|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и зашиты баз данных. |
|  |
| на тему: Разработка информационной системы для любителя кино. |
|  |
| студентки группы 090203-9o-19/1  специальности 09.02.03 Программирование в компьютерных системах |
| Максимовой Любови Юрьевны |

|  |  |  |
| --- | --- | --- |
| Студентка |  | Л.Ю. Максимова |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 1 |  | И.А. Миланова |

Москва

2022

**СОДЕРЖАНИЕ**

[**Введение** 3](#_gjdgxs)

[**1 Общая часть** 7](#_30j0zll)

[1.1 Назначение и цели создания системы 7](#_1fob9te)

[1.2 Обзор и анализ предметной области 7](#_3znysh7)

[1.3 Жизненный цикл базы данных 8](#_2et92p0)

[1.4 Выбор и характеристика СУБД 8](#_tyjcwt)

[1.5 Выбор и характеристика среды разработки приложения 9](#_3dy6vkm)

[**2 Специальная часть** 10](#_26in1rg)

[2.1 Постановка задачи 10](#_lnxbz9)

[2.2 Архитектура информационной системы 10](#_35nkun2)

[2.3 Логическая модель базы данных 11](#_1ksv4uv)

[2.4 Нормализация таблиц 11](#_44sinio)

[2.5 Описание таблиц 11](#_2jxsxqh)

[2.6 Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи) 13](#_z337ya)

[2.7 Разработка приложения 14](#_3j2qqm3)

[2.7.1 Диаграмма вариантов использования 14](#_1y810tw)

[2.7.2 Решение главной задачи проекта 14](#_2xcytpi)

[2.7.3 Тестирование приложения 20](#_3whwml4)

[2.7.4 Защита информационной системы 21](#_2bn6wsx)

[2.8 Инструкция пользователю 21](#_qsh70q)

[2.8.1 Общие сведения об информационной системе 22](#_3as4poj)

[2.8.2 Требования к техническим средствам 23](#_49x2ik5)

[2.8.3 Требования к программным средствам 23](#_2p2csry)

[2.8.4 Настройка информационной системы 23](#_147n2zr)

[**Заключение** 24](#_3o7alnk)

[**список использованной литературы** 26](#_23ckvvd)

[**приложения** 27](#_ihv636)

[Приложение А 27](#_32hioqz)

[Приложение Б 35](#_1hmsyys)

# Введение

Данный курсовой проект посвящен разработке информационной системы для любителя кино.

Каждый год появляется множество кинокартин от разных киностудий. Уровень заинтересованности к кино растет все больше, т.к. появляются новые технологии создания и съемки. Такие технологии могут перенести в мир, который невозможен в реальности. Именно это, вместе с нескучным сюжетом, захватывает при просмотре фильма.

Для хранения информации о фильмах и оценках пользователей создается база данных.

Основная цель курсовой работы – создание приложения для поиска фильма и ознакомления с ним.

Задачами курсовой работы являются:

* Анализ предметной области,
* Создание логической модели базы данных с помощью ER Assistant и создание базы данных с помощью MS SQL Server,
* Разработка приложения с помощью среды разработки Visual Studio 2022,
* Защита базы данных и приложения.

Актуальность информационной системы любителя кино заключается в необходимости такого сервиса, который предоставлял бы актуальную информацию о фильмах и новинках. Существующие аналоги имеют разного рода проблемы и, следовательно, становятся менее значимыми в обществе. Некоторые из этих проблем относятся к дизайну, удобству использован, а также количеству рекламы.

Объектом исследования является процесс просмотра информации об интересующем фильме.

Предметом исследования является возможность оценить фильм по пятибалльной шкале.

# 1 Общая часть

## 1.1 Назначение и цели создания системы

Цель создания информационной системы в том, чтобы пользователь сервиса мог искать интересующий его фильм через удобный поиск.

Назначение информационной системы состоит в просмотре актуальной информации о фильме и возможности его оценить.

## 1.2 Обзор и анализ предметной области

В силу специфичности разрабатываемой ИС, организация как таковая отсутствует и в ней нет необходимости, поэтому данную ИС необходимо рассматривать в лице тех, кто ею пользуется. Это администраторы и пользователи. Необходимо так же описать процесс выбора фильма в разрабатываемом сервисе.

В такое случае, можно сделать вывод, что деятельность разрабатываемой ИС заключается в предоставлении услуги просмотра актуальной информации об интересующем фильме, а также возможности поиска фильма и его оценивания.

Для успешного создания работоспособной и удобной информационной системы следует изучить предметную область и проанализировать имеющиеся данные.

## Жизненный цикл базы данных

Жизненный цикл базы данных – процесс проектрирования, реализации и поддержкки базы данных.

Жизненный цикл базы данных для данной информационной системы:

* 1. План разработки (4 дня),
  2. Описание требований к базе данных (2 дня),
  3. Проектирование логической модели (5 дней),
  4. Выбор СУБД (1 день),
  5. Проектирование физической модели (4 дня),
  6. Реализация (10 дней),
  7. Использование и поддержка.

## 1.4 Выбор и характеристика СУБД

СУБД (система управления базами данных) – это информационная среда для создания баз данных и управления хранимыми в них сведениями об объектах реального мира.

На данный момент существует большое количество СУБД, которые можно использовать для работы с различными приложениями. Выбор конкретной системы зависит от поставленных целей приложения и определяется многими факторами, но главный из них – возможность работы с построенной моделью данных. Поэтому одной из важнейших характеристик является тип модели (иерархический, сетевой, реляционный), который поддерживается СУБД. База данных для информационной системы должна быть реляционной. Помимо модели данных важным показателем является стоимость лицензии для разработки базы данных и стоимость поддержки данной СУБД.

В конечном результате, использование СУБД MS SQL Server является наиболее целесообразным, если предусматривается дальнейшее развитие приложения и последующее расширение БД. Большое количество сопровождающих приложений значительно облегчают процесс создания и администрирования БД. Высокая производительность и надежность является главной отличительной чертой данной СУБД. На рисунке 1 представлено стартовое окно Microsoft SQL Server Management Studio.

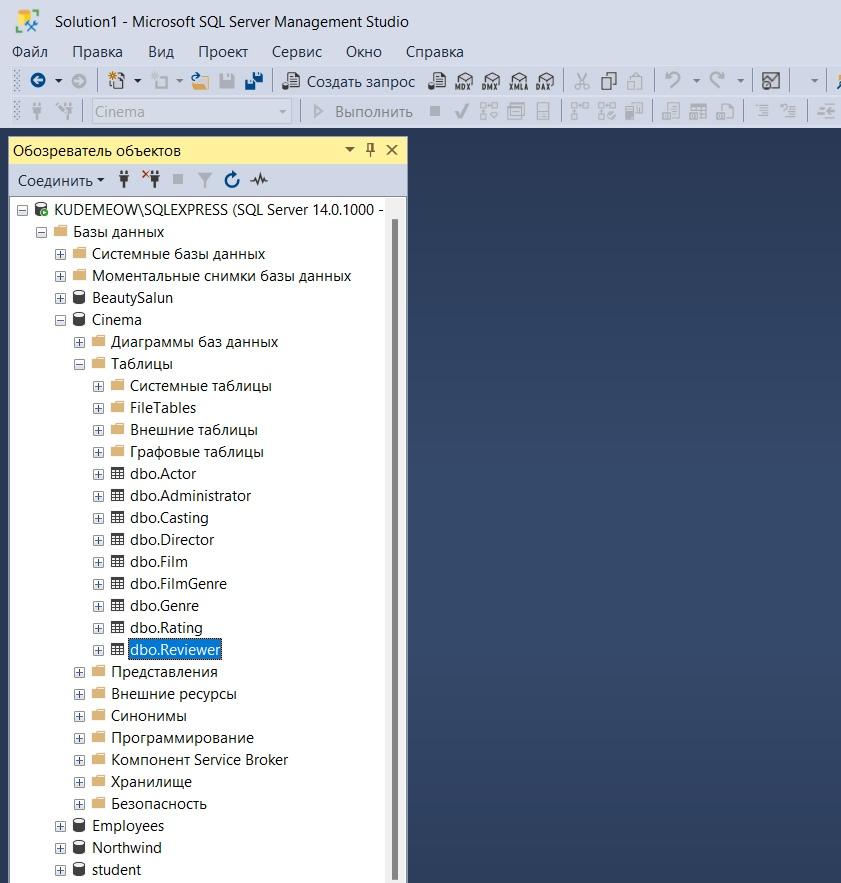


Рисунок 1 – Стартовое окно Microsoft SQL Server Management Studio

На сегодняшний день MS SQL Server – наиболее популярная СУБД, используемая для разработки настольных баз данных (БД).

MS SQL Server – реляционная СУБД, построенная на архитектуре клиент-сервер, представленном на Рисунке 2.

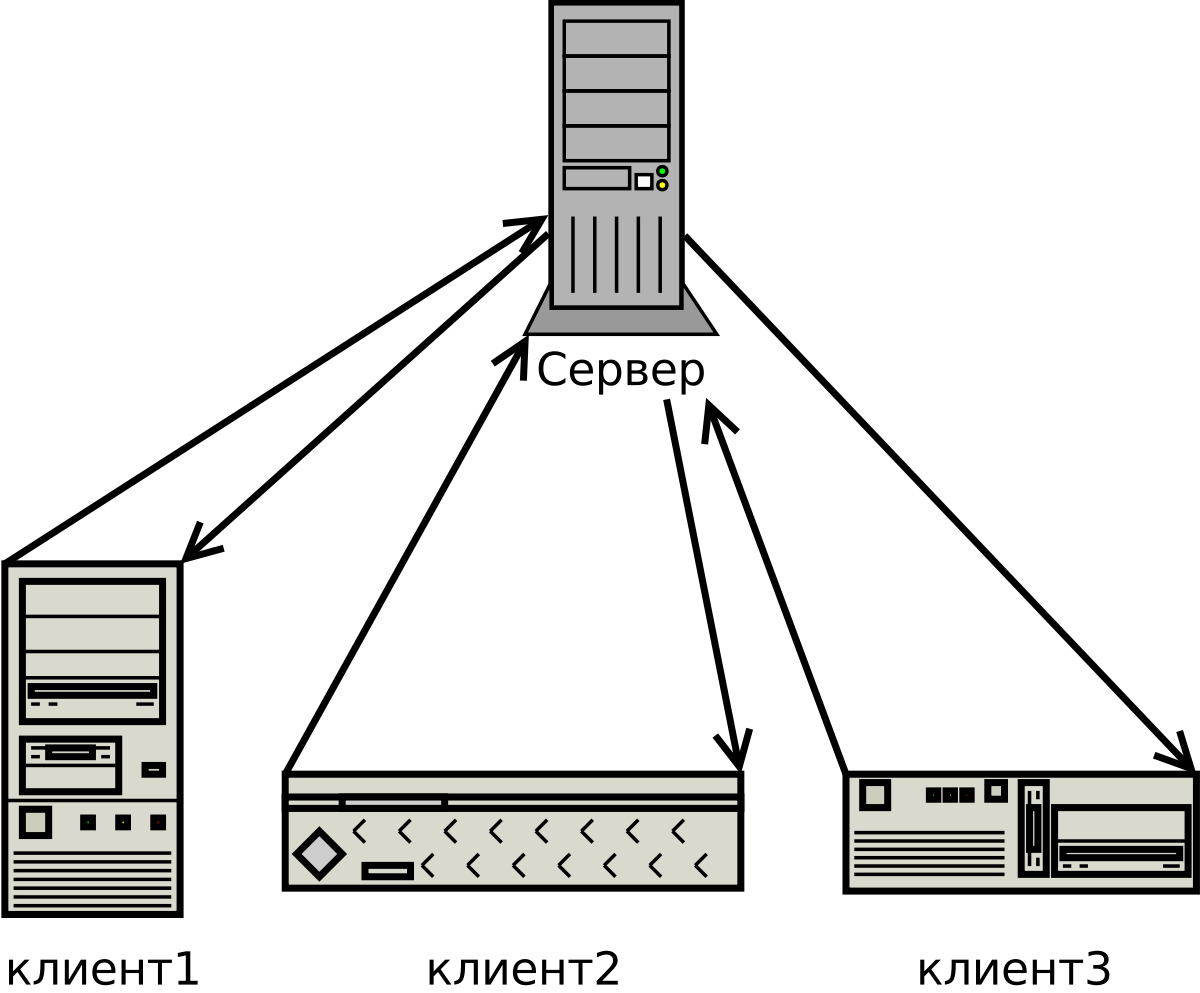


Рисунок 2 – Архитектура клиент-сервер

MS SQL Server успешно применяется для создания многопользовательских приложений, где файлы базы данных являются разделяемыми ресурсами в сети. Кроме того, в MS SQL Server реализована надёжная система защиты от несанкционированного доступа к файлам.

## 1.5 Выбор и характеристика среды разработки приложения

Для реализации задачи данного курсового проекта, а именно для создания информационной системы для любителя кино, был выбран язык программирования C# и среда разработки Microsoft Visual Studio 2022.

Microsoft Visual Studio – продукт компании Microsoft, который включает интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

Преимущества Visual Studio 2022:

1. Поколение компиляторов. Компилятор семейства Roslyn. Одно из главных преимуществ этих компиляторов – модульность. Она послужила ключом к организации компилятора в виде компонентной управляемой платформы,
2. Тестирование. Visual Studio 2022 позволяет использовать все возможности платформы Microsoft, для обеспечения контроля за качеством выпускаемого продукта.

C# – объектно-ориентированный язык программирования, который относится к семье языков с Cи-подобным синтаксисом. Язык имеет поддерживает перегрузку операторов, полиморфизм, атрибуты, делегаты, обобщённые типы и методы, итераторы, анонимные функции.

Основные достоинства С#:

* C# создавался параллельно с каркасом Framework .Net и полностью учитывает все его возможности,
* C# - объектно-ориентированный язык, в котором типы представлены в виде классов,
* C# является объектно-ориентированным языком, в котором существует возможность универсализации и наследования,
* Программисты C# получают такие же возможности работы с виртуальной машиной, что и программисты Java, благодаря платформе Framework .Net,
* Библиотека поддерживает удобное построение разных типов программных продуктов на C#, очень просто сохранять и получать информацию из БД.

# 2 Специальная часть

## 2.1 Постановка задачи

Спроектировать средствами MS SQL SERVER базу данных и создать средствами C# приложение, позволяющее:

1. Хранить информацию о фильмах,
2. Пользователю просматривать информацию о фильмах,
3. Оценивать фильмы,
4. Пользоваться поиском по названию фильма, жанру и режиссеру,
5. Администратору добавлять новые записи о фильмах,
6. Администратору изменять или удалять записи о фильмах.

## 2.2 Архитектура информационной системы

Разрабатываемая информационная система имеет клиент-серверную архитектуру, показанную на Рисунке 3.

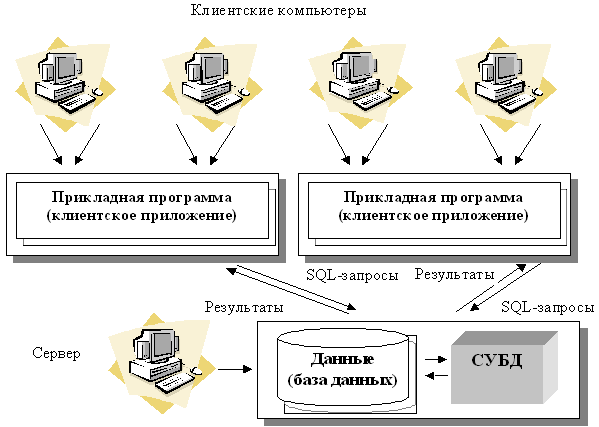


Рисунок 3 – Архитектура клиент-сервер

Архитектура клиент-сервер разделяет функции приложения пользователя (называемого клиентом) и сервера. Приложение-клиент формирует запрос к серверу, на котором расположена БД, на структурном языке запросов SQL.

Клиент и сервер взаимодействуют друг с другом в сети Интернет или любой другой компьютерной сети при помощи различных сетевых протоколов, анпример, IP-протокол, HTTP-протокол и т.д.

Сообщения, которые посылают клиенты называются HTTP запросы. Заопросы имеют специальные методы, которые говорят серверу, о том, как обрабатывать сообщение. А сообщения, которые отправляет сервер называются HTTP ответы.

Также стоит заметить, что в основе взаимодейтсвия клиент-сервер лежит прицнип того, что такое взаимодействие начинает клиент, сервер лишь отвечает клиенту и сообщает может ли он выполнить запрос.

Преимуществом модели взаимодействия клиент-сервер является то, что программный код клиентского приложения и серверного разделен. Если говорить про локальные компьютерные сети, то к приемуществам архитектуры можно отнести пониженные требования к машинам клиентов, т.к. большая часть вычислительных операций будет производиться на сервере, а также позволяет администратору сделать локальную сеть более защищенной.

К недостаткам взаимодействия клиент-сервер можно отнести то, что стоимость серверного оборудования значительно выше клиентского. Сервер должен обслуживать специально подготовленный человек.

## 2.3 Логическая модель базы данных

Логическая модель базы данных реализуемой информационной системы представлена на Рисунке 4.

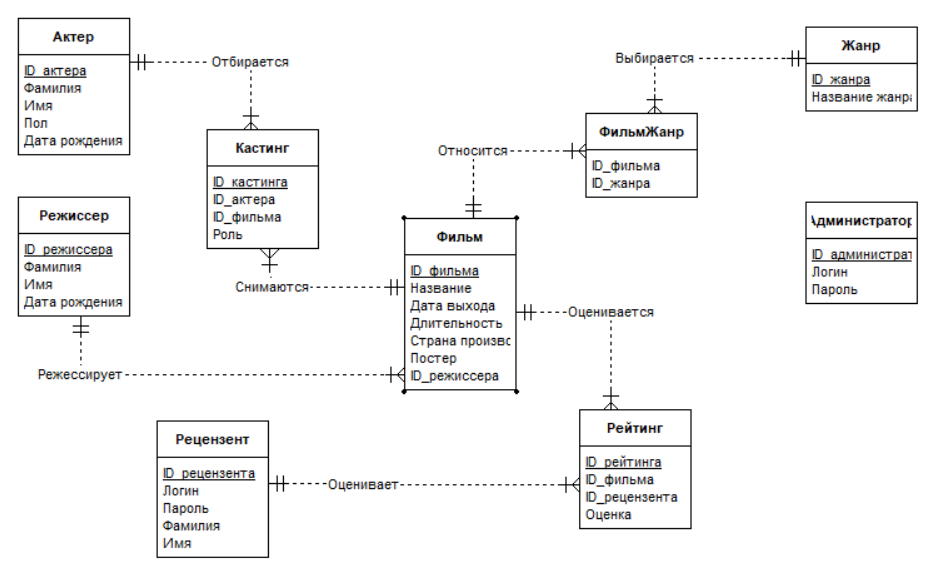


Рисунок 4 – Логическая модель базы данных

## 2.4 Нормализация таблиц

Нормализация – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности.

Нормализация нужна для:

* устранения аномалий,
* повышения производительности,
* повышения удобства управления данными.

Выделяют три формы нормализации:

1. первая нормальная форма. Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – атомарность значений. Подобная структура не является надежной,
2. вторая нормальная форма. Условием данной формы является отсутствие зависимости неключевых полей от части составного ключа,
3. третья нормальная форма. Третья НФ схожа по логике со второй НФ, но с некоторым отличием. Если вторая форма ликвидирует зависимости неключевых полей от части ключа, то третья нормальная форма исключает зависимость неключевых полей от других неключевых полей.

Все таблицы приведены к третьей нормальной форме, т.к. в них все атрибуты просты и неделимы. В таблицах не содержаться зависимости неключевых атрибутов от других неключевых атрибутов.

## 2.5 Описание таблиц

Таблица 1 – Actor (Актер)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Actor\_ID | Номер актера | int | PK(первичный ключ) |
| LastName | Фамилия | varchar (50) |  |
| FirstName | Имя | varchar (50) |  |

Продолжение Таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Sex | Пол | varchar (1) |  |
| DateOfBirth | Дата рождения | date |  |

Таблица 2 – Administrator(Администратор)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| ID\_Administrator | Номер администратора | int | PK(первичный ключ) |
| LoginA | Логин | varchar (100) |  |
| PasswordA | Пароль | varchar (10) |  |

Таблица 3 – Casting (Кастинг)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| ID\_Casting | Номер кастинга | int | PK (первичный ключ) |
| Actor\_ID | Номер актера | int |  |
| Film\_ID | Номер фильма | int |  |
| RoleActor | Роль | varchar (50) |  |

Таблица 4 – Director (Режиссер)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Director\_ID | Номер режиссера | int | PK (первичный ключ) |
| LastName | Фамилия | varchar (50) |  |
| FirstName | Имя | varchar (50) |  |
| DateOfBirth | Дата рождения | date |  |

Таблица 5 – Film (Фильм)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Film\_ID | Номер фильма | int | PK (первичный ключ) |
| Director\_ID | Номер режиссера | int |  |
| NameFilm | Название фильма | varchar (150) |  |
| ReleaseDate | Дата выхода | date |  |
| Duration | Длительность | time (7) |  |
| Country | Страна | varchar (50) |  |
| Posters | Постер | varbinary (MAX) |  |

Таблица 6 – FilmGenre (ФильмЖанр)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Film\_ID | Номер фильма | int | PK (первичный ключ) |
| Genre\_ID | Номер жанра | int | PK (первичный ключ) |

Таблица 7 – Genre (Жанр)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Genre\_ID | Номер жанра | int | PK (первичный ключ) |
| NameGenre | Название жанра | varchar (50) |  |

Таблица 8 – Reviewer (Рецензент/Пользователь)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| Reviewer\_ID | Номер актера | int | PK (первичный ключ) |
| LoginR | Фамилия | varchar (100) |  |
| PasswordR | Имя | varchar (100) |  |
| Familiya | Пол | nchar (100) |  |
| Imya | Дата рождения | nchar (100) |  |

Таблица 9 – Rating (Рейтинг)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название столбца** | **Описание** | **Тип** | **Примечание** |
| NumberRating | Номер актера | int | PK(первичный ключ) |
| Film\_ID | Фамилия | int |  |
| Reviewer\_ID | Имя | int |  |
| Score | Пол | int |  |

## 2.6 Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи)

Других объектов базы данных для данной информационной системы не представлено.

## 2.7 Разработка приложения

## 2.7.1 Диаграмма вариантов использования

На Рисунке 5 показана UseCase диаграмма (диаграмма вариантов использования) для данной информационной системы.

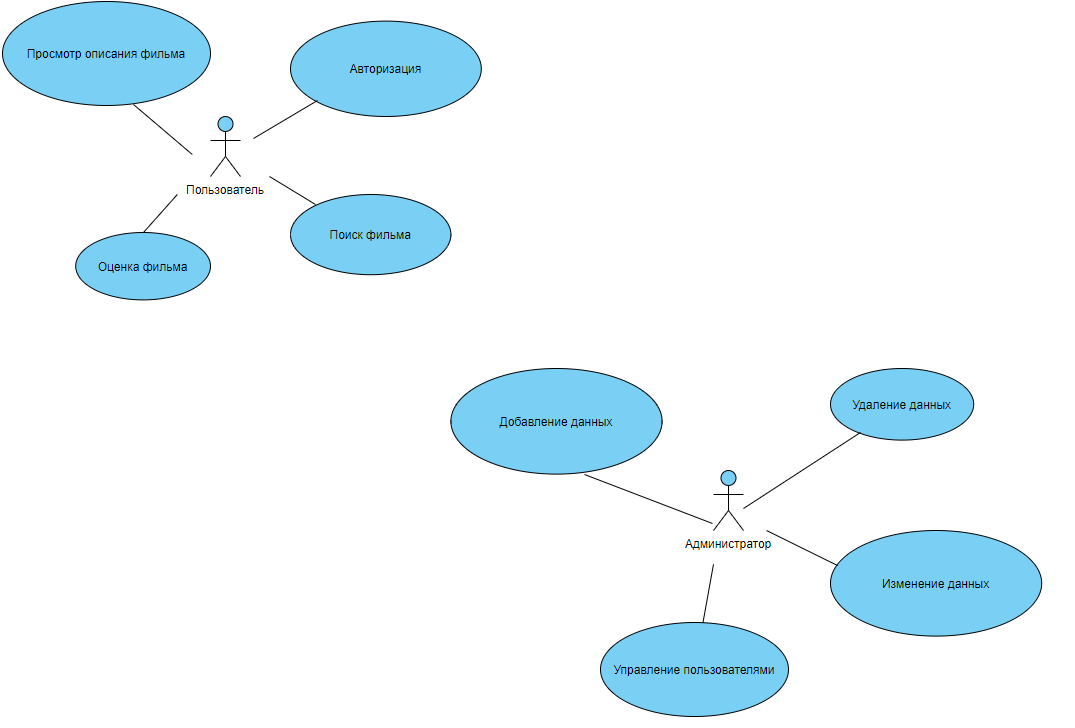
****

Рисунок 5 – диаграмма вариантов использования

### 2.7.2 Решение главной задачи проекта

Главная задача курсовой работы создать информационную систему для любителя кино.

Подзадачи:

1. Продумать способ вывода информации о фильме,
2. Разработать способ поиска фильма по нескольким атрибутам,
3. Осуществить оценку фильма по пятибалльной шкале,
4. Продумать случай изменения оценки с одной на другую,
5. Реализовать редактирование и удаление данных для администратора.

Вывод информации о фильме на главный экран пользователя происходит в элемент ListView посредством привязки данных Binding Path.

Код: реализации:

public WindowUserApp()

{

InitializeComponent();

CinemaEntities db = new CinemaEntities();

FilmsShow.ItemsSource = db.Film.ToList();

}

Поиск фильма происходит моментально по трем атрибутам: название фильма, жанр и режиссер.

Код реализации:

private void Searcher\_TextChanged(object sender, TextChangedEventArgs e)

{

string Text = Searcher.Text.ToLower().Trim();

FilmsShow.ItemsSource = films.Where(x=> Text.Split(' ').All(t =>

x.Director.LastName.ToLower().Contains(t)

|| x.Director.FirstName.ToLower().Contains(t))

|| x.Genre.Any(y => y.NameGenre.ToLower().Contains(Text))

|| x.NameFilm.ToLower().Contains(Text));

}

Оценка фильма добавляется в базу данных через указание столбцов и присваиваемых им значений. Для получения ID пользователя, который оценивает фильм, создается отдельный класс AppInfo, в котором имеется свойство currentUserId. Это свойство возвращает метод GetCurrentUser().

Код класса AppInfo:

static class AppInfo

{

public static int currentUserId { get; set; }

public static int GetCurrentUser() => currentUserId;

public static void SetReviewer(int reviewer) => currentUserId = reviewer;

}

Код добавления оценки в базу данных:

if (ratingId == 0)

{

Rating ratingCurrentFilm = new Rating();

ratingCurrentFilm.Film\_ID = currentFilmId;

ratingCurrentFilm.Score = choosenScore;

ratingCurrentFilm.Reviewer\_ID = AppInfo.GetCurrentUser();

db.Rating.Add(ratingCurrentFilm);

}

Заменить уже имеющуюся оценку на другую, в случае, если пользователь передумал или сохранил случайно, возможно поиском старой оценки и заменой ее на выбранную. С помощью метода Find() в базе данных отыскивается номер этого рейтинга и оценка заменяется на выбранную в окне оценивания.

Код, реализующий это:

else

{

Rating rating = db.Rating.Find(ratingId);

rating.Score = choosenScore;

}

Редактирование данных для администратора выполнено через элемент DataGrid. Этот элемент по умолчанию позволяет редактировать данные внутри двойным нажатием на ячейку. После редактирования по нажатию кнопки “Изменить” измененные данные сохраняются в базу данных. Сохранение реализовано с помощью метода SaveChanges().

Код редактирования:

private void EditClick(object sender, RoutedEventArgs e)

{

if ((sender as Button).DataContext is Film)

{

Tables.ItemsSource = CinemaEntities.GetContext().Film.ToList();

}

else if ((sender as Button).DataContext is Reviewer)

{

Tables.ItemsSource = CinemaEntities.GetContext().Reviewer.ToList();

}

else if ((sender as Button).DataContext is Administrator)

{

Tables.ItemsSource = CinemaEntities.GetContext().Administrator.ToList();

}

else if ((sender as Button).DataContext is Actor)

{

Tables.ItemsSource = CinemaEntities.GetContext().Actor.ToList();

}

else if ((sender as Button).DataContext is Casting)

{

Tables.ItemsSource = CinemaEntities.GetContext().Casting.ToList();

}

else if ((sender as Button).DataContext is Director)

{

Tables.ItemsSource = CinemaEntities.GetContext().Director.ToList();

}

else if ((sender as Button).DataContext is Rating)

{

Tables.ItemsSource = CinemaEntities.GetContext().Rating.ToList();

}

else if ((sender as Button).DataContext is Genre)

{

Tables.ItemsSource = CinemaEntities.GetContext().Genre.ToList();

}

CinemaEntities.GetContext().SaveChanges();

Tables.Items.Refresh();

}

По нажатию кнопки “Удалить” вызывается метод Remove(). Кнопка удаления расположена на каждой строке возле записи в таблице.

Код удаления:

private void DeleteClick(object sender, RoutedEventArgs e)

{

if((sender as Button).DataContext is Film film)

{

CinemaEntities.GetContext().Film.Remove(film);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Film.ToList(); ;

}

else if ((sender as Button).DataContext is Reviewer reviewer)

{

CinemaEntities.GetContext().Reviewer.Remove(reviewer);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Reviewer.ToList(); ;

}

else if((sender as Button).DataContext is Administrator administrator)

{

CinemaEntities.GetContext().Administrator.Remove(administrator);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Administrator.ToList(); ;

}

else if((sender as Button).DataContext is Actor actor)

{

CinemaEntities.GetContext().Actor.Remove(actor);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Actor.ToList(); ;

}

else if((sender as Button).DataContext is Casting casting)

{

CinemaEntities.GetContext().Casting.Remove(casting);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Casting.ToList(); ;

}

else if((sender as Button).DataContext is Director director)

{

CinemaEntities.GetContext().Director.Remove(director);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Director.ToList(); ;

}

else if((sender as Button).DataContext is Genre genre)

{

CinemaEntities.GetContext().Genre.Remove(genre);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Genre.ToList(); ;

}

else if((sender as Button).DataContext is Rating rating)

{

CinemaEntities.GetContext().Rating.Remove(rating);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Rating.ToList(); ;

}

Tables.Items.Refresh();

}

### 2.7.3 Тестирование приложения

Таблица 10 - Проверка требований к приложению

|  |  |  |
| --- | --- | --- |
| **Номер** | **Действие** | **Результат** |
| **1** | **2** | **3** |
| 1 | Проверка правильности ввода логина и пароля | Проверяет правильность ввода логина и пароля и разрешает перейти к главному окну. В случае неверного ввода выдает ошибку |
| 2 | Удаление и изменение данных в таблицах | Удаляет и изменяет данные |

### 2.7.4 Защита информационной системы

Защита информационной системы происходит посредством хранения в базе данных логинов и паролей пользователей и администраторов. Введенные данные сравниваются с данными в базе и, в случае успеха, происходит переход на главное окно в зависимости от роли.

## 2.8 Инструкция пользователю

Чтобы использование приложения стало доступным, пользователю нужно зарегистрироваться. Если пользователь уже зарегистрирован, то он вводит свои логин и пароль и жмет кнопку “Войти”. Если же пользователь новый, то зарегистрироваться он может по кнопке “Зарегистрироваться” внизу окна Авторизации. В окне Регистрации пользователь обязан заполнить все формы и нажать на кнопку “Сохранить”. После успешной авторизации пользователь переходит в Главное окно со всем функционалом.

Просмотр и выбор фильмов происходит через прокрутку на колесико мыши.

Для того, чтобы оценить фильм, пользователь должен нажать кнопку “Поставить оценку” и тогда откроется окно Оценивания. В нем представлена пятибалльная шкала оценки. Пользователь выбирает ту оценку, которую считает нужной и нажимает кнопку “Сохранить”.

### 2.8.1 Общие сведения об информационной системе

Данное приложение подходит для людей разной возрастной категории и подразумевает поиск фильма для просмотра в свободное время или для ознакомления с актуальной информацией о фильме, а также его оценке.

### 2.8.2 Требования к техническим средствам

Минимальные технические средства включают в себя:

* процессор Intel Core i3 или выше,
* видеокарту NVIDIA GTX 1050 или выше,
* клавиатуру,
* мышь,
* монитор,
* оперативная память размером от 2 Гб.

### 2.8.3 Требования к программным средствам

Требования программных средств:

* Операционная система не ниже Windows 7,
* Microsoft Visual Studio

### 2.8.4 Настройка информационной системы

Запускать лучше всего через Microsoft Visual Studio.

# Заключение

Данный курсовой проект посвящен созданию информационной системы для любителя кино.

За время работы над курсовым проектом:

1. проанализирована предметная область,
2. разработана база данных,
3. создано приложение для ознакомления с фильмами,
4. отточены навыки по работе с базой данных и созданию WPF приложений.

В приложении пользователи разделены на администраторов и обычных пользователей. Каждый из пользователей в зависимости от своей роли после прохождения авторизации переходит на главное окно. Имеется выход из главного окна и аккаунта.

Пользователь имеет право просматривать информацию о фильмах и оценивать их.

Администратор же имеет право редактировать и удалять записи из таблиц, если это необходимо, а также управляет аккаунтами пользователей.

Если говорить про расширение функционала приложения, то оно будет состоять:

* в большем количестве представляемых фильмов,
* возможности оставлять свое мнение о фильме в виде комментария,
* просмотра фильма в виде видео-ролика прямо в приложении.

# список использованной литературы

1. https://metanit.com/sharp/
2. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари , Бен Албахари. - М.: БХВ-Петербург, 2015. - 499 c.
3. Кариев, Ч. А. Разработка Windows-приложений на основе Visual C# (+ CD-ROM) / Ч.А. Кариев. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2007. - 768 c.
4. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL: полное руководство, 3-е издание = SQL: The Complete Reference, 3rd edition. – M.: «Вильямс», 2014. – 960 с.
5. Дунаев, В. Базы данных. Язык SQL для студента / В. Дунаев. - М.: БХВ-Петербург, 2012. - 320 c.

# приложения

## Приложение А

USE master

IF EXISTS (SELECT \* FROM SYS.DATABASES WHERE NAME = 'Cinema')

DROP DATABASE Cinema

GO

CREATE DATABASE Cinema ON PRIMARY(NAME = CinemaData,

FILENAME = 'D:\SQL\MSSQL14.SQLEXPRESS\MSSQL\DATA\CinemaData.mdf',

SIZE = 3 MB, MAXSIZE = 100, FILEGROWTH = 10)

LOG ON

(NAME = CinemaLog, FILENAME = 'D:\SQL\MSSQL14.SQLEXPRESS\MSSQL\DATA\CinemaLog.ldf',

SIZE = 3 MB, MAXSIZE = 100, FILEGROWTH = 10)

GO

USE Cinema

GO

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Actor' AND TYPE = 'U')

DROP TABLE Actor

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Director' AND TYPE = 'U')

DROP TABLE Director

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Casting' AND TYPE = 'U')

DROP TABLE Casting

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Film' AND TYPE = 'U')

DROP TABLE Film

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'FilmGenre' AND TYPE = 'U')

DROP TABLE FilmGenre

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Reviewer' AND TYPE = 'U')

DROP TABLE Reviewer

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Rating' AND TYPE = 'U')

DROP TABLE Rating

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Genre' AND TYPE = 'U')

DROP TABLE Genre

IF EXISTS (SELECT \* FROM SYSOBJECTS WHERE NAME = 'Administrator' AND TYPE = 'U')

DROP TABLE Administrator

GO

--создание таблиц

CREATE TABLE Actor(Actor\_ID int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

LastName varchar(50) NOT NULL,

FirstName varchar(50) NOT NULL,

Sex varchar(1) NOT NULL,

DateOfBirth date NOT NULL)

CREATE TABLE Genre(Genre\_ID int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

Genre varchar(50) NOT NULL)

CREATE TABLE Director(Director\_ID int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

LastName varchar(50) NOT NULL,

FirstName varchar(50) NOT NULL,

DateOfBirth date NOT NULL)

CREATE TABLE Film(Film\_ID int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

Director\_ID int NOT NULL,

NameFilm varchar(150) NOT NULL,

ReleaseDate date NOT NULL,

Duration time NOT NULL,

Country varchar(50) NOT NULL,

CONSTRAINT fk\_dirID FOREIGN KEY (Director\_ID) REFERENCES Director(Director\_ID))

CREATE TABLE FilmGenre(Film\_ID int NOT NULL,

Genre\_ID int NOT NULL,

CONSTRAINT fk\_genreID FOREIGN KEY (Genre\_ID) REFERENCES Genre(Genre\_ID),

CONSTRAINT fk\_filmID FOREIGN KEY (Film\_ID) REFERENCES Film(Film\_ID))

CREATE TABLE Casting(ID\_Casting int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

Actor\_ID int NOT NULL,

Film\_ID int NOT NULL,

RoleActor varchar(50) NOT NULL,

CONSTRAINT fk\_actorID FOREIGN KEY (Actor\_ID) REFERENCES Actor (Actor\_ID),

CONSTRAINT fk\_CasFilm FOREIGN KEY (Film\_ID) REFERENCES Film(Film\_ID))

CREATE TABLE Reviewer(Reviewer\_ID int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

LoginR varchar(100),

PasswordR varchar(10))

CREATE TABLE Rating(NumberRating int IDENTITY (1, 1) PRIMARY KEY NOT NULL,

Film\_ID int,

Reviewer\_ID int,

Score int,

CONSTRAINT fk\_RatFilm FOREIGN KEY (Film\_ID) REFERENCES Film(Film\_ID),

CONSTRAINT fk\_RatRew FOREIGN KEY (Reviewer\_ID) REFERENCES Reviewer(Reviewer\_ID))

CREATE TABLE Administrator(ID\_Administrator int PRIMARY KEY NOT NULL,

LoginA varchar(100),

PasswordA varchar(10))

INSERT INTO Actor(FirstName, LastName, Sex, DateOfBirth)

VALUES('Леонардо', 'Ди Каприо', 'М', CAST('1974-11-11' AS date)),

('Джордан', 'Пил', 'М', CAST('1979-02-21' AS date)),

('Мэттью', 'Макконахи', 'М', CAST('1969-11-04' AS date)),

('Уинстон', 'Дьюк', 'М', CAST('1986-11-15' AS date)),

('Райан', 'Гослинг', 'М', CAST('1980-11-12' AS date)),

('Брэд', 'Питт', 'М', CAST('1963-12-18' AS date)),

('Лупита', 'Нионго', 'Ж', CAST('1983-03-01' AS date)),

('Гвинет', 'Пэлтроу', 'Ж', CAST('1972-09-27' AS date)),

('Энн', 'Хэтэуэй', 'Ж', CAST('1982-11-12' AS date)),

('Джессика', 'Честейн', 'Ж', CAST('1977-03-24' AS date)),

('Скарлетт', 'Йоханссон', 'Ж', CAST('1984-11-22' AS date)),

('Джозеф', 'Гордон-Левитт', 'М', CAST('1981-02-17' AS date)),

('Зендея', '-', 'Ж', CAST('1996-09-01' AS date)),

('Тимоти', 'Шаламе', 'М', CAST('1995-12-27' AS date)),

('Тим', 'Роббинс', 'М', CAST('1958-10-16' AS date)),

('Морган', 'Фримен', 'М', CAST('1937-06-01' AS date)),

('Бен', 'Аффлек', 'М', CAST('1972-08-15' AS date)),

('Розамунд', 'Пайк', 'Ж', CAST('1979-01-27' AS date)),

('Чарли', 'Ханнэм', 'М', CAST('1980-04-10' AS date)),

('Колин', 'Фаррелл', 'М', CAST('1976-05-31' AS date)),

('Мирей', 'Инос', 'Ж', CAST('1975-09-22' AS date)),

('Пилу', 'Асбек', 'М', CAST('1982-03-02' AS date)),

('Майкл', 'Питт', 'М', CAST('1981-04-10' AS date)),

('Оскар', 'Айзек', 'М', CAST('1979-03-09' AS date)),

('Том', 'Харди', 'М', CAST('1977-09-15' AS date)),

('Кэри', 'Маллиган', 'Ж', CAST('1985-05-28' AS date))

INSERT INTO Genre(Genre)

VALUES('Научная фантастика'),

('Боевик'),

('Приключения'),

('Ужасы'),

('Драма'),

('Детектив'),

('Триллер'),

('Криминал')

INSERT INTO Director(FirstName, LastName, DateOfBirth)

VALUES('Дени', 'Вильнев', CAST('1967-10-03' AS date)),

('Кристофер', 'Нолан', CAST('1970-07-30' AS date)),

('Фрэнк', 'Дарабонт', CAST('1959-01-28' AS date)),

('Гай', 'Ричи', CAST('1968-09-10' AS date)),

('Дэвид', 'Финчер', CAST('1962-08-28' AS date)),

('Марк', 'Форстер', CAST('1969-11-27' AS date)),

('Руперт', 'Сандерс', CAST('1971-03-16' AS date)),

('Джордан', 'Пил', CAST('1979-02-21' AS date)),

('Николас', 'Виндинг Рефн', CAST('1970-09-29' AS date))

INSERT INTO Film(Director\_ID, NameFilm, ReleaseDate, Duration, Country)

VALUES(1, 'Дюна', CAST('2021-09-03' AS date), CAST('02:35:00' AS time), 'США, Канада, Венгрия'),

(3, 'Побег из Шоушенка', CAST('1994-09-10' AS date), CAST('02:22:00' AS time), 'США'),

(4, 'Джентельмены', CAST('2019-12-03' AS date), CAST('01:53:00' AS time), 'Великобритания, США'),

(2, 'Интерстеллар', CAST('2014-10-26' AS date), CAST('02:49:00' AS time), 'Великобритания, Канада, США'),

(5, 'Семь', CAST('1995-09-25' AS date), CAST('02:07:00' AS time), 'США'),

(2, 'Начало', CAST('2010-07-08' AS date), CAST('02:28:00' AS time), 'США, Великобритания'),

(6, 'Война миров Z', CAST('2013-06-02' AS date), CAST('01:56:00' AS time), 'США'),

(7, 'Призрак в доспехах', CAST('2017-03-16' AS date), CAST('01:46:00' AS time), 'США, Индия, Гонконг, Китай, Канада'),

(5, 'Исчезнувшая', CAST('2019-09-26' AS date), CAST('02:29:00' AS time), 'США'),

(8, 'Мы', CAST('2019-03-08' AS date), CAST('01:56:00' AS time), 'США, Китай, Япония'),

(9, 'Драйв', CAST('2011-05-20' AS date), CAST('01:40:00' AS time), 'США')

INSERT INTO FilmGenre(Film\_ID, Genre\_ID)

VALUES(1, 1),

(1, 2),

(1, 3),

(2, 5),

(2, 6),

(3, 2),

(3, 8),

(4, 1),

(4, 3),

(4, 5),

(4, 6),

(5, 5),

(5, 6),

(5, 8),

(6, 2),

(6, 1),

(6, 7),

(6, 3),

(6, 5),

(7, 4),

(7, 2),

(7, 5),

(7, 1),

(7, 3),

(7, 7),

(8, 2),

(8, 1),

(8, 5),

(8, 3),

(8, 7),

(9, 7),

(9, 6),

(9, 5),

(10, 4),

(10, 7),

(10, 6),

(11, 2),

(11, 5),

(11, 8),

(11, 7)

INSERT INTO Casting(Actor\_ID, Film\_ID, RoleActor)

VALUES(14, 1, 'Пол Атрейдес'),

(13, 1, 'Чани'),

(24, 1, 'Лето Атрейдес'),

(16, 2, 'Эллис Бойд "Рэд" Рэддинг'),

(15, 2, 'Энди Дюфрейн'),

(19, 3, 'Реймонд Смит'),

(3, 3, 'Мики Пирсон'),

(20, 3, 'Тренер'),

(3, 4, 'Купер'),

(9, 4, 'Амелия Бренд'),

(10, 4, 'Мёрф'),

(14, 4, 'Том'),

(6, 5, 'Дэвид Миллс'),

(16, 5, 'Уильям Сомерсет'),

(8, 5, 'Трэйси Миллс'),

(1, 6, 'Доминик Кобб'),

(12, 6, 'Артур'),

(25, 6, 'Имс'),

(6, 7, 'Джерри Лейн'),

(21, 7, 'Карэн Лейн'),

(11, 8, 'Мотоко Кусанаги'),

(22, 8, 'Бато'),

(23, 8, 'Кудзэ'),

(17, 9, 'Николас Данн'),

(18, 9, 'Эми Эллиотт-Данн'),

(4, 10, 'Гэбриел "Гэйб" Уилсон, Абрахам'),

(7, 10, 'Аделаида Уилсон, Рэд'),

(2, 10, 'Умирающий кролик, голос в доме забав'),

(5, 11, 'Водитель'),

(26, 11, 'Айрин')

INSERT INTO Reviewer(LoginR, PasswordR)

VALUES(NULL, NULL)

INSERT INTO Rating(Film\_ID, Reviewer\_ID, Score) --оценка по 5-бальной шкале

VALUES(NULL, NULL, NULL)

## Приложение Б

namespace WpfApp1

{

static class AppInfo

{

public static int currentUserId { get; set; }

public static int GetCurrentUser() => currentUserId;

public static void SetReviewer(int reviewer) => currentUserId = reviewer;

}

}

<Window x:Class="Kursach.Enter"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Авторизация" Height="490" Width="545">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#b48764"/>

<GradientStop Color="#887359" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Label Content="Авторизация" HorizontalAlignment="Center" Height="70" VerticalAlignment="Top" Background="{x:Null}" Foreground="#FFF4EBEB" FontSize="60" FontFamily="Bahnschrift Light Condensed" Margin="0,30,0,0"/>

<TextBox x:Name="EnterLogin" HorizontalAlignment="Left" Height="30" Margin="170,150,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<Label Content="Логин" HorizontalAlignment="Left" Height="48" Margin="56,132,0,0" VerticalAlignment="Top" Width="139" Foreground="#FFF4EBEB" FontSize="40" FontFamily="Bahnschrift Light Condensed"/>

<Label Content="Пароль" HorizontalAlignment="Left" Height="51" Margin="56,186,0,0" VerticalAlignment="Top" Width="156" Foreground="#FFF4EBEB" FontSize="40" FontFamily="Bahnschrift Light Condensed"/>

<PasswordBox x:Name="EnterPassword" HorizontalAlignment="Left" Height="30" Margin="170,202,0,0" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<Button x:Name="Enter1" Content="Войти" Height="30" Width="100" FontSize="25" FontFamily="Bahnschrift Light Condensed" Background="{x:Null}" BorderBrush="Black" Margin="231,271,195,117" Click="ClickEnter" IsDefault="True"/>

<Button x:Name="Registration" Content="Зарегистрироваться" Height="30" FontSize="25" FontFamily="Bahnschrift Light Condensed" Background="{x:Null}" Foreground="Black" Margin="144,344,108,44" BorderBrush="Black" Click="ClickRegistration"/>

</Grid>

</Window>

using System.Windows;

using System.Linq;

using WpfApp1;

using System.Windows.Input;

namespace Kursach

{

public partial class Enter : Window

{

public Enter()

{

InitializeComponent();

}

public void ClickEnter(object sender, RoutedEventArgs e)

{

using (CinemaEntities db = new CinemaEntities())

{

var sqlReviewer = (from reviewer in db.Reviewer

where reviewer.LoginR == EnterLogin.Text && reviewer.PasswordR == EnterPassword.Password

select reviewer).FirstOrDefault();

if (sqlReviewer != null)

{

AppInfo.SetReviewer(sqlReviewer.Reviewer\_ID);

WindowUserApp mainApp = new WindowUserApp();

this.Close();

mainApp.Show();

}

else

{

var sqlAdmin = (from administrator in db.Administrator

where administrator.LoginA == EnterLogin.Text && administrator.PasswordA == EnterPassword.Password

select administrator).FirstOrDefault();

if (sqlAdmin != null)

{

WindowAdminApp mainAppAdmin = new WindowAdminApp();

this.Close();

mainAppAdmin.Show();

}

else

{

MessageBox.Show("Пользователь не найден", "", MessageBoxButton.OK);

}

}

}

}

private void ClickRegistration(object sender, RoutedEventArgs e)

{

Registration registration = new Registration();

registration.Show();

}

}

}

<Window x:Class="WpfApp1.Rate"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Оценивание фильма" Height="376" Width="378">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#b48764"/>

<GradientStop Color="#887359" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Label Content="Выберите оценку" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" HorizontalAlignment="Left" Height="48" Margin="33,145,0,0" VerticalAlignment="Top" Width="176"/>

<Label Content="Оценивание фильма" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" HorizontalAlignment="Center" Height="48" Margin="0,10,0,0" VerticalAlignment="Top" Width="200"/>

<StackPanel x:Name="RadioButtons">

<RadioButton Checked="ChooseScore\_Checked" Content="1" HorizontalAlignment="Left" Height="35" Margin="289,60,0,0" VerticalAlignment="Top" Width="47" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" />

<RadioButton Checked="ChooseScore\_Checked" Content="2" HorizontalAlignment="Left" Height="35" Margin="289,15,0,0" VerticalAlignment="Top" Width="47" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" />

<RadioButton Checked="ChooseScore\_Checked" Content="3" HorizontalAlignment="Left" Height="35" Margin="289,15,0,0" VerticalAlignment="Top" Width="47" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" />

<RadioButton Checked="ChooseScore\_Checked" Content="4" HorizontalAlignment="Left" Height="35" Margin="289,15,0,0" VerticalAlignment="Top" Width="47" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" />

<RadioButton Checked="ChooseScore\_Checked" Content="5" HorizontalAlignment="Left" Height="35" Margin="289,15,0,0" VerticalAlignment="Top" Width="47" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="28" />

</StackPanel>

<Button x:Name="RateButton" Content="Сохранить" HorizontalAlignment="Center" Height="34" Margin="0,302,0,0" VerticalAlignment="Top" Width="150" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="25" Click="RateButton\_Click"/>

</Grid>

</Window>

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using WpfApp1;

namespace WpfApp1

{

/// <summary>

/// Логика взаимодействия для Rate.xaml

/// </summary>

public partial class Rate : Window

{

int currentFilmId;

int ratingId = 0;

public Rate(int CurrentFilmId)

{

InitializeComponent();

currentFilmId = CurrentFilmId;

using (CinemaEntities db = new CinemaEntities())

{

Rating lastRating = db.Rating.Where(rating => rating.Film\_ID == CurrentFilmId &&

rating.Reviewer\_ID == AppInfo.currentUserId).FirstOrDefault();

choosenScore = (lastRating is null) ? 0 : (int)lastRating.Score;

if (choosenScore != 0)

{

ratingId = lastRating.NumberRating;

(RadioButtons.Children[choosenScore - 1] as RadioButton).IsChecked = true;

}

}

}

int choosenScore;

private void ChooseScore\_Checked(object sender, RoutedEventArgs e)

{

choosenScore = int.Parse((sender as RadioButton).Content.ToString());

}

private void RateButton\_Click(object sender, RoutedEventArgs e)

{

if (choosenScore != 0)

{

using (CinemaEntities db = new CinemaEntities())

{

if (ratingId == 0)

{

Rating ratingCurrentFilm = new Rating();

ratingCurrentFilm.Film\_ID = currentFilmId;

ratingCurrentFilm.Score = choosenScore;

ratingCurrentFilm.Reviewer\_ID = AppInfo.GetCurrentUser();

db.Rating.Add(ratingCurrentFilm);

}

else

{

Rating rating = db.Rating.Find(ratingId);

rating.Score = choosenScore;

}

db.SaveChanges();

this.Close();

}

}

}

}

}

<Window x:Class="Kursach.Registration"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Регистрация" Height="482" Width="571">

<Grid Margin="0,0,0,0">

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#b48764"/>

<GradientStop Color="#887359" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Label Content="Регистрация" HorizontalAlignment="Left" Height="79" Width="264" VerticalAlignment="Top" Background="{x:Null}" Foreground="#FFF4EBEB" FontSize="60" FontFamily="Bahnschrift Light Condensed" Margin="149,26,0,0"/>

<TextBox x:Name="LoginReg" HorizontalAlignment="Left" Height="30" Margin="223,254,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<Label Content="Логин" HorizontalAlignment="Left" Height="48" Margin="17,245,0,0" VerticalAlignment="Top" Width="93" Foreground="#FFF4EBEB" FontSize="35" FontFamily="Bahnschrift Light Condensed"/>

<Label Content="Пароль" HorizontalAlignment="Left" Height="51" Margin="17,293,0,0" VerticalAlignment="Top" Width="111" Foreground="#FFF4EBEB" FontSize="35" FontFamily="Bahnschrift Light Condensed"/>

<Button x:Name="Save" Content="Сохранить" HorizontalAlignment="Left" Height="35" Margin="354,380,0,0" VerticalAlignment="Top" Width="182" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="25" Click="ClickSave"/>

<PasswordBox x:Name="PasswordReg" HorizontalAlignment="Left" Height="30" Margin="223,304,0,0" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<TextBox x:Name="Familiya" HorizontalAlignment="Left" Height="30" Margin="223,158,0,0" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<TextBox x:Name="Imya" HorizontalAlignment="Left" Height="30" VerticalAlignment="Top" Width="222" BorderBrush="Black" Background="{x:Null}" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="223,206,0,0"/>

<Label Content="Имя" HorizontalAlignment="Left" Height="48" Margin="17,197,0,0" VerticalAlignment="Top" Width="61" Foreground="#FFF4EBEB" FontSize="35" FontFamily="Bahnschrift Light Condensed"/>

<Label Content="Фамилия" HorizontalAlignment="Left" Height="48" Margin="17,149,0,0" VerticalAlignment="Top" Width="122" Foreground="#FFF4EBEB" FontSize="35" FontFamily="Bahnschrift Light Condensed"/>

</Grid>

</Window>

using System.Windows;

using WpfApp1;

using System.Linq;

namespace Kursach

{

public partial class Registration : Window

{

public Registration()

{

InitializeComponent();

}

public void ClickSave(object sender, RoutedEventArgs e)

{

using (CinemaEntities db = new CinemaEntities())

{

var LogRew = (from reviewer in db.Reviewer

where reviewer.LoginR == LoginReg.Text

select reviewer).FirstOrDefault();

var LogAdm = (from administator in db.Administrator

where administator.LoginA == LoginReg.Text

select administator).FirstOrDefault();

var FamiliyaR = (from reviewer in db.Reviewer

where reviewer.Familiya == Familiya.Text

select reviewer).FirstOrDefault();

var ImyaR = (from reviewer in db.Reviewer

where reviewer.Imya == Imya.Text

select reviewer).FirstOrDefault();

if (LogRew != null || LogAdm != null)

{

MessageBox.Show("Такой пользователь уже существует. Введите другой логин", "", MessageBoxButton.OK);

}

else

{

Reviewer reviewer = new Reviewer()

{

LoginR = LoginReg.Text,

PasswordR = PasswordReg.Password,

Familiya = Familiya.Text,

Imya = Imya.Text

};

if (!string.IsNullOrWhiteSpace(LoginReg.Text) && !string.IsNullOrWhiteSpace(PasswordReg.Password)

&& !string.IsNullOrWhiteSpace(Familiya.Text) && !string.IsNullOrWhiteSpace(Imya.Text))

{

db.Reviewer.Add(reviewer);

db.SaveChanges();

this.Close();

}

else

{

MessageBox.Show("Проверьте правильность ввода данных", "", MessageBoxButton.OK);

}

}

}

}

}

}

<Window x:Class="WpfApp1.WindowAdminApp"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

mc:Ignorable="d"

Title="Окно администратора" Height="550" Width="810">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="30"/>

<RowDefinition/>

<RowDefinition Height="30"/>

</Grid.RowDefinitions>

<DockPanel Grid.Row="0">

<Button Width="100" HorizontalAlignment="Left" Click="OpenFilms">

<TextBlock Text="Таблица фильмов" TextTrimming="CharacterEllipsis"/>

</Button>

<Button Width="100" HorizontalAlignment="Left" Click="OpenUsers">

<TextBlock Text="Таблица пользователей" TextTrimming="CharacterEllipsis"/>

</Button>

<Button Width="100" HorizontalAlignment="Left" Click="OpenAdmins">

<TextBlock Text="Таблица администраторов" TextTrimming="CharacterEllipsis"/>

</Button>

<Button Content="Таблица актеров" Width="100" HorizontalAlignment="Left" Click="OpenActors"/>

<Button Width="100" HorizontalAlignment="Left" Click="OpenCasting">

<TextBlock Text="Таблица кастингов" TextTrimming="CharacterEllipsis"/>

</Button>

<Button Content="Таблица жанров" Width="100" HorizontalAlignment="Left" Click="OpenGenres"/>

<Button Width="100" HorizontalAlignment="Left" Click="OpenDirectors">

<TextBlock Text="Таблица режиссеров" TextTrimming="CharacterEllipsis"/>

</Button>

<Button Width="100" HorizontalAlignment="Left" Click="OpenRates">

<TextBlock Text="Таблица рейтинга" TextTrimming="CharacterEllipsis"/>

</Button>

</DockPanel>

<DataGrid x:Name="Tables" Grid.Row="1">

<DataGrid.Columns>

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Удалить" Name="Delete" Click="DeleteClick"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel Grid.Row="2">

<Button x:Name="Change" Content="Изменить" Width="120" HorizontalAlignment="Left" Margin="0,0,0,0" Click="EditClick"/>

<Button x:Name="Add" Content="Добавить" Width="120" HorizontalAlignment="Left" Margin="70,0,0,0" Click="AddClick"/>

<Button x:Name="Exit" Content="Выйти" Width="120" HorizontalAlignment="Left" Margin="370,0,0,0" Click="ExitClick"/>

</DockPanel>

</Grid>

</Window>

using Kursach;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace WpfApp1

{

public partial class WindowAdminApp : Window

{

List<Film> films;

List<Reviewer> reviewers;

List<Actor> actors;

List<Administrator> administrators;

List<Casting> castings;

List<Director> directors;

List<Rating> ratings;

List<Genre> genres;

public WindowAdminApp()

{

InitializeComponent();

}

private void OpenFilms(object sender, RoutedEventArgs e)

{

films = CinemaEntities.GetContext().Film.ToList();

Tables.ItemsSource = films;

}

private void OpenUsers(object sender, RoutedEventArgs e)

{

reviewers = CinemaEntities.GetContext().Reviewer.ToList();

Tables.ItemsSource = reviewers;

}

private void OpenAdmins(object sender, RoutedEventArgs e)

{

administrators = CinemaEntities.GetContext().Administrator.ToList();

Tables.ItemsSource = administrators;

}

private void OpenActors(object sender, RoutedEventArgs e)

{

actors = CinemaEntities.GetContext().Actor.ToList();

Tables.ItemsSource = actors;

}

private void OpenCasting(object sender, RoutedEventArgs e)

{

castings = CinemaEntities.GetContext().Casting.ToList();

Tables.ItemsSource = castings;

}

private void OpenGenres(object sender, RoutedEventArgs e)

{

genres = CinemaEntities.GetContext().Genre.ToList();

Tables.ItemsSource = genres;

}

private void OpenDirectors(object sender, RoutedEventArgs e)

{

directors = CinemaEntities.GetContext().Director.ToList();

Tables.ItemsSource = directors;

}

private void OpenRates(object sender, RoutedEventArgs e)

{

ratings = CinemaEntities.GetContext().Rating.ToList();

Tables.ItemsSource = ratings;

}

private void DeleteClick(object sender, RoutedEventArgs e)

{

if((sender as Button).DataContext is Film film)

{

CinemaEntities.GetContext().Film.Remove(film);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Film.ToList(); ;

}

else if ((sender as Button).DataContext is Reviewer reviewer)

{

CinemaEntities.GetContext().Reviewer.Remove(reviewer);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Reviewer.ToList(); ;

}

else if((sender as Button).DataContext is Administrator administrator)

{

CinemaEntities.GetContext().Administrator.Remove(administrator);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Administrator.ToList(); ;

}

else if((sender as Button).DataContext is Actor actor)

{

CinemaEntities.GetContext().Actor.Remove(actor);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Actor.ToList(); ;

}

else if((sender as Button).DataContext is Casting casting)

{

CinemaEntities.GetContext().Casting.Remove(casting);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Casting.ToList(); ;

}

else if((sender as Button).DataContext is Director director)

{

CinemaEntities.GetContext().Director.Remove(director);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Director.ToList(); ;

}

else if((sender as Button).DataContext is Genre genre)

{

CinemaEntities.GetContext().Genre.Remove(genre);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Genre.ToList(); ;

}

else if((sender as Button).DataContext is Rating rating)

{

CinemaEntities.GetContext().Rating.Remove(rating);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Rating.ToList(); ;

}

Tables.Items.Refresh();

}

private void AddClick(object sender, RoutedEventArgs e)

{

if ((sender as Button).DataContext is Film film)

{

CinemaEntities.GetContext().Film.Add(film);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Film.ToList();

}

else if ((sender as Button).DataContext is Reviewer reviewer)

{

CinemaEntities.GetContext().Reviewer.Add(reviewer);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Reviewer.ToList();

}

else if ((sender as Button).DataContext is Administrator administrator)

{

CinemaEntities.GetContext().Administrator.Add(administrator);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Administrator.ToList();

}

else if ((sender as Button).DataContext is Actor actor)

{

CinemaEntities.GetContext().Actor.Add(actor);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Actor.ToList();

}

else if ((sender as Button).DataContext is Casting casting)

{

CinemaEntities.GetContext().Casting.Add(casting);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Casting.ToList();

}

else if ((sender as Button).DataContext is Director director)

{

CinemaEntities.GetContext().Director.Add(director);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Director.ToList();

}

else if ((sender as Button).DataContext is Rating rating)

{

CinemaEntities.GetContext().Rating.Add(rating);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Rating.ToList();

}

else if ((sender as Button).DataContext is Genre genre)

{

CinemaEntities.GetContext().Genre.Add(genre);

CinemaEntities.GetContext().SaveChanges();

Tables.ItemsSource = CinemaEntities.GetContext().Genre.ToList();

}

CinemaEntities.GetContext().SaveChanges();

Tables.Items.Refresh();

}

private void EditClick(object sender, RoutedEventArgs e)

{

if ((sender as Button).DataContext is Film)

{

Tables.ItemsSource = CinemaEntities.GetContext().Film.ToList();

}

else if ((sender as Button).DataContext is Reviewer)

{

Tables.ItemsSource = CinemaEntities.GetContext().Reviewer.ToList();

}

else if ((sender as Button).DataContext is Administrator)

{

Tables.ItemsSource = CinemaEntities.GetContext().Administrator.ToList();

}

else if ((sender as Button).DataContext is Actor)

{

Tables.ItemsSource = CinemaEntities.GetContext().Actor.ToList();

}

else if ((sender as Button).DataContext is Casting)

{

Tables.ItemsSource = CinemaEntities.GetContext().Casting.ToList();

}

else if ((sender as Button).DataContext is Director)

{

Tables.ItemsSource = CinemaEntities.GetContext().Director.ToList();

}

else if ((sender as Button).DataContext is Rating)

{

Tables.ItemsSource = CinemaEntities.GetContext().Rating.ToList();

}

else if ((sender as Button).DataContext is Genre)

{

Tables.ItemsSource = CinemaEntities.GetContext().Genre.ToList();

}

CinemaEntities.GetContext().SaveChanges();

Tables.Items.Refresh();

}

private void ExitClick(object sender, RoutedEventArgs e)

{

Enter enter = new Enter();

this.Close();

enter.Show();

}

}

}

<Window x:Class="Kursach.WindowUserApp"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp1"

xmlns:vm="clr-namespace:Kursach"

xmlns:sys="clr-namespace:System.Runtime;assembly=mscorlib"

mc:Ignorable="d"

Title="Кино-сервис" Height="650" Width="1000" ResizeMode="NoResize">

<Window.Resources>

<Style x:Key="style">

<Setter Property="GridViewColumnHeader.Visibility" Value="Collapsed"/>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#b48764"/>

<GradientStop Color="#887359" Offset="1"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<DockPanel Grid.Row="0" Margin="10">

<TextBlock Text="Поиск фильма по названию/имени режиссера/жанру:" Width="520" TextAlignment="Center" FontSize="28" FontFamily="Bahnschrift Light Condensed" Foreground="White"/>

<TextBox x:Name="Searcher" TextChanged="Searcher\_TextChanged" TextAlignment="Left" Background="{x:Null}" Foreground="White" BorderThickness="1" FontFamily="Bahnschrift Light Condensed" FontSize="28" Width="410"/>

</DockPanel>

<ListView BorderThickness="0" Grid.Row="1" Background="{x:Null}" x:Name="FilmsShow" ScrollViewer.CanContentScroll="False">

<!--<ListView.ItemContainerStyle>

<Style TargetType="ListViewItem">

<Setter Property="Selector.IsEnabled" Value="False"/>

</Style>

</ListView.ItemContainerStyle>-->

<ListView.View>

<GridView ColumnHeaderContainerStyle="{StaticResource style}">

<GridView.Columns>

<GridViewColumn>

<GridViewColumn.CellTemplate>

<DataTemplate>

<DockPanel>

<Border Height="250" Width="200">

<Image Source ="{Binding Path = Posters}"/>

</Border>

</DockPanel>

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn>

<GridViewColumn.CellTemplate>

<DataTemplate>

<StackPanel x:Name="Discription">

<TextBlock Text="{Binding Path = NameFilm, StringFormat='Название: {0}'}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock Text="{Binding Path = Director, StringFormat='Режиссер: {0}'}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock Text="{Binding Path = ReleaseDate, StringFormat='Дата выхода: {0:D}', ConverterCulture=ru}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock Text=" Жанр:" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<ItemsControl ItemsSource="{Binding Path = Genre}" Height="Auto" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding Path = NameGenre, StringFormat='{}{0} '}"/>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

<TextBlock Text=" Актеры:" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

<ItemsControl ItemsSource="{Binding Path = Casting}">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Vertical"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding Path=Actor.LastName, StringFormat='{}{0}'}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock Text="{Binding Path=Actor.FirstName, StringFormat='{}{0} - '}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="0, 5, 10, 0"/>

<TextBlock Text="{Binding Path=RoleActor}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="0, 5, 10, 0"/>

</StackPanel>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

<TextBlock Text="{Binding Path = Duration, StringFormat='Продолжительность: {0}'}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock Text="{Binding Path = Country, StringFormat='Страна: {0}'}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20" Margin="30, 5, 10, 0"/>

<TextBlock x:Name="ScoreShow" Text="{Binding Path = rateScore}" Height="50" Width="50" Margin="500,20, 0, 0" Background="{x:Null}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="30"/>

<Button Content="Поставить оценку" Height="25" VerticalAlignment="Top" Width="150" Click="RateClick" Margin="400,0, 0, 0" BorderBrush="Black" Background="{x:Null}" Foreground="White" FontFamily="Bahnschrift Light Condensed" FontSize="20"/>

</StackPanel>

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

</GridView.Columns>

</GridView>

</ListView.View>

</ListView>

<DockPanel Grid.Row="2">

<Button Content="Выйти" HorizontalAlignment="Right" Margin="0,7,30,0" Height="34" Grid.Row="1" VerticalAlignment="Top" Width="125" Click="ExitClick"/>

</DockPanel>

</Grid>

</Window>

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using WpfApp1;

namespace Kursach

{

public partial class WindowUserApp : Window

{

List<Film> films;

public WindowUserApp()

{

InitializeComponent();

CinemaEntities db = new CinemaEntities();

FilmsShow.ItemsSource = db.Film.ToList();

}

private void ExitClick(object sender, RoutedEventArgs e)

{

Enter enter = new Enter();

this.Close();

enter.Show();

}

private void RateClick(object sender, RoutedEventArgs e)

{

int idFilm = ((sender as Button).DataContext as Film).Film\_ID;

Rate rate = new Rate(idFilm);

rate.ShowDialog();

}

private void Searcher\_TextChanged(object sender, TextChangedEventArgs e)

{

string Text = Searcher.Text.ToLower().Trim();

FilmsShow.ItemsSource = films.Where(x=> Text.Split(' ').All(t =>

x.Director.LastName.ToLower().Contains(t)

|| x.Director.FirstName.ToLower().Contains(t))

|| x.Genre.Any(y => y.NameGenre.ToLower().Contains(Text))

|| x.NameFilm.ToLower().Contains(Text));

}

}

}