Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
| *К защите допустить*: |
| Заведующая кафедрой ПОИТ |
| \_\_\_\_\_\_\_\_\_\_\_\_Н.В. Лапицкая |

Пояснительная записка

к дипломному проекту

на тему

**ПРОГРАММНОЕ СРЕДСТВО «оНЛАЙН-КИНОТЕАТР С ИНТЕЛЛЕКТУАЛЬНЫМ ПОМОЩНИКОМ» НА БАЗЕ ПЛАТФОРМЫ ASP.NET CORE**

БГУИР ДП 1-40 01 01 01 001 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А.В. Абраменко |
| Руководитель |  | Н.П. Можей |
| Консультанты: |  |  |
| *от кафедры ПОИТ* |  | Н.П. Можей |
| *по экономической части* |  | А.А. Горюшкин |
|  |  |  |
| Нормоконтролер |  | А.В. Манцевич |
|  |  |  |
| Рецензент |  |  |

Минск 2021

**РЕФЕРАТ**

ПРОГРАММНОЕ СРЕДСТВО «ОНЛАЙН-КИНОТЕАТР С ИНТЕЛЛЕКТУАЛЬНЫМ ПОМОЩНИКОМ» НА БАЗЕ ПЛАТФОРМЫ ASP.NET CORE: дипломный проект / А.В. Абраменко – Минск: БГУИР, 2021, пз – п.з. – 140 с., чертежей (плакатов) – 6 л. формата А1.

Объектом разработки является программное средство, предназначенное для просмотра фильмов в онлайне, а также их подбор на основе пожеланий пользователя при помощи интеллектуального помощника.

Цель работы – совмещение онлайн-кинотеатра и интеллектуального помощника для подбора кино в рамках одного программного средства.

В первом разделе проводится анализ литературных источников, по итогам которого была изучена история онлайн-кинотеатров, особенности их проектирования, а также изучены принципы работы интеллектуальных помощников. Дополнительно проводится анализ аналогов, по итогам которого были выявлены преимущества и недостатки многих онлайн-кинотеатров и программных средств с интеллектуальным помощником для подбора кино. Выдвигаются общие требования к созданию программного средства.

Во втором разделе проводится моделирование программного средства, а также разработка функциональных требований.

Третий раздел посвящён разработке архитектуры программного средства, модели базы данных. Вдобавок проводится разработка алгоритмов программного средства.

В четвёртом разделе рассматривается язык программирования C# и платформа ASP.NET Core, взаимодействие с базой данных, а также рассматриваются основные компоненты программного средства.

В пятом разделе содержится информация о тестировании разработанного программного средства на соответствие функциональным требованиям.

Шестой раздел содержит руководство по установке программного средства, а также руководство по использованию.

В седьмом разделе приведено технико-экономическое обоснование разработки и внедрения программного средства.

Заключение содержит краткие выводы по дипломному проекту.

Дипломный проект является завершённым, поставленная задача решена  
в полной мере. Планируется дальнейшее развитие программного средства и  
расширение его функциональности. Проект выполнен самостоятельно, проведён анализ оригинальности в системе «Антиплагиат». Процент оригинальности составляет 91,93 %.

Министерство образования Республики Беларусь

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| Факультет | КС и С | | | Кафедра | | | ПОИТ | | | | | | | | | | |
| Специальность | 1-40 01 01 | | | Специализация | | | | | | 01 | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | |  | | | | | Н.В.Лапицкая | | | |
| « | | | | | | | | |  | | » | |  | | | 20 | г. |
| ЗАДАНИЕ | | | | | | | | | | | | | | | | | |
| по дипломному проекту студента | | | | | | | | | | | | | | | | | |
| Абраменко Алексея Викторовича | | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | | |
| 1. Тема проекта: | | Программное средство «Онлайн-кинотеатр с интеллектуальным | | | | | | | | | | | | | | | |
| **помощником» на базе платформы ASP.NET Core** | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | « | 06 | » | | апреля | | | | 2021 г. | | | № | 765-c | |
| 2. Срок сдачи студентом законченной работы | | | | | 01 июня 2021 года | | | | | | | | | | | | |
| 3. Исходные данные к проекту | | | Тип операционной системы – ОС Windows 10; | | | | | | | | | | | | | | |
| Языки программирования – C#, JavaScript; | | | | | | | | | | | | | | | | | |
| Перечень выполняемых функций: отображение подробной информации о фильмах, | | | | | | | | | | | | | | | | | |
| возможность просмотра фильмов в онлайне, подбор фильмов на основе пожеланий | | | | | | | | | | | | | | | | | |
| пользователя; | | | | | | | | | | | | | | | | | |
| Назначение разработки: просмотр фильмов в онлайне, а также их подбор на основе | | | | | | | | | | | | | | | | | |
| пожеланий пользователя при помощи интеллектуального помощника | | | | | | | | | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | | |
| 1 Анализ литературных источников, прототипов и формирование требований к | | | | | | | | | | | | | | | | | |
| проектируемому программному средству | | | | | | | | | | | | | | | | | |
| 2 Моделирование предметной области и разработка функциональных требований | | | | | | | | | | | | | | | | | |
| 3 Проектирование программного средства | | | | | | | | | | | | | | | | | |
| 4 Разработка программного средства | | | | | | | | | | | | | | | | | |
| 5 Тестирование программного средства | | | | | | | | | | | | | | | | | |
| 6 Руководство по установке и использованию программного средства | | | | | | | | | | | | | | | | | |
| 7 Технико-экономическое обоснование разработки и использования программного | | | | | | | | | | | | | | | | | |
| средства | | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | | |
| Список использованных источников | | | | | | | | | | | | | | | | | |
| Приложение А Текст программы | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |

|  |  |
| --- | --- |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения | |
| вида и типа материала) |  |
| Диаграмма развёртывания программного средства. Плакат – формат А1, лист 1. | |
| Диаграмма UML программного средства. Плакат – формат А1, лист 1. | |
| База данных программного средства. Плакат – формат А1, лист 1. | |
| Движение данных при регистрации. Схема данных – формат А1, лист 1. | |
| Контроллер фильмов. Схема алгоритма – формат А1, лист 1 | |
| Программное средство «Онлайн-кинотеатр с интеллектуальным помощником». Схема | |
| программы – формат А1, лист 1. | |
|  | |
|  | |

6. Содержание задания по технико–экономическому обоснованию

|  |  |  |
| --- | --- | --- |
| Описание функций, назначения и потенциальных пользователей программного средства | | |
| Расчёт затрат на разработку программного средства | | |
| Оценка результата от использования программного средства | | |
| Расчёт показателей эффективности инвестиций в разработку программного средства | | |
| Выводы по технико-экономическому обоснованию | | |
| Задание выдал |  | / А. А. Горюшкин / |
|  | | |
|  | | |
|  | | |

# КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной |  |  |  |
| области, разработка технического задания | 15-20 | 23.03–01.04 |  |
| Разработка функциональных требований, |  |  |  |
| проектирование архитектуры программы | 15-20 | 02.04–08.04 |  |
| Разработка схемы программы, алгоритмов, |  |  |  |
| схемы данных | 15-20 | 09.04–15.04 |  |
| Разработка программного средства | 15-20 | 16.04–29.04 |  |
| Тестирование и отладка | 10 | 30.04–13.05 |  |
| Оформление пояснительной записки |  |  |  |
| и графического материала | 20 | 14.05–31.05 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 22 марта 2021 г. | | Руководитель | |  | /Н. П. Можей/ |
| Задание принял к исполнению | |  | | / А. В. Абраменко / | |

**СОДЕРЖАНИЕ**

[Введение 8](#_Toc72236305)

[1 Анализ литературных источников, прототипов и формирование требований к проектируемому программному средству 10](#_Toc72236306)

[1.1 Анализ литературных источников 10](#_Toc72236307)

[1.2 Аналоги, их недостатки и достоинства 14](#_Toc72236308)

[1.3 Цели и задачи дипломного проекта. Формирование требований к приложению 24](#_Toc72236309)

[1.4 Выводы раздела 32](#_Toc72236310)

[2 Моделирование предметной области и разработка функциональных требований 33](#_Toc72236311)

[2.1 Функциональная модель программного средства 33](#_Toc72236312)

[2.2 Спецификация функциональных требований 40](#_Toc72236313)

[2.3 Выводы раздела 56](#_Toc72236314)

[3 Проектирование программного средства 57](#_Toc72236315)

[3.1 Разработка архитектуры приложения 57](#_Toc72236316)

[3.2 Разработка даталогической и физической моделей базы данных 59](#_Toc72236317)

[3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей 65](#_Toc72236318)

[3.4 Выводы раздела 71](#_Toc72236319)

[4 Разработка программного средства 72](#_Toc72236320)

[4.1 Язык программирования C# и платформа ASP.NET Core 72](#_Toc72236321)

[4.2 Взаимодействие с базой данных 74](#_Toc72236322)

[4.3 Основные компоненты программного средства 75](#_Toc72236323)

[4.4 Выводы раздела 80](#_Toc72236324)

[5 Тестирование программного средства 81](#_Toc72236325)

[5.1 Выводы раздела 86](#_Toc72236326)

[6 Руководство по установке и использованию программного средства 87](#_Toc72236327)

[6.1 Установка программного средства 87](#_Toc72236328)

[6.2 Руководство по использованию программного средства 87](#_Toc72236329)

[6.3 Выводы раздела 100](#_Toc72236330)

[7 Технико-экономическое обоснование разработки и использования программного средства 101](#_Toc72236331)

[7.1 Описание функций, назначения и потенциальных пользователей программного средства 101](#_Toc72236332)

[7.2 Расчёт затрат на разработку программного средства 102](#_Toc72236333)

[7.3 Оценка результата от использования программного средства 105](#_Toc72236334)

[7.4 Расчёт показателей эффективности инвестиций в разработку программного средства 106](#_Toc72236335)

[7.5 Выводы технико-экономического обоснования 107](#_Toc72236336)

[Заключение 109](#_Toc72236337)

[Список использованных источников 110](#_Toc72236338)

[Приложение А 111](#_Toc72236339)

**ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В настоящем отчёте применяются следующие определения и сокращения.

*Интернет* – всемирная система объединённых компьютерных сетей для хранения и передачи информации.

*ИИ* – искусственный интеллект.

*МО* – машинное обучение.

*Никнейм* – псевдоним, используемый пользователем в сети Интернет.

*OC* – Операционная Система – комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

*СУБД* – Система Управления Базами Данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

*Фреймворк* – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

*Скроллинг* – форма представления информации, при которой содержимое двигается в вертикальном или горизонтальном направлении.

*Пагинация* – в веб-приложениях под пагинацией понимают постраничный вывод с показом ограниченной части информации на одной веб-странице и возможностью переключения между страницами.

*ПО* – Программное Обеспечение.

*MVC* – Model-View-Controller (Модель-Представление-Контроллер) – архитектурный паттерн, подразумевающий разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента.

Введение

На сегодняшний день просмотр фильмов и сериалов является одним из самых частых видов времяпрепровождения. Всё из-за того, что кино способно подарить невероятные эмоции наравне с чтением любой книги. Однако, в то время как при прочтении книги ты сам представляешь себе весь мир, всех персонажей, в кино ты можешь это наблюдать благодаря серьёзному подходу к съёмкам: выбор места, подбор актёров, создание эксклюзивного саундтрека, – благодаря современным технологиям создания спецэффектов, благодаря камерам, снимающих в высоком разрешении. В общем, современные фильмы могут быть невероятно качественными и незабываемыми.

В результате в интернете одними из самых популярных сайтов являются те, которые о кино: информационные или для непосредственного просмотра. Каждый день такими сайтами пользуются миллионы людей, пытаясь скрасить свой вечер просмотром фильма или сериала. Однако ещё до появления интернета во всём мире люди любили ходить в кинотеатры и делали это довольно часто, так как на то время это было единственной возможностью просмотреть какой-нибудь фильм, который можно будет потом обсуждать с друзьями и знакомыми. Всё это нам говорит о том, что просмотр кино занимал и будет занимать определённое место в жизни каждого человека.

Целью данного дипломного проекта является создание программного средства «Онлайн-кинотеатр с интеллектуальным помощником», где пользователи смогут просматривать фильмы в онлайне, а также искать их на основе своих предпочтений при помощи интеллектуального помощника. Актуальность темы обусловлена ростом популярности онлайн-кинотеатров и в связи со сложностью выбора подходящего для просмотра фильма из-за их огромного количества и ежегодного пополнения.

В первом разделе отчета приведены данные анализа литературных источников. Также, в первом разделе проводится обзор существующих аналогов программных средств типа «Онлайн-кинотеатр с интеллектуальным помощником», выделяются положительные и отрицательные особенности существующих прототипов. На основе анализа аналогов и анализа информации, полученной из литературных источников, выдвигаются общие требования к созданию программного средства.

Во втором разделе проводится моделирование предметной области программного средства, а также разработка спецификации функциональных требований.

Третий раздел посвящён разработке архитектуры программного средства, а также модели базы данных. Также третий раздел описывает технологии, необходимые для разработки программного средства.

Итогом выполнения дипломного проекта является программное средство, которое полностью соответствует разработанной спецификации требований. Использование данного программного средства позволяет без затруднений просматривать фильмы в онлайне, а также предоставляет их подбор на основе пожеланий пользователя.

1 Анализ литературных источников, прототипов и формирование требований к проектируемому программному средству

1.1 Анализ литературных источников

**1.1.1** История становления онлайн-кинотеатров

Первоначально онлайн-кинотеатры задумывались как необычный заменитель домашней коллекции видео: пиратские или лицензионные видеокассеты и DVD. И на данный момент рынок носителей для просмотра фильмов практически полностью утратил свою популярность и стал почти неактуальным, однако всё же люди приобретают коллекционные издания любимых серий фильмов.

Изначально онлайн-кинотеатры появились в США в начале 2007 года, а уже после начали распространяться и по всему миру. Первым же среди легальных онлайн-кинотеатров на постсоветском пространстве считается компания Tvigle, основанная в конце 2007 года. И следом стали появляться и другие: «ivi» (февраль 2010 года), «Megogo» (2011 год), «Okko» (2011 год) и так далее [1].

Многие факторы повлияли на судьбу онлайн-кинотеатров на русскоязычном рынке. Например, одним из важных факторов является проникновение качественной интернет-связи на территорию России, что позволило там начать запуск массовых сервисов по распространению – кино, музыки, игр и так далее [1].

В ходе развития онлайн-кинотеатров выделились следующие бизнес-модели:

* AVoD (advertising video on demand) – рекламная модель, где перед просмотром видео требуется просмотреть рекламный ролик, а иногда и несколько;
* TVoD (transactional video on demand) – платная модель, где пользователь арендует единицу контента на ограниченное время или на неограниченное, но в таком случае зачастую требуется доплатить определённую сумму;
* SVoD (subscription video on demand) – платная модель, где по подписке пользователь получает доступ к библиотеке контента на ограниченное время;
* EST (electronic sell through) – платная модель, которая основывается на приобретении пользователем цифровой копии контента [2].

Изначально русскоязычный рынок онлайн-кинотеатров был ориентирован на рекламную бизнес-модель. Однако в 2017 году впервые рекламная бизнес-модель уступила перед платной, так как доходы от второй оказались выше, чем от первой [1]. На март 2021 года доля платной бизнес-модели в структуре выручки онлайн-кинотеатров составила 78% [2].

Говоря об онлайн-кинотеатрах, нельзя забывать и об обычных кинотеатрах, которые существуют уже более века. В традиционном варианте премьера кино проходит сначала в кинотеатрах, затем, спустя определённый срок (до полугода), появляется на DVD или Blu-Ray. Потом наступает черёд онлайн-кинотеатров, а спустя ещё несколько лет фильм показывают на общественном телевидении. Однако в наши дни фильм может появиться в онлайн-кинотеатре сразу после проката в кино, одновременно с ним или с выходом DVD. Случается, премьера проходит параллельно и на большом экране, и в Интернете, однако подобное сильно зависит от договорённости с правообладателями [3].

Одним из важных факторов привлечения аудитории онлайн-кинотеатра является эксклюзивный контент. И уже сегодня среди онлайн-кинотеатров наблюдается борьба на получение максимально большего количества эксклюзивных фильмов и сериалов, чтобы заманивать побольше людей именно в свою среду. Однако это выгодно и подписчикам онлайн-кинотеатров, так как они получают не только качественный сервис, но и оригинальный контент [4].

Одним из ярких примеров погони за эксклюзивностью контента на российском рынке онлайн-кинотеатров является «ivi», который начал производить собственные сериалы и фильмы. В мае 2019 года было создано подразделение «ivi originals», которым выделили 1 млрд. российских рублей на создание уникальных фильмов и сериалов. И уже сегодня из собственного производства у «ivi» в ассортименте имеется 3 полнометражных фильма и 6 сериалов, и всё это, исходя из статистики онлайн-кинотеатра, умеренно повлияло на рост ежегодной аудитории.

Если говорить о выручке с онлайн-кинотеатров на русскоязычном рынке, то стоит начать с 2013 года, когда рынок онлайн-кинотеатров вырос более чем в два раза, до 1.65 млрд. российских рублей. В 2015 году же данный рынок оценивался в 5.9 млрд. российских рублей (рост на 15% по сравнению с 2014 годом). Объём русскоязычного рынка онлайн-кинотеатров в 2017 году составил 7.7 млрд. рублей (рост по сравнению с 2016 годом на 50%) [1]. И напоследок, в 2018 году составил 16.903 млрд. российских рублей, в 2019 году – 25.669 млрд. российских рублей, в 2020 году – 38.942 млрд. российских рублей [2].

Таким образом история показывает, что из аналога домашней фильмотеки, онлайн-кинотеатры превратились прибыльный вид ведения бизнеса в Интернете, в котором наблюдается сильная конкуренция. Возрастающий ежегодный доход говорит о непрерывном увеличении аудитории, а соответственно и популярности, которая в наши дни крайне важна.

**1.1.2** Особенности проектирования онлайн-кинотеатров

Одним из самых важных факторов, влияющих на проектирование онлайн-кинотеатра, является выбранная бизнес-модель: рекламная (AVoD) или платная (TVoD, SVoD, EST).

В случае выбора рекламной бизнес-модели следует продумать следующие аспекты:

* на каких веб-страницах будет размещена реклама, а на каких не будет;
* расположение рекламы на веб-странице: количество и размеры блоков с рекламой;
* реакция на блокировщик рекламы: никак не оповещать пользователя, оповестить пользователя с просьбой выключить блокировщик или заблокировать возможность пользования онлайн-кинотеатром до момента отключения блокировщика.

В случае выбора платной бизнес-модели следует продумать следующее:

* соблюдение строго одного вида платного VoD или нескольких с предоставлением пользователю выбор;
* количество, качество и стоимость подписок в случае наличия TVoD или SVoD;
* способы оплаты для пользователя;
* наличие акций и их влияние на контент;
* наличие и возможность ввода промокодов для уменьшения суммы оплаты;
* технологии шифрования и защиты контента.

Если же абстрагироваться от бизнес-модели, то при проектировании онлайн-кинотеатра важно учесть:

* возможность легко и быстро находить пользователю нужный ему фильм;
* возможность выделения новинок среди контента для акцентирования внимания пользователя на них;
* количество информации предоставляемой пользователю о фильме;
* возможность без трудностей начинать просмотр фильма;
* возможность выбора качества (разрешения видео) просматриваемого фильма;
* возможность скачивания пользователем фильма;
* возможность выставления пользователем оценок фильму;
* возможность комментирования пользователем фильма;
* возможность пользователю подписаться на уведомления о выходе новинок, которые будут приходить ему на электронную почту.

Однако в наше время к важным аспектам онлайн-кинотеатра стал относиться и подбор фильмов на основе разных характеристик:

* количества просмотров;
* средней оценки;
* года премьеры;
* жанра;
* режиссёра;
* и так далее.

Таким образом при проектировании онлайн-кинотеатра нужно многое учесть, чтобы в итоге получился удобный и многофункциональный продукт для просмотра фильмов, способный удовлетворить пользователей. А аспекты, изложенные ранее, могут применяться при разработке технического задания к программному средству.

**1.1.3** Принципы работы интеллектуальных помощников

В наши дни одной из основных развивающихся технологий является ИИ.

Искусственный интеллект (ИИ) позволяет компьютерам обучаться на собственном опыте, адаптироваться к задаваемым параметрам и выполнять те задачи, которые раньше были под силу только человеку. В большинстве случаев реализации ИИ – от компьютерных шахматистов до беспилотных автомобилей – крайне важна возможность глубокого обучения и обработки естественного языка. Благодаря этим технологиям компьютеры можно «научить» выполнению определенных задач с помощью обработки большого объема данных и выявления в них закономерностей [6].

Машинное обучение (МО) – обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Различают два типа обучения:

* обучение по прецедентам, или индуктивное обучение, основано на выявлении общих закономерностей по частным эмпирическим данным;
* дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний [7].

Интеллектуальные помощники как раз и основаны на технологиях ИИ и МО и активно начинают входить в нашу жизнь. В каждом телефоне, где предустановлены Google-приложения, имеется Google Assistant (интеллектуальный помощник от компании Google), который может выполнять как и лёгкие функции: поставить будильник на определённое время, позвонить определённому человеку, показать ближайшие рестораны и кафе, – так и сложные функции: распознать песни рядом, купить билет на рейс.

Стандартным способом взаимодействия пользователя с интеллектуальным помощником является ввод текста первым и вывод текста вторым. Однако для повышения удобства работы с «помощником» в наши дни задействуются следующие технологии:

* голосовой ввод: ввод данных голосом вместо взаимодействия при помощи ввода текста;
* понимание естественного языка: возможность использовать не только ограниченный набор команд (словарь) и правила их произношения (синтаксис), а обращаться к устройству как в настоящем разговоре;
* голосовой вывод: вместо вывода текста интеллектуальный помощник зачитывает информацию вслух;
* интеллектуальная интерпретация: кроме непосредственного вводимого пользователем запроса ассистент использует дополнительную информацию (например, контекст действия или прошлое поведение), чтобы лучше понять свою задачу;
* агентские функции: совершение действие, которые пользователь не запрашивал, но которые, по оценке интеллектуального помощника, будут полезны.

Последние две технологии требуют, что «помощник» активно узнавал своего пользователя, чтобы иметь возможность подстраивать своё поведение под его потребности [8].

Одними из самых популярных голосовых интеллектуальных помощников являются:

* Alexa от компании Amazon ;
* Siri от компании Apple;
* Google Assistant от компании Google;
* Cortana от компании Microsoft.

Все перечисленные голосовые интеллектуальные помощники обладают своими особенностями и направлены на решение определённых задач, хоть и имеют достаточно общих функций [9].

Говоря о будущем интеллектуальных помощников, следует сказать, что их возможности взаимодействия с человеком, без сомнения, станут еще больше. Однако полностью самостоятельными в общении с ним в обозримой перспективе станут не всех из них – некоторым в ответах на определенные запросы по-прежнему должны будут помогать специалисты отделов поддержки. Таким образом, пока пользователям в некоторых случаях предстоит иметь дело с гибридным подходом в применении интеллектуальных помощников [10].

1.2 Аналоги, их недостатки и достоинства

**1.2.1** Описание функций «Онлайн-кинотеатра с интеллектуальным помощником» для поиска аналогов

Так как в результате поиска полные аналоги программных средств типа «Онлайн-кинотеатр с интеллектуальным помощником» не были выявлены, в данном разделе будут рассмотрены два вида частичных аналогов: программное средство типа онлайн-кинотеатр и программное средство с интеллектуальным помощником для подбора фильмов.

Программное средство типа онлайн-кинотеатр должно обладать следующими функциями:

* возможность регистрации нового профиля;
* возможность авторизации в существующий профиль;
* возможность просмотра списка имеющихся для просмотра фильмов с отображение краткой информацией о каждом;
* возможность поиска фильма по названию;
* возможность просмотра подробной информации о фильме;
* возможность просмотра видео фильма;
* возможность выбора качества просматриваемого видео.

Программное средство с интеллектуальным помощником для подбора фильмов должно выполнять подбор фильмов на основе:

* предустановленных характеристик (например, любимый жанр или любимый актёр);
* фильмов, помеченных как просмотренные;
* оценённых фильмов.

**1.2.2** Онлайн-кинотеатр «HDrezka.me» [11]

Данный онлайн-кинотеатр впервые появился в Интернете в 2013 году. В качестве бизнес-модели была выбрана рекламная.

Интерфейс главной страницы представлен на рисунке 1.1. Здесь пользователю отображаются новинки среди фильмов, сериалов, мультфильмов и аниме, также отображаются предустановленные тематические каталоги с фильмами.

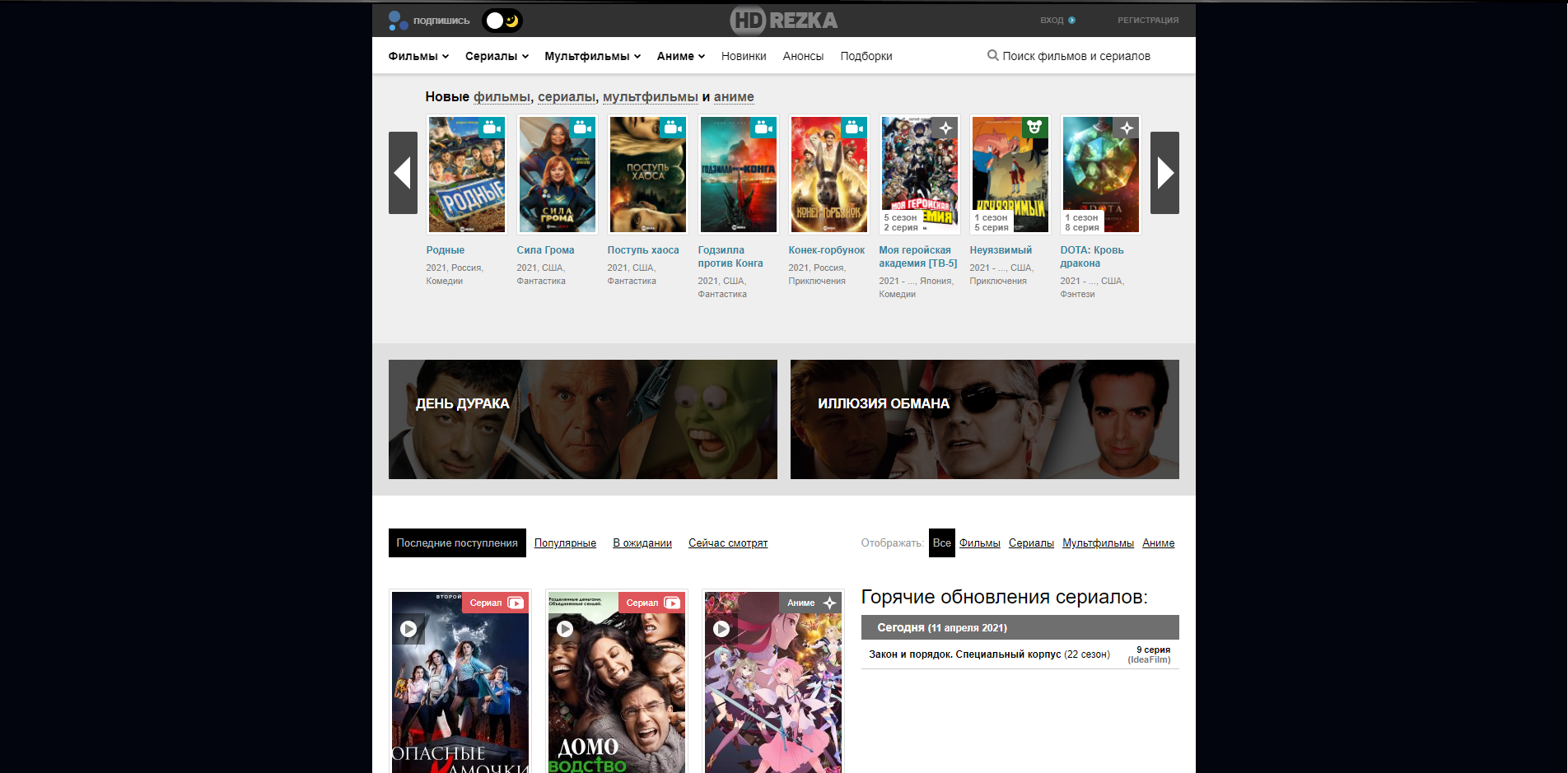


Рисунок 1.1 - Интерфейс главной страницы на апрель 2021 года онлайн-кинотеатра «HDrezka.me»

В онлайн-кинотеатре имеется большое количество фильмов, а также сериалов, мультфильмов и аниме, которые доступны для просмотра абсолютно бесплатно. Имеется страница, интерфейс которой представлен на рисунке 1.2, со списком доступных для просмотра фильмов с краткой информацией о каждом и с возможностью просмотра более детальной, но всё же не полной, информации о фильме при наведении на соответствующую иконку у соответствующего фильма.

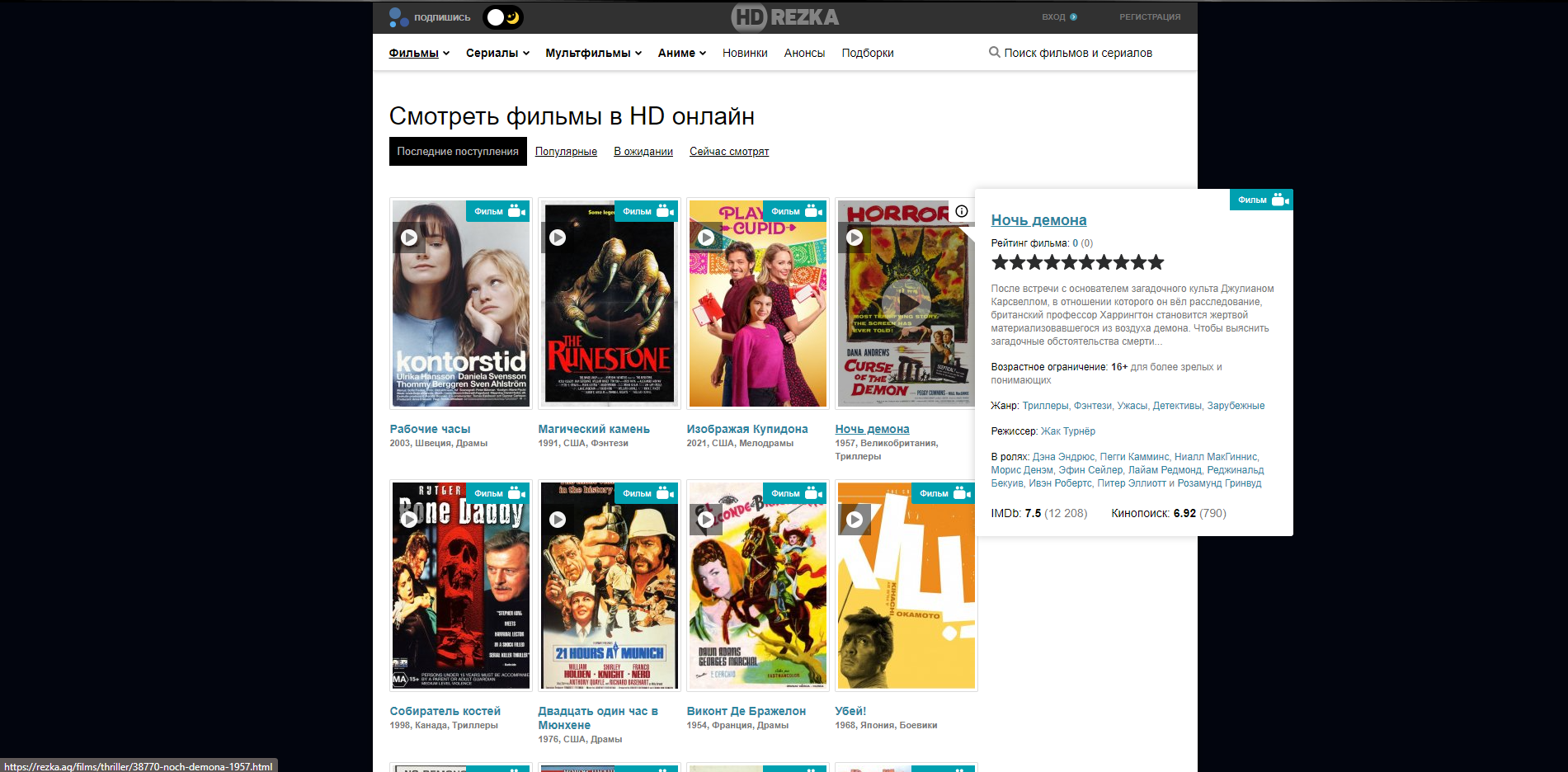


Рисунок 1.2 - Интерфейс страницы со списком доступных для просмотра фильмов на апрель 2021 года онлайн-кинотеатра «HDrezka.me»

Отображаемый список доступных для просмотра фильмов может быть отфильтрован по следующим критериям:

* жанр фильма;
* год премьеры фильма;
* компания-издатель фильма;
* новизна;
* популярность;
* просматриваемые на данный момент другими пользователями.

Также имеется возможность поиска фильмов по названию.

В случае выбора определённого фильма пользователю будет отображена соответствующая страница (см. рисунок 1.3), на которой возможны следующие действия:

* просмотр подробной информации о фильме;
* просмотр видео фильма;
* оценка фильма по десятибалльной шкале;
* написание отзыва о фильме;
* быстрый переход на другие имеющиеся для просмотра фильмы;
* добавление фильма в закладки (доступно только для авторизованных пользователей).

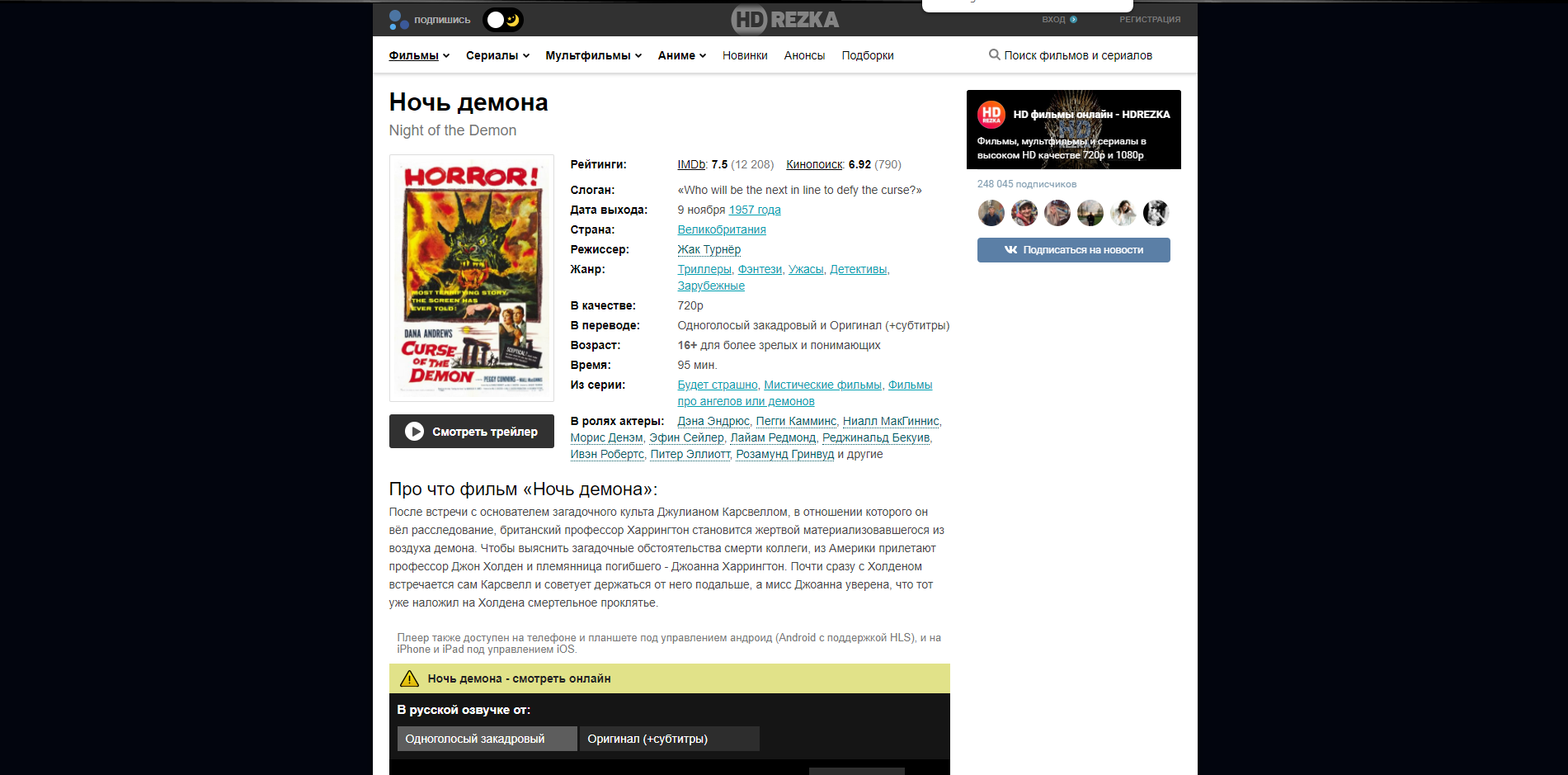


Рисунок 1.3 - Интерфейс страницы фильма «Ночь демона» на апрель 2021 года онлайн-кинотеатра «HDrezka.me»

В видеоплеере доступен выбор качества видео с максимальным разрешением Ultra HD (3840x2160). Также имеется возможность выбора озвучки фильма, если имеются альтернативы. А у некоторых фильмов имеется функция скачивания видео.

Последним стоит отметить, что данный онлайн-кинотеатр доступен для использования только через веб-браузер, самостоятельного приложения, например для телевизоров, не имеется.

У данного онлайн-кинотеатра можно выделить следующие преимущества:

* большое количество доступных для просмотра фильмов, сериалов, мультфильмов и аниме, список которых пополняется каждый день;
* просмотр всего контента является абсолютно бесплатным;
* просматривать весь контент можно будучи неавторизованным;
* лёгкий и интуитивно понятный дизайн;
* наличие разрешения Ultra HD у видео фильмов;
* возможность выбора озвучки у видео фильма;
* возможность скачивания видео фильма;
* лояльное отношение к блокировщику рекламы.

Недостатки:

* отсутствие возможности просмотра контента сайта на телевизорах с помощью самостоятельного приложения;
* наличие двух рекламных роликов в начале просмотра каждого видео в случае отсутствия у пользователя блокировщика рекламы;
* недостаточно обширный функционал по фильтрации фильмов.

В итоге можно сказать, что «HDrezka.me» является одним из лучших бесплатных онлайн-кинотеатров, доступных русскоязычным пользователям.

**1.2.2** Онлайн-кинотеатр «Okko» [12]

«Okko» – один из самых больших онлайн-кинотеатров в России, который доступен в Интернете с 2011 года. Основан на платной бизнес-модели: совмещение SVoD и EST.

Интерфейс главной страницы представлен на рисунке 1.4. Здесь пользователю в первую очередь отображаются новинки, доступные для просмотра, с краткой информацией о каждой. Далее идут имеющиеся подписки, а после них уже списки фильмов по категориям без краткой информации о каждом.

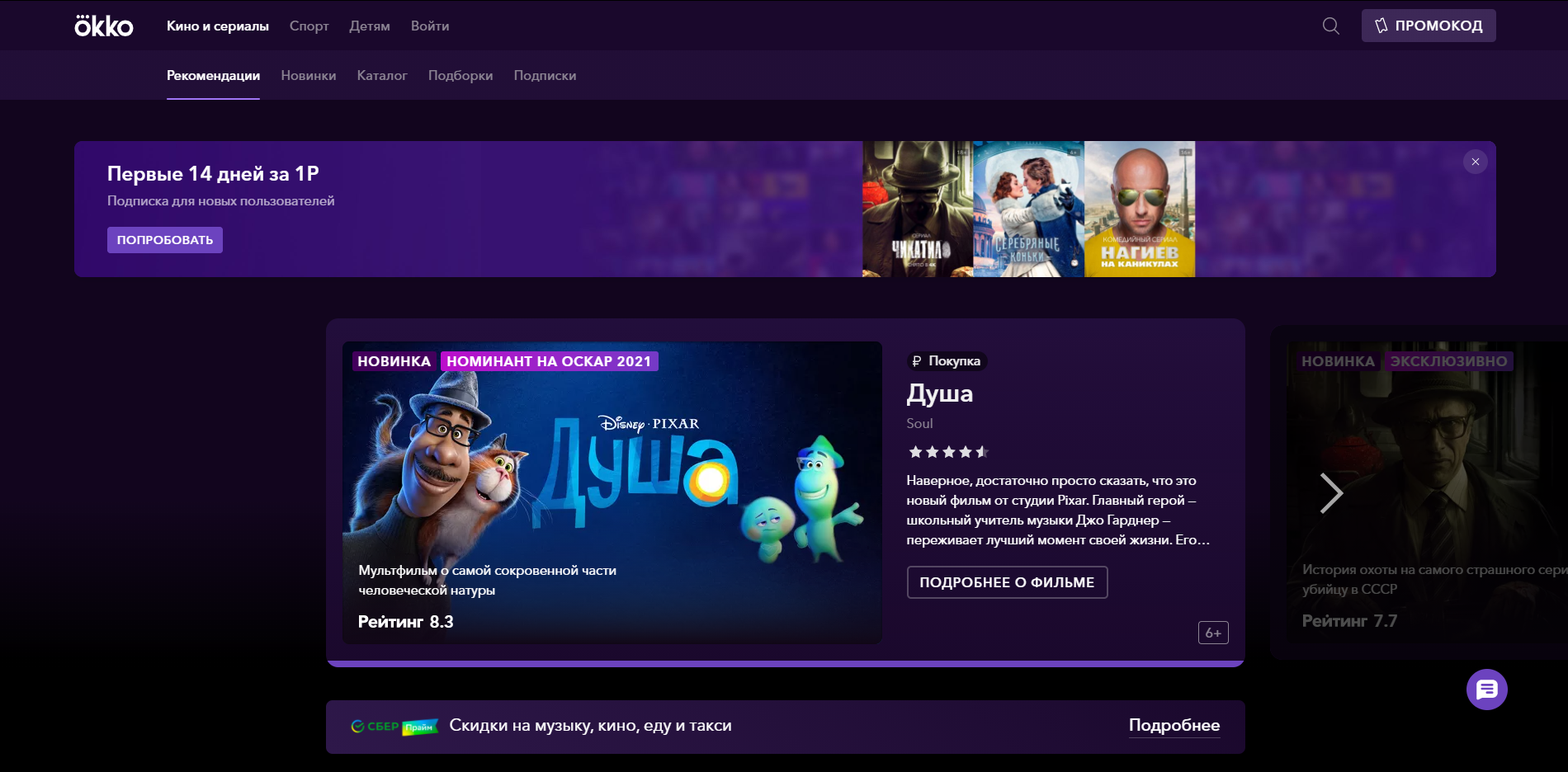


Рисунок 1.4 - Интерфейс главной страницы на апрель 2021 года онлайн-кинотеатра «Okko»

Данный онлайн-кинотеатр предлагает более 100000 фильмов, мультфильмов и сериалов. Доступный контент пополняется каждый день и иногда имеются новинки, которые все ещё идут в кинотеатрах. Имеется возможность смотреть фильмы со звуком Dolby Atmos и Dolby Digital Plus и с разрешением видео до Ultra HD 4K и даже до Ultra HD 8K (7680×4320).

В случае желания пользователя просматривать фильмы по подписке, «Okko» предлагает большое количество вариантов:

* подписка «Лайт»: 55 000 фильмов и сериалов на срок 1 месяц за 199 российских рублей;
* подписка «Оптимум»: 80 000 фильмов и сериалов на срок 1 месяц за 399 российских рублей;
* подписка «Премиум»: 90 000 фильмов и сериалов на срок 1 месяц за 799 российских рублей;
* подписка «Фильмы в Ultra HD 4K»: более 1000 фильмов с разрешением видео Ultra HD 4K на срок 1 месяц за 199 российских рублей;
* и так далее.

Все виды подписок могут осуществляться на срок более чем 1 месяц: на 3 месяца и на 12 месяцев. Осуществление подписки на длительный период является выгодным для пользователя, так как благодаря этому он экономит некоторую часть денег по сравнению со случаем ежемесячного осуществления подписки.

Также стоит отметить, что данный онлайн-кинотеатр даёт возможность просматривать трансляции различных спортивных матчей (от футбола до смешанных боевых искусств), и всё это по специальной подписке.

Приятным бонусом для пользователей является наличие бесплатной подписки, у которой в наличии фильмы об искусстве, образовательный и развлекательный контент.

Важно отметить, что имеются фильмы, которые недоступны ни по одной из имеющихся подписок. В таком случае пользователь вынужден купить на неограниченный срок желаемый фильм, и здесь имеется два варианта:

* за 199 российских рублей за один фильм с разрешением видео HD (1280x720) со стереозвуком;
* за 299 российских рублей за один фильм с разрешением видео Full HD (1920x1080) со стереозвуком.

Интерфейс страницы фильма, который куплен или доступен по подписке, представлен на рисунке 1.5.

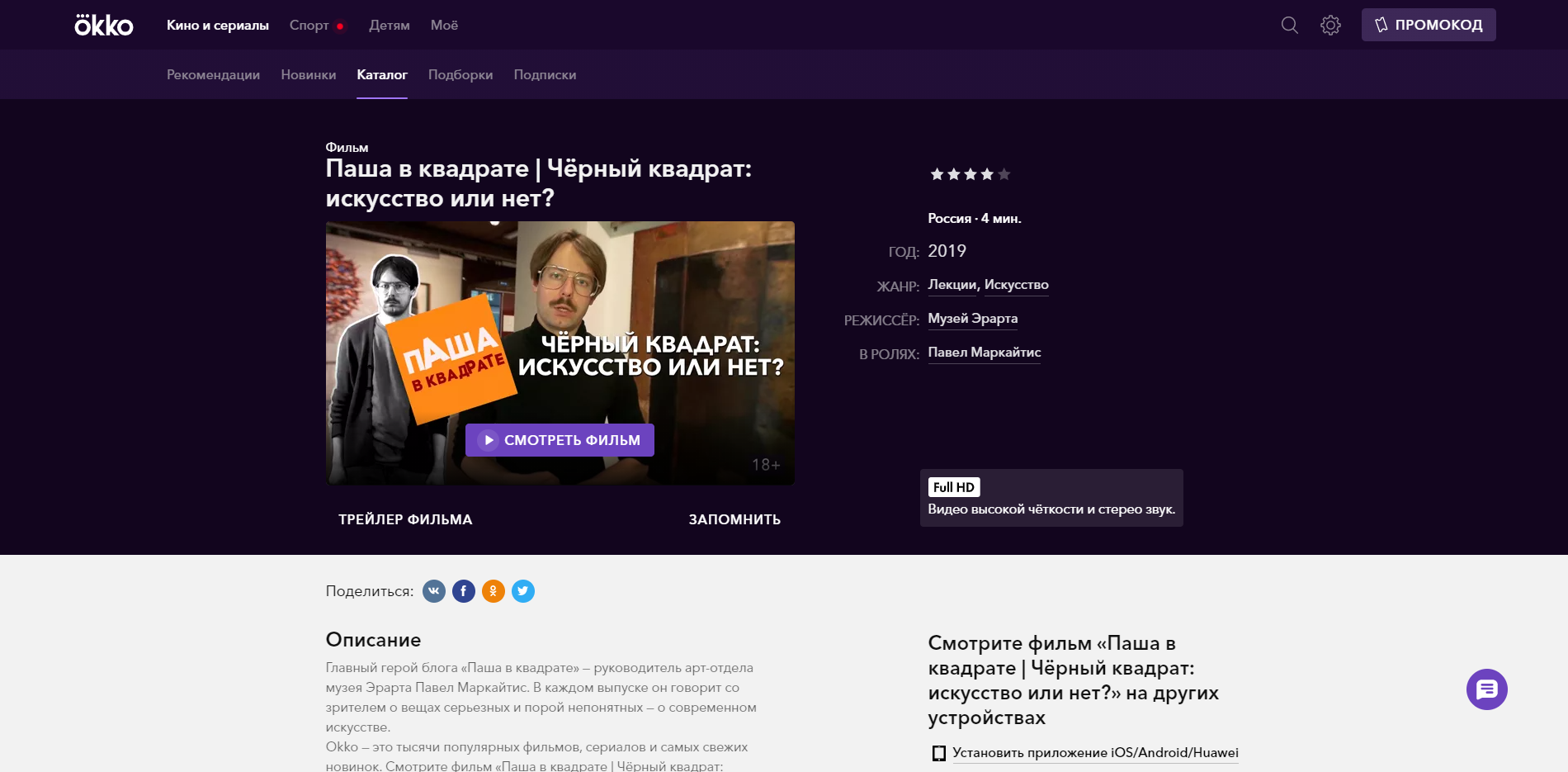


Рисунок 1.5 - Интерфейс страницы фильма «Чёрный квадрат: искусство или нет»" на апрель 2021 года онлайн-кинотеатра «Okko»

На странице фильма возможно:

* просмотреть подробную информацию о фильме;
* просмотреть видео фильма;
* просмотреть трейлер фильма;
* добавить фильм в закладки;
* быстро перейти на другие похожие фильмы.

Стоит выделить, что «Okko» не предоставляет возможности оставлять комментарии под фильмом или же выставлять ему оценку.

В онлайн-кинотеатре имеется возможность поиска фильмов, актёров, режиссёров по названию, а отображаемый список доступных для просмотра фильмов может быть отфильтрован по жанру.

Последним стоит отметить, что просмотр контента «Okko» доступен не только с помощью веб-браузера стационарного компьютера или ноутбука, а ещё и с помощью телевизоров с функцией Smart TV (Samsung, Sony, LG, Philips, Panasonic и другие популярные модели), на игровых консолях (PlayStation 3, 4 и 5 и Series X|S) и на мобильных устройствах (Android и iOS) в качестве самостоятельного и бесплатного для скачивания приложения. Приобретённый контент разрешается смотреть на одном профиле одновременно на 5 устройствах, которые поддерживают приложение «Okko».

В результате у данного онлайн-кинотеатра можно выделить следующие преимущества:

* большое количество доступного для просмотра контента, список которого пополняется каждый день;
* возможность просмотра фильмов на телевизорах, игровых консолях и телефонах благодаря самостоятельному приложению;
* приобретённый контент разрешается смотреть на одном профиле на 5 устройствах одновременно;
* возможность просмотра фильмов с качественным звуком по технологии Dolby Digital и с разрешением видео до Ultra HD 8K;
* наличие эксклюзивного контента, включая новинки, которые на момент добавления все ещё идут в кинотеатрах;
* наличие акций и скидок;
* возможность уменьшения размера оплаты при вводе соответствующего промокода;
* отсутствие какой-либо рекламы.

Недостатки:

* почти весь контент является платным;
* для просмотра любого видео требуется авторизация;
* отсутствие выбора озвучки у видео фильма;
* отсутствие возможности комментирования и оценивания фильмов;
* скудный функционал по фильтрации фильмов.

**1.2.3** Программное средство «Кинонавигатор» [13]

Базовая логика «Кинонавигатора» такая же, как у всех рекомендательных программных средств. Он собирает пользовательские оценки фильмов из разных источников и просит пользователя оценивать фильмы, которые он видел. Это позволяет составить его индивидуальный профиль предпочтений и на основании оценок других пользователей спрогнозировать, насколько ему понравится тот или иной фильм. После этого интеллектуальным помощником составляется список фильмов с наивысшими прогнозными оценками, эти фильмы и попадают в рекомендации. Предварительно пользователю предлагается составить краткую характеристику о себе с точки зрения кинозрителя.

Для осуществления любых действий в программном средстве авторизация не требуется, так как автоматически создаётся гостевой профиль, в котором сохраняется вся активность пользователя.

Интерфейс главной страницы представлен на рисунке 1.6.

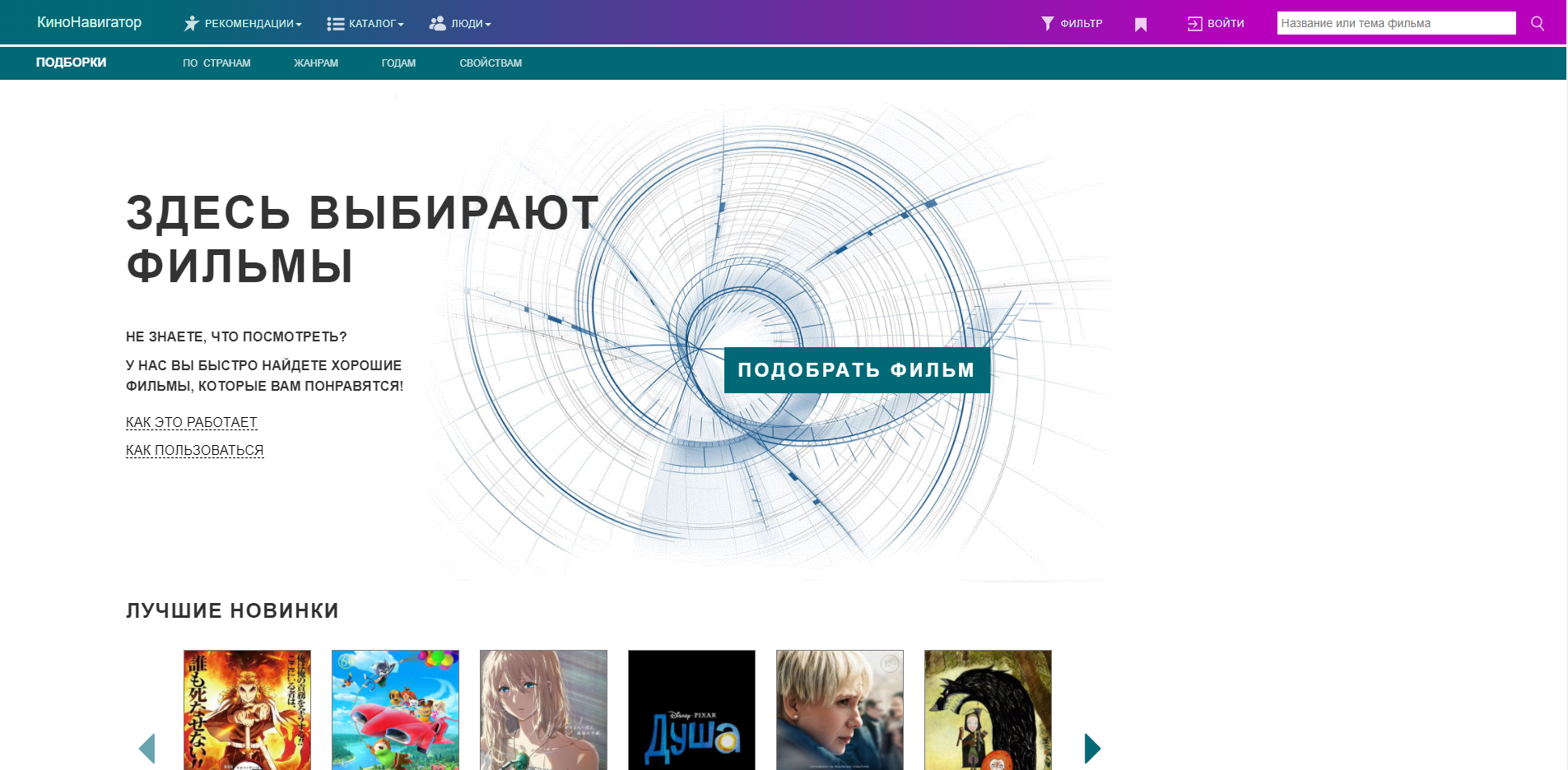


Рисунок 1.6 - Интерфейс главной страницы на апрель 2021 года программного средства «Кинонавигатор»

На главной странице пользователю предлагается сразу же составить краткую характеристику о себе, нажав на кнопку «Подобрать фильм», или сразу же узнать о том, как работает программное средство и как им пользоваться. Ниже предлагается просмотреть списки фильмов с краткой информацией о каждом, разбитые по категориям, и с возможностью сразу оценить любой из них.

Интерфейс формы составления краткой характеристики о себе представлен на рисунке 1.7. Пользователю предлагается указать следующую информацию:

* частоту просмотра фильмов;
* отношение к необычному виду юмора;
* отношение к фильмам с открытым финалом (непонятно, чем всё закончится);
* отношение к аниме;
* пол пользователя;
* возраст пользователя.

Вся указанная информация будет учитываться интеллектуальным помощником при подборе фильмов.



Рисунок 1.7 - Интерфейс формы заполнения характеристики пользователя на апрель 2021 года программного средства «Кинонавигатор»

После заполнения формы пользователю будет отображена страница с первым рекомендуемым фильмом.

Данное программное средство хранит огромное количество фильмов, а также персон, участвовавших в их создании. Информация, доступная по каждому фильму является обширной и подробной, позволяя пользователю побольше узнать о том, что его интересует.

«Кинонавигатор» предоставляет возможность осуществления детальной фильтрации списка фильмов на основе следующих характеристик:

* год премьеры;
* жанр;
* страна-производитель;
* полученные награды;
* популярность;
* рейтинг;
* простота просмотра;
* реалистичность происходящего в фильме;
* и так далее.

После выбора фильма пользователю будет отображена соответствующая страница (см. рисунок 1.8), на которой возможны следующие действия:

* просмотр крайне подробной информации о фильме;
* просмотр трейлера фильма;
* оценка фильма по пятибалльной шкале;
* написание комментария о фильме;
* добавление в закладки;
* просмотр списка похожих фильмов;
* переход на другой рекомендуемый фильм.

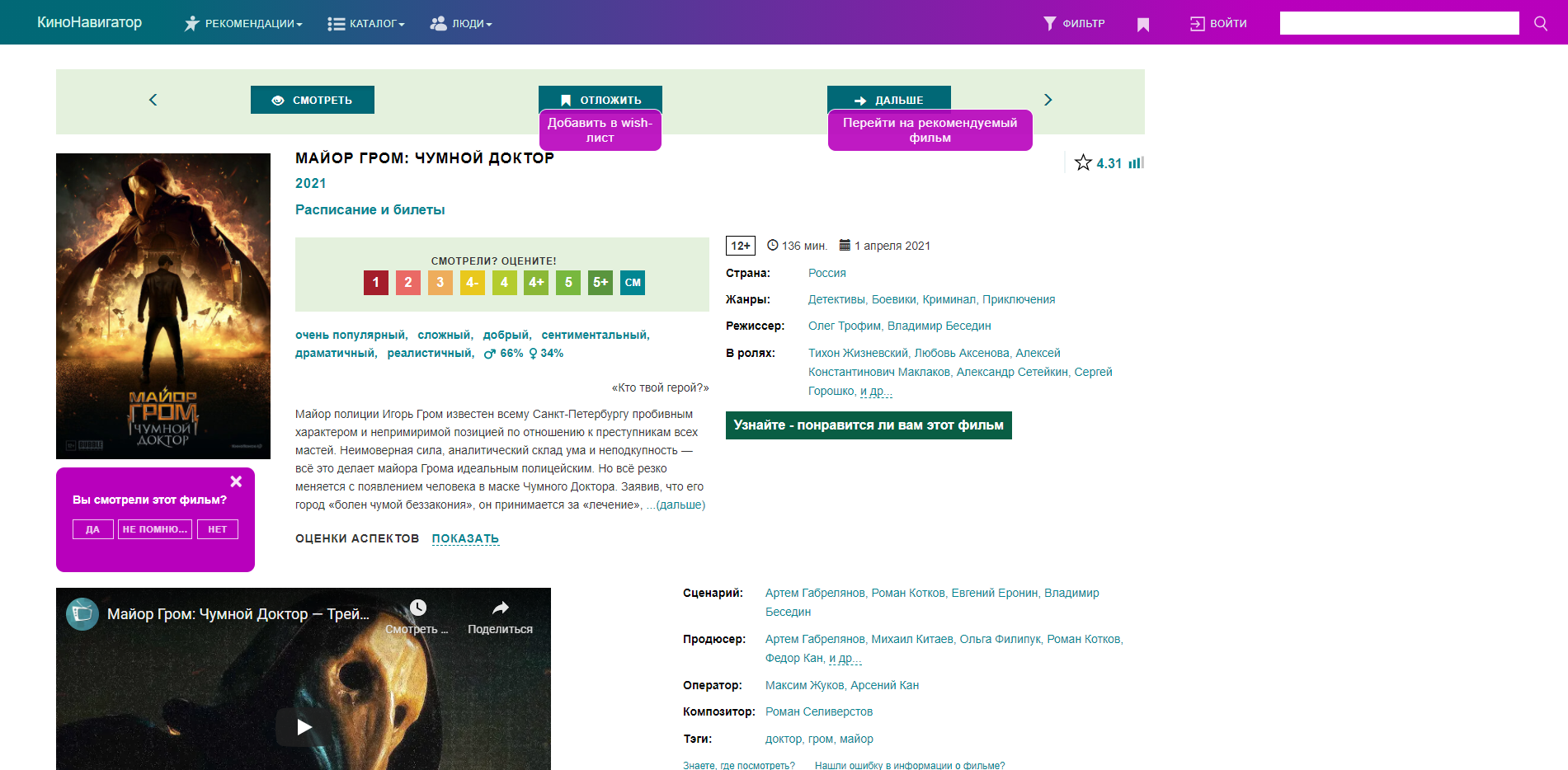


Рисунок 1.8 - Интерфейс страницы фильма «Майор Гром: Чумной доктор» на апрель 2021 года программного средства «Кинонавигатор»

Следует отметить, что интеллектуальный помощник достаточно точно выполняет подбор фильмов на основе всей активности пользователя, что указывает на хорошую проработанность системы данного программного средства.

У «Кинонавигатор» можно выделить следующие преимущества:

* лёгкий и интуитивно понятный дизайн;
* автоматическое создание гостевого профиля;
* обширное количество фильмов;
* обширный функционал фильтрации;
* крайне подробная информация о каждом фильме;
* качественный подбор фильмов;
* программное средство является абсолютное бесплатным в использовании;
* отсутствие какой-либо рекламы.

Недостатки:

* отсутствие возможности просмотра фильмов.

В итоге можно сказать, что «Кинонавигатор» является отличным программным средством с крайне точным подбором фильмов благодаря проработанному интеллектуальному помощнику.

**1.2.4** Другие аналоги

Также в качестве аналога был рассмотрен онлайн-кинотеатр «ivi», отличительной чертой которого является наличие сразу 3-х бизнез-моделей: AVoD, SVoD и EST. Дополнительно в данном онлайн-кинотеатре присутствуют фильмы и сериалы собственного производства, список которых будет пополняться каждый год. В остальном же функциональность «ivi» схожа с уже рассмотренным аналогом «Okko», однако объём функциональности у «ivi» всё-таки больше. Например, в нём пользователи имеют возможность писать отзывы и рецензии к предоставляемому контенту, или же здесь представлена необычная система рейтинга, который состоит не просто из числа от 1 до 10, а ещё из 4-х характеристик, которые помогут пользователю заранее определить сильные стороны фильма или сериала: режиссура, сюжет, зрелищность и актёры.

Ещё одним рассмотренным аналогом является онлайн-кинотеатр «Megogo». В плане функциональности данный аналог крайне схож с «Okko». Выделить же можно то, что «Megogo» запустил собственную студию сурдоперевода и открыл специальный раздел на сайте с тысячами фильмов для людей с нарушением слуха.

**1.2.5** Результат анализа аналогов

Результатом проделанного анализа приведённых и других аналогов стало объединение их преимуществ в создаваемом программном средстве.

1.3 Цели и задачи дипломного проекта. Формирование требований к приложению

**1.3.1** Назначение разработки

Подводя итоги всего описанного ранее, можно определить основные пользовательские требования и сформировать техническое задание для программного средства, разрабатываемого в данном проекте.

Программное средство «Онлайн-кинотеатр с интеллектуальным помощником» предназначено для просмотра фильмов в онлайне, а также их подбора на основе пожеланий пользователя при помощи интеллектуального помощника.

Основными целями создания данного программного средства являются:

* предоставление возможности просмотра подробной информации о фильмах;
* предоставление возможности просмотра фильмов в онлайне;
* предоставление возможности подбора фильмов для пользователя на основе его пожеланий при помощи интеллектуального помощника.

К особенностям разрабатываемого программного средства, в сравнении с рассмотренными ранее аналогами, можно отнести следующее:

* совмещение функциональности онлайн-кинотеатра и интеллектуального помощника в одном программном средстве;
* отображение под видеоплеером списка фильмов, которые подобны просматриваемому на данный момент, исходя из анализа интеллектуального помощника.

**1.3.2** Состав выполняемых функций

**1.3.2.1** Регистрация

При прохождении регистрации требуется удостовериться в корректности и уникальности введённого пользователем никнейма. В случае, если пользователь с таким никнеймом уже существует, регистрация считается неуспешной и должно отобразиться сообщение об ошибке. После успешной регистрации пользователь должен быть автоматически авторизирован и должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**1.3.2.2** Авторизация

При прохождении авторизации требуется удостовериться в корректности введённого пользователем никнейма, а также в существовании пользователя с таким никнеймом и введённым паролем. В случае, если пользователь с таким никнеймом и паролем не существует, авторизация считается неуспешной и должно отобразиться сообщение об ошибке. После успешной авторизации должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**1.3.2.3** Отображение списка фильмов, доступных для просмотра

В списке фильмов должна отображаться краткая информация о каждом из них. Краткая информация о фильме содержит:

* изображение постера;
* название фильма;
* год премьеры;
* жанры;
* рейтинг.

При отображении должна присутствовать возможность сортировки и фильтрации списка фильмов. Кроме того, при выборе фильма из отображаемого списка должно происходить перенаправление на отображение подробной информации о фильме. Также должна иметься возможность перенаправления на подбор фильмов на основе пожеланий пользователя.

Для пользователя с определённым статусом дополнительно имеется возможность редактирования и удаления выбранного из отображаемого списка фильма.

**1.3.2.4** Сортировка списка фильмов

После сортировки должно происходить отображение списка фильмов, в котором порядок элементов изменён на основе выбранного критерия. Критериями, по которым может осуществляться сортировка, являются:

* название фильма;
* год премьеры;
* рейтинг;
* дата добавления фильма.

**1.3.2.5** Фильтрация списка фильмов

После фильтрации должно происходить отображение списка фильмов, элементы которого соответствуют выбранным критериям. Критериями, по которым может осуществляться фильтрация, являются:

* название фильма;
* диапазон года премьеры;
* жанры;
* диапазон продолжительности;
* возрастные ограничения;
* диапазон рейтинга.

**1.3.2.6** Отображение подробной информации о фильме

При просмотре подробной информации о фильме должна присутствовать возможность просмотреть фильм в онлайне, просмотреть список комментариев о фильме. Подробная информация о фильме содержит:

* изображение постера;
* название фильма;
* слоган фильма;
* жанры;
* страны-производители;
* компании-производители;
* дата премьеры;
* бюджет;
* сборы;
* продолжительность;
* возрастное ограничение;
* рейтинг;
* количество просмотров;
* описание;
* список комментариев;
* список участников фильма с указанием их роли.

По нажатии на одного из участников фильма должно происходить перенаправление на отображение подробной информации о выбранном участнике фильма.

Для пользователя с определённым статус дополнительно имеется возможность добавления фильма в каталог «Буду смотреть» или его непосредственное удаление из него, возможность выставления оценки фильму по десятибалльной системе, а также возможность его комментирования.

**1.3.2.7** Просмотр фильма в онлайне

Во время просмотра пользователь может выполнять следующие действия:

* приостановка просмотра фильма;
* продолжение просмотра фильма;
* перемотка фильма на выбранный момент;
* регулировка звука;
* выбор качества видео;
* переход на полноэкранный режим видео;
* выход из полноэкранного режима видео.

**1.3.2.8** Подбор фильмов на основе пожеланий пользователя

При подборе должна учитываться информация о пользователе, а также должны учитываться все оценённые и добавленные в каталог «Буду смотреть» фильмы. После подбора фильмов на основе критериев, выбранных пользователем, должно выполняться отображение подробной информации о наиболее подходящем фильме. Критериями, на основе которых пользователь может выполнять подбор, являются:

* диапазон года премьеры;
* жанры;
* диапазон продолжительности;
* возрастные ограничения;
* диапазон рейтинга.

**1.3.2.9** Добавление фильма

При добавлении фильма требуется проверить корректность введённых данных. В случае, если имеются некорректно заполненные данные, добавление считается неуспешным и должно отобразиться сообщение об ошибке. В случае успешного добавления должно происходить перенаправление на отображение подробной информации о добавленном фильме.

**1.3.2.10** Редактирование фильма

При редактировании информации о фильме все поля и значения должны быть корректно загружены и отображены. До окончания редактирования должна иметься возможность отменить внесение изменений. В случае, если имеются некорректно отредактированные данные, редактирование считается неуспешным и должно происходить отображение сообщения об ошибке. После успешного редактирования новая версия информации о фильме должна быть сохранена и отображена.

**1.3.2.11** Удаление фильма

Перед удалением пользователь должен подтвердить выполнение данной функции. По окончании удаления должен быть отображён обновлённый список фильмов, доступных для просмотра.

**1.3.2.12** Отображение списка комментариев о фильме

В списке комментариев о фильме должна содержаться информация о каждом из них. Информация о комментарии содержит:

* никнейм пользователя, который написал комментарий;
* дату написания комментария;
* текст комментария.

**1.3.2.13** Комментирование фильма

При комментировании требуется проверить корректность введённых данных. В случае, если данные введены некорректно, комментирование считается неуспешным и должно отобразиться сообщение об ошибке. После успешного комментирования должен быть отображён обновлённый список комментариев о фильме.

**1.3.2.14** Оценивание фильма

По окончании оценивания рейтинг фильма должен быть пересчитан и сохранен в базе данных. После чего отображаемое значение рейтинга фильма должно быть обновлено.

**1.3.2.15** Добавление фильма в каталог «Буду смотреть»

После осуществления данной функции добавленный в каталог фильм должен отображаться в списке фильмов, добавленных в каталог «Буду смотреть». Также после осуществления данной функции пользователь должен иметь возможность удалить фильм из каталога «Буду смотреть».

**1.3.2.16** Удаление фильма из каталога «Буду смотреть»

После осуществления данной функции удалённый из каталога фильм больше не должен отображаться в списке фильмов, добавленных в каталог «Буду смотреть». Также после осуществления данной функции пользователь должен иметь возможность обратно добавить фильм в каталог «Буду смотреть».

**1.3.2.17** Просмотр подробной информации об участнике

Подробная информация об участнике содержит:

* изображение участника;
* имя участника;
* фамилия участника;
* рост;
* день рождения;
* пол;
* список фильмов с данным участником.

По нажатии на один из фильмов с данным участником должно происходить перенаправление на отображение подробной информации о выбранном фильме.

Для пользователя с определённым статусом дополнительно имеется возможность редактирования и удаления отображаемого участника.

**1.3.2.18** Добавление участника

При добавлении участника требуется проверить корректность введённых данных. В случае, если имеются некорректно заполненные данные, добавление считается неуспешным и должно отобразиться сообщение об ошибке. В случае успешного добавления должно происходить перенаправление на отображение подробной информации о добавленном участнике.

**1.3.2.19** Редактирование участника

При редактировании информации об участнике все поля и значения должны быть корректно загружены и отображены. До окончания редактирования должна иметься возможность отменить внесение изменений. В случае, если имеются некорректно отредактированные данные, редактирование считается неуспешным и должно происходить отображение сообщения об ошибке. После успешного редактирования новая версия информации об участнике должна быть сохранена и отображена.

**1.3.20** Удаление участника

Перед удалением пользователь должен подтвердить выполнение данной функции. По окончании удаления должен быть отображён список фильмов, доступных для просмотра.

**1.3.2.21** Просмотр профиля

В профиле пользователя должна отображаться подробная информация о нём. Подробная информация о пользователе содержит:

* никнейм пользователя;
* дата регистрации;
* пол;
* дата рождения;
* любимые жанры;
* любимые участники фильмов;
* любимые фильмы.

Также должна присутствовать возможность отредактировать профиль, возможность просмотреть список фильмов, добавленных в каталог «Буду смотреть» и возможность просмотреть список оценённых фильмов.

**1.3.2.22** Редактирование профиля

При редактировании профиля пользователя все поля и значения должны быть корректно загружены и отображены. До окончания редактирования должна иметься возможность отменить внесение изменений. В случае, если имеются некорректно отредактированные данные, редактирование считается неуспешным и должно происходить отображение сообщения об ошибке. После успешного редактирования новая версия профиля должна быть сохранена и отображена.

**1.3.2.23** Отображение списка фильмов, добавленных в каталог «Буду смотреть»

В списке должна отображаться краткая информация о каждом фильме, добавленном в каталог «Буду смотреть». При отображении должна присутствовать возможность сортировки и фильтрации данного списка фильмов. Кроме того, при выборе фильма из отображаемого списка должно происходить перенаправление на отображение подробной информации о фильме.

**1.3.2.24** Отображение списка фильмов, оценённых пользователем

В списке должна отображаться краткая информация о каждом фильме, оценённом пользователем. При отображении должна присутствовать возможность сортировки и фильтрации данного списка фильмов. Кроме того, при выборе фильма из отображаемого списка должно происходить перенаправление на отображение подробной информации о фильме.

**1.3.2.25** Выход из профиля

Выход из профиля должен быть возможен в любой момент времени пользования программным средством. После выхода из профиля пользователь должен быть перенаправлен на отображение списка фильмов, доступных для просмотра.

**1.3.3** Требования к входным данным

Входные данные для программного средства должны быть представлены в виде вводимого пользователем с клавиатуры текста, документов соответствующего формата и содержания и опций, предоставляемых пользовательским интерфейсом приложения.

В бизнес-логике программного средства должны быть реализованы проверки входных данных на корректность, и в случае их некорректности, пользователь должен получать соответствующее уведомление с просьбой ввести данные необходимого формата.

**1.3.4** Требования к выходным данным

Выходные данные должны быть представлены посредством отображения информации при помощи различных элементов реализованного и доступного пользовательского интерфейса.

**1.3.5** Требования к надежности

Для обеспечения надежности приложения требуется обеспечить бесперебойное питания технического средства, и своевременные проверки оборудования на наличие вирусных программ.

Время восстановления после отказа, вызванного сбоем электропитания технических средств, или не фатальным сбоем операционной системы, не должно превышать 60-ти минут при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, или фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Программное средство не должно непредвиденно прерывать свою работу. Отказы программного средства вследствие некорректных действий пользователя при взаимодействии с интерфейсом недопустимы.

**1.3.6** Технические требования

Архитектура всей системы должна отвечать следующим требованиям:

1) централизованная база данных;

2) организация доступа к компонентам системы через внешний канал связи (Интернет);

3) разделение бизнес логики, обработки и представления данных;

4) безопасность;

5) надёжность.

Для обеспечения работы системы требуются технические средства для размещения базы данных и серверной части системы. Должна быть обеспечена круглосуточная работа приложения.

Требования к техническому обеспечению серверной части системы:

1) Pentium 2000 МГц или более быстродействующий процессор;

2) оперативная память в объеме 4Гбайт или более;

3) свободное место на жестком диске в объеме не менее 20 ГБ;

4) постоянное подключение к сети Интернет.

Требования к программному обеспечению серверной части:

1) OC Windows 10 сборкой 0.18000 или выше;

2) СУБД Microsoft SQL Server версии 11.0 или выше.

Требования к техническому и программному обеспечению устройства клиента:

1) стабильное подключение к сети Интернет;

2) браузер с поддержкой HTML5 и JavaScript.

1.4 Выводы раздела

В данном разделе был проведён анализ литературных источников и анализ аналогов, сформированы требования к программному средству.

2 Моделирование предметной области и разработка функциональных требований

2.1 Функциональная модель программного средства

Функциональная модель программного средства представлена в виде  
диаграммы вариантов использования и информационной модели предметной  
области. Варианты использования отражают функциональность системы в ответ на внешние воздействия с точки зрения получения значимого результата  
для пользователей. Информационная модель предметной области в дальнейшем будет использоваться при проектировании базы данных для программного средства.

**2.1.1** Варианты использования программного средства

Варианты использования данного программного средства различаются в зависимости от статуса пользователя, который зависит от следующих факторов:

* была ли пользователем пройдена авторизация;
* роль авторизованного пользователя.

Все возможные статусы, которые предусмотрены в программном средстве, представлены на рисунке 2.1.



Рисунок 2.1 - Все возможные статусы пользователя

Любой пользователь, использующий приложение, изначально имеет статус «Гость». Такой пользователь не имеет доступа к:

* подбору фильмов на основе своих пожеланий;
* комментированию и оцениванию фильма;
* добавлению фильма в каталог «Буду смотреть»;
* просмотру и редактированию профиля;
* отображению списка фильмов, добавленных в каталог «Буду смотреть», или списка фильмов, оценённых пользователем;
* отображению списка фильмов, оценённых пользователем;
* добавлению, редактированию и удалению фильма или участника;

Для смены данного статуса на другой требуется пройти авторизацию. Все действия, которые может выполнять пользователь со статусом «Гость» представлены на рисунке 2.2.



Рисунок 2.2 - Диаграмма вариантов использования программного средства пользователем со статусом «Гость»

На представленной диаграмме можно заметить, что пользователь со статусом «Гость» в любой момент имеет возможность перейти на:

* регистрацию;
* авторизацию;
* просмотр списка фильмов, доступных для просмотра.

При просмотре списка фильмов, доступных для просмотра пользователь может воспользоваться сортировкой данного списка или же его фильтрацией. Также из данного просмотра пользователь может перейти на просмотр подробной информации о фильме, где имеются следующие возможности:

* переход на просмотр подробной информации об участнике, который связан с текущим фильмом;
* просмотр фильма в онлайне;
* просмотр списка комментариев.

В случае прохождения пользователем авторизации, его статус меняется либо на «Пользователь», либо на «Администратор», а это, в свою очередь, зависит от роли пользователя, которая хранится в базе данных.

Пользователь со статусом «Пользователь» не имеет доступа только к

* добавлению, редактированию и удалению фильма;
* добавлению, редактированию и удалению участника.

Все действия, которые может выполнять пользователь со статусом «Пользователь», представлены на рисунке 2.3.

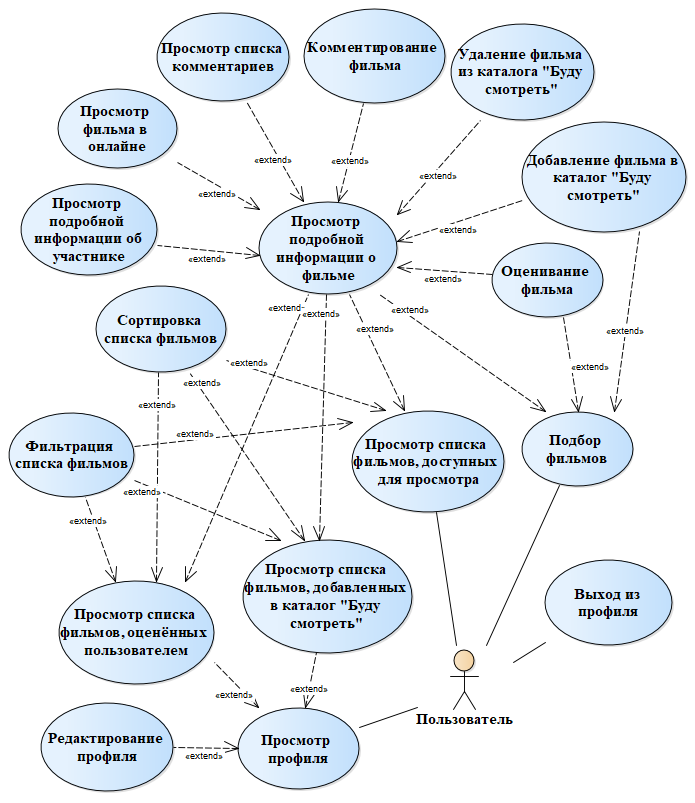


Рисунок 2.3 – Диаграмма вариантов использования программного средства пользователем со статусом «Пользователь»

На представленной диаграмме можно заметить, что пользователь со статусом «Пользователь» в любой момент имеет следующие возможности:

* выход из профиля;
* просмотр профиля;
* подбор фильма;
* просмотр списка фильмов, доступных для просмотра.

При просмотре профиля пользователь может отредактировать его, а также просмотреть список оценённых пользователем фильмов или список добавленных в каталог «Буду смотреть» фильмов, где доступна их сортировка и фильтрации и где имеется возможность перейти на просмотр подробной информации о фильме.

При подборе фильмов имеется возможность оценивание фильма, его добавление в каталог «Буду смотреть» или просмотр подробной информации о нём.

При просмотре списка фильмов, доступных для просмотра, пользователь может воспользоваться сортировкой данного списка или же его фильтрацией. Также из данного просмотра пользователь может перейти на просмотр подробной информации о фильме, где имеются следующие возможности:

* переход на просмотр подробной информации об участнике, который связан с текущим фильмом;
* просмотр фильма в онлайне;
* просмотр списка комментариев;
* комментирование фильма;
* добавление фильма в каталог «Буду смотреть» или же его удаление из него;
* оценивание фильма.

В программном средстве должна быть предусмотрена административная часть. К ней имеют доступ только те пользователи, которые обладают статусом «Администратор». Функциональность, доступная пользователям со статусом «Администратор», представлена на рисунке 2.4.

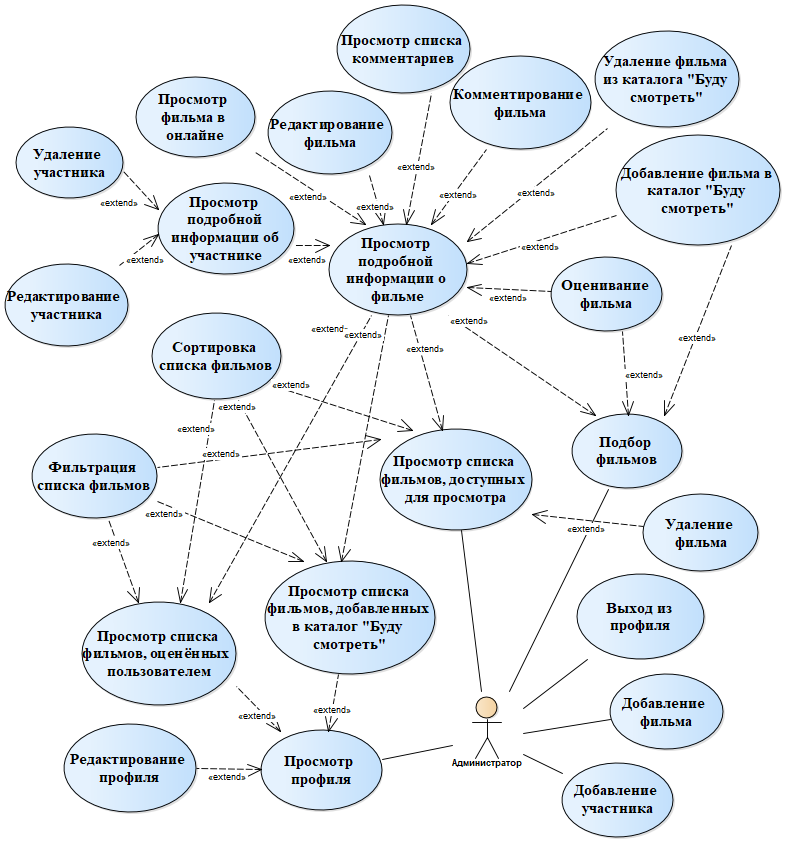


Рисунок 2.4 – Диаграмма вариантов использования программного средства пользователем со статусом «Администратор»

На представленной диаграмме можно заметить, что пользователь со статусом «Администратор» в любой момент имеет следующие возможности:

* выход из профиля;
* добавление фильма;
* добавление участника;
* просмотр профиля;
* подбор фильма;
* просмотр списка фильмов, доступных для просмотра.

При просмотре профиля пользователь может отредактировать его, а также просмотреть список оценённых пользователем фильмов или список добавленных в каталог «Буду смотреть» фильмов, где доступна их сортировка и фильтрации и где имеется возможность перейти на просмотр подробной информации о фильме.

При подборе фильмов имеется возможность оценивание фильма, его добавление в каталог «Буду смотреть» или просмотр подробной информации о нём.

При просмотре списка фильмов, доступных для просмотра, пользователь может воспользоваться сортировкой данного списка, его фильтрацией или же удалением фильма. Также из данного просмотра пользователь может перейти на просмотр подробной информации о фильме, где имеются следующие возможности:

* редактирование фильма;
* переход на просмотр подробной информации об участнике, который связан с текущим фильмом;
* просмотр фильма в онлайне;
* просмотр списка комментариев;
* комментирование фильма;
* добавление фильма в каталог «Буду смотреть» или же его удаление из него;
* оценивание фильма.

При просмотре подробной информации об участнике у пользователя имеется возможность отредактировать информацию о нём или же её удалить.

Таким образом, пользователь со статусом «Администратор» обладает возможностью пользоваться всеми функции программного средства.

**2.1.2** Разработка инфологической модели базы данных

Исходя из необходимости использования в проектируемом приложении базы данных, требуется разработать её инфологическую модель, за основу которой будет взята предметная область проекта.

Предментная область разрабатываемого программного средства включает в себя следующие сущности и их атрибуты:

1. пользователь:

* уникальный идентификатор;
* никнейм;
* хешированный пароль;
* идентификатор роли пользователя;
* дата регистрации;
* пол;
* дата рождения;

1. роль пользователя:

* уникальный идентификатор;
* наименование;

1. фильм:

* уникальный идентификатор;
* русскоязычное название;
* оригинальное название;
* путь к изображению постера;
* слоган;
* дата мировой премьеры;
* бюджет;
* сборы;
* продолжительность;
* идентификатор возрастного ограничения;
* количество просмотров;
* сумма всех оценок;
* рейтинг (средняя оценка фильма);
* описание;
* дата добавления;

1. возрастное ограничение:

* уникальный идентификатор;
* наименование;

1. жанр:

* уникальный идентификатор;
* наименование;

1. жанр фильма:

* идентификатор фильма;
* идентификатор жанра;

1. качество видео:

* уникальный идентификатор;
* наименование;

1. качество видео фильма:

* идентификатор фильма;
* идентификатор качества видео;
* путь к видео фильма;

1. страна:

* уникальный идентификатор;
* наименование;

1. страна-производитель фильма:

* идентификатор фильма;
* идентификатор страны;

1. компания:

* уникальный идентификатор;
* наименование;

1. компания-производитель фильма:

* идентификатор фильма;
* идентификатор компании;

1. участник:

* уникальный идентификатор;
* имя;
* фамилия;
* рост;
* день рождения;
* пол;
* путь к изображению участника;

1. роль:

* уникальный идентификатор;
* наименование;

1. роль участника фильма:

* идентификатор фильма;
* идентификатор участника;
* идентификатор роли;

1. просмотренный пользователем фильм:

* идентификатор пользователя;
* идентификатор фильма;
* просмотрен ли;
* оценка;

1. комментарий фильма:

* идентификатор комментария;
* идентификатор фильма;
* идентификатор пользователя;
* текст;
* дата добавления;

1. любимые фильмы:

* идентификатор пользователя;
* идентификатор фильма;

1. любимые участники:

* идентификатор пользователя;
* идентификатор участника;

1. любимые жанры:

* идентификатор пользователя;
* идентификатор жанра.

2.2 Спецификация функциональных требований

С учетом требований, определенных в подразделе 1.3, представим детализацию функций проектируемого программного средства.

Для детализации функций рассмотрим основные требования, предъявляемые к каждой функции программного средства как с точки зрения внутренней организации системы, так и с точки зрения взаимодействия системы с пользователем.

**2.2.1** Функция регистрации

Функция регистрации должна быть реализована с учётом следующих требований:

1. процесс регистрации инициируется пользователем системы со статусом «Гость»;
2. для регистрации пользователь обязан ввести уникальный никнейм, а также дважды пароль, и всё это должно быть проверено на корректность;
3. корректность введённых данных должна быть проверена при помощи встроенных инструментов разработки;
4. никнейм считается корректным, если его длина не превышает 16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита, цифры и пробельный символ;
5. пароль считается корректным, если состоит из 10-16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита и цифры;
6. в случае некорректности введённых данных, пользователю должно отобразиться сообщение об этом с предложением повторить попытку;
7. в случае неуникальности никнейма, пользователю должно отобразиться сообщение об этом с предложением изменить никнейм;
8. после успешной регистрации пользователь должен быть автоматически авторизирован и должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**2.2.2** Функция авторизация

Функция авторизации должна быть реализована с учётом следующих требований:

1. инициатором является пользователь со статусом «Гость»;
2. для авторизации пользователь обязан ввести никнейм и пароль, которые должны быть проверены на корректность;
3. корректность введённых данных должна быть проверена при помощи встроенных инструментов разработки;
4. никнейм считается корректным, если его длина не превышает 16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита, цифры и пробельный символ;
5. пароль считается корректным, если состоит из 10-16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита и цифры;
6. в случае некорректности введённых данных, пользователю должно отобразиться сообщение об этом с предложением повторить попытку;
7. необходимо удостовериться в существовании пользователя, с никнеймом и паролем, соответствующим введённым;
8. в случае, если пользователь с никнеймом и паролем, соответствующим введённым, не существует, должно отобразиться сообщение о некорректности введённых данных;
9. после успешной авторизации статус пользователя должен быть изменён на «Пользователь» или «Администратор», исходя из роли его профиля;
10. после успешной авторизации должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**2.2.3** Отображение списка фильмов, доступных для просмотра

Данная функция должна быть реализована с учётом следующих требований:

1. в списке фильмов должна отображаться краткая информация о каждом из них;
2. краткая информация о фильме содержит:

* изображение постера;
* русскоязычное название фильма;
* год премьеры;
* жанры – текст перечисления через запятую максимально трёх жанров;
* рейтинг – число, представленное десятичной дробью с двумя цифрами после запятой и фон которого зависит от значения числа: от 1 до 4 – красный, от 4.01 до 7 – серый, от 7.01 до 10 - зелёный;

1. для пользователя со статусом «Пользователь» или «Администратор» в краткой информации дополнительно должна отображаться оценка, которую пользователь выставил фильму, если таковая имеется;
2. должна присутствовать возможность сортировки списка фильмов на основе выбранного пользователем критерия;
3. должна присутствовать возможность фильтрации списка фильмов на основе выбранных пользователем критериев;
4. изначально список отсортирован по убывающей дате премьеры фильма;
5. должна быть реализована пагинация списка с отображением:

* текущей страницы пагинации;
* первой страницы пагинации;
* последней страницы пагинации;

1. на одной странице пагинации список должен содержать максимум 50 фильмов;
2. возле списка должно отображаться общее количество фильмов, которые в нём содержаться;
3. по нажатии название фильма должно происходить перенаправление на отображение подробной информации о нём;
4. для пользователя со статусом «Администратор» возле каждого фильма из списка должна отображаться кнопка «Удалить», по нажатию на которую будет задействована функция удаления фильма.

**2.2.4** Сортировка списка фильмов

Функция сортировки списка фильмов должна быть реализована с учётом следующих требований:

1. сортировка должна выполняться на основе одного из следующих критериев:

* название фильма;
* год премьеры фильма;
* рейтинг фильма;
* дата добавления фильма;

1. сортировка списка фильмов должна происходить при нажатии на выбранный из меню сортировки критерий;
2. сортировка может выполняться по возрастанию или убыванию выбранного критерия;
3. при нажатии на критерий, сортировка по которому выполнена на данный момент, должна выполняться новая сортировка, но уже по обратному типу: если была по возрастанию, то требуется выполнить по убыванию того же самого критерия, и так наоборот;
4. при нажатии на критерий, сортировка по которому на данный момент не выполнена, должна выполняться новая сортировка по возрастанию выбранного критерия;
5. после сортировки должно происходить отображение списка, порядок фильмов в котором изменён на основе выбранного пользователем критерия сортировки.

**2.2.5** Фильтрация списка фильмов

Функция фильтрации списка фильмов должна быть реализована с учётом следующих требований:

1. фильтрация должна выполняться на основе введённых или выбранных пользователем критериев;
2. критериями фильтрации являются:

* название фильма – поле ввода, которые состоит максимально из 255 символов, являться которыми могут прописные и заглавные буквы кириллического алфавита, прописные и заглавные буквы латинского алфавита, а также следующие символы: «- : , ! ? “ ”»;
* диапазон года премьеры – поле для выбора начала и конца фильтра по году премьеры фильма с шагом 1 год без возможности ввода значений;
* жанры – список жанров, в котором можно выбирать несколько вариантов одновременно;
* диапазон продолжительности – поле для выбора начала и конца фильтра по продолжительности фильма с шагом 5 минут без возможности ввода значений;
* возрастные ограничения – список возрастных ограничений, в котором можно выбирать несколько вариантов одновременно;
* диапазон рейтинга – поле для выбора начала и конца фильтра по рейтингу фильма с шагом 1 балл без возможности ввода значений;

1. фильтрация по введённому пользователем названию фильма должна выполняться на основе частичного соответствия русскоязычному или оригинальному названию фильма без учёта регистра;
2. фильтрация по диапазону года премьеры должна выполняться на основе вхождения года премьеры фильма в выбранный пользователем диапазон;
3. фильтрация по жанрам должна выполняться на основе полного соответствия жанра фильма одному из выбранных пользователем из списка;
4. фильтрация по диапазону продолжительности должна выполняться на основе вхождения продолжительности фильма в выбранный пользователем диапазон;
5. фильтрация по возрастным ограничениям должна выполняться на основе полного соответствия возрастного ограничения фильма одному из выбранных пользователем из списка;
6. фильтрация по диапазону рейтинга должна выполняться на основе вхождения рейтинга фильма в выбранный пользователем диапазон;
7. после фильтрации должно происходить отображение нового списка, фильмы в котором соответствуют всем выбранным пользователем критериям фильтрации.

**2.2.6** Отображение подробной информации о фильме

Данная функция должна быть реализована с учётом следующих требований:

1. подробная информация о фильме содержит:

* изображение постера;
* русскоязычное название фильма;
* оригинальное название фильма;
* слоган фильма;
* жанры – текст перечисления через запятую;
* страны-производители – текст перечисления через запятую;
* компании-производители – текст перечисления через запятую;
* дата премьеры – дата формата «dd/mm/yyyy», где «dd» - двухзначный день месяца, «mm» - двухзначный номер месяца, «yyyy» - четырёхзначное число года;
* бюджет;
* сборы;
* продолжительность;
* возрастное ограничение;
* рейтинг – число от 1 до 10, представленное десятичной дробью с двумя цифрами после запятой и цвет которого зависит от значения числа: от 1 до 4 – красный, от 4.01 до 7 – серый, от 7.01 до 10 - зелёный;
* количество просмотров;
* описание;
* список участников фильма;

1. в списке участников фильма должна отображаться краткая информация о каждом из них:

* изображение участника;
* имя и фамилия участника;
* роль участника в фильме;

1. список участников должен быть представлен таблицей с возможностью скроллинга без заголовков, где краткая информация об участнике занимает ровно одну строку и где максимально видимое количество строк равняется 8;
2. при нажатии имя и фамилию участника фильма должно происходить перенаправление на отображение подробной информации о выбранном участнике;
3. должна присутствовать возможность просмотреть фильм в онлайне;
4. должна присутствовать возможность просмотреть список комментариев о фильме;
5. для пользователя со статусом «Пользователь» или «Администратор» дополнительно должны быть предусмотрены следующие возможности:

* добавление фильма в каталог «Буду смотреть» или его непосредственное удаление из него;
* оценивание фильма по десятибалльной системе (от 1 до 10);
* комментирование фильма;

1. для пользователя со статусом «Администратор» дополнительно должна быть предусмотрена возможность редактирования фильма.

**2.2.7** Просмотр фильма в онлайне

Просмотр фильма в онлайне должен осуществляться благодаря встроенному видеоплееру, который, в свою очередь, должен поддерживать следующие функции:

1. приостановка видео;
2. возобновление видео;
3. перемотка видео на выбранный пользователем момент;
4. регулировка звука;
5. выбор качества видео;
6. переход в полноэкранный режим видеоплеера;
7. выход из полноэкранного режима;

В случае, когда пользователь поменял качество видео, видеоплеер должен остановить текущее воспроизведение и начать новое с самого начала с разрешением видео соответствующему выбранному пользователем.

**2.2.8** Подбор фильмов на основе пожеланий пользователя

Данная функция должна быть реализована с учётом следующих требований:

1. подбор фильмов может быть инициирован пользователем системы со статусом «Пользователь» или «Администратор»;
2. подбор фильмов осуществляется на основе критериев, выбранных пользователем;
3. критериями, на основе которых пользователь может выполнять подбор фильмов, являются:

* диапазон года премьеры – поле для выбора начала и конца фильтра по году премьеры фильма с шагом 1 год без возможности ввода значений;
* жанры – список жанров, в котором можно выбирать несколько вариантов одновременно;
* диапазон продолжительности – поле для выбора начала и конца фильтра по продолжительности фильма с шагом 5 минут без возможности ввода значений;
* возрастные ограничения – список возрастных ограничений, в котором можно выбирать несколько вариантов одновременно;
* диапазон рейтинга – поле для выбора начала и конца фильтра по рейтингу фильма с шагом 1 балл без возможности ввода значений;

1. дополнительно при подборе должна учитываться следующая информация:

* личная информация пользователя, указанная им же в своём профиле: дата рождения, пол, любимые жанры, любимые участники фильмов, любимые фильмы;
* все фильмы, добавленные в каталог «Буду смотреть»;
* все оценённые пользователем фильмы;

1. фильмы, добавленные в каталог «Буду смотреть» или оценённые пользователем, не могут являться результатом подбора;
2. после подбора фильмов на основе выбранных пользователем критериев должно происходить отображение краткой информации о первом фильме из списка подобранных;
3. должна иметься возможность отобразить следующий фильм из списка подобранных;
4. если список подобранных фильмов закончился, то должно отобразиться соответствующее сообщение;
5. если не был подобран ни один фильм, то должно отобразиться соответствующее сообщение;
6. краткая информация о подобранном фильме содержит:

* изображение постера;
* русскоязычное название фильма;
* год премьеры;
* жанры – текст перечисления через запятую максимально трёх жанров;
* рейтинг – число, представленное десятичной дробью с двумя цифрами после запятой и фон которого зависит от значения числа: от 1 до 4 – красный, от 4.01 до 7 – серый, от 7.01 до 10 - зелёный;

1. должна быть предусмотрена возможность добавления подобранного фильма в каталог «Буду смотреть», а также возможность оценивания фильма;
2. по нажатии на название фильма должно происходить перенаправление на отображение подробной информации о нём.

**2.2.9** Добавление фильма

Функция добавления фильма должна быть реализована с учётом следующих требований:

1. процесс добавления фильма может быть инициирован пользователем со статусом «Администратор»;
2. для добавления фильма обязательно требуется указать следующую информацию о фильме:

* изображение постера – загружаемый пользователем файл изображения в формате png;
* русскоязычное название – последовательность длинной от 3 до 100 символов, состоящая из букв кириллического алфавита, знаков препинания и пробельных символов;
* оригинальное название – последовательность длинной от 3 до 100 символов, состоящая из букв латинского алфавита, знаков препинания и пробельных символов;
* жанры – список жанров, в котором можно выбирать несколько вариантов одновременно;
* страны-производители – список стран-производителей, в котором можно выбирать несколько вариантов одновременно;
* компании-производители – список компаний-производителей, в котором можно выбирать несколько вариантов одновременно;
* дата премьеры – дата формата «dd/mm/yyyy», где «dd» – двухзначный день месяца, «mm» – двухзначный номер месяца, «yyyy» – четырёхзначное число года;

1. при добавлении дополнительно можно указать следующую информацию о фильме:

* слоган фильма – последовательность длинной от 3 до 100 символов, состоящая из букв латинского или кириллического алфавита, знаков препинания и пробельных символов;
* бюджет – число, максимальное количество цифр которого является 21 и которое выражается в долларах;
* сборы – число, максимальное количество цифр которого является 21 и которое выражается в долларах;
* продолжительность – число, максимальное количество цифр которого является 21 и которое выражается в минутах;
* возрастное ограничение – список возрастных ограничений, в котором можно выбрать только один вариант;
* описание – последовательность длинной от 0 до 2000 символов, состоящая из букв латинского и кириллического алфавита, знаков препинания и пробельных символов;
* список качества видео;
* список участников фильма;

1. список качества видео составляется путём поочерёдного выбора качества видео из списка доступных качеств видео и заполнения текстового поля информацией о файловом пути к видео;
2. список участников составляется путём поочерёдного выбора участника из списка доступных участников и роли фильма из списка доступных ролей;
3. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
4. после добавления фильм должен отобразиться в списке фильмов, доступных для просмотра.

**2.2.10** Редактирование фильма

Функция редактирования фильма должна быть реализована с учётом следующих требований:

1. процесс редактирования фильма может быть инициирован пользователем со статусом «Администратор»;
2. при редактировании информации о фильме все поля и значения должны быть корректно загружены и отображены;
3. информацией о фильме, которая является обязательной для указания в нём, является:

* изображение постера – загружаемый пользователем файл изображения в формате png;
* русскоязычное название – последовательность длинной от 3 до 100 символов, состоящая из букв кириллического алфавита, знаков препинания и пробельных символов;
* оригинальное название – последовательность длинной от 3 до 100 символов, состоящая из букв латинского алфавита, знаков препинания и пробельных символов;
* жанры – список жанров, в котором можно выбирать несколько вариантов одновременно;
* дата премьеры – дата формата «dd/mm/yyyy», где «dd» – двухзначный день месяца, «mm» – двухзначный номер месяца, «yyyy» – четырёхзначное число года;

1. информацией о фильме, которая является опциональной для указания в нём, является:

* слоган фильма – последовательность длинной от 3 до 100 символов, состоящая из букв латинского или кириллического алфавита, знаков препинания и пробельных символов;
* страны-производители – список стран-производителей, в котором можно выбирать несколько вариантов одновременно;
* компании-производители – список компаний-производителей, в котором можно выбирать несколько вариантов одновременно;
* бюджет – число, максимальное количество цифр которого является 21 и которое выражается в долларах;
* сборы – число, максимальное количество цифр которого является 21 и которое выражается в долларах;
* продолжительность – число, максимальное количество цифр которого является 21 и которое выражается в минутах;
* возрастное ограничение – список возрастных ограничений, в котором можно выбрать только один вариант;
* описание – последовательность длинной от 0 до 2000 символов, состоящая из букв латинского и кириллического алфавита, знаков препинания и пробельных символов;
* список качества видео;
* список участников фильма;

1. список качества видео составляется путём поочерёдного выбора качества видео из списка доступных качеств видео и заполнения текстового поля информацией о файловом пути к видео;
2. список участников составляется путём поочерёдного выбора участника из списка доступных участников и роли фильма из списка доступных ролей;
3. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
4. должна быть предусмотрена возможность отмены внесения изменений до окончания редактирования, в случае которого должно происходить перенаправление на отображение списка фильмов, доступных для просмотра;
5. по окончании редактирования новая версия информации о фильме должна быть сохранена в базу данных;
6. после редактирования должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**2.2.11** Удаление фильма

Функция удаления фильма должна быть реализована с учётом следующих требований:

1. процесс удаления фильма может быть инициирован пользователем со статусом «Администратор»;
2. перед удалением пользователь должен подтвердить необходимость выполнения данной функции;
3. после удаления должен быть отображён обновлённый список фильмов.

**2.2.12** Отображение списка комментариев о фильме

Данная функция должна быть реализована с учётом следующих требований:

1. в списке комментариев должна отображаться информация о каждом из них;
2. информация о комментарии содержит:

* никнейм пользователя, который написал комментарий;
* дату написания комментария;
* текст комментария;

1. список комментариев должен быть отсортирован по убыванию даты написания комментария;
2. список комментариев должен быть представлен таблицей с возможностью скроллинга без заголовков, где краткая информация о комментарии занимает ровно одну строку и где максимально видимое количество строк равняется 8;

**2.2.13** Комментирование фильма

Данная функция должна быть реализована с учётом следующих требований:

1. процесс комментирования фильма может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. для комментирования обязательно требуется ввести текст комментария, который является последовательностью длинной от 1 до 255 символов, состоящий из букв латинского или кириллического алфавита, знаков препинания и пробельных символов;
3. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
4. после комментирования отображаемый список комментариев фильма должен быть обновлён.

**2.2.14** Оценивание фильма

Функция оценивания фильма должна быть реализована с учётом следующих требований:

1. процесс комментирования фильма может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. по окончании оценивания рейтинг фильма должен быть пересчитан и сохранен в базе данных, после чего отображаемое значение рейтинга фильма должно быть обновлено.

**2.2.15** Добавление фильма в каталог «Буду смотреть»

Данная функция должна быть реализована с учётом следующих требований:

1. процесс добавления фильма в каталог «Буду смотреть» может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. после осуществления данной функции добавленный в каталог фильм должен отображаться в списке фильмов, добавленных в каталог «Буду смотреть», а также после осуществления данной функции пользователь должен иметь возможность удалить фильм из каталога «Буду смотреть».

**2.2.16** Удаление фильма из каталога «Буду смотреть»

Данная функция должна быть реализована с учётом следующих требований:

1. процесс удаления фильма из каталога «Буду смотреть» может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. после осуществления данной функции удалённый из каталога фильма больше не должен отображаться в списке фильмов, добавленных в каталог «Буду смотреть», а также после осуществления данной функции пользователь должен иметь возможность обратно добавить фильм в каталог «Буду смотреть».

**2.2.17** Просмотр подробной информации об участнике

Данная функция должна быть реализована с учётом следующих требований:

1. подробная информация об участнике содержит:

* изображение участника;
* имя участника;
* фамилия участника;
* рост – число, которое выражается в сантиметрах;
* день рождения – дата формата «dd/mm/yyyy», где «dd» - двухзначный день месяца, «mm» - двухзначный номер месяца, «yyyy» - четырёхзначное число года;
* пол;
* список фильмов с данным участником;

1. в списке фильмов с данным участником должна отображаться краткая информация о каждом из них:

* постер фильма;
* русскоязычное название фильма;
* роль участника в фильме;

1. список фильмов должен быть представлен таблицей с возможностью скроллинга без заголовков, где краткая информация о фильме занимает ровно одну строку и где максимально видимое количество строк равняется 8;
2. при нажатии русскоязычное название фильма должно происходить перенаправление на отображение подробной информации о выбранном фильме;
3. для пользователя со статусом «Администратор» дополнительно должна быть предусмотрена возможность редактирования и удаления участника.

**2.2.18** Добавление участника

Функция добавления фильма должна быть реализована с учётом следующих требований:

1. процесс добавления участника может быть инициирован пользователем со статусом «Администратор»;
2. для добавления участника обязательно требуется указать следующую информацию об участнике:

* изображение участника – загружаемый пользователем файл изображения в формате png;
* имя участника – последовательность длинной от 1 до 100 символов, состоящая из букв кириллического алфавита;
* фамилия участника – последовательность длинной от 1 до 100 символов, состоящая из букв кириллического алфавита;
* день рождения – дата формата «dd/mm/yyyy», где «dd» - двухзначный день месяца, «mm» - двухзначный номер месяца, «yyyy» - четырёхзначное число года;
* пол – вариант из списка, состоящий из трёх значений: «мужской», «женский», «не указан»;

1. при добавлении дополнительно можно указать следующую информацию о фильме:

* рост – число, которое выражается в сантиметрах;
* список фильмов с данным участником;

1. список фильмов с данным участником составляется путём поочерёдного выбора фильма из списка доступных фильмов и роли фильма из списка доступных ролей;
2. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
3. после добавления участник должен быть доступным для его выбора в списке всех участников.

**2.2.19** Редактирование участника

Функция редактирования фильма должна быть реализована с учётом следующих требований:

1. процесс редактирования фильма может быть инициирован пользователем со статусом «Администратор»;
2. при редактировании информации о фильме все поля и значения должны быть корректно загружены и отображены;
3. информацией об участнике, которая является обязательной для указания в нём, является:

* изображение участника – загружаемый пользователем файл изображения в формате png;
* имя участника – последовательность длинной от 1 до 100 символов, состоящая из букв кириллического алфавита;
* фамилия участника – последовательность длинной от 1 до 100 символов, состоящая из букв кириллического алфавита;
* день рождения – дата формата «dd/mm/yyyy», где «dd» - двухзначный день месяца, «mm» - двухзначный номер месяца, «yyyy» - четырёхзначное число года;
* пол – вариант из списка, состоящий из трёх значений: «мужской», «женский», «не указан»;

1. информацией об участнике, которая является опциональной для указания в нём, является:

* рост – число, которое выражается в сантиметрах;
* список фильмов с данным участником;

1. список фильмов с данным участником составляется путём поочерёдного выбора фильма из списка доступных фильмов и роли фильма из списка доступных ролей;
2. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
3. должна быть предусмотрена возможность отмены внесения изменений до окончания редактирования, в случае которого должно происходить перенаправление на отображение списка фильмов, доступных для просмотра;
4. по окончании редактирования новая версия информации об участнике должна быть сохранена в базу данных;
5. после редактирования должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

**2.2.20** Удаление участника

Функция удаления участника должна быть реализована с учётом следующих требований:

1. процесс удаления фильма может быть инициирован пользователем со статусом «Администратор»;
2. перед удалением пользователь должен подтвердить необходимость выполнения данной функции;
3. после удаления должен быть отображён список фильмов, доступных для просмотра.

**2.2.21** Просмотр профиля

Данная функция должна быть реализована с учётом следующих требований:

1. процесс просмотра профиля может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. при просмотре профиля должна отображаться следующая информация:

* никнейм пользователя;
* дата регистрации – дата формата «dd/mm/yyyy», где «dd» – двухзначный день месяца, «mm» – двухзначный номер месяца, «yyyy» – четырёхзначное число года;
* пол;
* дата рождения – дата формата «dd/mm/yyyy», где «dd» – двухзначный день месяца, «mm» – двухзначный номер месяца, «yyyy» – четырёхзначное число года;
* любимые жанры – текст перечисления через запятую;
* любимые участники фильмов – текст перечисления через запятую;
* любимые фильмы – текст перечисления через запятую;

1. должна присутствовать возможность перейти на редактирование профиля;
2. должна присутствовать возможность перейти на отображение списка фильмов, добавленных в каталог «Буду смотреть»;
3. должна присутствовать возможность перейти на отображение списка фильмов, оценённых пользователем.

**2.2.22** Редактирование профиля

Функция редактирования фильма должна быть реализована с учётом следующих требований:

1. процесс редактирования профиля может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. при редактировании информации о профиле все поля и значения должны быть корректно загружены и отображены;
3. информацией о профиле, которая является обязательной для указания в нём, является:

* никнейм пользователя;
* пароль;

1. никнейм считается корректным, если его длина не превышает 16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита, цифры и пробельный символ;
2. пароль считается корректным, если состоит из 10-16 символов, которыми могут быть заглавные и прописные буквы латинского алфавита и цифры;
3. информацией о профиле, которая является опциональной для указания в нём, является:

* пол – вариант из списка, состоящий из трёх значений: «мужской», «женский», «не указан»;
* дата рождения – дата формата «dd/mm/yyyy», где «dd» – двухзначный день месяца, «mm» – двухзначный номер месяца, «yyyy» – четырёхзначное число года;
* любимые жанры – список жанров, в котором можно выбирать несколько вариантов одновременно;
* любимые участники фильмов – список участников, в котором можно выбирать несколько вариантов одновременно;
* любимые фильмы – список фильмов, в котором можно выбирать несколько вариантов одновременно;

1. данные, введенные пользователем, должны проверяться на корректность, и если имеются некорректные данные, то должно отобразиться сообщение об ошибке;
2. должна быть предусмотрена возможность отмены внесения изменений до окончания редактирования, в случае которого должно происходить перенаправление на просмотр профиля;
3. по окончании редактирования новая версия информации об участнике должна быть сохранена в базу данных;
4. после редактирования должно происходить перенаправление на просмотр профиля.

**2.2.23** Отображение списка фильмов, добавленных в каталог «Буду смотреть»

Данная функция должна быть реализована с учётом следующих требований:

1. в списке фильмов, добавленных в каталог «Буду смотреть», должна отображаться краткая информация о каждом из них;
2. должна присутствовать возможность сортировки списка фильмов на основе выбранного пользователем критерия;
3. должна присутствовать возможность фильтрации списка фильмов на основе выбранных пользователем критериев.;
4. изначально список отсортирован по убывающей дате премьеры фильма;
5. должна быть реализована пагинация списка с отображением:

* текущей страницы пагинации;
* первой страницы пагинации;
* последней страницы пагинации;

1. на одной странице пагинации список должен содержать максимум 50 фильмов;
2. возле списка должно отображаться общее количество фильмов, которые в нём содержаться;
3. по нажатии название фильма должно происходить перенаправление на отображение подробной информации о нём.

**2.2.24** Отображение списка фильмов, оценённых пользователем

Данная функция должна быть реализована с учётом следующих требований:

1. в списке фильмов, оценённых пользователем, должна отображаться краткая информация о каждом из них;
2. должна присутствовать возможность сортировки списка фильмов на основе выбранного пользователем критерия;
3. должна присутствовать возможность фильтрации списка фильмов на основе выбранных пользователем критериев.;
4. изначально список отсортирован по убывающей дате премьеры фильма;
5. должна быть реализована пагинация списка с отображением:

* текущей страницы пагинации;
* первой страницы пагинации;
* последней страницы пагинации;

1. на одной странице пагинации список должен содержать максимум 50 фильмов;
2. возле списка должно отображаться общее количество фильмов, которые в нём содержаться;
3. по нажатии название фильма должно происходить перенаправление на отображение подробной информации о нём.

**2.2.25** Выход из профиля

Данная функция должна быть реализована с учётом следующих требований:

1. процесс выхода из профиля может быть инициирован пользователем со статусом «Пользователь» или «Администратор»;
2. возможность выхода из профиля должна быть доступной в любой момент времени, независимо от выполняемой программным средством функции;
3. после выхода из профиля статус пользователя должен быть сменён на «Гость»;
4. после выхода из профиля должно происходить перенаправление на отображение списка фильмов, доступных для просмотра.

2.3 Выводы раздела

В данном разделе была проведена разработка функциональной модели и была разработана спецификация функциональных требований.

3 Проектирование программного средства

3.1 Разработка архитектуры приложения

Для написания любого проекта очень важно, чтобы программа не только правильно и быстро работала, но и была хорошо организована. Сложность, как правило, растет гораздо быстрее размеров программы. И если не позаботиться об этом заранее, то довольно быстро наступает момент, когда ее становится тяжело контролировать. Правильная архитектура экономит очень много сил, времени и денег. Хорошая архитектура – это, прежде всего, выгодная архитектура, делающая процесс разработки и сопровождения программы более простым и эффективным. Программу с хорошей архитектурой легче расширять и изменять, а также тестировать, отлаживать и понимать.

Критерии правильной архитектуры:

* эффективность системы. В первую очередь программа должна решать поставленные задачи и выполнять свои функции в различных условиях. Сюда можно отнести такие характеристики, как надежность, безопасность, производительность, масштабируемость;
* гибкость системы. Любое приложение приходится менять со временем – изменяются требования, добавляются новые. Чем быстрее и удобнее можно внести изменения в существующий функционал, чем меньше проблем и ошибок это вызовет – тем более гибкая и конкурентоспособная система. Изменение одного фрагмента системы не должно влиять на ее другие фрагменты;
* расширяемость системы. Возможность добавлять в систему новые сущности и функции, не нарушая ее основной структуры;
* возможность повторного использования модулей системы;

На основании выдвинутых к программному средству функциональных требований, обзора существующих аналогов и анализа литературных источников, было принято решение о проектировании кроссплатформенного программного средства, функционирующего в сети Интернет и доступного с любого вида устройства. В качестве архитектурного стиля программного средства была выбрана комбинация клиент-серверного и объектно-ориентированного подходов.

Поскольку в качестве основной платформы функционирования приложения был выбран веб, логично предположить, что основной архитектурой приложения будет являться клиент-серверный подход, реализующий концепцию «клиент-сервер» с двумя независимыми друг от друга точками входа. Данный подход позволяет подключить к серверу то количество пользователей, которое позволяет техническое оснащение сервера. В качестве основного шаблона проектирования был принят шаблон MVC.  
MVC (Model-View-Controller) – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных  
компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. Визуальное представление шаблона MVC представлено на рисунке 3.1.



Рисунок 3.1 - Визуальное представление шаблона проектирования MVC

Каждый из трех компонентов шаблона выполняет строго отведенную  
для него функцию.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние. Модель даёт контроллеру представление данных, которые запросил пользователь. Модель данных будет одинаковой, вне зависимости от того, как мы хотим представлять их пользователю. Поэтому мы выбираем любой доступный вид для отображения данных. Модель содержит наиболее важную часть логики нашего приложения, логики, которая решает задачу, с которой мы имеем дело.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели. Вид обеспечивает различные способы представления данных, которые получены из модели. Он может быть  
шаблоном, который заполняется данными. Может быть несколько различных  
видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений. Контроллер управляет запросами пользователя. Его основная функция – вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

Веб-приложение обычно состоит из набора контроллеров, моделей и видов. Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуации.

Выбор шаблона MVC значительно облегчает разработку приложения с  
клиент-серверной архитектурой, поскольку сразу создает хорошо обозначенное разграничение между основными субъектами данного подхода: клиентом и сервером. К тому же MVC позволяет качественно и быстро отладить и протестировать приложение.

3.2 Разработка даталогической и физической моделей базы данных

На даталогическом уровне модель предметной области представляется в привязке к конкретной модели данных и описывает способ организации данных безотносительно их физического размещения в конкретной СУБД. В программном средстве, описываемом данным дипломным проектом, будет использована реляционная модель данных.

Реляционная модель данных основана на табличном способе представлении данных, где каждая строка таблицы представляет собой запись с уникальным идентификатором, который именуется ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута.

В реляционной модели данных логическая структура хранения отличается от физической. Благодаря этому создаётся возможность управления физической системой хранения, не меняя данных, которые содержатся в логической структуре. Например, изменение имени файла базы данных не повлияет на хранящиеся в нем таблицы.

Модель даталогического уровня, которая представлена в таблицах 3.1-3.20, основана на инфологической модели, описание которой приведено в пункте 2.1.2.

Таблица 3.1 – Даталогическая структура сущности «User»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| Nickname | Уникальное имя пользователя | Текстовый |
| PasswordHash | Хешированный пароль | Текстовый |
| RoleId | Идентификатор роли пользователя | Текстовый |
| RegistrationDate | Дата регистрации пользователя | Дата/время |
| Gender | Пол пользователя | Логический |
| Birthday | Дата рождения пользователя | Дата/время |

Сущность «User», представленная в таблице 3.1, предназначена для хранения информации о профилях пользователей. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор пользователя. Данная сущность связана с сущностью «UserRole» через поле «RoleId», хранящее уникальный идентификатор роли.

Таблица 3.2 – Даталогическая структура сущности «UserRole»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| Name | Уникальное название роли | Текстовый |

Сущность «UserRole», представленная в таблице 3.2, предназначена для хранения информации о роли, которая может быть у пользователя. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор роли. Название роли пользователя, представленное атрибутом Name, также является уникальным.

Таблица 3.3 – Даталогическая структура сущности «Film»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| RussianName | Русскоязычное название фильма | Текстовый |
| OriginalName | Оригинальное название фильма | Текстовый |
| PosterUrl | Путь к изображению постера | Текстовый |
| Slogan | Слоган фильма | Текстовый |
| WorldPremier | Дата мировой премьеры фильма | Дата/время |
| Budget | Бюджет фильма | Числовой |
| Fees | Сборы фильма | Числовой |
| Duration | Продолжительность фильма | Числовой |
| AgeRatingId | Идентификатор возрастного ограничения фильма | Текстовый |
| Views | Количество просмотров фильма | Числовой |
| RatingsSum | Сумма всех оценок | Числовой |
| Rating | Рейтинг (средняя оценка) фильма | Числовой |
| Description | Описание фильма | Текстовый |
| AddingDate | Дата добавления фильма | Дата/время |

Сущность «Film», представленная в таблице 3.3, предназначена для хранения информации о фильме. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор фильма. Данная сущность связана с сущностью «AgeRating» через поле «AgeRatingId», хранящее уникальный идентификатор возрастного ограничения.

Таблица 3.4 – Даталогическая структура сущности «AgeRating»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| Name | Название возрастного рейтинга | Текстовый |

Сущность «AgeRating», представленная в таблице 3.4, предназначена для хранения информации о возрастном ограничении, которое может быть у фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор роли. Название возрастного ограничения, представленное атрибутом Name, также является уникальным.

Таблица 3.5 – Даталогическая структура сущности «Genre»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Числовой |
| Name | Уникальное название жанра | Текстовый |

Сущность «Genre», представленная в таблице 3.5, предназначена для хранения информации о жанре, который может быть у фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор жанра. Название жанра, представленное атрибутом Name, также является уникальным.

Таблица 3.6 – Даталогическая структура сущности «FilmGenre»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| FilmId | Идентификатор фильма | Текстовый |
| GenreId | Идентификатор жанра | Числовой |

Сущность «FilmGenre», представленная в таблице 3.6, предназначена для хранения информации о жанре конкретного фильма. Первичным ключом данной сущности является уникальное сочетание полей «FilmId» и «GenreId», хранящих уникальный идентификатор фильма и уникальный идентификатор жанра соответственно.

Таблица 3.7 – Даталогическая структура сущности «VideoQuality»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Числовой |
| Name | Название качества видео | Текстовый |

Сущность «VideoQuality», представленная в таблице 3.7, предназначена для хранения информации о качестве видео, который может быть у фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор качества видео. Название качества видео, представленное атрибутом Name, также является уникальным.

Таблица 3.8 – Даталогическая структура сущности «FilmVideoQuality»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| FilmId | Идентификатор фильма | Текстовый |
| VideoQualityId | Идентификатор качества видео | Числовой |
| VideoUrl | Путь к видео | Текстовый |

Сущность «FilmVideoQuality», представленная в таблице 3.8, предназначена для хранения информации о качестве видео конкретного фильма и пути к нему. Первичным ключом данной сущности является уникальное сочетание полей «FilmId» и «VideoQualityId», хранящих уникальный идентификатор фильма и уникальный идентификатор качества видео соответственно.

Таблица 3.9 – Даталогическая структура сущности «Country»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Числовой |
| Name | Уникальное название страны | Текстовый |

Сущность «Country», представленная в таблице 3.9, предназначена для хранения информации о стране, которая может быть связана с созданием фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор страны. Название страны, представленное атрибутом Name, также является уникальным.

Таблица 3.10 – Даталогическая структура сущности «FilmCountry»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| FilmId | Идентификатор фильма | Текстовый |
| CountryId | Идентификатор страны | Числовой |

Сущность «FilmCountry», представленная в таблице 3.10, предназначена для хранения информации о стране-производителе конкретного фильма. Первичным ключом данной сущности является уникальное сочетание полей «FilmId» и «CountryId», хранящих уникальный идентификатор фильма и уникальный идентификатор страны соответственно.

Таблица 3.11 – Даталогическая структура сущности «Company»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Числовой |
| Name | Уникальное название компании | Текстовый |

Сущность «Country», представленная в таблице 3.11, предназначена для хранения информации о компании, которая может быть связана с созданием фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор компании. Название компании, представленное атрибутом Name, также является уникальным.

Таблица 3.12 – Даталогическая структура сущности «FilmCompany»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| FilmId | Идентификатор фильма | Текстовый |
| CompanyId | Идентификатор компании | Числовой |

Сущность «FilmCompany», представленная в таблице 3.12, предназначена для хранения информации о компании-производителе конкретного фильма. Первичным ключом данной сущности является уникальное сочетание полей «FilmId» и «CompanyId», хранящих уникальный идентификатор фильма и уникальный идентификатор компании соответственно.

Таблица 3.13 – Даталогическая структура сущности «Participater»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| Name | Имя участника | Текстовый |
| Surname | Фамилия участника | Текстовый |
| Growth | Рост участника | Числовой |
| Birthday | День рождения | Дата/время |
| Gender | Пол участника | Логический |
| PhotoUrl | Путь к изображению участника | Текстовый |

Сущность «Participater», представленная в таблице 3.13, предназначена для хранения информации об участнике фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор участника.

Таблица 3.14 – Даталогическая структура сущности «Role»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Числовой |
| Name | Уникальное название роли в фильме | Текстовый |

Сущность «Role», представленная в таблице 3.14, предназначена для хранения информации о роли, которая может быть связана с участником фильма. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор роли. Название роли, представленное атрибутом Name, также является уникальным.

Таблица 3.15 – Даталогическая структура сущности «FilmParticipaterRole»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| FilmId | Идентификатор фильма | Текстовый |
| ParticipaterId | Идентификатор участника | Текстовый |
| RoleId | Идентификатор роли | Числовой |

Сущность «FilmParticipaterRole», представленная в таблице 3.15, предназначена для хранения информации об участнике конкретного фильма и его роли в нём. Первичным ключом данной сущности является уникальное сочетание полей «FilmId», «ParticipaterId» и «RoleId», хранящих уникальный идентификатор фильма, уникальный идентификатор участника и уникальный идентификатор роли соответственно.

Таблица 3.16 – Даталогическая структура сущности «WatchedFilm»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| UserId | Идентификатор пользователя | Текстовый |
| FilmId | Идентификатор фильма | Текстовый |
| Watched | Просмотрен ли фильм | Логический |
| Rating | Оценка фильму | Числовой |

Сущность «WatchedFilm», представленная в таблице 3.16, предназначена для хранения информации о просмотренном фильме конкретного пользователя. Первичным ключом данной сущности является уникальное сочетание полей «UserId» и «FilmId», хранящих уникальный идентификатор пользователя и уникальный идентификатор фильма соответственно.

Таблица 3.17 – Даталогическая структура сущности «Comment»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| Id | Уникальный идентификатор | Текстовый |
| UserId | Идентификатор пользователя | Текстовый |
| FilmId | Идентификатор фильма | Текстовый |
| Text | Текст комментария к фильму | Логический |
| AddingDate | Дата добавления комментария | Дата/время |

Сущность «Comment», представленная в таблице 3.17, предназначена для хранения информации о комментарии конкретного пользователя о конкретном фильме. Первичным ключом данной сущности является поле Id, хранящее уникальный идентификатор комментария.

Таблица 3.18 – Даталогическая структура сущности «FavouriteFilms»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| UserId | Идентификатор пользователя | Текстовый |
| FilmId | Идентификатор фильма | Текстовый |

Сущность «FavoutiteFilms», представленная в таблице 3.18, предназначена для хранения информации о любимом фильме конкретного пользователя. Первичным ключом данной сущности является уникальное сочетание полей «UserId» и «FilmId», хранящих уникальный идентификатор пользователя и уникальный идентификатор фильма соответственно.

Таблица 3.19 – Даталогическая структура сущности «FavouriteParticipaters»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| UserId | Идентификатор пользователя | Текстовый |
| ParticipaterId | Идентификатор участника | Текстовый |

Сущность «FavoutiteParticipaters», представленная в таблице 3.19, предназначена для хранения информации о любимом участнике конкретного пользователя. Первичным ключом данной сущности является уникальное сочетание полей «UserId» и «ParticipaterId», хранящих уникальный идентификатор пользователя и уникальный идентификатор участника соответственно.

Таблица 3.20 – Даталогическая структура сущности «FavouriteGenres»

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение атрибута | Тип атрибута |
| UserId | Идентификатор пользователя | Текстовый |
| GenreId | Идентификатор жанра | Числовой |

Сущность «FavoutiteGenres», представленная в таблице 3.20, предназначена для хранения информации о любимом жанре конкретного пользователя. Первичным ключом данной сущности является уникальное сочетание полей «UserId» и «GenreId», хранящих уникальный идентификатор пользователя и уникальный идентификатор жанра соответственно.

Физический уровень моделирования БД описывает конкретные таблицы, связи, индексы, методы хранения, настройки производительности, безопасности. Описывать физическую модель требуется с помощью терминов и типов данных присущих конкретной СУБД.

3.3 Разработка алгоритма приложения и алгоритмов отдельных модулей

Статус пользователя определяет функции, которые ему доступны в программном средстве, поэтому функции регистрации и авторизации являются крайне важными.

Алгоритм регистрации пользователя представлен на рисунке 3.2. Для выполнения регистрации пользователю требуется корректно заполнить все обязательные поля формы, после чего нажать на соответствующую кнопку регистрации. После этого данные, введённые пользователем, отравятся серверной части программного средства, где они снова будут проверены на корректность. Если данные окажутся некорректными, то произойдёт отправка клиентской части программного средства сообщения о неуспешной регистрации, после чего произойдёт его вывод пользователю. Если же данные корректны, то начнётся процесс добавления пользователя в базу данных, в случае неуспеха которого повторятся действия, которые совершаются при отправке пользователем некорректных данных. В случае успеха добавления произойдёт отправка, а после и вывод сообщения об успешной регистрации.



Рисунок 3.2 - Схема алгоритма регистрации пользователя

В случае успешной регистрации, пользователь будет перенаправлен на страницу авторизации. Сразу после успешной регистрации пользователь может авторизоваться на сайте.

Алгоритм авторизации пользователя представлен на рисунке 3.3.



Рисунок 3.3 - Схема алгоритма авторизации пользователя

Как видно из схемы алгоритма, для выполнения авторизации пользователю требуется корректно заполнить обязательные поля формы, после чего нажать на соответствующую кнопку авторизации. После этого данные, введённые пользователем, отравятся серверной части программного средства, где они снова будут проверены на корректность. Если данные окажутся некорректными, то произойдёт отправка клиентской части программного средства сообщения о неуспешной авторизации, после чего произойдёт его вывод пользователю. Если же данные корректны, то начнётся процесс поиска пользователя в базе данных, в случае неуспеха которого повторятся действия, которые совершаются при отправке пользователем некорректных данных. В случае успеха поиска произойдёт отправка, а после и вывод сообщения об успешной авторизации.

В случае успешной авторизации, пользователю станут доступны функции согласно статусу его профиля.

Пользователь имеет возможность редактировать данные своего профиля. Алгоритм редактирования профиля пользователя приведён на рисунке 3.4. Для выполнения редактирования профиля пользователь должен изменить значения полей, после чего нажать на соответствующую кнопку редактирования профиля. После этого данные, введённые пользователем, отравятся серверной части программного средства, где они снова будут проверены на корректность. Если данные окажутся некорректными, то произойдёт отправка клиентской части программного средства сообщения о неуспешном редактировании профиля, после чего произойдёт его вывод пользователю. Если же данные корректны, то начнётся процесс поиска пользователя в базе данных, в случае неуспеха которого повторятся действия, которые совершаются при отправке пользователем некорректных данных. В случае успеха поиска произойдёт изменение данных пользователя в базе данных, затем произойдёт отправка, а после и вывод сообщения об успешном редактировании профиля.



Рисунок 3.4 - Схема алгоритма редактирования профиля

В случае удачного редактирования профиля, пользователь будет перенаправлен на просмотр профиля.

Пользователь со статусом «Администратор» может удалить доступный для просмотра фильм. Алгоритм удаления фильма приведён на рисунке 3.5.



Рисунок 3.5 - Схема алгоритма удаления фильма

Для выполнения данной функции требуется нажать на соответствующую кнопку возле нужного фильма из списка доступных для просмотра фильмов, после чего подтвердить выполнение данного действия. После этого данные о выбранном пользователе фильме отравятся серверной части программного средства. Далее начнётся процесс поиска фильма в базе данных, в случае неуспеха которого произойдёт отправка клиентской части программного средства сообщения о неуспешном удалении фильма, после чего произойдёт его вывод пользователю. В случае успеха поиска произойдёт удаление информации о фильме из базы данных, затем произойдёт отправка, а после и вывод сообщения об успешном удалении фильма.

В случае успешного удаления фильма пользователь будет перенаправлен на просмотр списка фильмов, доступных для просмотра, где будет отображено сообщение об успешном удалении фильма.

Согласно алгоритмам, приведённым ранее, будет реализована соответствующая функциональность в разрабатываемом программном средстве.

3.4 Выводы раздела

В данном разделе была разработана архитектура приложения, даталогическая и физическая модели базы данных, алгоритм приложения и алгоритмы отдельных модулей.

4 Разработка программного средства

4.1 Язык программирования C# и платформа ASP.NET Core

**4.1.1** Язык программирования C#

Для написания программного средства был выбран язык программирования C#. На сегодняшний момент данный язык программирования является одним из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 9.0, которая вышла 10 ноября 2020 года вместе с релизом .NET 5.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Данный язык программирования является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

* поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому при определенных условиях мы можем сделать отдельные модули одного приложения на отдельных языках;
* кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент - .NET 5 поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen;
* мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET;
* разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core;
* производительность. Согласно ряду тестов веб-приложения, на .NET в ряде категорий сильно опережают веб-приложения, построенные с помощью других технологий. Приложения на .NET в принципе отличаются высокой производительностью.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

Код на C# компилируется в приложения или сборки с расширениями exe или dll на языке CIL. Далее при запуске на выполнение подобного приложения происходит JIT-компиляция (Just-In-Time) в машинный код, который затем выполняется. При этом, поскольку наше приложение может быть большим и содержать кучу инструкций, в текущий момент времени будет компилироваться лишь та часть приложения, к которой непосредственно идет обращение. Если мы обратимся к другой части кода, то она будет скомпилирована из CIL в машинный код. При том уже скомпилированная часть приложения сохраняется до завершения работы программы. В итоге это повышает производительность.

**4.1.2** Платформа ASP.NET Core

ASP.NET – платформа разработки веб-приложений, в состав которой входят: веб-сервисы, программная инфраструктура, модель программирования. Данная платформа входит в состав платформы .NET, благодаря чему появляется большое обилие возможностей для использования.

ASP.NET Core является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

ASP.NET Core полностью является opensource, то есть весь исходный код доступен для просмотра. ASP.NET Core может работать поверх кросс-платформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, с помощью ASP.NET Core мы можем создавать кросс-платформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, то теперь ограничения использования только этой операционной системой не имеется. То есть мы можем запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC.

**4.1.3** Выводы

Благодаря выбору языка программирования C#, разрабатываемое программное средство будет гибким, масштабируемым и расширяемым, что упростит задачу по его поддержке и улучшении. Являясь частью платформы .NET, данный язык программирования так же привнесёт хорошую производительность, которая является важным фактором для любого программного средства.

Использование платформы ASP.NET Core делает разрабатываемое программное средство кроссплатформенным, что увеличит потенциальное количество пользователей. И так как в качестве шаблона проектирования был выбран шаблон MVC, то данная платформа становится более подходящей для использования, так как она включает в себя программную модель для создания программных средств именно по данному шаблону.

4.2 Взаимодействие с базой данных

В качестве систему управления базами данных (СУБД) была выбрана система Microsoft SQL Server 2012. Данная реляционная СУБД использует язык запросов Transact-SQL и предоставляет достаточно широкий перечень возможностей для взаимодействия с базой данных: формирование запросов, осуществление поиска необходимых данных, синхронизация информации, а также выполнение аналитической обработки данных и получение разнообразных отчетов. При выборе СУБД немаловажную роль сыграло наличие возможности тесного взаимодействия данной системы с платформой программирования .NET.

Платформа .NET предлагает широкий выбор методов взаимодействия с базой данных, во многом благодаря широкой коллекции библиотек, дополнений и расширений, позволяющих оптимизировать взаимодействие для решения конкретной задачи. Однако можно выделить два основных подхода к взаимодействию:

* взаимодействию в терминах, близких к структуре самой базы данных. При этом подходе работа с базой данных максимально приближена к ней, для доступа к хранимым данным необходимо непосредственно написание запросов на языке базы данных. Передача сведений осуществляется посредством объектов, эквивалентным результатам работы самой базы данных. Подобный подход увеличивает время на разработку, однако дает большие возможности для контроля выполняемых действий;
* использование объектно-ориентированных технологий доступа к данным (ORM). Данная технология программирования связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». В результате взаимодействие с базой данных максимально приближена к терминам платформы .NET. Знание и написание запросов к базе данных не требуется, могут предоставляться возможности для отложенной загрузки данных и их автоматического обновления в базе данных. Данный подход на платформе .NET реализуется с помощью компонентов LINQ to SQL, NHibernate, ADO.NET, Entity Framework, Entity Framework Core и других.

Для данного проекта было принято решение о целесообразности использования технологии ORM, поскольку по результату проектирования базы данных было получено большое количество связанных таблиц, а использование данной технологии значительно облегчит получение связанных данных. А отложенная загрузка позволит загружать данные лишь по необходимости, что положительно скажется на быстродействии системы. Выбрана же была технология Entity Framework Core, так как она хорошо подходит для работы с платформой ASP.NET Core.

4.3 Основные компоненты программного средства

**4.3.1** Класс «Film»

Для представления всей информации, которая может быть у фильма, используется класс «Film», код которого представлен на рисунке 4.1.

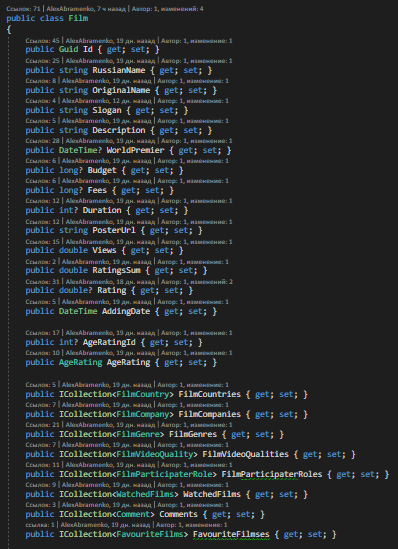


Рисунок 4.1 – Код класса «Film»

На представленном рисунке можно отметить следующие составляющие класса «Film»:

1. Id – идентификатор фильма;
2. RussianName – русскоязычное название фильма;
3. OriginalName – оригинальное название фильма;
4. Slogan – слоган фильма;
5. Description – описание фильма;
6. WorldPremier – мировая премьера фильма;
7. Budget – бюджет фильма;
8. Fees – сборы фильма;
9. Duration – продолжительность фильма;
10. PosterUrl – путь к файлу с постером фильма;
11. Views – просмотры фильма;
12. RatingsSum – сумма всех оценок всех пользователей об этом фильме;
13. Ratings – рейтинг фильма;
14. AddingDate – дата добавления фильма;
15. AgeRatingId – идентификатор возрастного ограничения;
16. AgeRating – объект класса «AgeRating», в котором хранится информация о возрастном ограничении, идентификатор которого равен значению в AgeRatingId;
17. FilmCountries – список стран-производителей, которые связаны с фильмом;
18. FilmCompanies – список компаний-производителей, которые связаны с фильмом;
19. FilmGenres – список жанров фильма;
20. FilmVideoQualities – список с информацией, каждый элемент которого содержит качество видео фильма и путь к соответствующему видео;
21. FilmParticipaterRoles – список с информацией об участниках фильма и их ролях в нём;
22. WatchedFilms – список с информацией, каждый элемент которого содержит информацию о пользователе, которые добавили данный фильм в каталог «Буду смотреть», и его оценке фильму если же он оценил его;
23. Comments – список с информацией, каждый элемент которого содержит информацию о пользователе, который написал комментарий к фильму, и сам текст комментария;
24. FavouriteFilmses – список с информацией о пользователях, которые отметили фильм как любимый.

**4.3.2** Интерфейс класса «IFilmsListViewModel»

В программном средстве требуется отображать списки фильмов, которые по своей составляющей имеют немного отличий, из-за чего для представления общих данных используется интерфейс класса «IFilmsListViewModel», код которого представлен на рисунке 4.2.

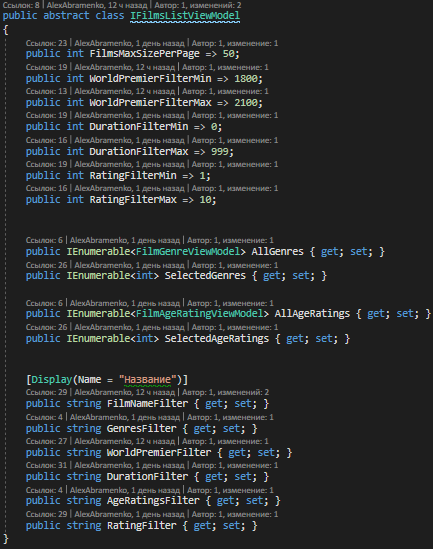


Рисунок 4.2 – Код интерфейса класса «IFilmsListViewModel»

На представленном рисунке можно отметить следующие составляющие класса «Film»:

1. FilmsMaxSizePerPage – максимальное количество фильмов в списке на одной странице;
2. WorldPremierFilterMin – минимальный год для фильтрации фильмов по мировой премьере;
3. WorldPremierFilterMax – максимальный год для фильтрации фильмов по мировой премьере;
4. DurationFilterMin – минимальная продолжительность для фильтрации фильмов по продолжительности;
5. DurationFilterMax – максимальная продолжительность для фильтрации фильмов по продолжительности;
6. RatingFilterMin – минимальное значение рейтинга для фильтрации фильмов по рейтингу;
7. RatingFilterMax – максимальное значение рейтинга для фильтрации фильмов по рейтингу;
8. AllGenres – список всех жанров, которые хранятся в базе данных;
9. SelectedGenres – список жанров, выбранных для фильтрации фильмов по жанрам;
10. AllAgeRatings – список всех возрастных ограничений, которые хранятся в базе данных;
11. SelectedAgeRatings – спсок возрастных ограничений, выбранный для фильтрации фильмов по возрастному ограничению;
12. FilmNameFilter – строка, по которой будет происходить фильтрация фильма по русскоязычному или оригинальному названиям;
13. GenresFilter – строка с идентификаторами жанров, выбранных пользователем, для фильтрации фильмов по жанрам;
14. WorldPremierFilter - строка, содержащая начальный и конечный диапазоны годов, для фильтрации фильмов по мировой премьере;
15. DurationFilter – строка, содержащая начальный и конечный диапазоны продолжительности, для фильтрации фильмов по продолжительности;
16. AgeRatingsFilter – строка с идентификаторами возрастных ограничений, выбранный пользователем, для фильтрации фильмов по возрастному ограничению;
17. RatingFilter – строка, содержащая начальный и конечный диапазоны значений рейтинга, для фильтрации фильмов по рейтингу.

**4.3.3** Класс «FilmListViewModel»

Для представления информации, которая нужна для отображения списка фильмов, используется класс «FilmListViewModel», код которого представлен на рисунке 4.1.

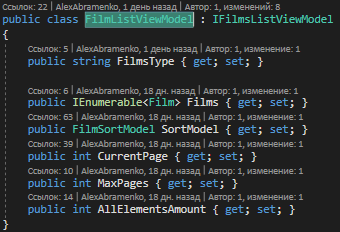


Рисунок 4.3 – Код класса «FilmListViewModel»

На представленном рисунке можно отметить следующие составляющие класса «Film»:

1. Films – список фильмов, каждый элемент которого представлен объектом класса «Film», в котором содержится вся информация о фильме;
2. SortModel – текущий критерий сортировки списка фильмов;
3. CurrentPage – номер текущей страницы пагинации;
4. MaxPages – максимальное количество страниц пагинации в рамках текущей фильтрации;
5. AllElementsAmount – общее количество фильмов, которые удовлетворяют текущей фильтрации.

4.4 Выводы раздела

В данном разделе был рассмотрен язык программирования C# и платформа ASP.NET Core, взаимодействие с базой данных, основные компоненты программного средства.

5 Тестирование программного средства

Одним из важнейших этапов разработки программного обеспечения является тестирование. Проведение тестирования позволяет повысить качество конечного продукта, а также выполнить проверку работоспособности разрабатываемого программного средства.

Методы тестирования программного обеспечения подразделяются на структурные и функциональные. Структурное тестирование основывается на детальном изучении логики программы и подборе тестов, обеспечивающих максимально возможное число проверяемых операторов, логических ветвлений и условий. Данный вид тестирования находит применение на более ранних этапах разработки программного средства. Наиболее распространены следующие виды структурного тестирования:

1. по критерию:

* путей;
* ветвей;

1. базовых путей;
2. циклов.

Функциональное тестирование является основным видом тестирования ПО. Каждая функция программы тестируется (проверяется на правильность в некоторых точках) и при этом делается вывод об ее правильности. Функциональное тестирование не является альтернативой структурному тестированию, поскольку позволяет обнаружить другие классы ошибок:

* некорректные или отсутствующие функции;
* ошибки интерфейса;
* неточности во внешних структурах данных.

В отличие от структурного, функциональное тестирование используется на более поздних этапах тестирования и может выполняться вручную или с использованием средств автоматизации. Целями функционального тестирования являются:

* поиск в тестируемом программном обеспечении ошибок;
* документирование найденных ошибок с целью дальнейшего их исправления;
* определение соответствия тестируемого программного продукта предъявляемым к нему требованиям и принятие объективного заключения о возможности поставки протестированного ПО заказчику.

В силу рассмотренных выше видов тестирования, для проведения тестирования программного средства, в том числе для оценки его работоспособности, было выбрано функциональное тестирование программного средства. Для его проведения был разработан ряд тестовых сценариев. Тестирование программного средства производилась на персональном компьютере с установленной операционной системой Windows 10.

Разработанные текстовые сценарии для пользователя, который не авторизован в программном средстве, представлены в таблице 5.1

Таблица 5.1 – Тестовые сценарии неавторизованного пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последовательность действий | Ожидаемый  результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Просмотр списка фильмов, доступных для просмотра. | 1. Открыть главную страницу программного средства. | Отображение страницы со списком фильмов, доступных для просмотра. | Тест успешно пройден. |
| 2 | Сортировка списка фильмов, доступных для просмотра. | 1. Открыть главную страницу программного средства.  2. Нажать на один из доступных параметров сортировки. | Отображение отсортированного списка фильмов, доступных для просмотра. | Тест успешно пройден. |
| 3 | Фильтрация списка фильмов, доступных для просмотра. | 1. Открыть главную страницу программного средства.  2. Выбрать параметры фильтрации.  3. Нажать на кнопку «Отфильтровать». | Отображение отфильтрованного списка фильмов, доступных для просмотра. | Тест успешно пройден. |
| 4 | Просмотр подробной информации о фильме. | 1. Открыть главную страницу программного средства.  2. Нажать на название одного из фильмов в списке. | Отображение страницы с подробной информацией о выбранном фильме | Тест успешно пройден. |
| 5 | Просмотр фильма в онлайне. | 1. Открыть главную страницу программного средства.  2. Нажать на название одного из фильмов в списке.  3. Нажать на кнопку пуск видеоплеера. | Видео фильма начало воспроизведение. | Тест успешно пройден. |

Разработанные тестовые сценарии регистрации и авторизации представлены в таблице 5.2.

Таблица 5.2 – Тестовые сценарии регистрации и авторизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последовательность действий | Ожидаемый  результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Регистрация. | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Регистрация».  3. Ввести корректные никнейм и пароль в соответствующие поля.  4. Нажать кнопку «Зарегистрироваться». | Автоматическая авторизация в новый профиль. После чего перенаправление на страницу со списком фильмов, доступных для просмотра. | Тест успешно пройден. |
| 2 | Регистрация. Валидация ошибок. | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Регистрация».  3. Ввести некорректные никнейм и пароль в соответствующие поля.  4. Нажать кнопку «Зарегистрироваться». | Отображение сообщения о некорректности введённых данных. | Тест успешно пройден. |
| 3 | Авторизация. | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Авторизация».  3. Ввести корректные никнейм и пароль в соответствующие поля.  4. Нажать кнопку «Авторизироваться». | Авторизация пользователя. После чего перенаправление на страницу со списком фильмов, доступных для просмотра. | Тест успешно пройден. |
| 4 | Авторизация. Валидация ошибок | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Авторизация». | Отображение сообщения о некорректности введённых данных. | Тест успешно пройден. |

Продолжение таблицы 5.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
|  |  | 3. Ввести корректные никнейм и пароль в соответствующие поля.  4. Нажать кнопку «Авторизироваться». |  |  |

Последующие тестовые сценарии требует предварительной авторизации для их успешного выполнения. Последовательность действий для успешного прохождения авторизации представлена в 3-м тест-кейсе таблицы 5.2 в столбце «Последовательность действий». Разработанные тестовые сценарии, требующие прохождения пользователем авторизации, доступны к рассмотрению в таблице 5.3.

Таблица 5.3 – Тестовые сценарии авторизованного пользователя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № тест-кейса | Тестируемая функциональность | Последовательность действий | Ожидаемый  результат | Полученный результат |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Просмотр профиля. | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Мой профиль». | Отображение страницы с информацией о профиле пользователе. | Тест успешно пройден. |
| 2 | Просмотр списка фильмов, добавленных в каталог «Буду смотреть». | 1. Открыть главную страницу программного средства.  2. Нажать на кнопку «Список фильмов из каталога “Буду смотреть”». | Отображение страницы со списком фильмов, добавленных в каталог «Буду смотреть». | Тест успешно пройден. |
| 3 | Просмотр списка фильмов, оценённых пользователем. | 1. Открыть главную страницу программного средства.  2. Нажать на кнопку «Список оценённых фильмов». | Отображение страницы со списком фильмов, оценённых пользователем. | Тест успешно пройден. |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 4 | Добавление фильма в каталог «Буду смотреть» | 1. Открыть главную страницу программного средства.  2. Нажать на название одного из фильмов в списке.  3. Нажать на кнопку «Добавить в каталог «Буду смотреть»». | Фильм добавлен в каталог «Буду смотреть». После чего текст кнопки меняется на «Удалить из каталога «Буду смотреть»». | Тест успешно пройден. |
| 5 | Удаление фильма из каталога «Буду смотреть» | 1. Открыть главную страницу программного средства.  2. Нажать на кнопку «Список фильмов из каталога “Буду смотреть”».  3. Нажать на кнопку «Удалить из каталога «Буду смотреть»» | Фильм удалён из каталога «Буду смотреть». После чего текст кнопки меняется на «Добавить в каталог «Буду смотреть»». | Тест успешно пройден. |
| 6 | Оценивание фильма. | 1. Открыть главную страницу программного средства.  2. Нажать на кнопку «Список фильмов из каталога «Буду смотреть»».  3. Нажать на соответствующее значение оценки. | Оценка добавлена. После чего происходит обновление рейтинга фильма и его просмотров. | Тест успешно пройден. |
| 7 | Комментирование фильма | 1. Открыть главную страницу программного средства.  2. Нажать на название одного из фильмов в списке.  3. Написать текст комментария в соответствующем поле.  4. Нажать на кнопку «Комментировать» | Комментарий добавлен. После происходит чего обновление списка комментариев фильма. | Тест успешно пройден. |

Продолжение таблицы 5.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 8 | Подбор фильмов. | 1. Открыть главную страницу программного средства.  2. Выбрать пункт «Подобрать фильм». | Отображение страницы с подобранным фильмом | Тест успешно пройден. |

Были проверены основные функциональные возможности приложения. Все тесты были пройдены успешны, что свидетельствует о том, что приложение успешно справилось с функциональными испытаниями.

5.1 Выводы раздела

В данном разделе было произведено тестирование функциональных возможностей программного средства.

6 Руководство по установке и использованию программного средства

6.1 Установка программного средства

В данном разделе приведены основные сведения по работе с программным средством. Приложение данного дипломного проекта (а точнее, его клиентская часть) не требует установки и настройки на конечных устройствах пользователя, поскольку представляет собой веб-приложение.

Для корректной работы программного средства необходим один из следующих браузеров с соответствующей минимальной версией:

* Google Chrome 70;
* Opera 58;
* Mozilla Firefox 66;
* Apple Safari 12.0;
* Microsoft Edge 44.

6.2 Руководство по использованию программного средства

При открытии сайта происходит отображение главной страницы: страницы со списком фильмов, доступных для просмотра. Интерфейс данной страницы представлен на рисунке 6.1. Здесь пользователь имеет возможность:

* просмотреть список фильмов, доступных для просмотра, с краткой информацией о каждом;
* отсортировать список фильмов, путём нажатия на соответствующий вариант сортировки;
* отфильтровать список фильмов, путём выбора критериев фильтрации, после чего нажав на кнопку «Отфильтровать»;
* перейти на страницу с подробной информацией о фильме, по названию которого пользователь нажал в списке.

Стоит отметить, что на каждой странице разработанного программного средства имеется заголовок, с помощью которого можно в любой момент перейти на нужную пользователю страницу. Для неавторизованного пользователя данный заголовок содержит:

* название сайта «MovieOnline», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Фильмы», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Регистрация», по нажатии на который происходит перенаправление на страницу регистрации пользователя;
* пункт «Авторизация», по нажатии на который происходит перенаправление на страницу авторизации пользователя.

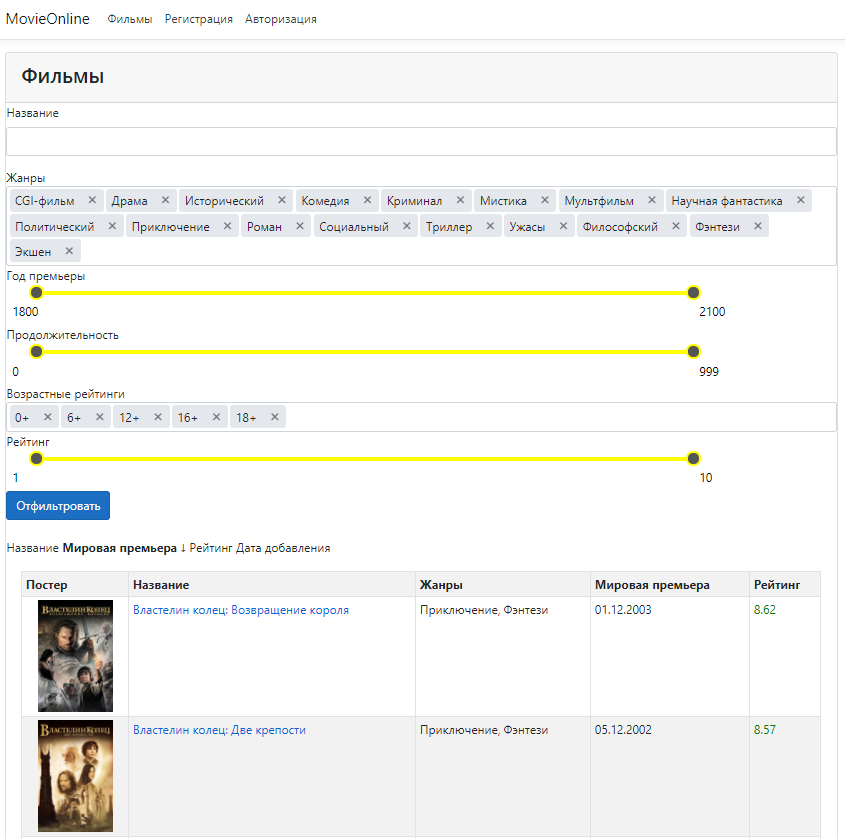


Рисунок 6.1 – Интерфейс главной страницы программного средства

При нажатии на название фильма из списка происходит перенаправление на страницу с подробной информацией о выбранном фильме. Интерфейс подробной информации о фильме, которая имеется на данной странице, представлен на рисунке 6.2. Здесь пользователь может узнать подробнее о выбранном фильме: от оригинального названия и стран-производителей до сборов и возрастного ограничения.



Рисунок 6.2 – Интерфейс подробной информации о фильме

Ниже подробной информации о фильме, представленной на рисунке 6.2, находится информация об участниках фильма, а также видеоплеер с возможностью смены качества видео. Всё это представлено на рисунке 6.3. Здесь пользователь может узнать о всех участниках фильма с их ролями в нём. При нажатии на имя и фамилию участника пользователь будет перенаправлен на страницу подробной информации о выбранном участнике. При нажатии на плеер начинается воспроизведение видео того качества, которое выбрано в меню под видеоплеером. Видеоплеер же обладает всеми функциями стандартными функциями: остановка видео, регулировка звука, переход в полноэкранный режим, перемотка видео. Также в самом конце располагается список комментариев о фильме, если же таковые имеются.

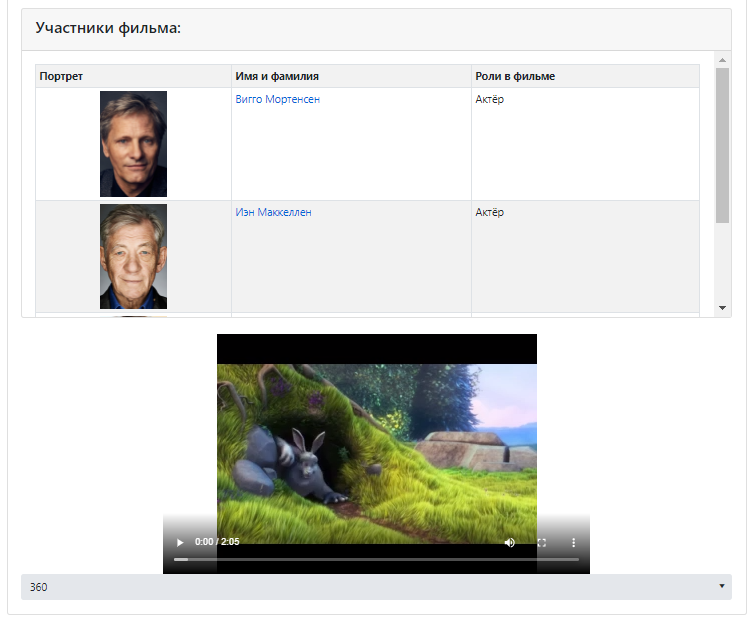


Рисунок 6.3 – Интерфейс информации об участниках фильма и видеоплеера с меню выбора качества видео

При нажатии на имя и фамилию участника происходит перенаправление на страницу с подробной информацией о выбранном участнике. Интерфейс подробной информации об участнике, которая имеется на данной странице, представлен на рисунке 6.4. Здесь пользователь может узнать подробнее о выбранном участнике: от даты рождения до роста. Также пользователь может просмотреть все фильмы, в которых он принимал участие, с указанием его ролей в них.

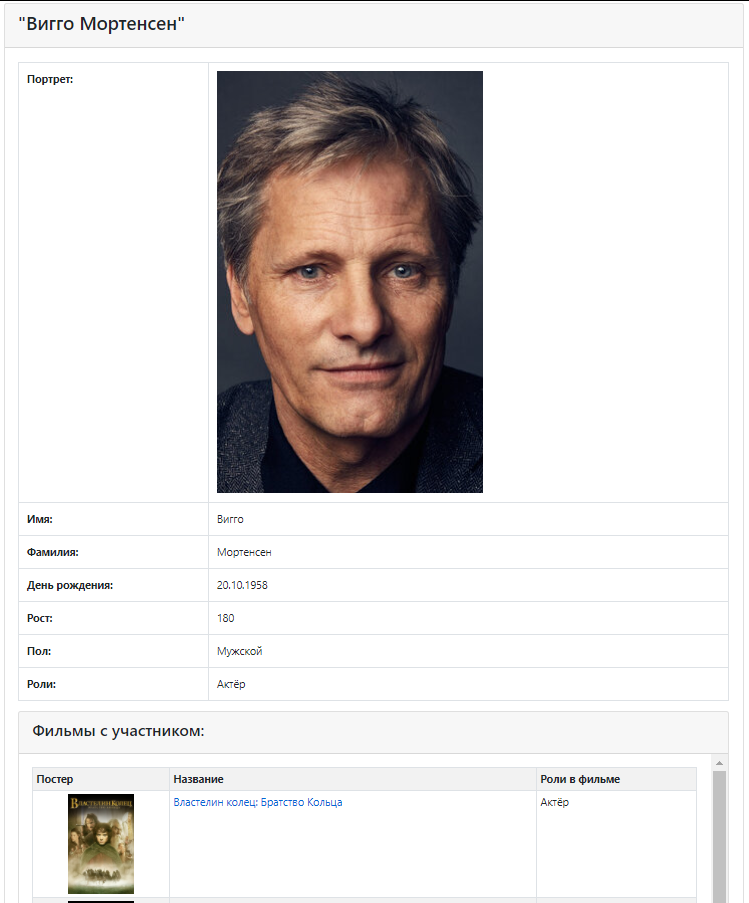


Рисунок 6.4 – Интерфейс страницы с подробной информацией об участнике

Для увеличения функциональности программного средства относительно неавторизованного пользователя требуется выполнить регистрацию для создания нового профиля или же авторизацию для входа в уже существующий.

При нажатии на пункт «Регистрация» в заголовке страницы пользователь будет перенаправлен на страницу регистрации, интерфейс которой представлен на рисунке 6.5. Здесь пользователь должен ввести корректные никнейм и пароль (который также требуется повторить в поле «Повторите пароль»). Для подтверждения регистрации требуется нажать на кнопку «Зарегистрироваться». Если введённые данные окажутся некорректными, то отобразятся сообщения с указанием конкретной ошибки в конкретном поле с некорректным значением. Интерфейс подобного случая представлен на рисунке 6.6.

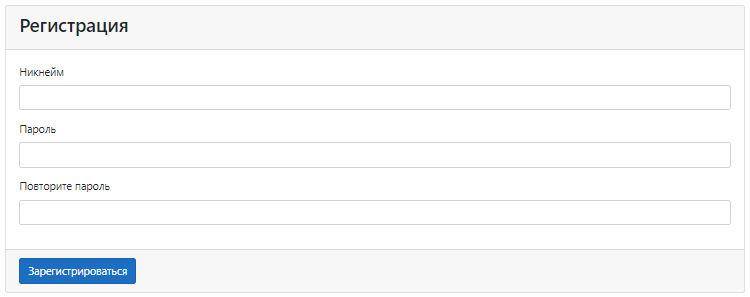


Рисунок 6.5 – Интерфейс страницы регистрации пользователя

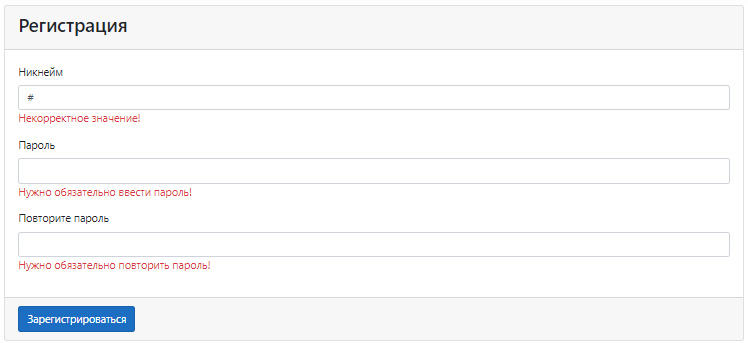


Рисунок 6.6 – Интерфейс страницы регистрации пользователя в случае ввода некорректных значений

Если все данные окажутся корректными, то пользователь будет зарегистрирован и автоматически авторизован, после чего будет перенаправлен на главную страницу программного средства.

При нажатии на пункт «Авторизация» в заголовке страницы пользователь будет перенаправлен на страницу регистрации, интерфейс которой представлен на рисунке 6.7. Здесь пользователь должен ввести корректные никнейм и пароль. Для подтверждения авторизации требуется нажать на кнопку «Авторизоваться». Если введённые данные окажутся некорректными, то отобразятся сообщения с указанием конкретной ошибки в конкретном поле с некорректным значением. Интерфейс подобного случая представлен на рисунке 6.8.

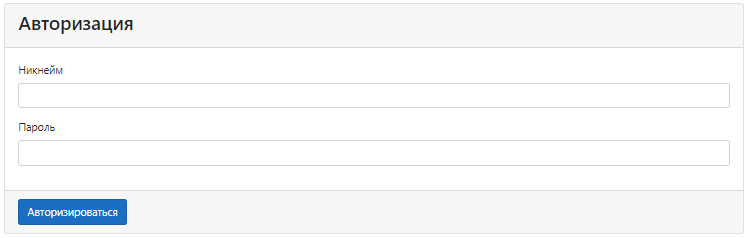


Рисунок 6.7 – Интерфейс страницы авторизации пользователя

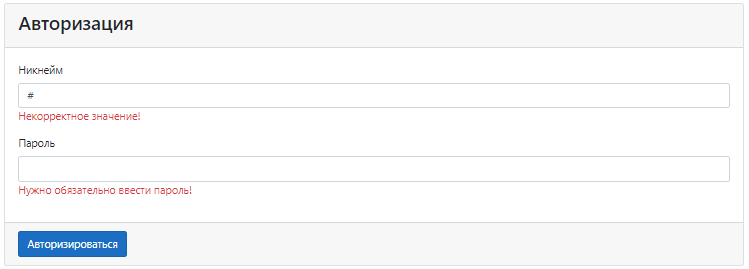


Рисунок 6.8 – Интерфейс страницы авторизации пользователя в случае ввода некорректных значений

После того, как пользователь будет авторизован, у него появятся новые функциональные возможности в программном средстве. Также изменится и заголовок на каждой странице программного средства, интерфейс которого представлен на рисунке 6.9.

Для авторизованного пользователя со статусом «Пользователь» данный заголовок содержит:

* название сайта «MovieOnline», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Фильмы», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Подобрать фильм», по нажатии на который осуществляется переход на страницу подбора фильма;
* пункт «Мой профиль», по нажатии на который происходит перенаправление на страницу с информацией о профиле текущего пользователя;
* пункт «Выход из профиля», по нажатии на который происходит выход пользователя из текущего профиля.



Рисунок 6.9 – Интерфейс заголовка для авторизованного пользователя со статусом «Пользователь»

При нажатии на пункт «Мой профиль» пользователь будет перенаправлен на страницу с информацией о своём профиле. Интерфейс данной страницы представлен на рисунке 6.10. Здесь пользователь может просмотреть информацию о своём профиле, перейти его редактирование, перейти на просмотр списка добавленных в каталог «Буду смотреть» фильмов или перейти на просмотр списка оценённых фильмов.

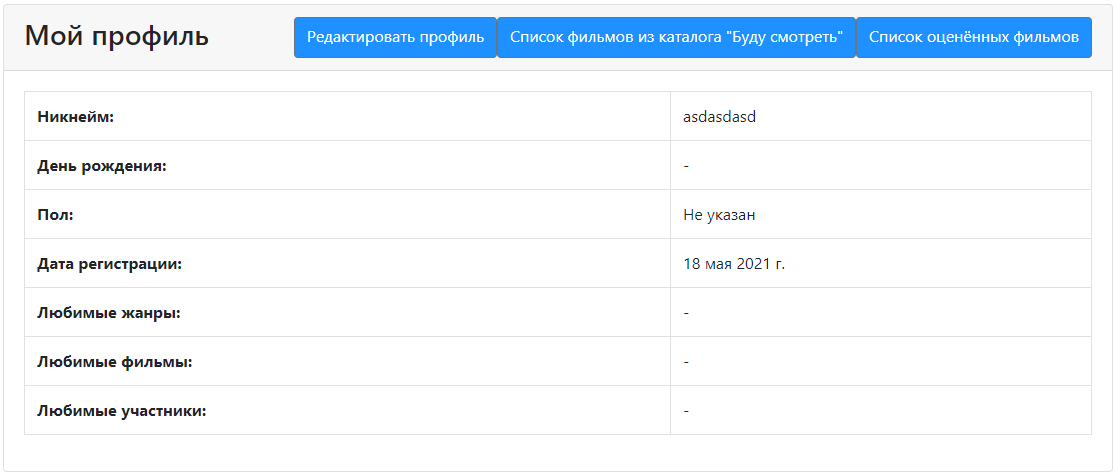


Рисунок 6.10 – Интерфейс страницы с подробной информацией о профиле текущего пользователя

При нажатии на кнопку «Редактировать профиль» пользователь будет перенаправлен на страницу редактирования профиля. Интерфейс данной страницы представлен на рисунке 6.11. Здесь пользователь может изменить никнейм, пароль и может указать день рождения, пол, любимые жанры, любимые фильмы, любимых участников. Вся эта информация будет использована при подборе фильма.

При нажатии на кнопку «Подтвердить» вся введённая информация будет сохранена, после чего пользователь будет перенаправлен на страницу подробной информации о своём профиле.

В случае заполненных данных интерфейс будет представлять из себя содержимое рисунка 6.12.

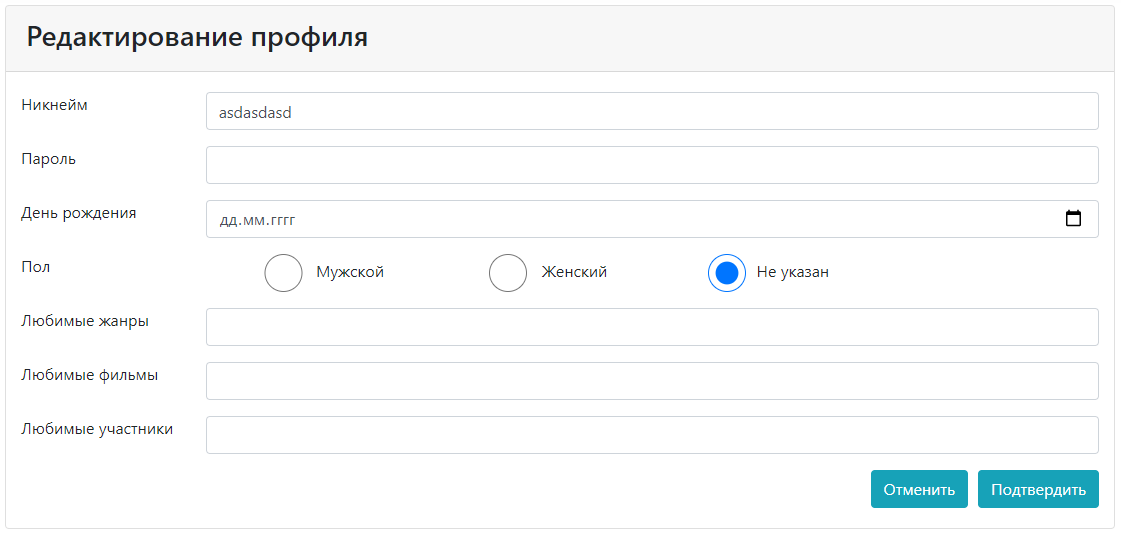


Рисунок 6.11 – Интерфейс страницы редактирования профиля пользователя

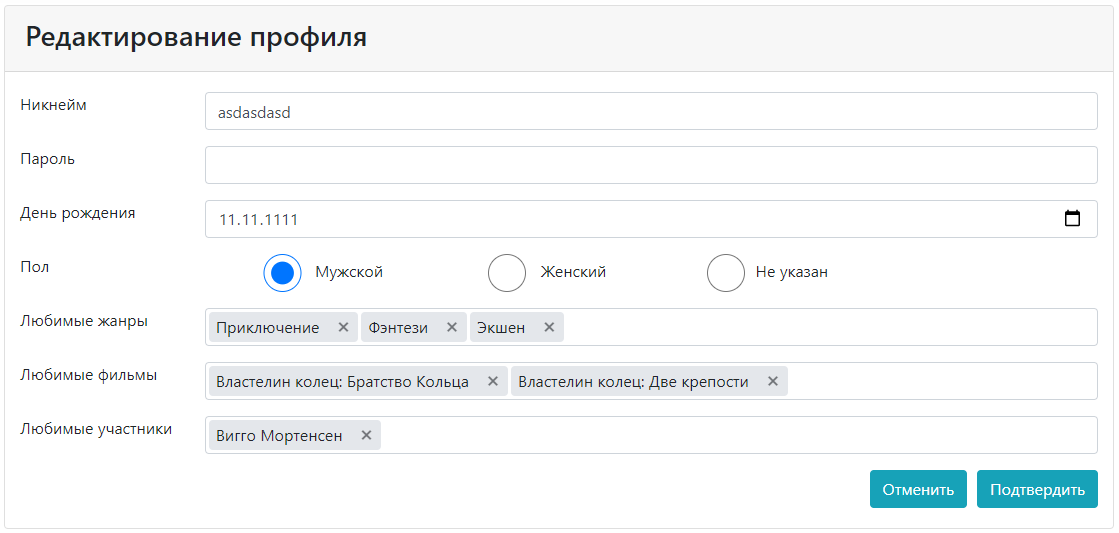


Рисунок 6.12 – Интерфейс страницы редактирования профиля пользователя с заполненными данными

При нажатии на кнопку «Список фильмов из каталога «Буду смотреть»» пользователь будет перенаправлен на страницу со списком фильмов, которые были добавлены в каталог «Буду смотреть». Интерфейс данной страницы идентичен интерфейсу главной страницы.

При нажатии на кнопку «Список оценённых фильмов» пользователь будет перенаправлен на страницу со списком оценённых фильмов. Интерфейс данной страницы идентичен интерфейсу главной страницы.

При нажатии на пункт «Подобрать фильм» пользователь будет перенаправлен на страницу подбора фильма. Интерфейс данной страницы представлен на рисунке 6.13. Здесь пользователь может осуществить фильтрацию для более точного подбора, просмотреть краткую информацию о подобранном фильме, перейти на страницу с подробной информацией о подобранном фильме, нажав на название фильма, а также добавить его в каталог «Буду смотреть» или же оценить его. Также пользователь может перейти к следующему подобранному фильму или же вернуться к предыдущему при нажатии на соответствующие кнопки «Следующий» и «Предыдущий».

Если же не удалось подобрать какой-либо фильм, то вместо информации о подобранном фильме будет находиться соответствующий текст «Нет подходящих фильмов».

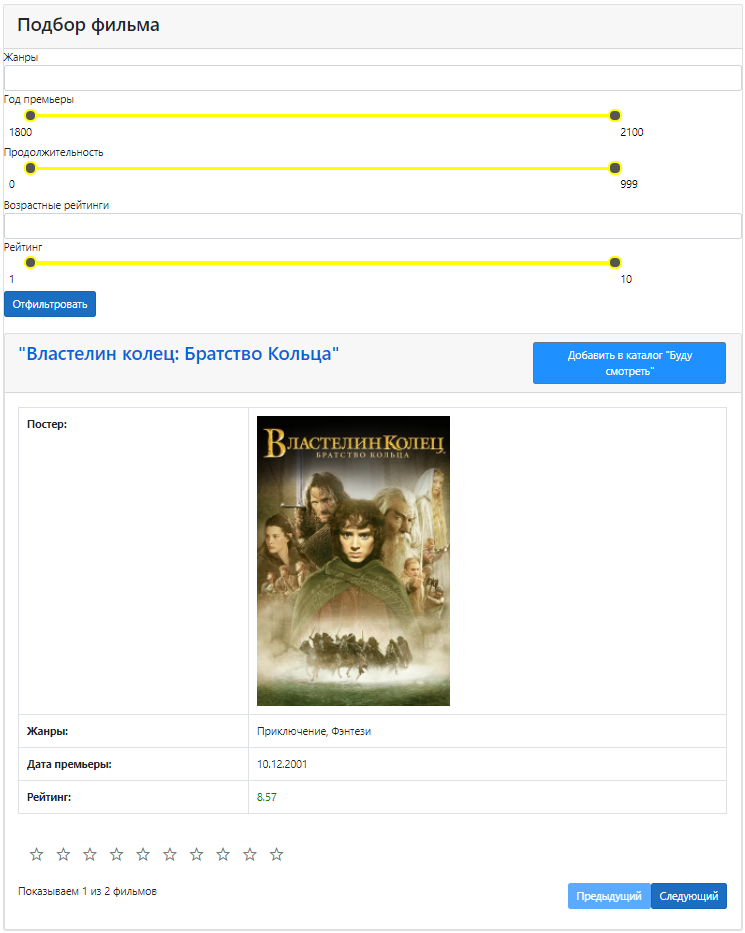


Рисунок 6.13 – Интерфейс страницы редактирования профиля пользователя с заполненными данными

Также стоит отметить, что у авторизованного пользователя появились новые возможности на странице подробной информации о фильме:

* добавление фильма в каталог «Буду смотреть»;
* удаление фильма из каталога «Буду смотреть»;
* оценивание фильма;
* комментирование фильма.

Если же авторизованный пользователь имеет статус «Администратор», то у него появятся все функциональные возможности в программном средстве. Также изменится и заголовок на каждой странице программного средства, интерфейс которого представлен на рисунке 6.14.

Для авторизованного пользователя со статусом «Администратор» данный заголовок содержит:

* название сайта «MovieOnline», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Фильмы», по нажатии на который осуществляется переход на главную страницу программного средства;
* пункт «Добавить фильм», по нажатии на который осуществляется переход на страницу добавления фильма;
* пункт «Добавить участника», по нажатии на который осуществляется переход на страницу добавления участника;
* пункт «Подобрать фильм», по нажатии на который осуществляется переход на страницу подбора фильма;
* пункт «Мой профиль», по нажатии на который происходит перенаправление на страницу с информацией о профиле текущего пользователя;
* пункт «Выход из профиля», по нажатии на который происходит выход пользователя из текущего профиля.



Рисунок 6.14 – Интерфейс заголовка для авторизованного пользователя со статусом «Администратор»

При нажатии на пункт «Добавить фильм» пользователь будет перенаправлен на страницу добавления фильма. Интерфейс данной страницы представлен на рисунке 6.15. Здесь пользователь может указать всю возможную информацию о фильме, установить в качестве постера любое изображение, может связать участников с фильмом с указанием их ролей в нём, связать качество видео с названием файла, которое представляет из себя видео. При нажатии на кнопку «Подтвердить» происходит добавление фильма, после чего происходит перенаправление на страницу подробной информации о добавленном фильме.

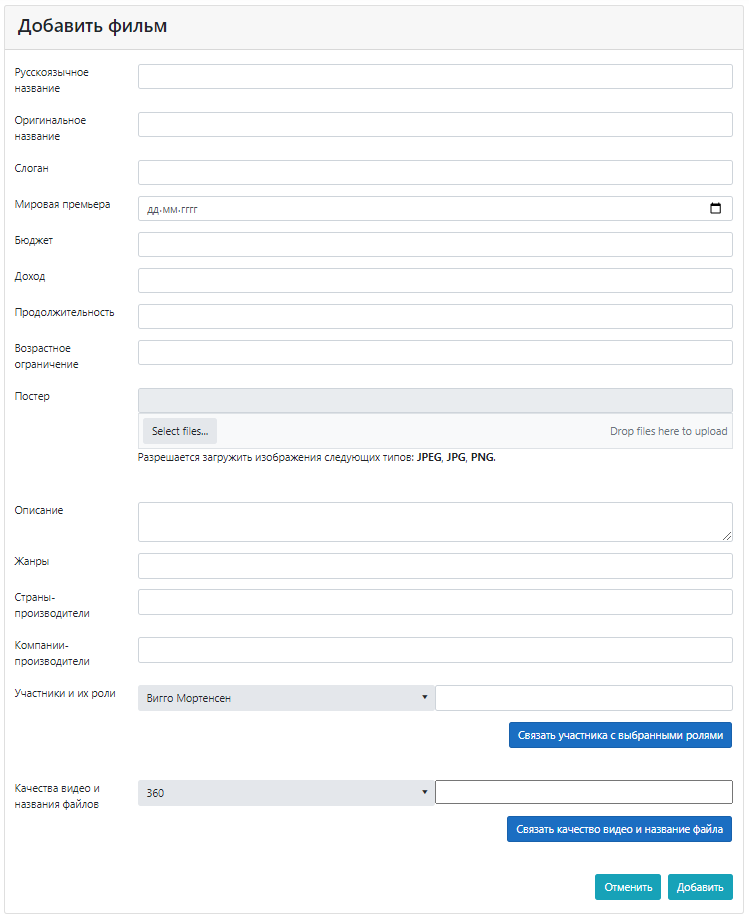


Рисунок 6.15 – Интерфейс страницы добавления фильма

При нажатии на пункт «Добавление участника» пользователь будет перенаправлен на страницу добавления участника. Интерфейс данной страницы представлен на рисунке 6.16. Здесь пользователь может указать всю возможную информацию об участнике, установить в качестве портрета любое изображение, может связать фильмы с участником с указанием его ролей в них. При нажатии на кнопку «Подтвердить» происходит добавление участника, после чего происходит перенаправление на страницу подробной информации о добавленном участнике.



Рисунок 6.16 – Интерфейс страницы добавления участника

Также на главной странице пользователь со статусом «Администратор» имеет возможность удалять фильмы, нажав на кнопку «Удалить», на странице подробной информации о фильме пользователь имеет возможность перейти на страницу редактирования данного фильма, нажав на кнопку «Редактировать», а на странице подробной информации об участнике имеется возможность перейти на страницу редактирования данного участника, нажав на кнопку «Редактировать», и возможность удалить участника, нажав на кнопку «Удалить».

В итоге программное средство работает стабильно. Интерфейс удобный и понятный. Предусмотрены различные проверки во избежание критических ситуаций и ошибок пользователя. Была учтена логика администрирования.

6.3 Выводы раздела

В данном разделе была рассмотрена установка программного средства и составлено руководство по его использованию. Она работает стабильно. Интерфейс удобный и понятный. Предусмотрены различные проверки во избежание критических ситуаций и ошибок пользователя. Была учтена логика администрирования.

7 Технико-экономическое обоснование разработки и использования программного средства

7.1 Описание функций, назначения и потенциальных пользователей программного средства

Разработанное в дипломном проекте программное средство предназначено для хранения фильмов и предоставления возможности их просмотра в онлайне, а также их подбора на основе пожеланий пользователя.

Применение разработанного программного средства пользователями позволит решать следующие задачи:

1. поиск и фильтрация фильмов по выбранным критериям;
2. просмотр актуальной информации о фильме;
3. просмотр видео фильма в выбранном качестве;
4. подбор фильмов на основе пожеланий пользователя.

Функции программного средства:

1. регистрация аккаунта;
2. авторизация в аккаунт;
3. предоставление возможности просмотра списка фильмов с краткой информацией о них;
4. предоставление возможности фильтрации просматриваемого списка фильмов;
5. предоставление возможности просмотра информации о фильме;
6. предоставление возможности просмотра видео фильма;
7. предоставление возможности подбора фильмов на основе пожеланий пользователя: жанра, длительности, возрастного рейтинга и т.п.;
8. предоставление возможности авторизованным пользователям комментирования фильмов;
9. предоставление возможности авторизованным пользователям добавления фильма в стандартный каталог под названием «Буду смотреть»;
10. предоставление возможности авторизованному пользователю просмотра списка фильмов в стандартном каталоге под названием «Буду смотреть»;
11. предоставление возможности авторизованному пользователю оценки фильма по десятибалльной шкале (от 1 до 10);
12. предоставление возможности авторизованному пользователю просмотра списка фильмов в своём каталоге под названием «Оцененные фильмы».

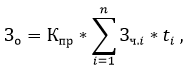
Основными пользователями программного средства являются организации, занятые в сфере культурного досуга, например:

1. Фонд «ЭФИР КЛУБ ЛЮБИТЕЛЕЙ КИНО», местоположение – Российская Федерация, город Москва;
2. Фонд «Клуб любителей интеллектуального кино», местоположение – Российская Федерация, город Москва;
3. Культурно-информационное учреждение "Мастерская социального кино", местоположение – Республика Беларусь, город Минск.

7.2 Расчёт затрат на разработку программного средства

**7.2.1** Затраты на основную заработную плату команды разработчиков

Расчёт основной заработной платы участников команды осуществляется по формуле



где *n* – количество исполнителей, занятых разработкой конкретного ПО;

Кпр – коэффициент, учитывающий процент премий, равный 1.25;

Зч.*i* – часовая заработная плата i-го исполнителя, р.;

*ti* – трудоёмкость работ, выполняемых i-м исполнителем, ч.

Часовая заработная плата рассчитывается как результат деления месячной заработной платы на 168 ч.

Заработная плата будет исходя из данных на рынке труда для каждого разработчика.

Для определения трудоёмкости нужно учитывать сложность разработки программного средства и объём выполняемых им функций.

Размер премии определяется как 25% от основной заработной платы.

Исходя из данных о зарплате сотрудников, представленных на сайте [it-academy.by](https://www.it-academy.by/employment/kariera/), была заполнена следующая таблица.

Таблица 7.21 – Расчёт затрат на основную заработную плату разработчиков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование должности разработчика | Вид выполняемой работы | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоёмкость работ, ч | Зарплата по тарифу, р. |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1. Бизнес-аналитик | Анализ и написание требований для программного средства | 2100 | 12.5 | 64 | 800 |
| 2. Администратор баз данных | Проектирование и реализация базы данных | 1000 | 5.95 | 48 | 285.6 |
| 3. Инженер-программист | Реализация серверной части программного средства | 2200 | 13.10 | 160 | 2096 |
| 4. Программист | Реализация клиентской части программного средства | 1800 | 10.71 | 96 | 1028.16 |
| 5. Специалист по тестированию программного обеспечения | Тестирование и отладка программного средства | 1700 | 10.12 | 48 | 485.76 |
| 6. Дизайнер | Проектирование пользовательского интерфейса программного средства | 1700 | 10.12 | 40 | 404.8 |
| Премия (25%) | | | | | 1275.08 |
| Всего основная заработная плата разработчиков | | | | | 6375.4 |

**7.2.2** Затраты на дополнительную заработную плату разработчиков

Включают выплаты, предусмотренные законодательством о труде, и определяется по формуле

где Зо – затраты на основную заработную плату, р.;

Нд – норматив дополнительной заработной платы равный 15%.

Имея Зо = 6375.4 р. и Нд = 15%, посчитаем Зд:

**7.2.3** Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование)

Определяются в соответствии с действующими законодательными актами по формуле

где Нсоц – норматив отчислений на социальные нужды равный 34.6%.

Имея Зо = 6375.4 р., Зд = 956.31 р. и Нсоц = 34% + 0.6% (для страховки) = 34.6%, посчитаем Рсоц:

**7.2.4** Прочие затраты

Под прочими затратами здесь подразумеваются: затраты на аренду рабочих помещений, электричество, лицензионное ПО, обеды разработчикам.

Прочие затраты рассчитываются по формуле

где Нпз – норматив прочих затрат равный 125%.

Имея Зо = 6375.4 р. и Нпз = 125%, посчитаем Рпз:

Итоговая сумма затрат на разработку программного средства находится путём суммирования всех рассчитанных статей затрат (табл. 7.2).

Таблица 7.22 – Затраты на разработку программного средства

|  |  |
| --- | --- |
| Наименование статьи затрат | Сумма, р. |
| 1. Основная заработная плата разработчиков | 6375.4 |
| 2. Дополнительная заработная плата разработчиков | 956.31 |
| 3. Отчисления на социальные нужды | 2536.43 |
| 4. Прочие затраты | 7969.25 |
| Общая сумма затрат на разработку | 17837.39 |

7.3 Оценка результата от использования программного средства

**7.3.1** Экономический эффект при разработке программного средства для свободной реализации на рынке информационных технологий

Экономический эффект представляет собой прибыль, полученную от соглашений с компаниями, проявившими желание опубликовать рекламу и получившими одобрение от администратора программного средства.

На основе данных о стоимости размещения рекламы, представленных на сайте [tutby.com](https://tutby.com/advert/price/), был составлен следующий план: каждый год планируется заключать приблизительно 10 соглашений: каждый длительностью в год и средней ценой в 4000 белорусских рублей.

Являясь резидентом Парка высоких технологий (ПВТ), организация освобождена от налога на прибыль, вследствие чего можно рассчитать прибыль от программного средства за год по формуле

где Ц – средняя цена соглашения с компанией на размещение рекламы на

период в один год, р.;

*N* – количество соглашений, заключённых за год;

НДС – налог на добавленную стоимость, %;

Зр – затраты на разработку и реализацию программного средства, р.

Налог на добавленную стоимость определяется по формуле

где Ндс – ставка налога на добавленную стоимость согласно действующему

законодательству равная 20%.

Являясь резидентом Парка высоких технологий (ПВТ), организация освобождена от налога на добавленную стоимость (НДС), вследствие чего получаем НДС = 0 р.

Имея Ц = 4000 р, *N* = 10, НДС = 0 р. и Зр = 17837.39 р., посчитаем П:

Для оценки эффективности затрат в разработку ПО необходимо рассчитать уровень рентабельности затрат по формуле

Имея организацию, которая освобождена от налога на прибыль, П = 22162.61 р. и Зр = 17837.39 р., посчитаем Ур:

Средняя процентная ставка по банковским депозитным вкладам на февраль 2021 составляет 10.57%.

Благодаря рассчитанному Ур = 124.4% можно придти к тому, что уровень рентабельности затрат превышает среднюю процентную ставку по банковским депозитным вкладам (124.4% - 100% = 24.4% > 10.57%), вследствие чего можно сделать вывод, что проект будет экономически эффективным.

**7.3.2** Расчёт срока окупаемости

Срок окупаемости рассчитывается по формуле

где Дп – срок расчётного периода равный 12 месяцам.

Имея П = 22162.61 р., Зр = 17837.39 р. и Дп = 12 месяцев, посчитаем Со:

Таким образом, для того, чтобы полностью окупить программное средство, потребуется не менее 10 месяцев.

7.4 Расчёт показателей эффективности инвестиций в разработку программного средства

Экономическую эффективность инвестирования в разработку данного программного средства можно отобразить через рентабельность инвестиций, которая вычисляется по формуле

Имея организацию, которая освобождена от налога на прибыль, П = 22162.61 р. и Зр = 17837.39 р., посчитаем Ри:

В данном случае инвестирование в разработку целесообразно так как рентабельность инвестиций превышает 110.57% (100% плюс ставка по банковским депозитам), из чего следует, что вложится в проект выгоднее чем внести деньги на банковский депозит.

7.5 Выводы технико-экономического обоснования

В результате обозначения решаемых задач и выполняемых функций программным средством был определён состав разработчиков, после чего была определено трудоёмкость для каждого из них. Вследствие полученных данных прогнозируемая длительность разработки программного средства составляет 50-60 дней.

Благодаря использованию актуальных данных о зарплатах разработчиков и данных рассчитанных ранее были определены зарплаты по тарифу для каждого разработчика, вследствие чего с учётом премии в 25% стало возможным рассчитать основную заработную плату разработчиков, значение которой получилось Зо = 6375.4 р.

После дополнительных расчётов: затраты на дополнительную заработную плату разработчиков Зд = 956.31 р., отчисления на социальные нужды Рсоц = 2536.43, прочие затраты Рпз = 7969.25 р. – была рассчитана общая сумма затрат на разработку Зр = 17837.39 р.

Далее, после определения экономического эффекта как заключение соглашений с компаниями на размещение рекламы на клиентской части программного средства, был выработан план на год и подсчитана прибыль за первый год П = 22162.61 р., которая превышает общую сумму затрат на разработку Зр = 17837.39 р, о чём нам говорит рассчитанный уровень затрат рентабельности Ур = 124.24%.

В итоге мы получаем, что разработанное программное средство «Онлайн-кинотеатр с интеллектуальным помощником» на базе платформы ASP.NET Core, за относительно небольшой по длительности период разработки, в рамках составленного плана окупится в течение 10 месяцев. А рассчитанный уровень рентабельности затрат указывает на экономическую эффективность проекта.

Заключение

В ходе разработки программного средства был проведён анализ предметной области, проанализированы литературные источники, а также существующие аналоги онлайн-кинотеатров и программных средств с интеллектуальным помощником для подбора кино.

Были проведены моделирование программного средства и разработка функциональных требований.

Было произведено проектирование программного средства, разработана программная архитектура, диаграмма развёртывания программного средства, а также разработаны алгоритмы функций программного средства, в том числе общий алгоритм работы приложения.

Было произведено тестирование программного средства, которое показало полное соответствие разработанного программного обеспечения спецификации требований.

В завершение, были разработаны методики использования программного средства пользователем.

Было проведено технико-экономическое обоснование дипломного проекта, которое показало, что разработка экономически выгодна.

Итогом выполнения дипломного проекта является программное средство для просмотра фильмов в онлайне и их подбора на основе пожеланий пользователя при помощи интеллектуального помощника. Разработанное программное средство полностью соответствует разработанной спецификации требований. При регулярном использовании данное программное средство гораздо облегчит поиск подходящего фильма и позволит сразу же приступить к его просмотру, что позволит сэкономить время пользователя, а также увеличит шансы на положительные эмоции от фильма благодаря качественному подбору.

Таким образом, все поставленные на данный дипломный проект задачи были успешно выполнены.

Работа над программным средством будет продолжаться. В качестве направлений улучшения разработанного программного средства являются: качество и количество информации, хранимой о каждом фильме, что позволит пользователю побольше узнать о том, что ему интересно, и позволит увеличить возможности фильтрации, и качество подбора интеллектуальным помощником.

Список использованных источников

[1] Telesputnik [Электронный ресурс]. – Электронные данные. – Режим доступа: https://telesputnik.ru/.

[2] Tadviser [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.tadviser.ru/.

[3] Корнихин С., В фокусе / С. Корнихин // Мир ПК. – 2013. – №6. – С. 45.

[4] Octagon [Электронный ресурс]. – Электронные данные. – Режим доступа: https://octagon.media/.

[5] Dp.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.dp.ru/.

[6] SAS [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.sas.com/.

[7] MachineLearning.ru SAS [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.machinelearning.ru/.

[8] Skillbox [Электронный ресурс]. – Электронные данные. – Режим доступа: https://skillbox.ru/.

[9] Rb.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: https://rb.ru/.

[10] If24 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.if24.ru/.

[11] HDrezka.me [Электронный ресурс]. – Электронные данные. – Режим доступа: https://rezka.ag/.

[12] Okko [Электронный ресурс]. – Электронные данные. – Режим доступа: https://okko.tv/

[13] Kinonavigator [Электронный ресурс]. – Электронные данные. – Режим доступа: https://kinonavigator.ru/

[14] Oracle.com [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.oracle.com/ru/.

Приложение А

**(обязательное)**

**Исходный код**

namespace MovieOnline.Web.Controllers

{

//[Authorize]

public class FilmController : Controller

{

private const int SELECTION\_DEFAULT\_TAKE = 3;

private readonly IUnitOfWork \_uow;

public FilmController(IUnitOfWork uow) => \_uow = uow;

[Route("[controller]/[action]")]

public IActionResult List(

string filmNameFilter = "",

string genresFilter = "All",

string worldPremierFilter = "",

string durationFilter = "",

string ageRatingsFilter = "All",

string ratingFilter = "",

FilmSortState sortOrder = FilmSortState.WorldPremierDesc,

int page = 1)

{

var model = CreateFilmListViewModel(filmNameFilter, genresFilter, worldPremierFilter, durationFilter, ageRatingsFilter, ratingFilter, sortOrder, page);

ViewBag.FilterByNameValue = null;

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(model);

}

[Route("[controller]/[action]")]

public IActionResult GetFilmsList(

string filmNameFilter = "",

string genresFilter = "All",

string worldPremierFilter = "",

string durationFilter = "",

string ageRatingsFilter = "All",

string ratingFilter = "",

FilmSortState sortOrder = FilmSortState.WorldPremierDesc,

int page = 1)

{

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

var model = CreateFilmListViewModel(filmNameFilter, genresFilter, worldPremierFilter, durationFilter, ageRatingsFilter, ratingFilter, sortOrder, page);

return PartialView("\_GetFilmsList", model);

}

[Route("[controller]/[action]/{id:guid}")]

public IActionResult Details(Guid id)

{

var model = new FilmDetailsViewModel();

model.Film = \_uow.FilmRepository.Find(id);

model.VideoQualities = model.Film.FilmVideoQualities

.Select(x => new FilmDetailsVideoQualityViewModel() { QualityName = x.VideoQuality.Name, VideoFileName = x.VideoName });

model.Comments = model.Film.Comments

.Select(x => new FilmCommentViewModel() { UserName = x.User.Nickname, AddingDate = x.AddingDate, Text = x.Text });

ViewBag.DidUserWatch = null;

if (!string.IsNullOrEmpty(User.Identity.Name))

{

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

var watchedFilm = user?.WatchedFilms.SingleOrDefault(x => x.FilmId == id);

if (watchedFilm == null)

{

ViewBag.WillUserWatch = false;

ViewBag.DidUserWatch = false;

}

else

{

ViewBag.WillUserWatch = watchedFilm.Watched.HasValue && !watchedFilm.Watched.Value && !watchedFilm.Rating.HasValue;

ViewBag.DidUserWatch = watchedFilm.Watched != null && watchedFilm.Watched.Value;

ViewBag.UserRating = watchedFilm.Rating;

}

}

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return model.Film == null ? (IActionResult)NotFound() : View(model);

}

[Authorize]

[HttpPost("[controller]/[action]/id={id:guid}&isDelete={isDelete:bool}")]

public IActionResult WillWatchFilm(Guid id, bool isDelete = false)

{

var film = \_uow.FilmRepository.Find(id);

if (film == null)

{

return Content("Error");

}

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var watchedFilm = user.WatchedFilms.Where(x => x.FilmId == id).SingleOrDefault();

if ((watchedFilm == null && isDelete) || (watchedFilm != null && (!isDelete || watchedFilm.Rating.HasValue)))

{

return Content("Error");

}

if (!isDelete)

{

watchedFilm = new WatchedFilms()

{

UserId = user.Id,

FilmId = film.Id,

Watched = false,

Rating = null

};

user.WatchedFilms.Add(watchedFilm);

}

else

{

user.WatchedFilms.Remove(watchedFilm);

}

\_uow.Commit();

return Content("");

}

[Authorize]

[HttpPost("[controller]/[action]/id={id:guid}&rating={rating:double}&isDelete={isDelete:bool}")]

public IActionResult RateFilm(Guid id, double rating, bool isDelete = false)

{

var film = \_uow.FilmRepository.Find(id);

if (film == null)

{

return Content("Error");

}

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var watchedFilm = user.WatchedFilms.Where(x => x.FilmId == id).SingleOrDefault();

var ratingsSum = film.RatingsSum;

var views = film.Views;

if (!isDelete)

{

if (watchedFilm == null)

{

watchedFilm = new WatchedFilms()

{

UserId = user.Id,

FilmId = film.Id,

Watched = true,

Rating = rating

};

user.WatchedFilms.Add(watchedFilm);

ratingsSum += rating;

views++;

}

else

{

ratingsSum = ratingsSum - (watchedFilm.Rating.HasValue ? watchedFilm.Rating.Value : 0) + rating;

watchedFilm.Watched = true;

watchedFilm.Rating = rating;

}

}

else if (watchedFilm != null)

{

user.WatchedFilms.Remove(watchedFilm);

views--;

}

\_uow.Commit();

NumberFormatInfo nfi = new NumberFormatInfo();

nfi.NumberDecimalSeparator = ".";

var newRating = ratingsSum / views;

return Content((views != 0 ? newRating.ToString("0.00", nfi) : "No rating") + "," + views.ToString());

}

[Authorize]

[HttpPost("[controller]/[action]/id={id:guid}&text={text}")]

public IActionResult CommentFilm(Guid id, string text)

{

var film = \_uow.FilmRepository.Find(id);

if (film == null)

{

return Content("Error");

}

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var comment = new Comment()

{

Id = Guid.NewGuid(),

UserId = user.Id,

FilmId = film.Id,

AddingDate = DateTime.Now,

Text = text

};

\_uow.CommentRepository.Add(comment);

\_uow.Commit();

return Content(user.Nickname + "," + comment.AddingDate);

}

[Authorize]

[Route("[controller]/[action]")]

public IActionResult FilmSelection(

string genresFilter = "All",

string worldPremierFilter = "",

string durationFilter = "",

string ageRatingsFilter = "All",

string ratingFilter = "")

{

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var model = CreateFilmSelectionListViewModel(user, genresFilter, worldPremierFilter, durationFilter, ageRatingsFilter, ratingFilter);

ViewBag.DidUserWatch = null;

if (model.SelectionInfo.CurrentFilm != null)

{

var watchedFilm =

user?.WatchedFilms.SingleOrDefault(x => x.FilmId == model.SelectionInfo.CurrentFilm.Id);

if (watchedFilm == null)

{

ViewBag.WillUserWatch = false;

ViewBag.DidUserWatch = false;

}

else

{

ViewBag.WillUserWatch = watchedFilm.Watched.HasValue && !watchedFilm.Watched.Value &&

!watchedFilm.Rating.HasValue;

ViewBag.DidUserWatch = watchedFilm.Watched != null && watchedFilm.Watched.Value;

ViewBag.UserRating = watchedFilm.Rating;

}

}

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(model);

}

[Authorize]

[Route("[controller]/[action]")]

public IActionResult GetFilmSelectionView(

string genresFilter = "All",

string worldPremierFilter = "",

string durationFilter = "",

string ageRatingsFilter = "All",

string ratingFilter = "")

{

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var model = CreateFilmSelectionListViewModel(user, genresFilter, worldPremierFilter, durationFilter, ageRatingsFilter, ratingFilter);

ViewBag.DidUserWatch = null;

if (model.SelectionInfo.CurrentFilm != null)

{

var watchedFilm =

user?.WatchedFilms.SingleOrDefault(x => x.FilmId == model.SelectionInfo.CurrentFilm.Id);

if (watchedFilm == null)

{

ViewBag.WillUserWatch = false;

ViewBag.DidUserWatch = false;

}

else

{

ViewBag.WillUserWatch = watchedFilm.Watched.HasValue && !watchedFilm.Watched.Value &&

!watchedFilm.Rating.HasValue;

ViewBag.DidUserWatch = watchedFilm.Watched != null && watchedFilm.Watched.Value;

ViewBag.UserRating = watchedFilm.Rating;

}

}

return PartialView("\_GetFilmsSelection", model);

}

[Authorize]

[Route("[controller]/[action]")]

public IActionResult GetFilmSelectionInfo(string ids, int index)

{

var user = \_uow.UserRepository.FindUserByNickname(User.Identity.Name);

if (user == null)

{

return NotFound();

}

var model = new FilmSelectionViewModel();

model.CurrentIndex = index;

model.FoundFilmsIds = ids.Split(",").Select(Guid.Parse).ToList();

if (index >= model.FoundFilmsIds.Count)

{

model.CurrentIndex--;

}

model.CurrentFilm = \_uow.FilmRepository.Find(model.FoundFilmsIds[model.CurrentIndex]);

ViewBag.DidUserWatch = null;

if (model.CurrentFilm != null)

{

var watchedFilm =

user?.WatchedFilms.SingleOrDefault(x => x.FilmId == model.CurrentFilm.Id);

if (watchedFilm == null)

{

ViewBag.WillUserWatch = false;

ViewBag.DidUserWatch = false;

}

else

{

ViewBag.WillUserWatch = watchedFilm.Watched.HasValue && !watchedFilm.Watched.Value &&

!watchedFilm.Rating.HasValue;

ViewBag.DidUserWatch = watchedFilm.Watched != null && watchedFilm.Watched.Value;

ViewBag.UserRating = watchedFilm.Rating;

}

}

return PartialView("\_GetFilmSelectionInfo", model);

}

private FilmListViewModel CreateFilmListViewModel(

string filmNameFilter,

string genresFilter,

string worldPremierFilter,

string durationFilter,

string ageRatingsFilter,

string ratingFilter,

FilmSortState sortOrder,

int currentPage)

{

var model = new FilmListViewModel() { CurrentPage = currentPage };

//Films

var films = \_uow.FilmRepository.ListWithShortInclude().ToList();

films = FilterFilmsByFilmName(model, films, filmNameFilter).ToList();

var allGenres = \_uow.GenreRepository.List().ToList();

model.AllGenres = allGenres.Select(x => new FilmGenreViewModel() { Id = x.Id, Name = x.Name }).ToList();

films = FilterFilmsByGenres(model, films, allGenres, genresFilter).ToList();

films = FilterFilmsByWorldPremier(model, films, worldPremierFilter).ToList();

films = FilterFilmsByDuration(model, films, durationFilter).ToList();

var allAgeRatings = \_uow.AgeRatingRepository.List().ToList();

model.AllAgeRatings = allAgeRatings

.Select(x => new FilmAgeRatingViewModel() {Id = x.Id, Name = x.Name})

.ToList();

films = FilterFilmsByAgeRatings(model, films, allAgeRatings, ageRatingsFilter).ToList();

films = FilterFilmsByRating(model, films, ratingFilter).ToList();

films = SortFilms(films, sortOrder).ToList();

model.SortModel = new FilmSortModel(sortOrder);

model.AllElementsAmount = films.Count;

model.MaxPages = films.Count() / model.FilmsMaxSizePerPage;

if (films.Count() % model.FilmsMaxSizePerPage != 0)

{

model.MaxPages++;

}

model.Films = films

.Skip((model.CurrentPage - 1) \* model.FilmsMaxSizePerPage)

.Take(model.FilmsMaxSizePerPage);

return model;

}

private FilmSelectionListViewModel CreateFilmSelectionListViewModel(

User user,

string genresFilter,

string worldPremierFilter,

string durationFilter,

string ageRatingsFilter,

string ratingFilter)

{

var model = new FilmSelectionListViewModel();

//Films

var films = \_uow.FilmRepository.ListWithAllInclude().ToList();

films = films.Where(x => x.WatchedFilms == null || (x.WatchedFilms != null && !x.WatchedFilms.Any(x => x.UserId == user.Id))).ToList();

if (user.FavouriteFilmses != null && user.FavouriteFilmses.Any())

{

films = films.Where(x => user.FavouriteFilmses.All(y => y.FilmId != x.Id)).ToList();

}

var allGenres = \_uow.GenreRepository.List().ToList();

model.AllGenres = allGenres.Select(x => new FilmGenreViewModel() { Id = x.Id, Name = x.Name }).ToList();

films = FilterFilmsByGenres(model, films, allGenres, genresFilter).ToList();

films = FilterFilmsByWorldPremier(model, films, worldPremierFilter).ToList();

films = FilterFilmsByDuration(model, films, durationFilter).ToList();

var allAgeRatings = \_uow.AgeRatingRepository.List().ToList();

model.AllAgeRatings = allAgeRatings

.Select(x => new FilmAgeRatingViewModel() { Id = x.Id, Name = x.Name })

.ToList();

films = FilterFilmsByAgeRatings(model, films, allAgeRatings, ageRatingsFilter).ToList();

films = FilterFilmsByRating(model, films, ratingFilter).ToList();

DoSelection(model, user, films);

model.AllElementsAmount = films.Count;

return model;

}

private void DoSelection(FilmSelectionListViewModel model, User user, List<Film> films)

{

var selectedFilmsIds = new List<Guid>();

IEnumerable<Guid> filmsWithFavoriteGenres = null,

filmsWithFavoriteParticipaters = null,

filmsByGenresOfFavoriteFilms = null,

filmsByAgeRatingOfFavoriteFilms = null,

filmsByGenresOfRatedFilms = null,

filmsByAgeRatingOfRatedFilms = null,

filmsByGenresOfWillWatchFilms = null,

filmsByAgeRatingOfWillWatchFilms = null;

if (user.FavouriteGenreses != null && user.FavouriteGenreses.Any())

{

filmsWithFavoriteGenres = films

.Where(x =>

x.FilmGenres

.Any(y => user.FavouriteGenreses

.Any(z => z.GenreId == y.GenreId)))

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier).ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

}

if (user.FavouriteParticipaterses != null && user.FavouriteParticipaterses.Any())

{

filmsWithFavoriteParticipaters = films

.Where(x =>

x.FilmParticipaterRoles

.Any(x => user.FavouriteParticipaterses

.Any(y => y.ParticipaterId == x.ParticipaterId)))

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier).ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

}

if (user.FavouriteFilmses != null && user.FavouriteFilmses.Any())

{

var genresIds = new List<int>();

var ageRatingsIds = new List<int>();

foreach (var favourite in user.FavouriteFilmses)

{

var film = favourite.Film;

genresIds.AddRange(film.FilmGenres.Select(x => x.GenreId));

if (film.AgeRatingId.HasValue)

{

ageRatingsIds.Add(film.AgeRatingId.Value);

}

}

var groupedGenresIds = genresIds.GroupBy(x => x).Select(x => new {Id = x.Key, Count = x.Count()}).ToList();

var resultGenres = groupedGenresIds.OrderBy(x => x.Count).Take(SELECTION\_DEFAULT\_TAKE).Select(x => x.Id).ToList();

filmsByGenresOfFavoriteFilms = films

.Where(x =>

x.FilmGenres

.Any(y => resultGenres

.Any(z => z == y.GenreId)))

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier).ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

var groupedAgeRatingsIds = ageRatingsIds.GroupBy(x => x).Select(x => new { Id = x.Key, Count = x.Count() }).ToList();

var resultAgeRating = groupedAgeRatingsIds.OrderBy(x => x.Count).Select(x => x.Id).Take(1).ToList();

if (resultAgeRating.Any())

{

filmsByAgeRatingOfFavoriteFilms = films

.Where(x =>

x.AgeRatingId.HasValue && x.AgeRatingId.Value == resultAgeRating.First())

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier)

.ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

}

}

if (user.WatchedFilms != null && user.WatchedFilms.Any())

{

var ratedFilms = user.WatchedFilms.Where(x => x.Rating.HasValue).Select(x => x.Film).ToList();

if (ratedFilms.Any())

{

var genresIds = new List<int>();

var ageRatingsIds = new List<int>();

foreach (var film in ratedFilms)

{

genresIds.AddRange(film.FilmGenres.Select(x => x.GenreId));

if (film.AgeRatingId.HasValue)

{

ageRatingsIds.Add(film.AgeRatingId.Value);

}

}

var groupedGenresIds = genresIds.GroupBy(x => x).Select(x => new { Id = x.Key, Count = x.Count() }).ToList();

var resultGenres = groupedGenresIds.OrderBy(x => x.Count).Take(SELECTION\_DEFAULT\_TAKE).Select(x => x.Id).ToList();

filmsByGenresOfRatedFilms = films

.Where(x =>

x.FilmGenres

.Any(y => resultGenres

.Any(z => z == y.GenreId)))

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier).ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

var groupedAgeRatingsIds = ageRatingsIds.GroupBy(x => x).Select(x => new { Id = x.Key, Count = x.Count() }).ToList();

var resultAgeRating = groupedAgeRatingsIds.OrderBy(x => x.Count).Select(x => x.Id).Take(1).ToList();

if (resultAgeRating.Any())

{

filmsByAgeRatingOfRatedFilms = films

.Where(x =>

x.AgeRatingId.HasValue && x.AgeRatingId.Value == resultAgeRating.First())

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier)

.ThenByDescending(x => x.Views)

.Select(x => x.Id)

.Take(SELECTION\_DEFAULT\_TAKE)

.ToList();

}

}

var willWatchFilms = user.WatchedFilms.Where(x => x.Watched.HasValue && !x.Watched.Value && !x.Rating.HasValue).Select(x => x.Film).ToList();

if (willWatchFilms.Any())

{

var genresIds = new List<int>();

var ageRatingsIds = new List<int>();

foreach (var film in willWatchFilms)

{

genresIds.AddRange(film.FilmGenres.Select(x => x.GenreId));

if (film.AgeRatingId.HasValue)

{

ageRatingsIds.Add(film.AgeRatingId.Value);

}

}

var groupedGenresIds = genresIds.GroupBy(x => x).Select(x => new { Id = x.Key, Count = x.Count() }).ToList();

var resultGenres = groupedGenresIds.OrderBy(x => x.Count).Take(SELECTION\_DEFAULT\_TAKE).Select(x => x.Id).ToList();

filmsByGenresOfWillWatchFilms = films

.Where(x =>

x.FilmGenres

.Any(y => resultGenres

.Any(z => z == y.GenreId)))

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier).ThenByDescending(x => x.Views)

.Select(x => x.Id)

.ToList();

var groupedAgeRatingsIds = ageRatingsIds.GroupBy(x => x).Select(x => new { Id = x.Key, Count = x.Count() }).ToList();

var resultAgeRating = groupedAgeRatingsIds.OrderBy(x => x.Count).Select(x => x.Id).Take(1).ToList();

if (resultAgeRating.Any())

{

filmsByAgeRatingOfWillWatchFilms = films

.Where(x =>

x.AgeRatingId.HasValue && x.AgeRatingId.Value == resultAgeRating.First())

.OrderByDescending(x => x.Rating).ThenByDescending(x => x.WorldPremier)

.ThenByDescending(x => x.Views)

.Select(x => x.Id)

.Take(SELECTION\_DEFAULT\_TAKE)

.ToList();

}

}

}

if (filmsWithFavoriteGenres != null && filmsWithFavoriteGenres.Any())

{

selectedFilmsIds.AddRange(filmsWithFavoriteGenres);

}

if (filmsWithFavoriteParticipaters != null && filmsWithFavoriteParticipaters.Any())

{

selectedFilmsIds.AddRange(filmsWithFavoriteParticipaters);

}

if (filmsByGenresOfFavoriteFilms != null && filmsByGenresOfFavoriteFilms.Any())

{

selectedFilmsIds.AddRange(filmsByGenresOfFavoriteFilms);

}

if (filmsByAgeRatingOfFavoriteFilms != null && filmsByAgeRatingOfFavoriteFilms.Any())

{

selectedFilmsIds.AddRange(filmsByAgeRatingOfFavoriteFilms);

}

if (filmsByGenresOfRatedFilms != null && filmsByGenresOfRatedFilms.Any())

{

selectedFilmsIds.AddRange(filmsByGenresOfRatedFilms);

}

if (filmsByAgeRatingOfRatedFilms != null && filmsByAgeRatingOfRatedFilms.Any())

{

selectedFilmsIds.AddRange(filmsByAgeRatingOfRatedFilms);

}

if (filmsByGenresOfWillWatchFilms != null && filmsByGenresOfWillWatchFilms.Any())

{

selectedFilmsIds.AddRange(filmsByGenresOfWillWatchFilms);

}

if (filmsByAgeRatingOfWillWatchFilms != null && filmsByAgeRatingOfWillWatchFilms.Any())

{

selectedFilmsIds.AddRange(filmsByAgeRatingOfWillWatchFilms);

}

model.SelectionInfo = new FilmSelectionViewModel()

{

CurrentIndex = 0

};

if (selectedFilmsIds.Any())

{

selectedFilmsIds = selectedFilmsIds.Distinct().ToList();

model.AllElementsAmount = selectedFilmsIds.Count();

model.SelectionInfo.FoundFilmsIds = selectedFilmsIds;

model.SelectionInfo.CurrentFilm =

\_uow.FilmRepository.Find(selectedFilmsIds[model.SelectionInfo.CurrentIndex]);

}

else

{

model.SelectionInfo.FoundFilmsIds = new List<Guid>();

}

}

private IEnumerable<Film> FilterFilmsByFilmName(

IFilmsListViewModel model,

IEnumerable<Film> films,

string filmNameFilter)

{

model.FilmNameFilter = "";

if (!string.IsNullOrEmpty(filmNameFilter))

{

filmNameFilter = filmNameFilter.Trim();

films = films

.Where(x => x.RussianName.Contains(filmNameFilter) || x.OriginalName.Contains(filmNameFilter))

.ToList();

model.FilmNameFilter = filmNameFilter;

}

return films;

}

private IEnumerable<Film> FilterFilmsByGenres(

IFilmsListViewModel model,

IEnumerable<Film> films,

IEnumerable<Genre> allGenres,

string genresFilter)

{

if (string.IsNullOrEmpty(genresFilter) || genresFilter == "All")

{

model.SelectedGenres = allGenres.Select(x => x.Id);

model.GenresFilter = genresFilter;

films = films.Where(x => x.FilmGenres.Any());

}

else

{

var selectedGenresId = genresFilter.Split(",").Select(int.Parse).ToList();

model.SelectedGenres = selectedGenresId;

model.GenresFilter = genresFilter;

films = films.Where(x => x.FilmGenres.Any(x => selectedGenresId.Contains(x.GenreId)));

}

return films;

}

private IEnumerable<Film> FilterFilmsByWorldPremier(

IFilmsListViewModel model,

IEnumerable<Film> films,

string worldPremierFilter)

{

model.DurationFilter = "";

if (!string.IsNullOrEmpty(worldPremierFilter))

{

var selectedWorldPremierArray = worldPremierFilter.Split(",");

if (selectedWorldPremierArray.Length == 2)

{

var worldPremierRangeStart = int.Parse(selectedWorldPremierArray[0]);

var worldPremierRangeEnd = int.Parse(selectedWorldPremierArray[1]);

if (worldPremierRangeStart != model.WorldPremierFilterMin || worldPremierRangeEnd != model.WorldPremierFilterMax)

{

if (worldPremierRangeStart >= model.WorldPremierFilterMin &&

worldPremierRangeStart <= model.WorldPremierFilterMax &&

worldPremierRangeStart <= worldPremierRangeEnd)

{

films = films.Where(x =>

x.WorldPremier.HasValue && x.WorldPremier.Value.Year >= worldPremierRangeStart &&

x.WorldPremier.Value.Year <= worldPremierRangeEnd);

model.WorldPremierFilter = worldPremierFilter;

}

}

}

}

return films;

}

private IEnumerable<Film> FilterFilmsByDuration(

IFilmsListViewModel model,

IEnumerable<Film> films,

string durationFilter)

{

model.DurationFilter = "";

if (!string.IsNullOrEmpty(durationFilter))

{

var selectedDurationArray = durationFilter.Split(",");

if (selectedDurationArray.Length == 2)

{

var durationRangeStart = int.Parse(selectedDurationArray[0]);

var durationRangeEnd = int.Parse(selectedDurationArray[1]);

if (durationRangeStart != model.DurationFilterMin || durationRangeEnd != model.DurationFilterMax)

{

if (durationRangeStart >= model.DurationFilterMin &&

durationRangeStart <= model.DurationFilterMax &&

durationRangeStart <= durationRangeEnd)

{

films = films.Where(x =>

x.Duration.HasValue && x.Duration.Value >= durationRangeStart &&

x.Duration.Value <= durationRangeEnd);

model.DurationFilter = durationFilter;

}

}

}

}

return films;

}

private IEnumerable<Film> FilterFilmsByAgeRatings(

IFilmsListViewModel model,

IEnumerable<Film> films,

IEnumerable<AgeRating> allAgeRatings,

string ageRatingFilter)

{

if (string.IsNullOrEmpty(ageRatingFilter) || ageRatingFilter == "All")

{

model.SelectedAgeRatings = allAgeRatings.Select(x => x.Id);

model.AgeRatingsFilter = ageRatingFilter;

films = films.Where(x => x.AgeRating != null);

}

else

{

var selectedAgeRatingsId = ageRatingFilter.Split(",").Select(int.Parse).ToList();

model.SelectedAgeRatings = selectedAgeRatingsId;

model.AgeRatingsFilter = ageRatingFilter;

films = films.Where(x => x.AgeRating != null && selectedAgeRatingsId.Contains(x.AgeRating.Id));

}

return films;

}

private IEnumerable<Film> FilterFilmsByRating(

IFilmsListViewModel model,

IEnumerable<Film> films,

string ratingFilter)

{

model.RatingFilter = "";

if (!string.IsNullOrEmpty(ratingFilter))

{

var selectedRatingArray = ratingFilter.Split(",");

if (selectedRatingArray.Length == 2)

{

var ratingRangeStart = int.Parse(selectedRatingArray[0]);

var ratingRangeEnd = int.Parse(selectedRatingArray[1]);

if (ratingRangeStart != model.RatingFilterMin || ratingRangeEnd != model.RatingFilterMax)

{

if (ratingRangeStart >= model.RatingFilterMin &&

ratingRangeStart <= model.RatingFilterMax &&

ratingRangeStart <= ratingRangeEnd)

{

films = films.Where(x =>

x.Rating.HasValue

&& x.Rating.Value >= ratingRangeStart

&& x.Rating.Value <= ratingRangeEnd);

model.RatingFilter = ratingFilter;

}

}

}

}

return films;

}

private IEnumerable<Film> SortFilms(IEnumerable<Film> films, FilmSortState sortOrder)

{

switch (sortOrder)

{

case FilmSortState.RussianNameAsc:

films = films.OrderBy(x => x.RussianName);

break;

case FilmSortState.RussianNameDesc:

films = films.OrderByDescending(x => x.RussianName);

break;

case FilmSortState.WorldPremierAsc:

films = films.OrderBy(x => x.WorldPremier);

break;

case FilmSortState.WorldPremierDesc:

films = films.OrderByDescending(x => x.WorldPremier);

break;

case FilmSortState.RatingAsc:

films = films.OrderBy(x => x.Rating);

break;

case FilmSortState.RatingDesc:

films = films.OrderByDescending(x => x.Rating);

break;

case FilmSortState.AddingDateAsc:

films = films.OrderBy(x => x.AddingDate);

break;

case FilmSortState.AddingDateDesc:

films = films.OrderByDescending(x => x.AddingDate);

break;

}

return films;

}

}

}

namespace MovieOnline.Web.Controllers

{

public class ParticipaterController : Controller

{

private readonly IUnitOfWork \_uow;

public ParticipaterController(IUnitOfWork uow) => \_uow = uow;

[Route("[controller]/[action]/{id:guid}")]

public IActionResult Details(Guid id)

{

var participater = \_uow.ParticipaterRepository.Find(id);

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return participater == null ? (IActionResult)NotFound() : View(participater);

}

}

}

namespace MovieOnline.Web.Areas.Admin.Controllers

{

[Area("Admin")]

[Authorize(Roles = "Admin")]

public class FilmController : Controller

{

private readonly IUnitOfWork \_uow;

public FilmController(IUnitOfWork uow) => \_uow = uow;

[HttpGet("[area]/[controller]/[action]")]

public IActionResult Create()

{

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(CreateFilmEditViewModel());

}

[HttpPost("[area]/[controller]/[action]")]

public IActionResult Create(FilmEditViewModel model)

{

var isValid = ModelState.IsValid

&& !string.IsNullOrEmpty(model.PosterUrl)

&& (model.SelectedGenres != null && model.SelectedGenres.Any())

&& (model.SelectedCountries != null && model.SelectedCountries.Any())

&& (model.SelectedCompanies != null && model.SelectedCompanies.Any());

if (!isValid)

{

return View(CreateFilmEditViewModel());

}

var film = new Film()

{

Id = Guid.NewGuid(),

RussianName = model.RussianName,

OriginalName = model.OriginalName,

Slogan = model.Slogan,

WorldPremier = model.WorldPremier,

Budget = model.Budget,

Fees = model.Fees,

Duration = model.Duration,

Description = model.Description,

PosterUrl = model.PosterUrl,

AgeRatingId = model.AgeRatingId

};

ParseSelectedParticipatersWithRoles(film, model.SelectedParticipatersWithRoles);

ParseSelectedVideoQualitiesWithVideoUrls(film, model.SelectedVideoQualitiesWithVideoUrls);

ParseSelectedGenres(film, model.SelectedGenres);

ParseSelectedCountries(film, model.SelectedCountries);

ParseSelectedCompanies(film, model.SelectedCompanies);

\_uow.FilmRepository.Add(film);

\_uow.Commit();

//return RedirectToAction("List", new RouteValueDictionary(new { area = "", controller = "Film", action = "List"}));

//return Content("/Film/List");

return Content("/Film/Details/" + film.Id);

}

[HttpGet("[area]/[controller]/[action]/{id:guid}")]

public IActionResult Edit(Guid id)

{

var film = \_uow.FilmRepository.Find(id);

if (film == null)

{

return NotFound();

}

var model = CreateFilmEditViewModel(film);

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(model);

}

[HttpPost("[area]/[controller]/[action]")]

public IActionResult Edit(FilmEditViewModel model)

{

var isValid = ModelState.IsValid

&& !string.IsNullOrEmpty(model.PosterUrl)

&& (model.SelectedGenres != null && model.SelectedGenres.Any())

&& (model.SelectedCountries != null && model.SelectedCountries.Any())

&& (model.SelectedCompanies != null && model.SelectedCompanies.Any());

if (!isValid)

{

return View(CreateFilmEditViewModel());

}

var film = \_uow.FilmRepository.Find(model.Id);

if (film == null)

{

return NotFound();

}

film.RussianName = model.RussianName;

film.OriginalName = model.OriginalName;

film.Slogan = model.Slogan;

film.WorldPremier = model.WorldPremier;

film.Budget = model.Budget;

film.Fees = model.Fees;

film.Duration = model.Duration;

film.Description = model.Description;

film.PosterUrl = model.PosterUrl;

film.AgeRatingId = model.AgeRatingId;

ParseSelectedParticipatersWithRoles(film, model.SelectedParticipatersWithRoles);

ParseSelectedVideoQualitiesWithVideoUrls(film, model.SelectedVideoQualitiesWithVideoUrls);

ParseSelectedGenres(film, model.SelectedGenres);

ParseSelectedCountries(film, model.SelectedCountries);

ParseSelectedCompanies(film, model.SelectedCompanies);

\_uow.FilmRepository.Update(film);

\_uow.Commit();

//return RedirectToAction("List", new RouteValueDictionary(new { area = "", controller = "Film", action = "List" }));

//return Content("/Film/List");

return Content("/Film/Details/" + film.Id);

}

[Route("[area]/[controller]/[action]")]

public IActionResult Delete(

Guid filmId,

string filmNameFilter = "",

string genresFilter = "All",

string worldPremierFilter = "",

string durationFilter = "",

string ageRatingsFilter = "All",

string ratingFilter = "",

FilmSortState sortOrder = FilmSortState.WorldPremierDesc,

int page = 1)

{

var film = \_uow.FilmRepository.Find(filmId);

if (film == null)

{

return NotFound();

}

\_uow.FilmRepository.Delete(film);

\_uow.Commit();

return RedirectToAction(

"GetFilmsList",

new RouteValueDictionary(new

{

area = "",

controller = "Film",

action = "GetFilmsList",

filmNameFilter,

genresFilter,

worldPremierFilter,

durationFilter,

ageRatingsFilter,

ratingFilter,

sortOrder,

page

}));

}

[Route("[area]/[controller]/[action]")]

public IActionResult GetSelect(string type, int index)

{

return PartialView("\_Select", CreateSelectViewModel(type, index));

}

[Route("[area]/[controller]/[action]")]

public IActionResult GetMultiSelect(string type, int index)

{

return PartialView("\_MultiSelect", CreateMultiSelectViewModel(type, index));

}

private FilmEditViewModel CreateFilmEditViewModel(Film film = null)

{

var model = new FilmEditViewModel();

model.AllAgeRatings = \_uow.AgeRatingRepository.List().Select(x => new FilmAgeRatingViewModel() { Id = x.Id, Name = x.Name });

model.AllGenres = \_uow.GenreRepository.List().Select(x => new FilmGenreViewModel() { Id = x.Id, Name = x.Name });

model.AllCompanies = \_uow.CompanyRepository.List().Select(x => new FilmCompanyViewModel() { Id = x.Id, Name = x.Name });

model.AllCountries = \_uow.CountryRepository.List().Select(x => new FilmCountryViewModel() { Id = x.Id, Name = x.Name });

model.AllParticipaters = \_uow.ParticipaterRepository

.List()

.Select(

x => new FilmParticipaterViewModel()

{

Id = x.Id,

FullName = x.GetFullName(),

BirthDay = x.GetBirthdayString(),

PhotoUrl = x.PhotoUrl

})

.OrderBy(x => x.FullName);

model.AllRoles = \_uow.RoleRepository.List().Select(x => new FilmRoleViewModel() { Id = x.Id, Name = x.Name });

model.AllVideoQualities = \_uow.VideoQualityRepository.List().Select(x => new FilmVideoQualityViewModel() { Id = x.Id, Name = x.Name });

if (film != null)

{

model.RussianName = film.RussianName;

model.OriginalName = film.OriginalName;

model.Slogan = film.Slogan;

model.PosterUrl = film.PosterUrl;

model.AgeRatingId = film.AgeRatingId;

model.WorldPremier = film.WorldPremier;

model.Budget = film.Budget;

model.Fees = film.Fees;

model.Duration = film.Duration;

model.Description = film.Description;

model.SelectedGenres = film.FilmGenres.Select(x => x.GenreId);

model.SelectedCountries = film.FilmCountries.Select(x => x.CountryId);

model.SelectedCompanies = film.FilmCompanies.Select(x => x.CompanyId);

model.SelectedParticipatersWithRoles = film.FilmParticipaterRoles.Select(x => (x.ParticipaterId, x.RoleId)).GroupBy(x => x.ParticipaterId).Select(x => x.Key.ToString() + ", " + string.Join(", ", x.Select(y => y.RoleId.ToString())));

model.SelectedVideoQualitiesWithVideoUrls = film.FilmVideoQualities.Select(x => (x.VideoQualityId, x.VideoName)).GroupBy(x => x.VideoQualityId).Select(x => x.Key.ToString() + ", " + string.Join(", ", x.Select(y => y.VideoName)));

}

return model;

}

private void ParseSelectedGenres(Film film, IEnumerable<int> genres)

{

film.FilmGenres = new List<FilmGenre>();

if (genres != null && genres.Any())

{

foreach (var genre in genres)

{

film.FilmGenres.Add(new FilmGenre()

{

FilmId = film.Id,

GenreId = genre

});

}

}

}

private void ParseSelectedCountries(Film film, IEnumerable<int> counties)

{

film.FilmCountries = new List<FilmCountry>();

if (counties != null && counties.Any())

{

foreach (var county in counties)

{

film.FilmCountries.Add(new FilmCountry()

{

FilmId = film.Id,

CountryId = county

});

}

}

}

private void ParseSelectedCompanies(Film film, IEnumerable<int> companies)

{

film.FilmCompanies = new List<FilmCompany>();

if (companies != null && companies.Any())

{

foreach (var company in companies)

{

film.FilmCompanies.Add(new FilmCompany()

{

FilmId = film.Id,

CompanyId = company

});

}

}

}

private void ParseSelectedParticipatersWithRoles(Film film, IEnumerable<string> participatersRoles)

{

if (participatersRoles!= null && participatersRoles.Any())

{

var participaterRoles = new List<(Guid participaterId, IEnumerable<int> rolesIds)>();

foreach (var item in participatersRoles)

{

var splittedItem = item.Split(",");

if (splittedItem.Length <= 1)

{

continue;

}

var participaterId = Guid.Parse(splittedItem[0]);

var rolesIds = new List<int>();

for (int i = 1; i < splittedItem.Length; i++)

{

rolesIds.Add(int.Parse(splittedItem[i]));

}

participaterRoles.Add((participaterId, rolesIds));

}

var participaterRolesCopy = new List<(Guid participaterId, IEnumerable<int> rolesIds)>(participaterRoles);

var groupedParticipatersRoles = participaterRolesCopy.GroupBy(x => x.participaterId);

participaterRoles.Clear();

foreach (var item in groupedParticipatersRoles)

{

var disctinctRoles = new List<int>();

foreach (var roles in item)

{

disctinctRoles.AddRange(roles.rolesIds);

}

disctinctRoles = disctinctRoles.Distinct().ToList();

participaterRoles.Add((item.Key, disctinctRoles));

}

film.FilmParticipaterRoles = new List<FilmParticipaterRole>();

foreach (var participaterRole in participaterRoles)

{

foreach (var roleId in participaterRole.rolesIds)

{

film.FilmParticipaterRoles.Add(new FilmParticipaterRole()

{

FilmId = film.Id,

ParticipaterId = participaterRole.participaterId,

RoleId = roleId

});

}

}

}

else

{

film.FilmParticipaterRoles = new List<FilmParticipaterRole>();

}

}

private void ParseSelectedVideoQualitiesWithVideoUrls(Film film, IEnumerable<string> videoQualitiesWithUrls)

{

if (videoQualitiesWithUrls != null && videoQualitiesWithUrls.Any())

{

var videoQualityWithUrls = new List<(int videoQualityId, IEnumerable<string> urls)>();

foreach (var item in videoQualitiesWithUrls)

{

var splittedItem = item.Split(",");

if (splittedItem.Length <= 1)

{

continue;

}

var videoQualityId = int.Parse(splittedItem[0]);

var urls = new List<string>();

for (int i = 1; i < splittedItem.Length; i++)

{

urls.Add(splittedItem[i]);

}

videoQualityWithUrls.Add((videoQualityId, urls));

}

film.FilmVideoQualities = new List<FilmVideoQuality>();

foreach (var videoQualityUrl in videoQualityWithUrls)

{

foreach (var url in videoQualityUrl.urls)

{

film.FilmVideoQualities.Add(new FilmVideoQuality()

{

FilmId = film.Id,

VideoQualityId = videoQualityUrl.videoQualityId,

VideoName = url

});

}

}

}

else

{

film.FilmVideoQualities = new List<FilmVideoQuality>();

}

}

private SelectViewModel CreateSelectViewModel(string type, int index)

{

var model = new SelectViewModel()

{

Type = type,

Index = index

};

switch (type)

{

case "Participater":

model.AllParticipaters = \_uow.ParticipaterRepository

.List()

.Select(

x => new FilmParticipaterViewModel()

{

Id = x.Id,

FullName = x.GetFullName(),

BirthDay = x.GetBirthdayString(),

PhotoUrl = x.PhotoUrl

})

.OrderBy(x => x.FullName);

break;

case "Role":

model.AllRoles = \_uow.RoleRepository.List().Select(x => new FilmRoleViewModel() { Id = x.Id, Name = x.Name });

break;

case "VideoQuality":

model.AllVideoQualities = \_uow.VideoQualityRepository.List().Select(x => new FilmVideoQualityViewModel() { Id = x.Id, Name = x.Name });

break;

}

return model;

}

private MultiSelectViewModel CreateMultiSelectViewModel(string type, int index)

{

var model = new MultiSelectViewModel()

{

Type = type,

Index = index

};

switch (type)

{

case "Participaters":

model.AllParticipaters = \_uow.ParticipaterRepository

.List()

.Select(

x => new FilmParticipaterViewModel()

{

Id = x.Id,

FullName = x.GetFullName(),

BirthDay = x.GetBirthdayString(),

PhotoUrl = x.PhotoUrl

})

.OrderBy(x => x.FullName);

break;

case "Roles":

model.AllRoles = \_uow.RoleRepository.List().Select(x => new FilmRoleViewModel() { Id = x.Id, Name = x.Name });

break;

case "VideoQualities":

model.AllVideoQualities = \_uow.VideoQualityRepository.List().Select(x => new FilmVideoQualityViewModel() { Id = x.Id, Name = x.Name });

break;

}

return model;

}

}

}

namespace MovieOnline.Web.Areas.Admin.Controllers

{

[Area("Admin")]

[Authorize(Roles = "Admin")]

public class ParticipaterController : Controller

{

private readonly IUnitOfWork \_uow;

public ParticipaterController(IUnitOfWork uow) => \_uow = uow;

[HttpGet("[area]/[controller]/[action]")]

public IActionResult Create()

{

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(CreateParticipaterEditViewModel());

}

[HttpPost("[area]/[controller]/[action]")]

public IActionResult Create(ParticipaterEditViewModel model)

{

var isValid = ModelState.IsValid

&& !string.IsNullOrEmpty(model.PhotoUrl);

if (!isValid)

{

return View(CreateParticipaterEditViewModel());

}

var participater = new Participater()

{

Id = Guid.NewGuid(),

FirstName = model.FirstName,

LastName = model.LastName,

Birthday = model.BirthDay,

Gender = model.Gender,

Growth = model.Growth,

PhotoUrl = model.PhotoUrl

};

ParseSelectedFilmsWithRoles(participater, model.SelectedFilmsWithRoles);

\_uow.ParticipaterRepository.Add(participater);

\_uow.Commit();

//return RedirectToAction("List", new RouteValueDictionary(new { area = "", controller = "Film", action = "List" }));

return Content("/Participater/Details/" + participater.Id);

}

[HttpGet("[area]/[controller]/[action]/{id:guid}")]

public IActionResult Edit(Guid id)

{

var participater = \_uow.ParticipaterRepository.Find(id);

if (participater == null)

{

return NotFound();

}

var model = CreateParticipaterEditViewModel(participater);

ViewBag.Username = User.Identity.Name ?? "anonymous";

ViewBag.UserRole = User.FindFirst(ClaimsIdentity.DefaultRoleClaimType)?.Value ?? "anonymous";

return View(model);

}

[HttpPost("[area]/[controller]/[action]")]

public IActionResult Edit(ParticipaterEditViewModel model)

{

var isValid = ModelState.IsValid

&& !string.IsNullOrEmpty(model.PhotoUrl);

if (!isValid)

{

return View(CreateParticipaterEditViewModel());

}

var participater = \_uow.ParticipaterRepository.Find(model.Id);

if (participater == null)

{

return NotFound();

}

participater.FirstName = model.FirstName;

participater.LastName = model.LastName;

participater.Birthday = model.BirthDay;

participater.Gender = model.Gender;

participater.Growth = model.Growth;

participater.PhotoUrl = model.PhotoUrl;

ParseSelectedFilmsWithRoles(participater, model.SelectedFilmsWithRoles);

\_uow.ParticipaterRepository.Update(participater);

\_uow.Commit();

//return RedirectToAction("List", new RouteValueDictionary(new { area = "", controller = "Film", action = "List" }));

return Content("/Participater/Details/" + participater.Id);

}

[Route("[area]/[controller]/[action]/{id:guid}")]

public IActionResult Delete(Guid id)

{

var participater = \_uow.ParticipaterRepository.Find(id);

if (participater == null)

{

return NotFound();

}

\_uow.ParticipaterRepository.Delete(participater);

\_uow.Commit();

return RedirectToAction(

"List",

new RouteValueDictionary(new

{

area = "",

controller = "Film",

action = "List"

}));

}

private ParticipaterEditViewModel CreateParticipaterEditViewModel(Participater participater = null)

{

var model = new ParticipaterEditViewModel();

model.AllFilms = \_uow.FilmRepository

.List()

.Select(

x => new ParticipaterFilmViewModel()

{

Id = x.Id,

RussianName = x.RussianName,

WorldPremierYear = x.WorldPremier.HasValue ? x.WorldPremier.Value.Year : (int?)null,

PosterUrl = x.PosterUrl

})

.OrderBy(x => x.RussianName);

model.AllRoles = \_uow.RoleRepository.List().Select(x => new ParticipaterRoleViewModel() { Id = x.Id, Name = x.Name });

if (participater != null)

{

model.FirstName = participater.FirstName;

model.LastName = participater.LastName;

model.BirthDay = participater.Birthday;

model.Gender = participater.Gender;

model.PhotoUrl = participater.PhotoUrl;

model.Growth = participater.Growth;

model.SelectedFilmsWithRoles = participater.FilmParticipaterRoles.Select(x => (x.FilmId, x.RoleId)).GroupBy(x => x.FilmId).Select(x => x.Key.ToString() + ", " + string.Join(", ", x.Select(y => y.RoleId.ToString())));

}

return model;

}

private void ParseSelectedFilmsWithRoles(Participater participater, IEnumerable<string> filmsRoles)

{

if (filmsRoles != null && filmsRoles.Any())

{

var filmRoles = new List<(Guid filmId, IEnumerable<int> rolesIds)>();

foreach (var item in filmsRoles)

{

var splittedItem = item.Split(",");

if (splittedItem.Length <= 1)

{

continue;

}

var filmId = Guid.Parse(splittedItem[0]);

var rolesIds = new List<int>();

for (int i = 1; i < splittedItem.Length; i++)

{

rolesIds.Add(int.Parse(splittedItem[i]));

}

filmRoles.Add((filmId, rolesIds));

}

var filmRolesCopy = new List<(Guid filmId, IEnumerable<int> rolesIds)>(filmRoles);

var groupedFilmsRoles = filmRolesCopy.GroupBy(x => x.filmId);

filmRoles.Clear();

foreach (var item in groupedFilmsRoles)

{

var disctinctRoles = new List<int>();

foreach (var roles in item)

{

disctinctRoles.AddRange(roles.rolesIds);

}

disctinctRoles = disctinctRoles.Distinct().ToList();

filmRoles.Add((item.Key, disctinctRoles));

}

participater.FilmParticipaterRoles = new List<FilmParticipaterRole>();

foreach (var filmRole in filmRoles)

{

foreach (var roleId in filmRole.rolesIds)

{

participater.FilmParticipaterRoles.Add(new FilmParticipaterRole()

{

FilmId = filmRole.filmId,

ParticipaterId = participater.Id,

RoleId = roleId

});

}

}

}

else

{

participater.FilmParticipaterRoles = new List<FilmParticipaterRole>();

}

}

}

}

namespace MovieOnline.Web.Areas.Admin.Controllers

{

[Area("Admin")]

[Authorize(Roles = "Admin")]

public class UploadController : Controller

{

private readonly IUnitOfWork \_uow;

private readonly IWebHostEnvironment \_webHostEnvironment;

public UploadController(IUnitOfWork uow, IWebHostEnvironment webHostEnvironment)

=> (\_uow, \_webHostEnvironment) = (uow, webHostEnvironment);

[Route("[area]/[controller]/[action]")]

public async Task<ActionResult> SaveFile(IEnumerable<IFormFile> files)

{

// The Name of the Upload component is "files"

if (files != null && files.Any())

{

var file = files.First();

var fileContent = ContentDispositionHeaderValue.Parse(file.ContentDisposition);

// Some browsers send file names with full path.

// We are only interested in the file name.

var fileName = Path.GetFileName(fileContent.FileName.ToString().Trim('"'));

//fileName = Guid.NewGuid() + "\_" + fileName;

var physicalPath = Path.Combine(\_webHostEnvironment.WebRootPath, "assets", "images", fileName);

// The files are not actually saved in this demo

using (var fileStream = new FileStream(physicalPath, FileMode.Create))

{

await file.CopyToAsync(fileStream);

}

}

// Return an empty string to signify success

return Content("");

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнительные сведения* | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР ДП 1-40 01 01 01 001 ПЗ | | | | | Пояснительная записка | | | | 140 с. | |
|  | | | | | Отзыв руководителя | | | |  | |
|  | | | | | Рецензия | | | |  | |
|  | | | | | Справка о внедрении | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
| ГУИР.751003-01 СП | | | | | Программное средство «Онлайн- | | | | Формат А1 | |
|  | | | | | кинотеатр с интеллектуальным | | | |  | |
|  | | | | | помощником». | | | |  | |
|  | | | | | Схема программы | | | |  | |
| ГУИР. 751003-01 СД | | | | | Движение данных при | | | | Формат А1 | |
|  | | | | | регистрации. | | | |  | |
|  | | | | | Схема данных | | | |  | |
| ГУИР. 751003-01 СА | | | | | Контроллер фильмов. | | | | Формат А1 | |
|  | | | | | Схема алгоритма | | | |  | |
| ГУИР. 751003-01 ПЛ | | | | | Диаграмма развёртывания | | | | Формат А1 | |
|  | | | | | программного средства. | | | |  | |
|  | | | | | Плакат | | | |  | |
| ГУИР. 751003-02 ПЛ | | | | | Диаграмма UML | | | | Формат А1 | |
|  | | | | | программного средства. | | | |  | |
|  | | | | | Плакат | | | |  | |
| ГУИР. 751003-03 ПЛ | | | | | База данных | | | | Формат А1 | |
|  | | | | | программного средства. | | | |  | |
|  | | | | | Плакат | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | БГУИР ДП 1-40 01 01 01 001 | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм | Лист | № докум. | Подп. | Дата | Программное средство «Онлайн-кинотеатр с интеллектуальным помощником» на базе платформы ASP.NET Core.  Ведомость дипломного  проекта | Литера | | | Лист | Листов |
| Разработал | | Абраменко |  |  |  | т |  | 140 | 140 |
| Проверил | | Можей |  |  | Кафедра ПОИТ  гр. 751003 | | | | |
| Т.контроль | | Можей |  |  |
| Н.контр. | | Манцевич |  |  |
| Утвердил | |  |  |  |