МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Направление специальности 1-40 01 01 «Программное обеспечение информационных технологий»(программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство для поиска фильмов и сериалов

Исполнитель

Студент(ка) 2 курса группы 4 Смолик Валерий Александрович

(Ф.И.О.)

Руководитель работы преп.-стажёр Чистякова Ю.А

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01Программное обеспечение информационных технологий  Студент: Смолик Валерий Александрович | Группа:\_\_4\_\_ |
| **Тема:** Программное средство для поиска фильмов и сериалов | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2023 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора сервиса:
  + Поддерживать работу c базой данных;
  + Добавлять новые фильмы.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Оставлять отзывы;
  + Выполнять поиск по жанрам, названию, рейтингу, году;
  + Добавление фильма в категорию: ”Просмотреть позже”

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2023 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2023 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2023 |  |
| 5 | Кодирование программного средства | 23.04.2023 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2023 |  |
| 7 | Оформление пояснительной записки | 7.05.2023 |  |
| 9 | Защита проекта | 20.05.2023 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2023**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *преп.-стажёр Чистякова Ю.А*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Смолик В.А.*

(дата и подпись студента)

Оглавление

[ВВЕДЕНИЕ 5](#_Toc135433310)

[1 Аналитический обзор литературы и постановка задачи 6](#_Toc135433311)

[1.1 Анализ прототипов 6](#_Toc135433312)

[1.1.1 Кинопоиск 6](#_Toc135433313)

[1.1.2 MEGOGO 8](#_Toc135433314)

[1.1 Постановка задачи 9](#_Toc135433315)

[2. Анализ требований к программному средству и разработка функциональных требований 10](#_Toc135433316)

[2.1 Описание средств разработки 10](#_Toc135433317)

[2.2 Описание разрабатываемой функциональности программного средства 11](#_Toc135433318)

[3 Проектирование программного средства 13](#_Toc135433319)

[3.1 Общая структура 13](#_Toc135433320)

[3.2 Диаграмма классов 15](#_Toc135433321)

[3.3 Проектирование баз данных 15](#_Toc135433322)

[3.4 Описание работы программы 18](#_Toc135433323)

[4 Создание программного средства 20](#_Toc135433324)

[4.1 Реализация работы с базой данных 20](#_Toc135433325)

[4.2 Реализация основных функций 21](#_Toc135433326)

[4.2.1 Окно регистрации 22](#_Toc135433327)

[4.2.2 Окно авторизации 23](#_Toc135433328)

[4.2.3 Основной функционал и возможности администратора 25](#_Toc135433329)

[4.2.4 Основной функционал и возможности администратора 25](#_Toc135433330)

[5 Тестирование 26](#_Toc135433331)

[5.1 Авторизация и регистрация 26](#_Toc135433332)

[5.2 Функции программного средства 27](#_Toc135433333)

[6.Руководство по использованию 28](#_Toc135433334)

[Заключение 29](#_Toc135433335)

[Список используемых источников 30](#_Toc135433336)

[ПРИЛОЖЕНИЕ А. 31](#_Toc135433337)

[ПРИЛОЖЕНИЕ Б 32](#_Toc135433338)

[ПРИЛОЖЕНИЕ В. 33](#_Toc135433339)

ВВЕДЕНИЕ

Сегодня вышло столько фильмов, как хороших, так и не очень, а новые фильмы и сериалы выпускаются с такой скоростью, что мы не только не успеваем смотреть их, но о некоторых из них даже не подозреваем. Связано это в первую очередь с развитием технологий, которые в разы упрощают съемку фильма.

Раньше фильмы выходили реже и людям приходилось идти в кинотеатр, чтобы посмотреть его, однако все изменило распространение интернета. Он позволил людям получить доступ к неограниченному количеству фильмов и сериалов, однако их нужно уметь правильно выбирать, чтобы не тратить время впустую.

Эту проблему и пытается решить данный курсовой проект.

Целью курсового проекта является приложение для поиска фильмов и сериалов – это программа, позволяющая удобно и быстро выбрать фильм или сериал по интересующему жанру, прочитать его описание, а также отзывы пользователей. Оно не является корпоративным. Данное программное средство полезно, в первую очередь, людям, которые часто смотрят фильмы, так как оно облегчит подбор фильма на вечер. Также позволит поделиться своими эмоциями после просмотра с другими пользователями.

Приложение имеет два типа пользователей: клиент и администратор.

У администратора есть возможность совершать удаление, добавление и редактирование данных в базе данных, также базовый функционал приложения, как у пользователя.

В свою очередь пользователь может просматривать каталог, искать фильмы по имени, выполнять сортировки, а так же сменять свои данные. При просмотре полной информации о фильме отображаются комментарии пользователей, а также клиент может добавлять фильм в категории “Понравившееся”, “Посмотреть позже”. Присутствует смена языка интерфейса, если пользователь не знает русский.

1 Аналитический обзор литературы и постановка задачи

**1.1** **Анализ прототипов**

В результате анализа аналогичных решений , а также целей и задач данного курсового проекта, были сформированы требования к данному программному средству.

* + 1. Кинопоиск

Одним из наиболее популярных аналогичных решений является «Кинопоиск» (имеет сайт-аналог).

Приложение «Кинопоиск» предназначено для любителей фильмов и сериалов. Имеет множество разных функций. К примеру, в данном программном средстве публикуются новости и интересные факты, связанные с киноиндустрией, также помимо того, что пользователь может узнать подробную информацию о фильме или сериале, его актерах, режиссерах, рейтинге и т.д., имеет качественный и удобный плеер.

|  |  |
| --- | --- |
| https://sun9-56.userapi.com/impg/03_aZzhruZYNqlgrNigqjmsmiHIuQWW6kcxDJA/aeh9akyB5tA.jpg?size=828x1792&quality=95&sign=198e54a825f0a5d041ef7607ccf804dc&type=album | https://sun9-21.userapi.com/impg/LtINwPAZsvkKbYvSyymwYSTyig6g2jtHQdk1zQ/wKXklzbv7tc.jpg?size=828x1792&quality=95&sign=379b7907fae12f95b103a62b1a0330ec&type=album |

Рисунок 1.1 – Окно Медиа «КиноПоиск»

Войдя в приложение и перейдя во вкладку Медиа, приложение предлагает списки фильмов-новинок, различных жанров. Пример представлен на рисунке 1.1.

Так же есть удобный поиск, который содержит не только поиск по фильмам и сериалам, странам, режиссерам, актерам и т.д., но и рекомендации, популярных персон, а также у кого из кино-звезд сегодня день рождения, что можно увидеть на рисунке 1.2.

|  |
| --- |
| https://sun9-75.userapi.com/impg/6rM0XXGWak2ZR8T23ZVaFjPdo6oWUFyucCZDEg/ixkckbaIUkE.jpg?size=828x1792&quality=95&sign=1cc3ae52432a0ecc22c01f25f3f09e37&type=album |

Рисунок 1.2 – Окно поиска приложения «КиноПоиск»

Каждая страница с фильмом содержит полную информацию о фильме, актерах и режиссерах, кассовых сборах, рейтингу и т.д. Также есть возможность добавить фильм/сериал в категорию посмотреть позже. Это можно увидеть на рисунке 1.3.

|  |  |  |
| --- | --- | --- |
| https://sun9-16.userapi.com/impg/4fF1jXg-_llUfUubV4lYV59uOqISe-3IpT5Fag/8ilPhrtRKQI.jpg?size=828x1792&quality=95&sign=5642ca5fdbb4629c346a588e62a8db5f&type=album | https://sun9-77.userapi.com/impg/8JnRepSohQG_PjYJa3HvcPGTsbjIOuB9QMbkTw/9e_b695MNfo.jpg?size=828x1792&quality=95&sign=45ac3df7095262359d762bd637b9d573&type=album | https://sun9-20.userapi.com/impg/Z-uLqoeSYibzyEYAdG_8kzYRPaZlEx1d7Gc3GA/aEq34fXUUGQ.jpg?size=828x1792&quality=95&sign=1bafa137172ae126101261824cc52f81&type=album |

Рисунок 1.3 – Страница информации о фильме приложения «КиноПоиск»

Минусов, кроме платной подписки для просмотра фильмов и сериалов, выявлено не было.

1.1.2 MEGOGO

Еще одним альтернативным решением является приложение «MEGOGO», которое может быть использовано как для подбора фильма/сериала, так и для их просмотра. При запуске приложения открывается главная страница, на которой расположены основные разделы, а также популярные фильмы. Присутствует возможность просмотра фильмов бесплатно, что будет подкупать пользователя использовать конкретно его. Главная страница представлена на рисунке 1.4.

|  |
| --- |
| https://sun9-43.userapi.com/impg/IAbQI_5Y1e82PdfX40jREALKSNaVDo1vOcSFoA/TXUnRhVTj1M.jpg?size=828x1792&quality=95&sign=d303269bbb0c6e6724792071201db73c&type=album |

Рисунок 1.4 – Главная страница «MEGOGO»

Присутствует удобный поиск с Топом просмотров MEGOGO, а также бургер-меню, которое при нажатии, позволяет перейти на страницу с фильтрами. Страница поиска и фильтров представлена на рисунке 1.5.

|  |  |
| --- | --- |
| https://sun9-42.userapi.com/impg/KCBMEaIQnGbvKEknUmoRwhIG0oLktw_A-pNZzw/aE78LsdqAtk.jpg?size=828x1792&quality=95&sign=3aaa0a01b1cf8807e6ef6c1776e45199&type=album | https://sun9-6.userapi.com/impg/A7SO35OaKUoBlO4CT_7F93HT6UOoUq_LBozjIw/fVTeGR2BL-o.jpg?size=828x1792&quality=95&sign=c4ea48ba629b9f949d31c358841dc8dd&type=album |

Рисунок 1.4 – Страница поиска и фильтров «MEGOGO»

«MEGOGO» является удобным, интуитивно-понятным приложением с хорошим UI/UX, однако на сегодняшний день большинство фильмов недоступны к просмотру, что является существенным недостатком.

* 1. Постановка задачи

После просмотра известных аналогичных решений, были выявлены преимущества и недостатки , на основе которых можно сформировать список требований , которые будут впоследствии предъявлены к приложению, разрабатываемому в данном курсовом проекте. Можно выделить следующие функции, которые должны выполняться при помощи программного средства:

* Управление администратором базой данных, а именно добавление, редактирование, удаление фильмов/сериалов;
* У пользователя должна быть возможность зарегистрироваться или войти в существующую учетную запись;
* Пользователь должен иметь возможность изменить существующую запись;
* Пароли пользователей обязательно должны быть зашифрованы;
* Возможность авторизированным пользователям изучать информацию о фильмах, выполнять сортировки и поиск;
* Возможность смены языка;
* Возможность добавления/удаления фильма в/из категорию “Понравившееся”;
* Возможность добавления/удаления фильма в/из категорию “Посмотреть позже”.

Таким образом, требования после изучения аналогов были расширены, что еще раз подтверждает важность данного этапа.

2. Анализ требований к программному средству и разработка функциональных требований

Перед непосредственной разработкой приложения обязательно необходимо сформулировать функциональные требования, для того чтобы установить каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований.

2.1 Описание средств разработки

При разработке проекта были использованы три паттерна: MVVM, Repository, Unit of Work, а также технологии, которые описаны ниже.

Microsoft SQL Server – система управления реляционными БД. SQL Server Management Studio – утилита из Microsoft SQL Server, необходимая для управления и администрирования всех компонентов Microsoft SQL Server.

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения, предоставляющий среду для разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности.

Технология WPF (Windows Presentation Foundation) является частью экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов. Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если при использовании ADO.NET необходимо создавать подключения, команды , транзакции , то в Entity Framework представлен более высокий уровень абстракции , который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Font Awesome — это шрифт с иконками, которые могут добавляться к любым элементам веб-страницы, чтобы повысить их наглядность и улучшить дизайн. Разработчик Дэйв Ганди собрал впечатляющую коллекцию из несколько сотен иконок подходящих под любые задачи.

Преимущества:

* Все иконки векторные и масштабируются в самых широких пределах, без ухудшения их качества;
* Иконки хорошо работают во всех браузерах, включая старый IE8;
* Иконки можно поворачивать, анимировать и комбинировать между собой, накладывая одну иконку поверх другой. Это ещё больше расширяет возможное количество вариантов.

Недостатки:

* Символы могут быть только одного цвета. Это ограничение частично обходится объединением иконок разного цвета друг с другом, а также с помощью эффектов CSS;
* Избыточность.

Extended WPF Toolkit™ от Xceed — это коллекция элементов управления, компонентов и утилит WPF для создания приложений Windows.

2.2 Описание разрабатываемой функциональности программного средства

Требования — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению (ПО), в результате анализа требований.

Цели разработки требований:

* обеспечение наиболее полного и точного отражения условий или возможностей, необходимых заказчику для решения его проблем и достижения бизнес-целей;
* снижение затрат на разработку, обслуживание и поддержку программного обеспечения;
* минимизация рисков переделки продукта, создания негодного ПО или срыва сроков сдачи проекта.

Классификация требований:

* Функциональные (Описание требуемого поведения системы в определенных условиях);
* Нефункциональные (Описание свойства или особенности, которым должна обладать система, или ограничение, которое должна соблюдать система);
* Бизнес-требование.

Приложение не является коммерческим, поэтому бизнес требования не могут быть сформированы.

**Функциональные**:

* Регистрация пользователя;
* Поиск по фильтрам;
* Возможность добавления/удалить фильма в категорию посмотреть позже;
* Возможность для администратора добавления, изменения, удаления фильма.

**Нефункциональные**

* При регистрации пароль должен соответствовать требованиям: не менее 3 символов;
* Приложение не должно загружаться дольше чем 9 секунды;
* Должен обрабатывать большое количество запросов пользователей без снижения производительности;
* Локализация.

Пользовательские требования — это требования, которые пользователи желают видеть в программном продукте.

Пользовательские требования:

1. Выполнять регистрацию и авторизацию;
2. Выполнять поиск фильма по названию;
3. Иметь возможность оставить комментарий;
4. Иметь возможность сортировки по жанрам, имени, году выхода;
5. Иметь возможность добавить фильм в категорию “Понравившееся”;
6. Иметь возможность добавить фильм в категорию “Посмотреть позже”;
7. Иметь возможность редактирования профиля.

Администратор имеет возможность:

1. Выполнять CRUD операции с БД;
2. Выполнять редактирования профиля.

На рисунке 2.1 представлена UML-диаграмма возможностей, которая отражает функциональность программного средства с точки зрения взаимодействия администратора с приложением( копия в Приложении Б)

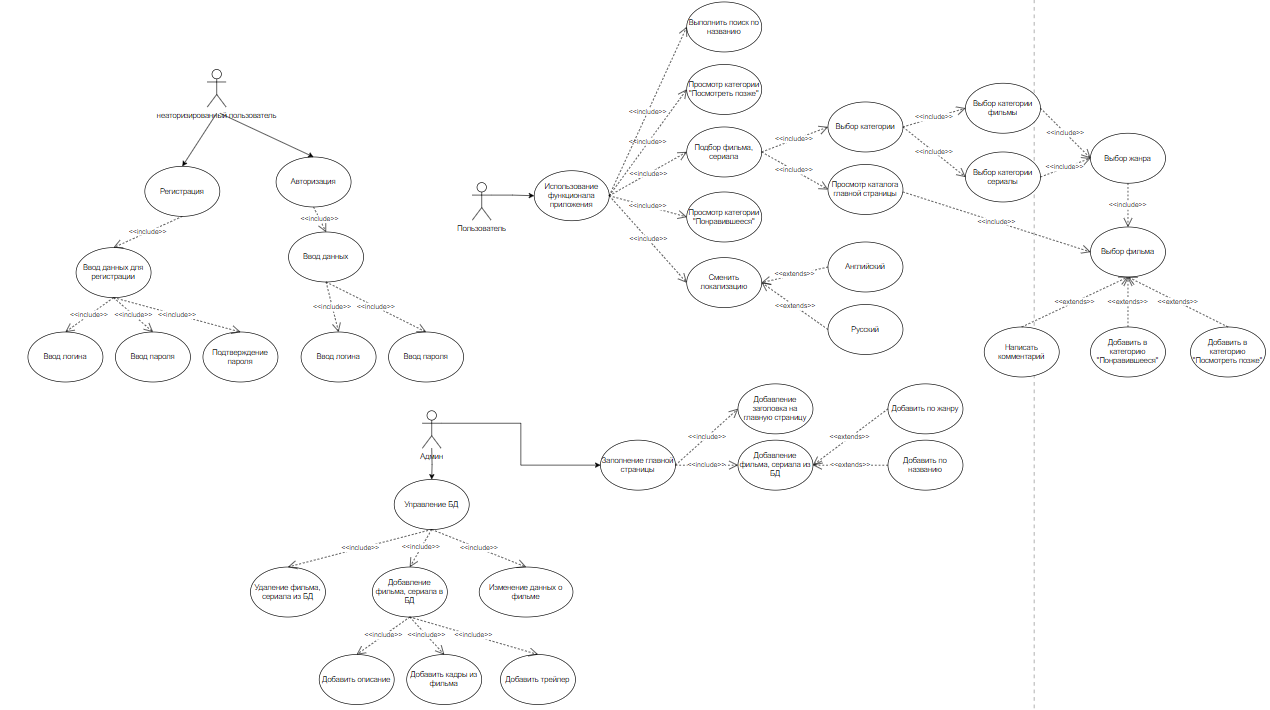


Рисунок 2.1 – UML-диаграмма

Из выше сказанного следует, что разрабатываемый продукт должен иметь удобный интерфейс для пользователя и администратора, и выполнять CRUD.

3 Проектирование программного средства

3.1 Общая структура

Приложение “GoKin” имеет структуру, представленную на рисунке 3.1.

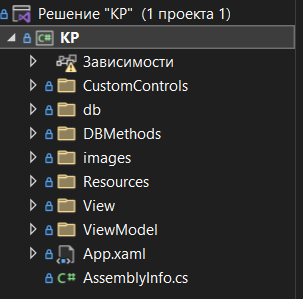


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.2.

Таблица 3.2 – Описание структуры папок и файлов проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Имя папки | | Содержание | |
| CustomControls | | Содержит внутри себя Usercontrol, для переопределения поведения PasswordBox-a | |
| db | | Здесь расположены модели, с которыми и происходит вся работа приложения, а также контекст для подключения к БД. | |
| DBMethods | Содержит внутри себя классы репозиториев, которые реализуют паттерн Repository, для работы с БД; клас хэширования; класс unitOfWork, класс, ответственный за все репозитории, реализует паттерн Unit of Work. | |
| images | Содержит внутри себя логотип, а также другие изображения. | |

Продолжение таблицы 3.2 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя папки | Содержание |
| Resources | Содержит внутри себя ResourceDictionaries, которые отвечают за стили приложения |
| View | Содержит все представления, которые отвечают за UI. |
| ViewModel | Содержит логику взаимодействия View⬄ViewMode⬄Model |
| App.xaml | Файл, который определяет ресурсы, которые являются общие для приложения |

Данная таблица помогает разобраться в общей структуре проектируемого приложения.

Структура содержимого папок “db” и “DBMethods” представлена на рисунке 3.3.

|  |
| --- |
|  |

Рисунок 3.3 – Структура папок “db” и “DBMethods”

В папке “db” имеется две вложенные папки “context”, который содержит строку подключения, и DbSet’s моделей данных из dbClasses.

В папке “dbClasses” содержаться модели данных:

* UserProfile.cs – модель с данными пользователя;
* Likes.cs – модель с понравившимися пользователю фильмами;
* WatchLater.cs – модель с фильмами, помещенные в категорию “Посмотреть позже”;
* Review.cs – модель, содержащая отзывы конкретного пользователя;
* FramesFromMovie.cs – модель, содержащая кадры из фильма;
* MiniItemInfo.cs – модель, содержащая краткую информацию о фильме;
* BigItemInfo.cs – модель, содержащая полную информацию о фильме.

В папке “DBMethods” содержаться две вложенные папки , одна из которых содержит класс HashMD5.cs. Данный класс используется для шифрации пароля, используя методы класса MD5.

Папка “Repositories” содержит классы с реализацией паттерна repository, а также интерфейса IRepository.cs, для моделей данных.

3.2 Диаграмма классов

Диаграмма классов – это UML-диаграмма, которая описывает систему, визуализируя различные типы объектов внутри системы и виды статических связей, которые существуют между ними. Он также иллюстрирует операции и атрибуты классов.

Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями и состоит из имени класса, атрибутов (свойств) класса, методы класса. Диаграмма классов представлена в приложении А

3.3 Проектирование баз данных

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе.

При разработке приложения была разработана и протестирована база данных “GoKin”. Для ее создания использовалось СУБД Microsoft SQL Server. Для взаимодействия базы данных с программным средством был использован Entity Framework.

В данный момент БД состоит из 7 таблиц, которые связаны между собой связями один ко многим или один к одному.

Поля таблицы framesFromMovie:

* ID – primary key, содержит уникальный индекс кадра;
* Frame – непосредственно сам кадр;
* BigItemInfoID – foreigh key, отвечающий за связь с таблицей bigItemInfos.

Поля таблицы bigItemInfos:

* ID – primary key, содержит уникальный индекс полной характеристики;
* Title – название фильма;
* TitleOrig – название на оригинальном языке;
* BigImg – Постер фильма;
* Year – дата премьеры в мире;
* Genre – жанр фильма;
* Country – Страны;
* Description – описание фильма;
* ratingIMDb – рейтинг IMDb;
* ratingKP – рейтинг Кинопоиск;
* fees – кассовые сборы в мире;
* directors – режиссер ;
* actors – актеры.

Поля таблицы userLikes:

* ID – primary key , уникальный ID лайка;
* userID- foreign key, ID пользователя , который лайкнул;
* bigItemInfoID – foreign key, ID понравившегося фильма(полная информация);
* miniItemInfoID – foreign key, ID понравившегося фильма(краткая информация).

Поля таблицы miniItemInfos

* ID – primary key, содержит уникальный индекс краткой характеристики;
* Name – имя фильма;
* Genre – жанр фильма;
* Year – дата премьеры в мире;
* SmallImg – Постер фильма;
* BigItemInfoID – foreighn key, который содержит ID полной информации.

Поля таблицы UserWatchLater:

* ID – primary key, содержит уникальный индекс фильма в категории посмотреть позже;
* userID- foreign key, ID пользователя , который добавил в категорию “посмотреть позже”;
* bigItemInfoID – foreign key, ID “Отложенного” фильма(полная информация);
* miniItemInfoID – foreign key, ID “Отложенного”фильма(краткая информация).

Поля таблицы UserProfiles:

* ID – primary key, содержит уникальный индекс пользователя;
* Login – логин пользователя;
* Email – email пользователя;
* Avatar – аватарка пользователя;
* Password – зашифрованный пароль.

Поля таблицы reviews:

* ID – primary key, содержит уникальный комментария пользователя;
* Avatar – аватарка пользователя;
* Login – логин пользователя;
* UserReviewTest – текст комментария;
* Date – дата оставления комментария;
* bigItemInfoID – foreign key, ID фильма, на который оставлен отзыв (полная информация);
* miniItemInfoID – foreign key, ID фильма, на который оставлен отзыв (краткая информация).

Связи между таблицами можно посмотреть на рисунке 3.4.

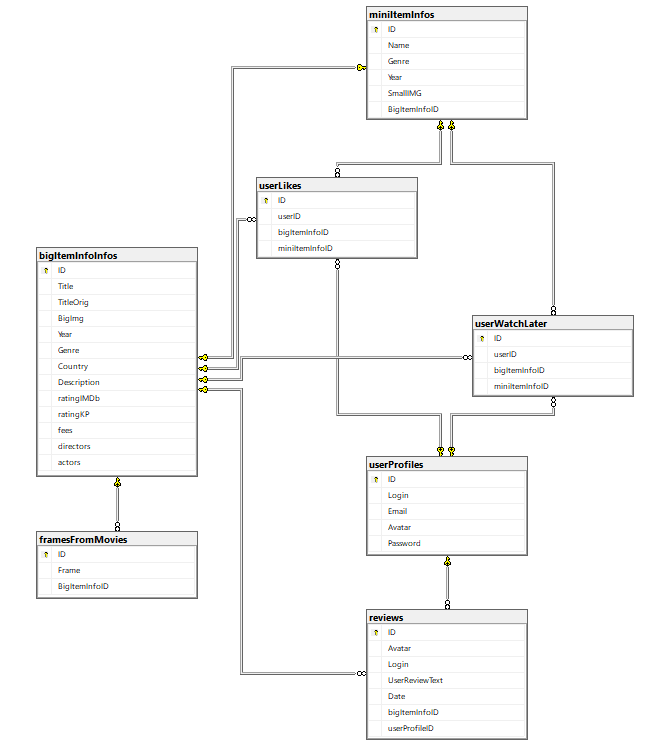


Рисунок 3.4 – Связи между таблицами в БД

Следует уделять время на проектирование отношений внутри базы данных, потому что при неправильном подходе будет сложно извлекать данные.

3.4 Описание работы программы

Первое, что увидит пользователь при запуске приложения – окно регистрации. Если у пользователя есть аккаунт , то он переходит в окно авторизации, а если нет то создает новый аккаунт и после этого попадает на страницу авторизации, после чего открывается основной окно. На диаграмме рисунок 3.5 описывается процесс авторизации пользователя.

|  |
| --- |
|  |

Рисунок 3.5 – Диаграмма последовательности авторизации

Схему взаимодействия окон в программном средстве можно посмотреть на рисунке 3.6. Структура окон “Gokin” позволяет пользователю быстро и удобно перемещаться по приложению, это связано с тем , что слева всегда открыто навигационное меню.

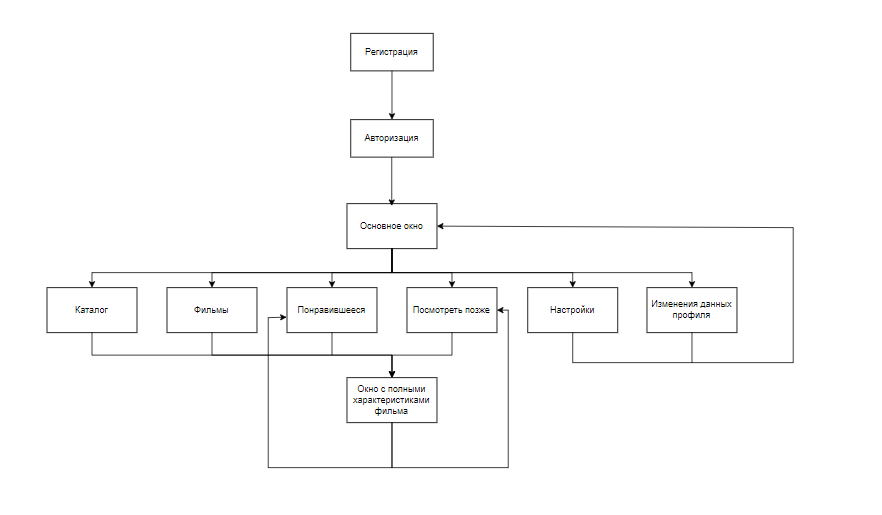


Рисунок 3.6 – Диаграмма последовательности авторизации

При каждом запуске программного средства пользователь должен выбрать необходимое действие , а именно регистрация или авторизация и в зависимости от ответа либо вести данные и авторизоваться, либо зарегистрироваться заново. Алгоритм работы авторизации отображен на блок-схеме(рисунок 3.6).

|  |
| --- |
|  |

Рисунок 3.7 – Блок схема работы авторизации и регистрации

После успешной авторизации будет открыто главное меню c аватаркой и логином пользователя.

4 Создание программного средства

4.1 Реализация работы с базой данных

В приложении “GoKin” используются паттерн Repository , который позволяет инкапсулировать логику работы с БД. Т.К. имеется большое количество моделей и соответственно большое количество репозиториев, было принято решение использовать паттерн Unit of Work, т.к. он поможет упростить работу с различными репозиториями и даст уверенность в том, что все репозитории будут использовать один и тот же DbContext.

Реализация паттерна представлена в классе “UnitOfWork” (реализация представлена в Приложении В). При открытие приложения создается объект, который позволяет взаимодействовать со всеми репозиториями.

При нажатии на кнопку добавлении фильма или сериала в базу данных, срабатывает привязанная команда, которая обращается к конкретному репозиторию и выполняет функцию Add(). После функции добавления обязательно должна быть вызвана функция Save() для того чтобы сохранить все изменения из локальной базы данных в глобальную базу данных.

Для того чтобы реализовать паттерн репозиторий, в первую очередь необходимо создать обобщенный интерфейс, который и будет содержать методы, реализацию которым даст репозиторий.

В интерфейсе объявлены следующие методы:

* Add(T user) – добавление объекта в репозиторий;
* Edit(T user) – изменение объета;
* Remove(T user) – удаление объекта;
* T GetById(int id) – получение объекта по id;
* T GetByLogin(string login) – получение объекта по логину;
* IEnumerable<T> GetAll() – получить все данные из таблицы;
* Save() – сохранить изменения;

Реализация интерфейса представлена на рисунке 4.1.

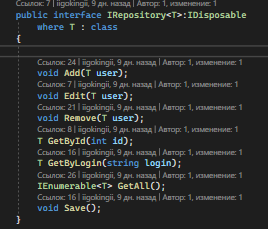


Рисунок 4.1 – Реализация интерфейса

Примером репозитория, который работает с сущностями типа UserProfile служит “UserProfileRepository”. Листинг класса в приложении В.

4.2 Реализация основных функций

Пользователь может воздействовать на программное средство путем использования комманд. Command можно использовать как к button , так и к radiobutton и т.д.

Команда определяется в ViewModel и затем через механизм Binding привязывается непосредственно к элементу управления.

Реализация обработки выполнена в классе “ViewModelCommandBase” , можно посмотреть на рисунке 4.2.

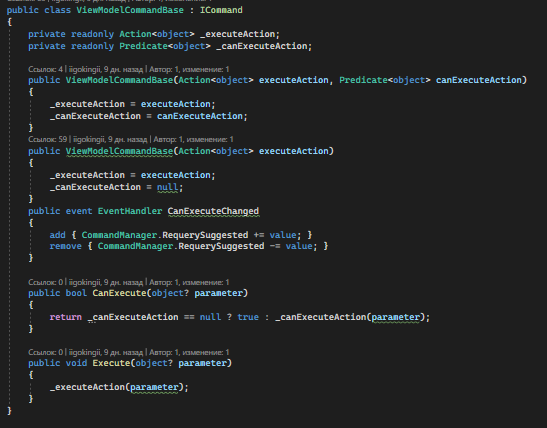


Рисунок 4.2 – Реализация ViewModelCommandBase

Конструктор ViewModelCommand может принимать два параметра: executeAction — это метод, который будет выполнен при вызове команды, и canExecuteAction - вызывается при каждой попытке выполнить команду и возвращает булевое значение, которое указывает, можно ли выполнить команду в данный момент. Если метод canExecuteAction не передан в конструкторе, то метод CanExecute всегда возвращает true.

Метод Execute вызывается при выполнении команды и вызывает метод executeAction, который был передан в конструкторе. Событие CanExecuteChanged вызывается, когда состояние команды изменяется (например, при изменении свойства, которое влияет на возможность выполнения команды). Это событие позволяет обновлять состояние элементов пользовательского интерфейса, связанных с командой, когда состояние команды изменяется.

Для уведомления всех команд о необходимости переопределить состояние выполнения используется статический класс CommandManager.

Рассмотрим использование команд на примере регистрации пользователя. Метод ExecuteLoginCommand команды RegistrationCommand будет запущена только тогда , когда метод CanExecuteLoginCommand вернет true. Пример реализации представлена на рисунке 4.3.

|  |
| --- |
|  |

Рисунок 4.3 – Реализация ExecuteLoginCommand

Для того чтобы вызвать команду, необходимо ее привязать к элементу управления . Пример привязки представлен на рисунке 4.4.

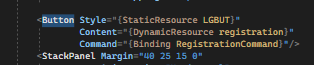


Рисунок 4.4 – Привязка ExecuteLoginCommand

Привязка дает возможность связывать View и ViewModel, путем использования ключевого слова Binding.

4.2.1 Окно регистрации

При запуске приложения пользователь попадает на страницу регистрации из которой , если у него есть аккаунт, он может сразу перейти на окно авторизации. При нажатии на кнопку регистрация , введенные пользователем данные проходят проверку на валидность и если данные не валидны то выводится сообщение, указывающее какое из полей было введено неверно. RegistrationView и RegistrationViewModel взаимодействуют через механизм связывания. Если данные пользователя прошли валидацию , то его вводы заносятся в бд, за тем лишь исключением , что пароль шифруется и ставится аватарка по умолчанию, которую , при желании можно сменить в пользовательских настройках. Пример привязки textbox и свойств во ViewModel представлен на рисунке 4.5

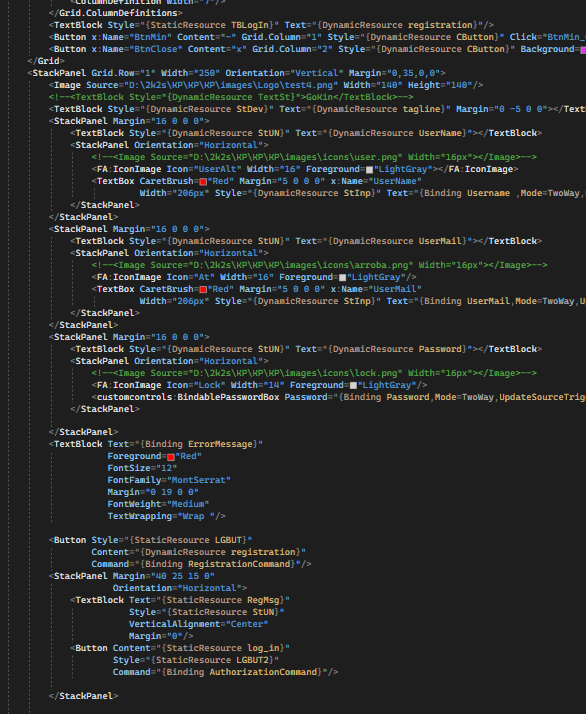


Рисунок 4.5 – Пример привязки textbox и свойств во ViewModel

Во время регистрации пользователь должен ввести логин, корректн.ый email, а также пароль.

4.2.2 Окно авторизации

При нажатии на выделенную кнопку авторизация или зарегистрировавшись в окне регистрация, пользователь попадает на окно авторизации. Здесь , так же как и в окне регистрации свойство и элементы управления связаны через binding. Что позволяет при нажатии на кнопку, к которой привязана команда , проверить корректность введенных данных и если данные не верны то вывести соответствующее сообщение.

При проверке введенного пароля и пароля из бд , введенный пароль хэшируется.

При совпадении паролей и логинов пользователь попадает на главную страницу из которой у него открывается доступ к функционалу приложения. Из формы авторизации данные передаются на главную страницу и отображаются в верхнем правом углу Возможные выборы пользователя из главноей страницы представлены на рисунке 4.6.

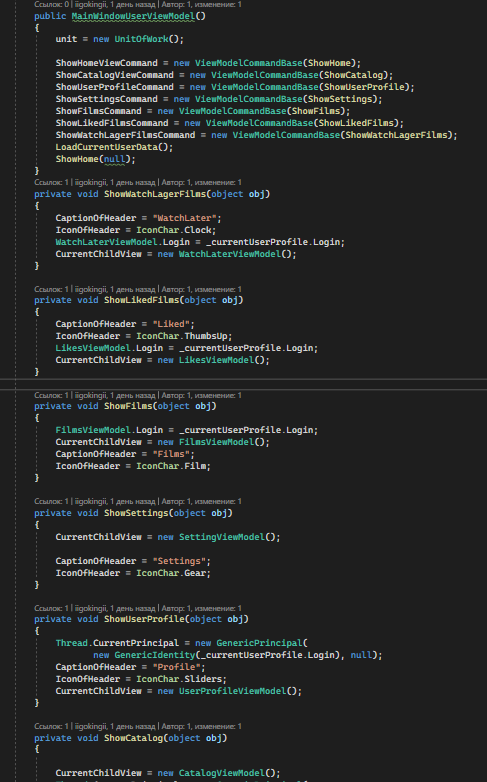


Рисунок 4.6 – Возможные выборы пользователя

При переходе в настройки , есть возможность сменить локализацию приложения, а при переходе в профиль можно изменить все ранее введенные данные. При нажатии на фильм из каталога будет открыта страница с полной информацией о фильме и если он понравился , то его можно добавить в категорюю “понравившееся”, а если сейчас на просмотр нет времени , то в категорию “посмотреть позже”. На рисунке 4.7 отображены методы добавления в категории “посмотреть позже” и “понравившееся”.

|  |
| --- |
|  |

Рисунок 4.7 – методы добавления в категории

В данных методах, как и везде по приложению используется паттерн Unit of work.

4.2.3 Основной функционал и возможности администратора

На страницу пользователя можно попасть путем ввода своего логина и пароля, ну или создать новый аккаунт. Пользователь может выбирать фильмы из разных окон , в которых фильмы отсортированы по различным критериям.

У пользователя есть так же возможность добавить фильмы в категории “посмотреть позже” и понравившееся.

У всех пользователей есть возможность оставить комментарий, который будет виден всем остальным.

4.2.4 Основной функционал и возможности администратора

На страницу администратора можно попасть путем ввода логина и пароля администратора. Основным функционал администратора, в отличии от пользователя является то, что он может выполнять добавление, удаление, редактирование данных.

5 Тестирование

5.1 Авторизация и регистрация

При запуске программного средства открывается окно регистрации и после него окно авторизации. При введении некорректных данных кнопка регистрации не будет активной, связано это с тем что при создании команды был использован метод CanExecuteLoginCommand который проверяет на валидность данные и если они оказываются некорректными блокирует выполнение команды .Рисунок 5.1.

|  |
| --- |
|  |

Рисунок 5.1 – Обработка данных в окне регистрация

Так же присутствует проверка на уникальность логина при регистрации и если логин уже есть в базе данных, выведется соответствующая ошибка.

При некорректном вводе данных в окне авторизации будет выведен текст в чем проблема. Рисунок 5.2.

|  |
| --- |
|  |

Рисунок 5.2 – Обработка данных в окне авторизации

Формы регистрации и авторизации работают корректно.

5.2 Функции программного средства

После авторизации, пользователь попадает на главную страницу, где он может выбрать заинтересовавший фильм и написать на нем комментарий. Так же пользователь может изменить данные своего профиля, но без введенного корректно пароля ничего изменить не получится. Пример неправильного изменения данных продемонстрирован на рисунке 5.3.

|  |
| --- |
|  |

Рисунок 5.3 – Неправильное изменение данных

На стороне администратора при добавлении новых фильмов присутствуют валидации данных, один из которых продемонстрирован на рисунке 5.4

|  |
| --- |
|  |

Рисунок 5.4 – Обработка данных в окне авторизации

Подобные проверки стоят и на других полях.

6.Руководство по использованию

Для того чтобы запустить приложение помимо установленной visual studio, должен быть установлен Microsoft SQL Server для работы с БД.

Прежде всего необходимо в классе DbAppContext изменить строку подкючения. Все остальное Entity Framework сделает за вас.

Запустить приложение, при регистрации вводить пароль длинее 3 символов, а Email что-то похожее на [vlks@mail.ru](mailto:vlks@mail.ru), далее вводите пароль и вас перенаправит на другую вкладку с авторизацией, где вы введете пароль и логин который только что вводили.

Перейдя на главную страницу вы можете изменить данные аккаунта , старый пароль является обязательным , остальные поля по усмотрению.

С помощью левого навигационного меню можно выбрать интересующий раздел.

При выборе страницы “Каталог” вы увидите все фильмы , которые есть на данный момент в базе данных, здесь же можно отфильтровать их имени, жанру, году. Если выбрали интересующий фильм нажмите на него, откроется полная информация о фильме, и появиться возможность добавить/убрать фильм из понравившегося или категории “посмотреть позже”.

Отсюда можно как вернуться обратно на страницу с каталогом , так и перейти на другие страницы. Например можно изменить локализацию в настройках или перейти во вкладку фильмы и поскролить по жанрам.

Так же есть возможность перейти во вкладку “Посмотреть позже” и вернуться к тому, на что когда-то времени не хватило.

С любой из страниц, на которых расположены фильмы, можно перейти к полному описанию.

Заключение

В ходе выполнения курсового проекта было создано программное средство для поиска фильмов и сериалов. Результатом является разработка простого , но в то же время удобного интерфейса, который может быть расширен , путем добавления фильмов и сериалов, введения рекомендаций, расширения основной базы данных. В ходе работы были проанализированы аналогичные решения конкурентов, были выявлены достоинства и недостатки. В самом конце работы проводилось тестирование программного средства.

В ходе разработки программного средства было выполнено следующее:

* Была создана база данных с применением подхода Code First, для хранения, добавления и редактирования фильмов;
* Была разработана архитектура ПС;
* Был разработан интерфейс программного средства;
* Была разработана функциональная часть;
* Проведено тестирование.
* Написано руководство пользователя

В соответствии с результатом выполненной работы можно сделать вывод, что приложение работает корректно, а также весь необходимый функционал был разработан.

Список используемых источников

1. Professor WEB [Электронный ресурс]. – ЭУ. – Режим доступа: <https://professorweb.ru/my/WPF/UI_WPF/level6/UI_WPF_index.php>. Дата доступа: 16.05.2023

2. WPF tutorial [Электронный ресурс] /Режим доступа: <https://www.wpf-tutorial.com/>. Дата доступа: 16.05.2023

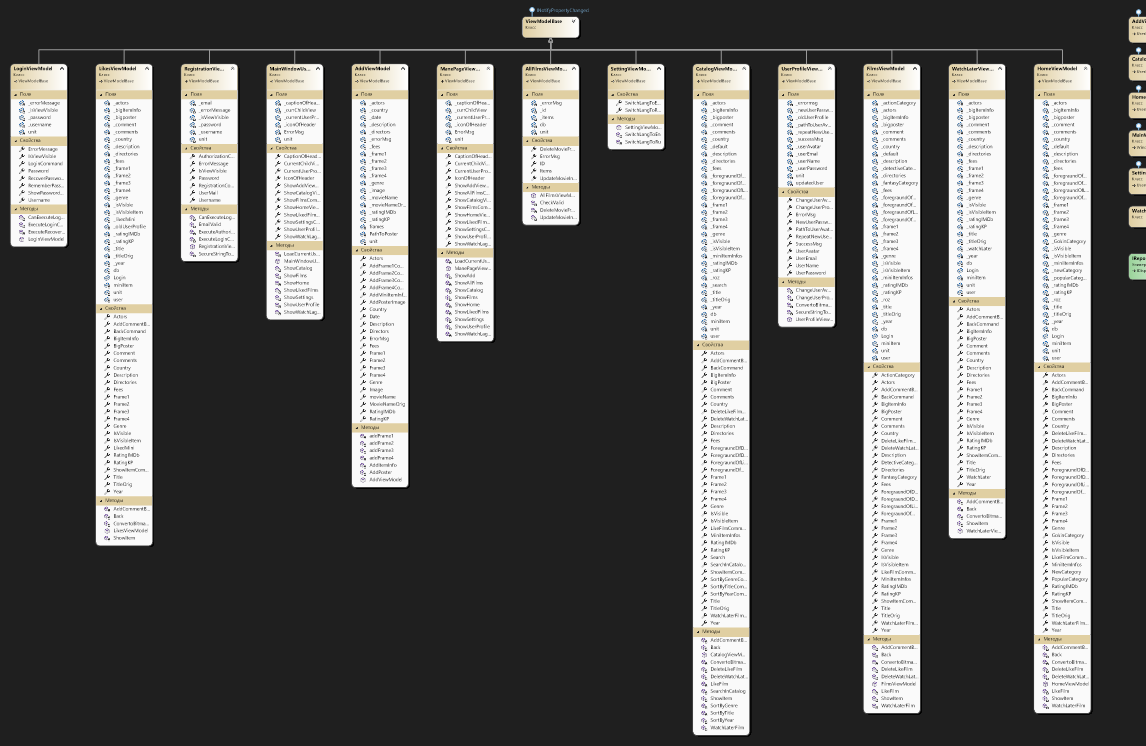
3. Metanit [Электронный ресурс]. – Руководство по WPF. – Режим доступа: <https://metanit.com/sharp/wpf/>. Дата доступа: 16.05.2023

4. Professor WEB [Электронный ресурс]. – Подстановка шрифтов. –Режим доступа: <https://professorweb.ru/my/WPF/UI_WPF/level6/6_4.php>. Дата доступа: 16.05.2023

5. Г. Буч, Д. Рамбо, И. Якобсон. Язык UML. Руководство пользователя. — 2-е. — М. : ДМК Пресс, 2006. — 496 с.

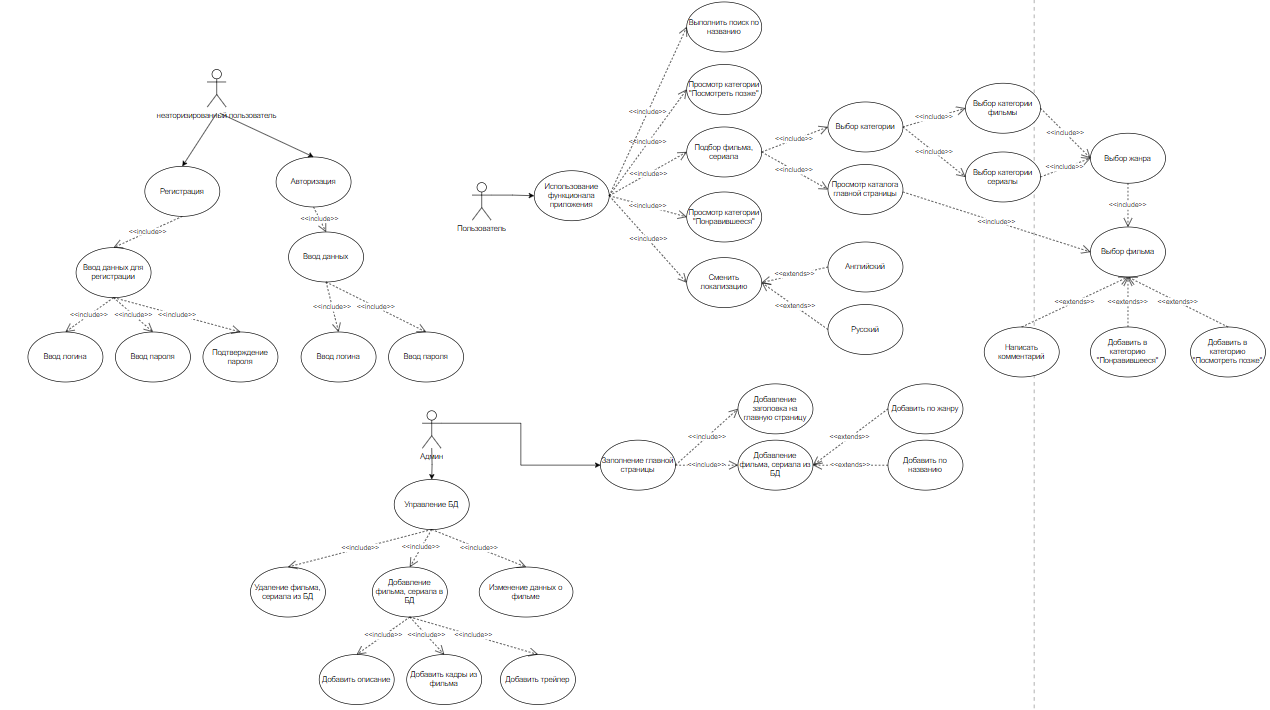
ПРИЛОЖЕНИЕ А.

Диаграмма классов ViewModel



ПРИЛОЖЕНИЕ Б

UML-диаграмма возможностей



ПРИЛОЖЕНИЕ В.

Листинг – класса UnitOfWork

|  |
| --- |
| using KP.db.context;  using KP.DBMethods.Repositories;  using KP.DBMethods.Repositories.UserProfileRepositor;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace KP.DBMethods.UnitOfWork  {  internal class UnitOfWork : IDisposable  {  DbAppContext db = new DbAppContext();  private UserProfileRepository \_userProfileReposit;  private MiniItemInfoRepository \_miniItemInfoRepository;  private FramesFromMovieRepository \_framesFromMovieRepository;  private BigItemInfoRepository \_bigItemInfoRepository;  private ReviewRepository \_reviewRepository;  private LikesRepository \_likesRepository;  private WatchLaterRepository \_watchLaterRepository;  public WatchLaterRepository WatchLaterRepository  {  get  {  if (\_watchLaterRepository == null)  \_watchLaterRepository = new WatchLaterRepository();  return \_watchLaterRepository;  }  }  public LikesRepository LikesRepository  {  get  {  if (\_likesRepository == null)  \_likesRepository = new LikesRepository();  return \_likesRepository;  }  }  public ReviewRepository ReviewRepository  {  get  {  if (\_reviewRepository == null)  \_reviewRepository = new ReviewRepository();  return \_reviewRepository;  }  }  public BigItemInfoRepository BigItemInfoRepository  {  get  {  if (\_bigItemInfoRepository == null)  \_bigItemInfoRepository = new BigItemInfoRepository();  return \_bigItemInfoRepository;  }  }  public UserProfileRepository Users  {  get  {  if(\_userProfileReposit == null)  \_userProfileReposit = new UserProfileRepository();  return \_userProfileReposit;  }  }  public MiniItemInfoRepository MiniItemInfoRepository  {  get  {  if (\_miniItemInfoRepository == null)  \_miniItemInfoRepository = new MiniItemInfoRepository();  return \_miniItemInfoRepository;  }  }  public FramesFromMovieRepository FramesFromMovieRepository  {  get  {  if (\_framesFromMovieRepository == null)  \_framesFromMovieRepository = new FramesFromMovieRepository();  return \_framesFromMovieRepository;  }  }  public void Save()  {  FramesFromMovieRepository.Save();  WatchLaterRepository.Save();  LikesRepository.Save();  MiniItemInfoRepository.Save();  ReviewRepository.Save();  BigItemInfoRepository.Save();  Users.Save();  }  private bool disponsed = false;  public virtual void Disponse(bool disposing)  {  if (!this.disponsed)  {  if (disposing)  {  db.Dispose();  }  }  this.disponsed = true;  }  public void Dispose()  {  Disponse(true);  GC.SuppressFinalize(this);  }  }  } |

Листинг – класс UserProfileRepository

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net;  using System.Text;  using System.Threading.Tasks;  using KP.db.context;  using KP.dbClasses;  using KP.DBMethods.HashPasswordMD5;  using KP.View.login;  namespace KP.DBMethods.Repositories.UserProfileRepositor  {  internal class UserProfileRepository : IRepository<UserProfile>  {  private DbAppContext db;  public UserProfileRepository()  {  this.db = new DbAppContext();  }  public void Add(UserProfile user)  {  db.userProfiles.Add(user);  }  public bool AuthenticateUser(NetworkCredential credential)  {  bool validUser;  UserProfile? us = db.userProfiles.FirstOrDefault(p => p.Login == credential.UserName && p.Password == HashMD5.HashPasswordWithMD5(credential.Password));  validUser = us==null? false : true;  return validUser;  }  //Todo  public void Edit(UserProfile \_user)  {  UserProfile user= db.userProfiles.FirstOrDefault(p => p.Login == \_user.Login && p.Email == \_user.Email && p.ID == \_user.ID);  if (user != null)  {  user.reviews = \_user.reviews;  user.Email = \_user.Email;  user.Avatar = \_user.Avatar;  user.ID = \_user.ID;  user.Login = \_user.Login;  }  }  public IEnumerable<UserProfile> GetAll()  {  return db.userProfiles.ToList();  }  /////////////TODO  public UserProfile GetById(int id)  {  return db.userProfiles.FirstOrDefault(p => p.ID.Equals(id));  }  public UserProfile GetByLogin(string login)  {  return db.userProfiles.FirstOrDefault(p => p.Login == login);  }  //////////////////    public void Remove(UserProfile user)  {  try  {  db.userProfiles.Remove(user);  }  catch (Exception e) { }  }  public void Save()  {  try  {  this.db.SaveChanges();  }  catch(Exception ex) { }    }  private bool disponsed = false;  public virtual void Disponse(bool disposing)  {  if (!this.disponsed)  {  if (disposing)  {  db.Dispose();  }  }  this.disponsed = true;  }  public void Dispose()  {  Disponse(true);  GC.SuppressFinalize(this);  }  }  } |