», «Жанр», «Страна» и «Файл»;
* нажать на кнопку «Загрузить»;
* переход на страницу библиотеки пользователя.

Заключение

В заключении можно сказать, что на данный момент реализована авторизация и регистрация пользователя. Разработан основной пользовательский интерфейс с шапкой приложения и сайдбаром. Также была создана база данных.

Дальнейшей разработкой будет парсинг информации о фильмах из «КиноПоиск АПИ» и добавления новых фильмов пользователем. Также нужно реализовать сортировку фильмов в библиотеке пользователя и страницу профиля, где пользователь сможет просмотреть или изменить свою информацию.

В дальнейшем продумывается функционал для опытных пользователей, такой как поменять цвет интерфейса и поменять фон приложения.

В целом, разработка данного веб-приложения является многоступенчатым процессом, и на данный момент уже достигнуты значимые этапы. Однако, с анализом предметной области, можно сделать вывод о том, что проект имеет большой потенциал для дальнейшего развития и расширения функционала.

Список использованных источников

1. «Каталогизация» [Электронный ресурс] – Режим доступа: [Каталогизация (библиотеки и музеи) — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8_%D0%B8_%D0%BC%D1%83%D0%B7%D0%B5%D0%B8)#:~:text=%D0%9A%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F%20%D0%B2%20%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D1%87%D0%BD%D0%BE%D0%BC%20%D0%B4%D0%B5%D0%BB%D0%B5%20%D0%B8,%D0%BF.).) (дата обращения 20.09.2023)
2. «Plex» [Электронный ресурс] – Режим доступа: <https://www.plex.tv/>(дата обращения 29.09.2023)
3. «Kodi Media Player» [Электронный ресурс] – Режим доступа: <https://kodi.tv/> (дата обращения 29.09.2023)
4. «Stremio» [Электронный ресурс] – Режим доступа: <https://www.stremio.com/> (дата обращения 29.09.2023)
5. «All My Movies» [Электронный ресурс] – Режим доступа: <https://allmymovies.ru/> (дата обращения 29.09.2023)
6. «MyFilms» [Электронный ресурс] – Режим доступа: <https://vse-analogi.ru/app/myfilms>(дата обращения 29.09.2023)
7. «Extreme Movie Manager» [Электронный ресурс] – Режим доступа: <http://www.binaryworks.it/extrememoviemanager/index.aspx> (дата обращения 29.09.2023)
8. «Figma» [Электронный ресурс] – Режим доступа: <https://vc.ru/design/184941-figma-vse-chto-vam-nuzhno-znat> (дата обращения 07.11.2023).
9. «React» [Электронный ресурс] – Режим доступа: <https://react.dev> (дата обращения 07.11.2023).
10. «Node.JS» [Электронный ресурс] – Режим доступа: <https://habr.com/ru/companies/ruvds/articles/422893/> (дата обращения 07.11.2023).
11. «PostgreSQL» [Электронный ресурс] – Режим доступа: <https://www.postgresql.org/about/> (дата обращения 07.11.2023).

Приложение А

Инициализация базы данных

CREATE DATABASE videocatalog

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

COMMENT ON DATABASE videocatalog

IS 'Диплом';

Приложение Б

Главный index.js

import React, { createContext } from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import UserMovie from './movie/UserMovie';

export const Context = createContext(null)

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

      <Context.Provider value={{

        user: new UserMovie()

      }}>

        <App />

      </Context.Provider>

);

App.js

import React, { useContext, useEffect, useState } from 'react';

import { RouterProvider } from 'react-router-dom';

import './App.css';

import router from './router';

function App() {

  return (

    <>

      <RouterProvider router={router} />

    </>

  );

}

export default App;

Страница профиля

import React, { useContext } from 'react';

import { Button } from 'react-bootstrap';

import { useNavigate } from 'react-router-dom';

import { LOGIN\_ROUTE } from '../utils/consts';

import { Context } from '..';

const Profile = () => {

    const {user} = useContext(Context)

    const navigate = useNavigate();

    const logout = async () => {

        localStorage.removeItem('token')

        navigate(LOGIN\_ROUTE)

    }

    return (

        <div className="d-flex justify-content-center align-items-center">

            <Button onClick={logout} size='lg' style={{radius: 2000, backgroundColor: '#E84A5F', color: 'white'}} className='mt-3' variant={"outline-succes"}>

                        Выйти

            </Button>

        </div>

    )

}

export default Profile;

Страница авторизации

import React, { useContext, useState } from 'react';

import {Container, Form, Button} from 'react-bootstrap'

import Card from "react-bootstrap/Card"

import { NavLink, useNavigate } from 'react-router-dom';

import { MAIN\_ROUTE, REGISTRATION\_ROUTE } from '../utils/consts';

import { login } from '../http/userAPI'

import { Context } from '..';

import '../style/Login.css'

const Login = () => {

    const {user} = useContext(Context)

    const navigate = useNavigate();

    const [Login, setLogin] = useState('')

    const [Password, setPassword] = useState('')

    const click = async () => {

        try {

            let data;

            data = await login(Login, Password);

            console.log(data);

            user.setUser(data)

            console.log(user.user);

            user.setIsAuth(true)

            navigate(MAIN\_ROUTE)

        } catch (e) {

            alert(e.response.data.message)

        }

    }

    return (

        <Container

            className="d-flex justify-content-center align-items-center"

            style={{height: window.innerHeight - 54}}

        >

            <Card style={{width: 400, border: 0}} className="p-5">

                <h2 className="m-auto">Авторизация</h2>

                <Form className="d-flex flex-column">

                        <Form.Control

                            size='lg'

                            className="mt-3"

                            placeholder="Введите ваш логин..."

                            value={Login}

                            onChange={e => setLogin(e.target.value)}

                        />

                        <Form.Control

                            size='lg'

                            className="mt-3"

                            type="password"

                            placeholder="Введите ваш пароль..."

                            value={Password}

                            onChange={e => setPassword(e.target.value)}

                        />

                    <Button onClick={click} size='lg' style={{radius: 2000, backgroundColor: '#E84A5F', color: 'white'}} className='mt-3' variant={"outline-succes"}>

                        Войти

                    </Button>

                    <div style={{textAlign: 'center'}}>

                        Нет аккаунта? <NavLink to={REGISTRATION\_ROUTE}>Заргистрируйтесь!</NavLink>

                    </div>

                </Form>

            </Card>

        </Container>

    )

}

export default Login;

Страница регистрации

import React, { useContext, useState } from 'react';

import {Container, Form, Button} from 'react-bootstrap'

import Card from "react-bootstrap/Card"

import { NavLink, useNavigate } from 'react-router-dom';

import { LOGIN\_ROUTE } from '../utils/consts';

import { registration } from '../http/userAPI';

import { Context } from '..';

import '../style/Login.css'

const Registration = () => {

    const {user} = useContext(Context)

    const navigate = useNavigate()

    const [Login, setLogin] = useState('')

    const [Password, setPassword] = useState('')

    const [Email, setEmail] = useState('')

    const [Birthday, setBirthday] = useState('')

    const click = async () => {

        console.log("Login:", Login);

        console.log("Password:", Password);

        console.log("Email:", Email);

        console.log("Birthday:", Birthday);

        try{

            let data;

            data = await registration(Login, Password, Email, Birthday);

            user.setUser(user)

            user.setIsAuth(false)

            navigate(LOGIN\_ROUTE)

        } catch (e) {

            alert(e.response.data.message)

        }

    }

    const formatBirthday = (value) => {

        // Функция для форматирования даты в формат "гггг-мм-дд"

        const regex = /^(\d{0,4})(\d{0,2})(\d{0,2})$/;

        const matches = value.match(regex);

        if (matches) {

            const formattedDate = matches

                .slice(1, 4) // Отбрасываем полное совпадение

                .map((match, index) => (index === 2 ? match.padStart(2, '0') : match)) // Дополняем день до двух цифр

                .join('-'); // Формируем дату

            return formattedDate;

        }

        return value;

    };

    const handleBirthdayChange = (e) => {

        // Обработчик изменения значения в поле даты

        const formattedDate = formatBirthday(e.target.value);

        setBirthday(formattedDate);

    };

    return (

        <Container

            className="d-flex justify-content-center align-items-center"

            style={{height: window.innerHeight - 54}}

        >

            <Card style={{width: 400, border: 0}} className="p-5">

                <h2 className="m-auto">Регистрация</h2>

                <Form className="d-flex flex-column">

                        <Form.Control

                            className="mt-3"

                            size='lg'

                            placeholder="Введите ваш логин..."

                            value={Login}

                            onChange={e => setLogin(e.target.value)}

                        />

                        <Form.Control

                            className="mt-3"

                            size='lg'

                            type="password"

                            placeholder="Введите ваш пароль..."

                            value={Password}

                            onChange={e => setPassword(e.target.value)}

                        />

                        <Form.Control

                            size='lg'

                            type='email'

                            className="mt-3"

                            placeholder="Введите ваш Email..."

                            value={Email}

                            onChange={e => setEmail(e.target.value)}

                        />

                        <h2 style={{marginTop: 10, marginBottom: 0, marginLeft: 0, fontSize: 16}}>Дата рождения</h2>

                        <Form.Control

                            size='lg'

                            type='date'

                            className="mt-3"

                            placeholder="Введите вашу дату рождения..."

                            value={Birthday}

                            onChange={handleBirthdayChange}

                        />

                    <Button onClick={click} size='lg' style={{radius: 2000, backgroundColor: '#E84A5F', color: 'white'}} className='mt-3' variant={"outline-success"}>

                        Регистрация

                    </Button>

                    <div style={{textAlign: 'center'}}>

                        Есть аккаунт? <NavLink to={LOGIN\_ROUTE}>Войдите!</NavLink>

                    </div>

                </Form>

            </Card>

        </Container>

    )

}

export default Registration;

Шапка и сайдбар

import React, { useContext, useEffect } from 'react'

import { Outlet, useNavigate } from 'react-router-dom';

import { Context } from '..';

import { LOGIN\_ROUTE } from '../utils/consts';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

import { LIBRARY\_ROUTE, MAIN\_ROUTE, PROFILE\_ROUTE } from '../utils/consts';

import '../style/MainLayout.css'

function MainLayout() {

    const {user} = useContext(Context)

    const navigate = useNavigate()

    useEffect(() => {

        console.log(user.isAuth)

        if(!user.isAuth){

            navigate(LOGIN\_ROUTE)

        }

      }, [user.isAuth]);

    const profileClick = async () => {

        navigate(PROFILE\_ROUTE)

    }

    const mainClick = async () => {

        navigate(MAIN\_ROUTE)

    }

    const libraryClick = async () => {

        navigate(LIBRARY\_ROUTE)

    }

    return (

        <>

            <div class="header">

                <Navbar style={{ backgroundColor: '#FF847C', borderRadius: 10, paddingTop: 15, paddingBottom: 15, paddingLeft: 5, paddingRight: 5 }}>

                    <Container style={{ padding: 0 }}>

                        <Navbar.Brand onClick={() => mainClick()} style={{ color: 'white' }}>ЛОГОТИП</Navbar.Brand>

                        <Nav className="justify-contend-end">

                            <div onClick={profileClick} class="cursor-pointer">

                                <svg xmlns="http://www.w3.org/2000/svg" width="38" height="38" viewBox="0 0 38 38" fill="none">

                                    <g clip-path="url(#clip0\_70\_125)">

                                        <rect x="-6" y="-6" width="50" height="50" rx="10" fill="white" />

                                        <path d="M23.75 13.4583C23.75 14.928 23.1661 16.3376 22.1268 17.3768C21.0876 18.4161 19.678 19 18.2083 19C16.7386 19 15.329 18.4161 14.2897 17.3768C13.2505 16.3376 12.6666 14.928 12.6666 13.4583C12.6666 11.9886 13.2505 10.579 14.2897 9.53974C15.329 8.50048 16.7386 7.91663 18.2083 7.91663C19.678 7.91663 21.0876 8.50048 22.1268 9.53974C23.1661 10.579 23.75 11.9886 23.75 13.4583Z" fill="black" />

                                        <path d="M27.9142 31.6667H9.41607C8.77957 31.6667 8.42015 30.9684 8.81757 30.4713C10.583 28.2593 14.7092 23.75 19 23.75C23.3177 23.75 27.0164 28.3195 28.5427 30.5156C28.8911 31.0175 28.5269 31.6667 27.9157 31.6667H27.9142Z" fill="black" stroke="black" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" />

                                    </g>

                                    <defs>

                                        <clipPath id="clip0\_70\_125">

                                            <rect width="38" height="38" rx="10" fill="white" />

                                        </clipPath>

                                    </defs>

                                </svg>

                            </div>

                        </Nav>

                    </Container>

                </Navbar>

            </div>

            <div class="sidebar">

                <nav class="sidebar-nav">

                    <div onClick={() => mainClick()} class='main-div cursor-pointer'>

                        <svg xmlns="http://www.w3.org/2000/svg" width="38" height="38" viewBox="0 0 38 38" fill="none" class="svg-sidebar">

                            <path d="M10.8569 4.75H6.78617C5.66162 4.75 4.75 5.66162 4.75 6.78617V10.8569C4.75 11.9815 5.66162 12.8931 6.78617 12.8931H10.8569C11.9815 12.8931 12.8931 11.9815 12.8931 10.8569V6.78617C12.8931 5.66162 11.9815 4.75 10.8569 4.75Z" fill="black" />

                            <path d="M10.8569 14.9293H6.78617C5.66162 14.9293 4.75 15.8409 4.75 16.9654V21.0362C4.75 22.1607 5.66162 23.0723 6.78617 23.0723H10.8569C11.9815 23.0723 12.8931 22.1607 12.8931 21.0362V16.9654C12.8931 15.8409 11.9815 14.9293 10.8569 14.9293Z" fill="black" />

                            <path d="M10.8569 25.1069H6.78617C5.66162 25.1069 4.75 26.0185 4.75 27.1431V31.2138C4.75 32.3384 5.66162 33.25 6.78617 33.25H10.8569C11.9815 33.25 12.8931 32.3384 12.8931 31.2138V27.1431C12.8931 26.0185 11.9815 25.1069 10.8569 25.1069Z" fill="black" />

                            <path d="M21.0361 4.75H16.9654C15.8408 4.75 14.9292 5.66162 14.9292 6.78617V10.8569C14.9292 11.9815 15.8408 12.8931 16.9654 12.8931H21.0361C22.1607 12.8931 23.0723 11.9815 23.0723 10.8569V6.78617C23.0723 5.66162 22.1607 4.75 21.0361 4.75Z" fill="black" />

                            <path d="M31.2139 4.75H27.1431C26.0186 4.75 25.1069 5.66162 25.1069 6.78617V10.8569C25.1069 11.9815 26.0186 12.8931 27.1431 12.8931H31.2139C32.3384 12.8931 33.25 11.9815 33.25 10.8569V6.78617C33.25 5.66162 32.3384 4.75 31.2139 4.75Z" fill="black" />

                            <path d="M21.0361 14.9293H16.9654C15.8408 14.9293 14.9292 15.8409 14.9292 16.9654V21.0362C14.9292 22.1607 15.8408 23.0723 16.9654 23.0723H21.0361C22.1607 23.0723 23.0723 22.1607 23.0723 21.0362V16.9654C23.0723 15.8409 22.1607 14.9293 21.0361 14.9293Z" fill="black" />

                            <path d="M21.0361 25.1069H16.9654C15.8408 25.1069 14.9292 26.0185 14.9292 27.1431V31.2138C14.9292 32.3384 15.8408 33.25 16.9654 33.25H21.0361C22.1607 33.25 23.0723 32.3384 23.0723 31.2138V27.1431C23.0723 26.0185 22.1607 25.1069 21.0361 25.1069Z" fill="black" />

                            <path d="M31.2139 14.9293H27.1431C26.0186 14.9293 25.1069 15.8409 25.1069 16.9654V21.0362C25.1069 22.1607 26.0186 23.0723 27.1431 23.0723H31.2139C32.3384 23.0723 33.25 22.1607 33.25 21.0362V16.9654C33.25 15.8409 32.3384 14.9293 31.2139 14.9293Z" fill="black" />

                            <path d="M31.2139 25.1069H27.1431C26.0186 25.1069 25.1069 26.0185 25.1069 27.1431V31.2138C25.1069 32.3384 26.0186 33.25 27.1431 33.25H31.2139C32.3384 33.25 33.25 32.3384 33.25 31.2138V27.1431C33.25 26.0185 32.3384 25.1069 31.2139 25.1069Z" fill="black" />

                        </svg>

                        <h2>Главная</h2>

                    </div>

                    <div onClick={() => libraryClick()} class='main-div cursor-pointer'>

                        <svg xmlns="http://www.w3.org/2000/svg" width="38" height="38" viewBox="0 0 38 38" fill="none" class="svg-sidebar">

                            <path d="M7.91663 11.0833H30.0833M7.91663 19H30.0833M7.91663 26.9167H30.0833" stroke="black" stroke-width="5" stroke-linecap="round" stroke-linejoin="round" />

                        </svg>

                        <h2>Библиотека</h2>

                    </div>

                </nav>

            </div>

            <div class="main">

                <div class='containers'>

                    <Outlet />

                </div>

            </div>

        </>

    );

}

export default MainLayout;

Роутинг страниц

import { createBrowserRouter } from "react-router-dom";

import Login from "./pages/Login";

import Registration from "./pages/Registration";

import Main from "./pages/Main";

import Movie from "./pages/Movie"

import Library from "./pages/Library";

import Profile from "./pages/Profile";

import MainLayout from "./components/MainLayout";

import { LIBRARY\_ROUTE, LOGIN\_ROUTE, MAIN\_ROUTE, MOVIE\_ROUTE, PROFILE\_ROUTE, REGISTRATION\_ROUTE } from "./utils/consts";

const router = createBrowserRouter([

 {

   path: '/',

   element: <MainLayout/>,

   children: [

      {

         path: MOVIE\_ROUTE,

         element: <Movie/>

      },

      {

         path: MAIN\_ROUTE,

         element: <Main/>

      },

      {

        path: LIBRARY\_ROUTE,

        element: <Library />

      },

      {

        path: PROFILE\_ROUTE,

        element: <Profile />

      }

   ]

 },

 {

   path: LOGIN\_ROUTE,

   element: <Login/>

 },

 {

   path: REGISTRATION\_ROUTE,

   element: <Registration/>

 },

])

export default router;

Приложение В

Авторизация и регистрация

const {User} = require('../models/models')

const ApiError = require('../error/ApiError')

const bcrypt = require('bcrypt')

const jwt = require('jsonwebtoken')

const generateJwt = (ID\_Usera, Login) => {

    return jwt.sign(

        {ID\_Usera, Login},

         process.env.SECRET\_KEY,

        {expiresIn: '24h'})

}

class UserController {

    async registration(req, res) {

        const {Login, Password, Birthday, Email} = req.body

        if(!Login || !Password){

            return next(ApiError.badRequest('Неккоретный Логин или Пароль'))

        }

        const candidate = await User.findOne({where: {Login}})

        if(candidate){

            return next(ApiError.badRequest('Пользователь с таким Логином уже существует'))

        }

        const hashPassword = await bcrypt.hash(Password, 5)

        const user = await User.create({Login, Password: hashPassword, Birthday, Email})

        const token = generateJwt(user.ID\_Usera, user.Login)

        return res.json({token})

    }

    async login(req, res, next){

        const {Login, Password} = req.body

        const user = await User.findOne({where: {Login}})

        if(!user){

            return next(ApiError.internal('Пользователь не найден'))

        }

        let comparePassword = bcrypt.compareSync(Password, user.Password)

        if(!comparePassword){

            return next(ApiError.internal('Указан неверный пароль'))

        }

        const token = generateJwt(user.ID\_Usera, user.Login)

        return res.json({token})

    }

    async check(req, res, next){

        const token = generateJwt(req.user.ID\_Usera, req.user.Login)

        return res.json({token})

    }

    async getProfile(req,res){

        const {Login, Birthday, Email} = req.user

        return res.json({

            Login: Login,

            Birthday: Birthday,

            Email: Email,

        });

    }

}

module.exports = new UserController()

Роутинг авторизации и регистрации пользователя

const Router = require('express')

const router = new Router()

const userController = require('../controllers/userControllers')

const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration', userController.registration)

router.post('/login', userController.login)

router.get('/auth', authMiddleware, userController.check)

router.get('/profile', userController.getProfile)

module.exports = router

Подключение к базе данных

const {Sequelize} = require('sequelize');

module.exports = new Sequelize(

    process.env.DB\_NAME,

    process.env.DB\_USER,

    process.env.DB\_PASSWORD,

    {

        dialect: "postgres",

        host: process.env.DB\_HOST,

        port: process.env.DB\_PORT

    }

);

Описание модели базы данных

const sequelize = require('../db')

const {DataTypes} = require('sequelize')

const User = sequelize.define('Users', {

    ID\_Usera: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    Login: {type: DataTypes.STRING, unique: true},

    Password: {type: DataTypes.STRING},

    Birthday: {type: DataTypes.DATE},

    Email: {type: DataTypes.TEXT, unique: true}

})

const Zhanr = sequelize.define('Zhanrs', {

    ID\_Zhanra: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    Name: {type: DataTypes.STRING}

})

const Request = sequelize.define('Requests', {

    ID\_Request: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    ID\_Filma: {type: DataTypes.INTEGER},

    ID\_Usera: {type: DataTypes.INTEGER},

    FilePath: {type: DataTypes.STRING}

})

const Film\_Zhanr = sequelize.define('Film\_Zhanrs', {

    ID\_FilmaZhanr: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    ID\_Zhanra: {type: DataTypes.INTEGER},

    ID\_Filma: {type: DataTypes.INTEGER}

})

const Film\_FilmMaker = sequelize.define('Film\_FilmMakers', {

    ID\_FilmFMaker: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    ID\_Filma: {type: DataTypes.INTEGER},

    ID\_FilmMaker: {type: DataTypes.INTEGER}

})

const Film\_Actor = sequelize.define('Film\_Actors', {

    ID\_FilmActor: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    ID\_Actora: {type: DataTypes.INTEGER},

    ID\_Filma: {type: DataTypes.INTEGER}

})

const FilmMaker = sequelize.define('Film\_Makers', {

    ID\_FilmMaker: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    Surname: {type: DataTypes.STRING},

    FirstName: {type: DataTypes.STRING},

    Birthday: {type: DataTypes.DATE},

    Post: {type: DataTypes.STRING},

    Sex: {type: DataTypes.STRING},

    Photo: {type: DataTypes.STRING}

})

const Film = sequelize.define('Films', {

    ID\_Filma: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    Name: {type: DataTypes.STRING},

    DataVihoda: {type: DataTypes.INTEGER},

    Country: {type: DataTypes.STRING},

    AgeRestriction: {type: DataTypes.STRING},

    Description: {type: DataTypes.TEXT},

    Rating: {type: DataTypes.DOUBLE},

    Photo: {type: DataTypes.STRING}

})

const Actor = sequelize.define('Actors', {

    ID\_Actora: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

    Surname: {type: DataTypes.STRING},

    FirstName: {type: DataTypes.STRING},

    Birthday: {type: DataTypes.DATE},

    Sex: {type: DataTypes.STRING},

    Photo: {type: DataTypes.STRING}

})

User.hasMany(Request, {foreignKey: 'ID\_Usera'})

Request.belongsTo(User, {foreignKey: 'ID\_Usera'})

Zhanr.hasMany(Film\_Zhanr, {foreignKey: 'ID\_Zhanra'})

Film\_Zhanr.belongsTo(Zhanr, {foreignKey: 'ID\_Zhanra'})

Actor.hasMany(Film\_Actor, {foreignKey: 'ID\_Actora'})

Film\_Actor.belongsTo(Actor, {foreignKey: 'ID\_Actora'})

FilmMaker.hasMany(Film\_FilmMaker, {foreignKey: 'ID\_FilmMaker'})

Film\_FilmMaker.belongsTo(FilmMaker, {foreignKey: 'ID\_FilmMaker'})

Film.hasMany(Film\_Zhanr, {foreignKey: 'ID\_Filma'})

Film\_Zhanr.belongsTo(Film, {foreignKey: 'ID\_Filma'})

Film.hasMany(Request, {foreignKey: 'ID\_Filma'})

Request.belongsTo(Film, {foreignKey: 'ID\_Filma'})

Film.hasMany(Film\_Actor, {foreignKey: 'ID\_Filma'})

Film\_Actor.belongsTo(Film, {foreignKey: 'ID\_Filma'})

Film.hasMany(Film\_FilmMaker, {foreignKey: 'ID\_Filma'})

Film\_FilmMaker.belongsTo(Film, {foreignKey: 'ID\_Filma'})

module.exports = {

    User,

    Request,

    Film,

    Zhanr,

    Film\_Zhanr,

    Film\_Actor,

    Film\_FilmMaker,

    FilmMaker,

    Actor

}