СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой данной курсовой работы является «Проектирование и разработка реляционной базы данных информационной системы для «книоманов»».

На сегодняшний день базы данных занимают одно из первых мест среди различных структур хранения данных. Базы данных используются в приложениях, написанных для облегчения работы мелких и крупных предприятий, учреждений. База данных является эффективно организованной структурой хранения данных, которая предоставляет пользователю значительные возможности при работе с информацией, находящейся в ней.

В настоящее время многие люди увлекаются различными фильмами: ходят в кино, смотрят их по телевизору или в интернете. Однако хороших сайтов, на которых вы могли бы отслеживать выход новых фильмов или же находить самые популярные не так уж и много. Кроме того, многим пользователям хотелось бы иметь удобный способ отслеживать просмотренные ими фильмы, а также делить своим мнением с другими.

Целью данной курсовой работы является создание системы, которая помогла бы пользователям с этими задачами, а пользователь имел к ней доступ в любой момент времени. Такая система должна облегчить поиск интересных фильмов, а также поможет не забыть уже просмотренные фильмы.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

Инфологическая модель данных – описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

1.1 Анализ предметной области и выявление необходимого набора сущностей

Перед началом разработки базы данных, необходимо определить основные цели, задачи и правила для решаемой проблемы, после чего приступать к проектированию. Поэтому сформулируем краткое описание поставленной задачи.

Наименование задачи – разработка приложения для поиска фильмов и создания собственного списка фильмов.

Функции пользователя:

- Поиск фильмов на сайте.
- Выставление оценок.
- Комментирование фильмов.
- Создание собственной базы фильмов.
- Создание заметок к фильмам.

Основные бизнес-правила:

- Аккаунты пользователей должны быть защищены от взлома.
- Пользователь может восстановить доступ к своему аккаунту.
- Только пользователь может управлять своими данными.
- Комментарии могут оставлять только пользователи подтвердившие свои аккаунты.

Список фильмов пользователя.

Для построения информационной системы требуется для начала выделить необходимы набор сущностей, которые описывают эту систему. Данный набор должен удовлетворять всем условиям на проектирование системы.

Определим минимальный набор сущностей, необходимый для проектирования информационной системы для «книоманов». Для определения первичного набора сущностей будет проведён анализ технического задания и предметной области.

Для подобного приложения необходимо описание такой сущности как фильм. В данной сущности должны быть заложены описательные характеристики фильма, однозначно идентифицирующие его.

Связанные сущности с фильмом это – жанры, участники, рейтинг и страны.

Кроме того, не менее важной сущностью является пользователь. Для них есть две дополнительные сущности — это оценки и фильмы пользователей.

1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Атрибут – поименованная характеристика сущности.

Атрибутом сущности является любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей.

Для каждой сущности, выделенной в пункте 1.1. необходимо определить атрибуты.

1 Сущность – «фильм»:

Атрибуты: [Name], [Budget], [Time], [CreateYearId], [MPAAId], [Discription], [RetingId], [Photo].

2 Сущность – «жанр»:

Атрибуты: [GenersName].

3 Сущность – «участник»:

Атрибуты: [Name], [Href].

4 Сущность – «рейтинг»:

Атрибуты: [UserStar], [AllStar].

5 Сущность – «страны»:

Атрибуты: [Name].

6 Сущность – «пользователь»:

Атрибуты: [UserName], [NormalizedUserName], [Email], [NormalizedEmail], [EmailConfirmed], [PasswordHash], [SecurityStamp], [ConcurrencyStamp], [PhoneNumber], [PhoneNumberConfirmed], [TwoFactorEnabled], [LockoutEnd], [LockoutEnabled], [AccessFailedCount], [Year], [Photo], [StatusComentId].

7 Сущность – «оценки»:

Атрибуты: [Star].

8 Сущность – «фильмы пользователей»:

Атрибуты: [Discription].

1.3 Определение связей между объектами

Следующим этапом в проектировании инфологической модели является установление связи между сущностями.

Связь – это ассоциирование двух или более сущностей. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается имя конца связи, степень конца связи (сколько экземпляров данной сущности связывается), обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Для реализации информационной системы «киноианов» необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему в совокупности и определить взаимное влияние объектов, составляющих систему.

1. [StatusComent] – [AspNetUsers]: (1:M).
2. [AspNetRoles] – [AspNetUserRoles]: (1:M).
3. [AspNetUsers] – [AspNetUserRoles]: (1:M).
4. [Films] – [Coment]: (1:M).
5. [AspNetUsers] – [Coment]: (1:M).
6. [MPAA] – [Films]: (1:M).

7. [Reting] – [Films]: (1:M).
8. [CreateYear] – [Films]: (1:M).
9. [Films] – [FilmsToCountrys]: (1:M).
10. [Country] – [FilmsToCountrys]: (1:M).
11. [Films] – [FilmsToGenres]: (1:M).
12. [Genre] – [FilmsToGenres]: (1:M).
13. [Films] – [ParticipantsBuffer]: (1:M).
14. [Participants] – [ParticipantsBuffer]: (1:M).
15. [StatusParticipants] – [ParticipantsBuffer]: (1:M).
16. [Films] – [UserFilm]: (1:M).
17. [StatusView] – [UserFilm]: (1:M).
18. [UserFilm] – [UserStar]: (1:M).
19. [AspNetUsers] – [UserStar]: (1:M).

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model).

По правилам построения ER-диаграмм в нотации Crow's Foot (рус. «воронья лапка») сущность изображается в виде прямоугольника. Связь изображается линией, которая связывает две сущности, участвующие в отношении. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Атрибуты сущности записываются внутри прямоугольника, изображающего сущность.

На основе проведенного проектирования, в частности на основе инфологической схемы, приведенной на рис. 1.1, получим ER-диаграмму, проектируемой базы данных, представленную в приложении А на рис. А.2.

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 Построение набора необходимых отношений базы данных

Для построения схемы реляционной базы данных необходимо определить совокупность отношений, составляющих базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

На основе полученной в первом пункте концептуальной модели можно определить набор необходимых отношений базы данных.

На рисунке 2.1 представлены отношения для базы данных информационной системы университета.

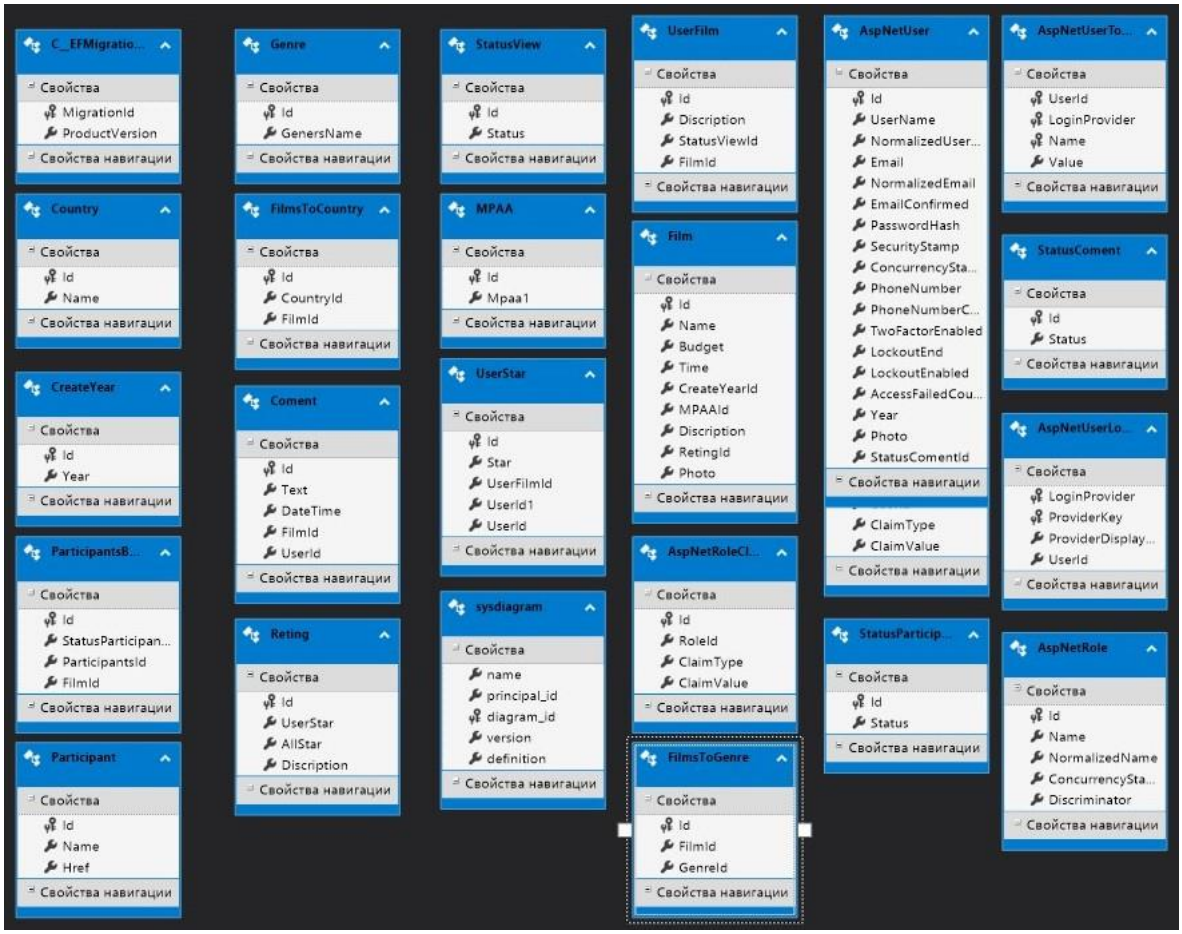

Рисунок 2.1 – Набор необходимых отношений базы данных

Схема реляционной базы данных представлена в приложении В.

2.2 Задание первичных и внешних ключейопределенных отношений

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам. Каждое отношение должно обладать хотя бы одним ключом. В таблице 2.1 определены первичные и внешние ключи для отношений [4].

Таблица 2.1 – Первичные и внешние ключи отношений

№ п/п	**Название таблицы**	**Первичный ключ**	**Внешние ключи**
1	AspNetRoles	[Id]	
2	AspNetUserRoles	[Id]	[UserId] [RoleId]
3	AspNetUsers	[Id]	[StatusComentId]
4	Coment	[Id]	[FilmId] [UserId]
5	Country	[Id]	
6	CreateYear	[Id]	
7	Films	[Id]	[RetingId] [MPAAId] [CreateYearId]
8	FilmsToCountrys	[Id]	[CountryId] [FilmId]
9	FilmsToGenres	[Id]	[FilmId] [GenreId]
10	Genre	[Id]	
11	MPAA	[Id]	
12	Participants	[Id]	
13	ParticipantsBuffer	[Id]	[StatusParticipantsId] [ParticipantsId] [FilmId]
14	Reting	[Id]	

Продолжение таблицы 2.1

№ п/п	Название таблицы	Первичный ключ	Внешние ключи
15	StatusComent	[Id]	
16	StatusParticipants	[Id]	
17	StatusView	[Id]	
18	UserFilm	[Id]	[StatusViewId] [FilmId]
19	UserStar	[Id]	[UserFilmId] [UserId]

2.3 Приведение отношения бд к третьей нормальной форме

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации [4].

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в 1НФ. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым, исключив из базы данных избыточность и аномалии обновления.

Определение 3НФ – неключевые атрибуты не должны определять другие неключевые атрибуты.

В спроектированной базе данных почти все отношения находятся в третьей нормальной форме, за исключением таблицы Participants, ее целесообразно оставить во превой нормальной форме.

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Ограничение целостности для внешних ключей состоит в том, что значение внешнего ключа должно быть равным значению первичного ключа цели; либо быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.

Условиями целостности называется набор правил, используемых для поддержания допустимых межтабличных связей и запрета на случайное изменение или удаление связанных данных. Следует устанавливать целостность данных только при выполнении следующих условий: связываемое поле из главной таблицы является полем первичного ключа и имеет уникальный индекс, связанные поля имеют один и тот же тип данных.

Для автоматического обновления связанных полей (удаления записей) при обновлении (удалении) в главной таблице, следует устанавливать обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Ограничение целостности, накладываемые на разрабатываемую систему:

- ключевое поле отношения должно быть уникальным;
- внешний ключ должен быть повторяющимся, то есть соответствовать уникальному ключу в своем отношении.

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, а также чтобы были заполнены все обязательные поля в таблицах, т.е. те поля, которые не могут содержать значения NULL.

2.5 Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении В. Полученная ER-диаграмма построена по методу Crow's Foot (рус. «воронья лапка»). Средства моделирования Crow's Foot специально разработаны для построения реляционных информационных систем. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются знаком ключа, внешние ключи обозначаются знаком ссылки.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных MS SQL Express. Это обусловлено тем, что, данная СУБД имеет большую функциональность, множество средств для поддержки и работы с ней, развитую инфраструктуру для интеграции баз данных в пользовательские приложения.

В создаваемой базе данных будут использоваться следующие типы данных:

- INT – Целочисленный тип. Размер – 4 байта.
- NVARCHAR – Строковый тип переменной длины.
- VARBINARY – Двоичные данные.
- BIT – Битовый тип. Используется как логический тип.
- DATE – Тип, определяющий дату.
- REAL – Число с плавающей точкой. Размер – 4 байт.
- DECIMAL – Число с плавающей точкой. Размер – 8 байт.

Опишем все таблицы, которые будут созданы в базе данных.

Таблица [AspNetUsers] содержит список всех пользоваетлей. Ее структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы [AspNetUsers]

Имя атрибута	**Тип**	**Описание**
[Id]	NVARCHAR (450)	Идентификатор пользователя
[UserName]	NVARCHAR (256)	Имя пользователя
[NormalizedUserName]	NVARCHAR (256)	Имя пользователя
[Email]	NVARCHAR (256)	Почта
[NormalizedEmail]	NVARCHAR (256)	Почта
[EmailConfirmed]	BIT	Подтверждение почты
[PasswordHash]	NVARCHAR (MAX)	Хэш пароля
[SecurityStamp]	NVARCHAR (MAX)	Токен
[StatusComentId]	INT	Идентификатор статуса комментариев
[LockoutEnd]	DATETIMEOFFSET (7)	Конец действия токена

Продолжение таблицы 3.1

Имя атрибута	**Тип**	**Описание**
[LockoutEnabled]	BIT	Авторизация пользователя
[AccessFailedCount]	INT	Колличество неудачных попыток входа
[Year]	DATETIME2 (7)	Год рождения
[Photo]	VARBINARY (MAX)	Фото

Таблица [AspNetRoleClaims] содержит информацию об утверднениях для этой роли. Ее структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы [AspNetRoleClaims]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[RoleId]	NVARCHAR (450)	Идентификатор роли
[ClaimType]	NVARCHAR (MAX)	Тип утверждения
[ClaimValue]	NVARCHAR (MAX)	Утверждение

Таблица [AspNetRoles] содержит информацию о ролях. Ее структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы [AspNetRoles]

Имя атрибута	**Тип**	**Описание**
[Id]	NVARCHAR (450)	Идентификатор для данной таблицы.
[Name]	NVARCHAR (256)	Название роли
[NormalizedName]	NVARCHAR (256)	Название роли
[ConcurrencyStamp]	NVARCHAR (MAX)	Уникальная печать
[Discriminator]	NCHAR (256)	Класс упровляющий ролями

Таблица [AspNetUserClaims] содержит утверждения для пользователей. Ее структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы [AspNetUserClaims]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[UserId]	NVARCHAR (450)	Идентификатор пользователя
[ClaimType]	NVARCHAR (MAX)	Тип утвержединя
[ClaimValue]	NVARCHAR (MAX)	Утверждение

Таблица [Coment] содержит информацию комментария к фильмам. Ее структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы [Coment]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Text]	NVARCHAR (MAX)	Текст комментария
[DateTime]	DATETIME2 (7)	Время добавления
[FilmId]	INT	Идентификатор фильма
[UserId]	NVARCHAR (450)	Идентификатор пользоваетля

Таблица [Country] содержит информацию о странах. Ее структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы [Country]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Name]	NVARCHAR (MAX)	Название страны

Таблица [AspNetUserRoles] служит для соединения пользователей с их ролями. Ее структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы [AspNetUserRoles]

Имя атрибута	**Тип**	**Описание**
[UserId]	NVARCHAR (450)	Идентификатор пользователя
[RoleId]	NVARCHAR (450)	Идентификатор роли

Таблица [CreateYear] служит для хранения информации о годах создания фильмов. Ее структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы [CreateYear]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Year]	INT	Год

Таблица [Films] служит для хранения информации о фильмах. Ее структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы [Films]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Name]	NVARCHAR (MAX)	Название фильма
[Budget]	NVARCHAR (MAX)	Бюджет
[Time]	DATETIME2 (7)	Продолжительность фильма
[CreateYearId]	INT	Идентификатор года
[MPAAId]	INT	Идентификатор возростного рейтинга
[Discription]	NVARCHAR (MAX)	Описание
[RetingId]	INT	Идентификатор рейтинга
[Photo]	NVARCHAR (MAX)	Фото

Таблица [Genre] служит для хранения типов гражданств. Ее структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы [Genre]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[GenersName]	NVARCHAR (256)	Название жанра

Таблица [FilmsToCountrys] служит для связи фильмов и стран. Ее структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы [FilmsToCountrys]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[CountryId]	INT	Идентификатор страны
[FilmId]	INT	Идентификатор фильма

Таблица [FilmsToGenres] служит для связи фильмов и жанров. Ее структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы [FilmsToGenres]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[FilmId]	INT	Идентификатор фильма
[GenreId]	INT	Идентификатор жанра

Таблица [MPAA] служит для хранения. Ее структура приведена в таблице 3.13.

Таблица 3.13 – Характеристика атрибутов таблицы [MPAA]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Mpaa]	NVARCHAR (MAX)	Возростной рейтинг

Таблица [Participants] служит для хранения информации об участниках. Ее структура приведена в таблице 3.14.

Таблица 3.14 – Характеристика атрибутов таблицы [Participants]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы.
[Name]	NVARCHAR (MAX)	ФИО Участинка
[Href]	NVARCHAR (MAX)	Ссылка

Таблица [ParticipantsBuffer] служит для связи фильмов участиников и их статусов. Ее структура приведена в таблице 3.15.

Таблица 3.15 – Характеристика атрибутов таблицы [ParticipantsBuffer]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[StatusParticipantsId]	INT	Идентификатор статуса участника
[ParticipantsId]	INT	Идентификатор участника
[FilmId]	INT	Идентификатор фильма

Таблица [Reting] служит для хранения информации рейтинге фильма. Ее структура приведена в таблице 3.16.

Таблица 3.16 – Характеристика атрибутов таблицы [Reting]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[UserStar]	REAL	Оценка пользователей
[AllStar]	REAL	Оценка в мире
[Discription]	NVARCHAR (MAX)	Описание

Таблица [StatusComent] служит для хранения информации о возможности комментирования. Ее структура приведена в таблице 3.17.

Таблица 3.17 – Характеристика атрибутов таблицы [StatusComent]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[Status]	NVARCHAR(MAX)	Статус

Таблица [StatusParticipants] служит для хранения информации о статусах участников. Ее структура приведена в таблице 3.18.

Таблица 3.18 – Характеристика атрибутов таблицы [StatusParticipants]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[Status]	NVARCHAR (MAX)	статус

Таблица [StatusView] служит для хранения информации о статусах просмотра фильмов. Ее структура приведена в таблице 3.19.

Таблица 3.19 – Характеристика атрибутов таблицы [StatusView]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[Status]	NVARCHAR (MAX)	Статус

Таблица [UserFilm] служит для хранения информации о фильмах которые понравелись пользоваетлю. Ее структура приведена в таблице 3.20.

Таблица 3.20 – Характеристика атрибутов таблицы [UserFilm]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[Discription]	NVARCHAR (MAX)	Заметка
[StatusViewId]	INT	Статус просмотра
[FilmId]	INT	Идентификатор фильма

Таблица [UserStar] служит для хранения информации об оценках пользоваетлей. Ее структура приведена в таблице 3.21.

Таблица 3.21 – Характеристика атрибутов таблицы [UserStar]

Имя атрибута	**Тип**	**Описание**
[Id]	INT	Идентификатор для данной таблицы
[Star]	REAL	Оценка
[UserFilmId]	INT	Идентификатор фильма
[UserId]	NVARCHAR (450)	Идентификатор пользоваетля

Для подсчета среднего рейтинга пользователей для фильма был релаизован триггер. При изменении оценки или добавлении новой он пересчитывает среднюю оценку для фильма. Далее приводится реализация данного триггера:

```
CREATE TRIGGER [CountReting]
    ON [dbo].[UserStar]
    FOR INSERT, UPDATE
    AS
    BEGIN
        SET NOCOUNT ON
        UPDATE dbo.Reting SET UserStar = (SELECT AVG(star.Star)
FROM dbo.UserStar star, dbo.UserFilm fil, inserted ins
```

```
        WHERE fil.FilmId = (SELECT FilmId FROM dbo.UserFilm fil,
inserted ins where ins.UserFilmId = fil.Id))
        where Reting.Id = (SELECT film.RetingId FROM dbo.Films film,
dbo.UserFilm fil, inserted ins where fil.FilmId = film.Id AND ins.UserFilmId =
fil.Id)
    END
```

4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

1 Запрос, возвращающий список всех фильмов.

```
SELECT [e].[Id], [e].[Budget], [e].[CreateYearId], [e].[Discription],
[e].[MPAAId], [e].[Name], [e].[Photo], [e].[RetingId], [e].[Time], [e.Year].[Id],
[e.Year].[Year], [e.Reting].[Id], [e.Reting].[AllStar], [e.Reting].[Discription],
[e.Reting].[UserStar]
    FROM [Films] AS [e]
    INNER JOIN [CreateYear] AS [e.Year] ON [e].[CreateYearId] =
[e.Year].[Id]
    INNER JOIN [Reting] AS [e.Reting] ON [e].[RetingId] = [e.Reting].[Id]
    ORDER BY [e].[Id]
```

2 Запрос, возвращающий информацию о пользователе.

```
SELECT TOP(1) [e].[Id], [e].[AccessFailedCount], [e].[ConcurrencyStamp],
[e].[Email], [e].[EmailConfirmed], [e].[LockoutEnabled], [e].[LockoutEnd],
[e].[NormalizedEmail], [e].[NormalizedUserName], [e].[PasswordHash],
[e].[PhoneNumber], [e].[PhoneNumberConfirmed], [e].[Photo], [e].[SecurityStamp],
[e].[StatusComentId], [e].[TwoFactorEnabled], [e].[UserName], [e].[Year]
    FROM [AspNetUsers] AS [e]
    WHERE [e].[UserName] = @_Name_0
```

3 Запрос, выводящий список самых популярных фильмов на сайте.

```
SELECT [e].[Id], [e].[Budget], [e].[CreateYearId], [e].[Discription],
[e].[MPAAId], [e].[Name], [e].[Photo], [e].[RetingId], [e].[Time], [e.Year].[Id],
[e.Year].[Year], [e.Reting].[Id], [e.Reting].[AllStar], [e.Reting].[Discription],
[e.Reting].[UserStar]
    FROM [Films] AS [e]
    INNER JOIN [CreateYear] AS [e.Year] ON [e].[CreateYearId] =
[e.Year].[Id]
    INNER JOIN [Reting] AS [e.Reting] ON [e].[RetingId] = [e.Reting].[Id]
    ORDER BY (
      SELECT COUNT(*)
      FROM [UserFilm] AS [u]
      WHERE [e].[Id] = [u].[FilmId]
```

```
) DESC, [e].[Id]
```

4 Запрос, выводящий список самых популярных фильмов в мире.

```
SELECT [E].[ID], [E].[BUDGET], [E].[CREATEYEARID],
[E].[DISCRIPTION], [E].[MPAAID], [E].[NAME], [E].[PHOTO], [E].[RETINGID],
[E].[TIME], [E.YEAR].[ID], [E.YEAR].[YEAR], [E.RETING].[ID],
[E.RETING].[ALLSTAR], [E.RETING].[DISCRIPTION],
[E.RETING].[USERSTAR]
    FROM [FILMS] AS [E]
    INNER JOIN [CREATEYEAR] AS [E.YEAR] ON
[E].[CREATEYEARID] = [E.YEAR].[ID]
    INNER JOIN [RETING] AS [E.RETING] ON [E].[RETINGID] =
[E.RETING].[ID]
    ORDER BY [E.RETING].[ALLSTAR] DESC, [E].[ID]
```

5 Запрос, возвращающий список фильмов по жанру.

```
SELECT [E.FILMSTOGENRES].[ID], [E.FILMSTOGENRES].[FILMID],
[E.FILMSTOGENRES].[GENREID], [F.GENRE].[ID],
[F.GENRE].[GENERSNAME]
    FROM [FILMSTOGENRES] AS [E.FILMSTOGENRES]
    INNER JOIN [GENRE] AS [F.GENRE] ON
[E.FILMSTOGENRES].[GENREID] = [F.GENRE].[ID]
    INNER JOIN (
      SELECT DISTINCT [E0].[ID]
      FROM [FILMS] AS [E0]
      INNER JOIN [CREATEYEAR] AS [E.YEAR0] ON
[E0].[CREATEYEARID] = [E.YEAR0].[ID]
      INNER JOIN [RETING] AS [E.RETING0] ON [E0].[RETINGID] =
[E.RETING0].[ID]
    ) AS [T] ON [E.FILMSTOGENRES].[FILMID] = [T].[ID]
    ORDER BY [T].[ID]
```

6 Запрос, выводящий список фильмов с конкретным годом.

```
SELECT [E].[ID], [E].[BUDGET], [E].[CREATEYEARID],
[E].[DISCRIPTION], [E].[MPAAID], [E].[NAME], [E].[PHOTO], [E].[RETINGID],
[E].[TIME], [E.RETING].[ID], [E.RETING].[ALLSTAR],
[E.RETING].[DISCRIPTION], [E.RETING].[USERSTAR], [E.YEAR].[ID],
[E.YEAR].[YEAR]
```

```
        FROM [FILMS] AS [E]
        INNER JOIN [RETING] AS [E.RETING] ON [E].[RETINGID] =
[E.RETING].[ID]
        INNER JOIN [CREATEYEAR] AS [E.YEAR] ON
[E].[CREATEYEARID] = [E.YEAR].[ID]
        WHERE CONVERT(VARCHAR(11), [E.YEAR].[YEAR]) =
@__TOSTRING_0
        ORDER BY [E].[ID]
```

7 Запрос, выводящий список фильмов пользователя.

```
    SELECT [E.USERSTARS].[ID], [E.USERSTARS].[STAR],
[E.USERSTARS].[USERFILMID], [E.USERSTARS].[USERID],
[E.USERSTARS].[USERID1], [U.USERFILM].[ID],
[U.USERFILM].[DISCRIPTION], [U.USERFILM].[FILMID],
[U.USERFILM].[STATUSVIEWID], [U.USERFILM.STATUSVIEW].[ID],
[U.USERFILM.STATUSVIEW].[STATUS]
        FROM [USERSTAR] AS [E.USERSTARS]
        INNER JOIN [USERFILM] AS [U.USERFILM] ON
[E.USERSTARS].[USERFILMID] = [U.USERFILM].[ID]
        INNER JOIN [STATUSVIEW] AS [U.USERFILM.STATUSVIEW] ON
[U.USERFILM].[STATUSVIEWID] = [U.USERFILM.STATUSVIEW].[ID]
        INNER JOIN (
          SELECT TOP(1) [E0].[ID]
          FROM [ASPNETUSERS] AS [E0]
          WHERE [E0].[USERNAME] =
@__HTTPCONTEXT_USER_IDENTITY_NAME_0
          ORDER BY [E0].[ID]
        ) AS [T] ON [E.USERSTARS].[USERID] = [T].[ID]
        ORDER BY [T].[ID], [U.USERFILM.STATUSVIEW].[ID]
```

8 Запрос, выводящий список фильмов пользователя с конкретным статусом.

```
    SELECT [E.USERSTARS].[ID], [E.USERSTARS].[STAR],
[E.USERSTARS].[USERFILMID], [E.USERSTARS].[USERID],
[E.USERSTARS].[USERID1], [U.USERFILM].[ID],
[U.USERFILM].[DISCRIPTION], [U.USERFILM].[FILMID],
[U.USERFILM].[STATUSVIEWID], [U.USERFILM.STATUSVIEW].[ID],
[U.USERFILM.STATUSVIEW].[STATUS]
```

```
        FROM [USERSTAR] AS [E.USERSTARS]
        INNER JOIN [USERFILM] AS [U.USERFILM] ON
[E.USERSTARS].[USERFILMID] = [U.USERFILM].[ID]
        INNER JOIN [STATUSVIEW] AS [U.USERFILM.STATUSVIEW] ON
[U.USERFILM].[STATUSVIEWID] = [U.USERFILM.STATUSVIEW].[ID]
        INNER JOIN (
          SELECT TOP(1) [E0].[ID]
          FROM [ASPNETUSERS] AS [E0]
          WHERE [E0].[USERNAME] =
@__HTTPCONTEXT_USER_IDENTITY_NAME_0
          ORDER BY [E0].[ID]
        ) AS [T] ON [E.USERSTARS].[USERID] = [T].[ID]
        ORDER BY [T].[ID], [U.USERFILM.STATUSVIEW].[ID]
```

9 Запрос, изменяющий пароль пользователя.

```
        UPDATE [ASPNETUSERS] SET [ACCESSFAILEDCOUNT] = @P0,
[CONCURRENCYSTAMP] = @P1, [EMAIL] = @P2, [EMAILCONFIRMED] =
@P3, [LOCKOUTENABLED] = @P4, [LOCKOUTEND] = @P5,
[NORMALIZEDEMAIL] = @P6, [NORMALIZEDUSERNAME] = @P7,
[PASSWORDHASH] = @P8, [PHONENUMBER] = @P9,
[PHONENUMBERCONFIRMED] = @P10, [PHOTO] = @P11,
[SECURITYSTAMP] = @P12, [STATUSCOMENTID] = @P13,
[TWOFACTORENABLED] = @P14, [USERNAME] = @P15, [YEAR] = @P16
        WHERE [ID] = @P17 AND [CONCURRENCYSTAMP] = @P18;
        SELECT @@ROWCOUNT;
```

10 Запрос, выводящий информацию об одном фильме.

```
        SELECT [E.PARTICIPANTSBUFFERS].[ID],
[E.PARTICIPANTSBUFFERS].[FILMID],
[E.PARTICIPANTSBUFFERS].[PARTICIPANTSID],
[E.PARTICIPANTSBUFFERS].[STATUSPARTICIPANTSID],
[P.PARTICIPANT].[ID], [P.PARTICIPANT].[HREF],
[P.PARTICIPANT].[NAME], [P.STATUSPARTICIPANTS].[ID],
[P.STATUSPARTICIPANTS].[STATUS]
        FROM [PARTICIPANTSBUFFER] AS [E.PARTICIPANTSBUFFERS]
        INNER JOIN [PARTICIPANTS] AS [P.PARTICIPANT] ON
[E.PARTICIPANTSBUFFERS].[PARTICIPANTSID] = [P.PARTICIPANT].[ID]
```

```
            INNER JOIN [STATUSPARTICIPANTS] AS
[P.STATUSPARTICIPANTS] ON
[E.PARTICIPANTSBUFFERS].[STATUSPARTICIPANTSID] =
[P.STATUSPARTICIPANTS].[ID]
            INNER JOIN (
               SELECT DISTINCT [T3].*
               FROM (
                  SELECT TOP(1) [E2].[ID]
                  FROM [FILMS] AS [E2]
                  INNER JOIN [CREATEYEAR] AS [E.YEAR2] ON
[E2].[CREATEYEARID] = [E.YEAR2].[ID]
                  INNER JOIN [RETING] AS [E.RETING2] ON [E2].[RETINGID] =
[E.RETING2].[ID]
                  LEFT JOIN [MPAA] AS [E.MPAA2] ON [E2].[MPAAID] =
[E.MPAA2].[ID]
                  WHERE [E2].[ID] = @__TOINT32_0
                  ORDER BY [E2].[ID]
               ) AS [T3]
            ) AS [T4] ON [E.PARTICIPANTSBUFFERS].[FILMID] = [T4].[ID]
            ORDER BY [T4].[ID]
```

5 ВЫБОР И ОБОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Так как задачей является создание именно веб-приложения, для разработки использовались языки разметки HTML и CSS.

HTML (HyperText Markup Language) — язык разметки (маркировки) гипертекста. Гипертекст своим развитием обязан интернету, хоть и созда-вался он совсем не для того. HTML дает возможность производить переход от одной части текста к другой, и, что замечательно, эти части могут хра-ниться на совершенно разных компьютерах.

HTML не стоит путать с языками программирования, он создан специально для разметки вэб-страниц. Именно язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе. Важно заметить, что не только различные браузеры, но и различные их версии могут по-разному воспринимать и отображать на экране код. Следовательно, некоторые элементы корректно выглядящие в браузере Opera могут выглядеть иначе в Internet Explorer и других браузерах.

CSS – это формальный язык, служащий для описания оформления внешнего вида документа, созданного с использованием языка разметки (HTML, XHTML, XML). Название происходит от английского Cascading Style Sheets, что означает «каскадные таблицы стилей».

Назначение CSS – отделять то, что задает внешний вид страницы, от ее содержания. Если документ создан только с использованием HTML, то в нем определяется не только каждый элемент, но и способ его отображения (цвет, шрифт, положение блока и т. д.). Если же подключены каскадные таблицы стилей, то HTML описывает только очередность объектов. А за все их свойства отвечает CSS. В HTML достаточно прописывать класс, не перечисляя все стили каждый раз.

Для реализации сервреной части приложения был выбран ASP.NET Core а для хранения данных MS SQL Express.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

1. Создавать веб-приложения и службы, приложения IoT и серверные части для мобильных приложений.
2. Использовать избранные средства разработки в Windows, macOS и Linux.
3. Выполнять развертывания в облаке или локальной среде.

4. Работать в .NET Core или .NET Framework.

Для работы с базой данных был выбран EF core.

Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping - отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц 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 работаем с объектами.

ORM — Object-Relational Mapping или в переводе на русский объектно-реляционное отображение. Это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

В качестве среды разработки была выбрана Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управ-ляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

6.1 Разработка и построение интерфейса

Приложения разделено на две основные части: веб-приложение с которым работает клиент и приложение для администратора написанное с использованием Windows forms.

Главная форма приложения является объектом класса Form1, наследуемый от класса Window, определенного в .NET Framework. Создание всех компонентов формы, в частности главного меню, управляющих элементов и окон сообщений происходит в методе по мере их вызова, соответствующими им конструкторами.

Все основные таблицы для представления данных были выполнены в виде DataGrid, что упрощает понимание и просмотр информации, т.к. она представляется в табличном виде.

Дочерние (вспомогательные) формы выполнены в виде диалоговых окон. Основной упор при проектировании интерфейса приложения был сделан на привлекательность и понятность для конечного пользователя.

При проектировании приложения были учтены все возможные случаи некорректной работы программы, поэтому большинство нештатных ситуаций сопровождается оповещениями с описанием проблемы.

Скриншоты главной и некоторых диалоговых окон представлены в приложении С.

Интерфейс клиентского приложения выполнен с помощью html css. Рендерингом html страниц занимается движок Razor, а для стилизации интерфейса использовался bootstrap.

6.2 Построение главного меню и кнопок панели инструментов

Главное навигационное меню сайта представлено ссылками на различные группы фильмов, а также на личный кабинет пользователя и регистрационную форму. Данные пункты выполнены в виде одной панели, которая реагирует на нажатия пользователя и сопровождает все страницы приложения.

6.3 Выполнение программного кода в среде Microsoft Visual C#

Опишем работу приложения с базой данных. Все необходимые интерфейсы для работы с базами данных находятся в классе DBContext. Подключение к базе данных начинается с формирования строки подключения и последующим созданием контекста на основе данной строки:

```
1.  services.AddDbContext<DBContext>(options =>
2.  options.UseSqlServer(Configuration.GetConnectionString("C
onnection")));
```

Все запросы по работе с базой данных обращаются к контексту и строятся с помощью LINQ to Entity [2]. В следствии чего упрощается построение запросов, вызов процедур и фильтрация результатов. Пример текста запроса, возвращающего полную информацию о фильме выглядит следующим образом:

```
1.  var film = await dBContext.Films
2.  .Include(e => e.FilmsToCountrys)
3.  .ThenInclude(e => e.Country)
4.  .Include(e => e.FilmsToGenres)
5.  .ThenInclude(e => e.Genre)
6.  .Include(e => e.MPAA)
7.  .Include(e => e.Reting)
8.  .Include(e => e.Year)
9.  .Include(e => e.ParticipantsBuffers)
10. .ThenInclude(e => e.StatusParticipants)
11. .Include(e => e.ParticipantsBuffers)
12. .ThenInclude(e => e.Participant)
13. .Where(e => e.Id ==
Convert.ToInt32(RouteData.Values["Id"])).FirstOrDefaultAsync();
```

Результат выполненного запроса представляет собой объект класс Film, который позже можно использовать как модель для страницы.

Также в приложении есть несколько сервисов например SendEmailAsync занимается рассылкой электронных писем.

Хранение настроек приложения обеспечивает файл appsettings.json в нем находятся основные настройки, а также сторка подключения к бд.

ЗАКЛЮЧЕНИЕ

В результате выполненной работы, была создана база данных для «книоманов», а также эффективно работающее с этой базой данных веб-приложение.

Разработанная база данных удовлетворяет всем требованиям, предъявленным в задании, и позволяет без проблем хранить и извлекать требуемую информацию.

Созданное приложение позволяет упростить поиск интересных фильмов, а также создать собственную базу из них.

В приложении реализованы запросы, позволяющие пользователю выбрать всю необходимую информацию по заданным критериям, осуществлять поиск данных и формировать отчеты.

Благодаря разделению приложения на несколько частей была существенно повышена безопасность так как в клиенте для пользователя нет функций по удалению данных или нарушению их целостности.

Разработанная система реагирует на ошибочный ввод данных, а также способна определять возникающие ошибки и уведомлять об этом пользователя, чтобы в любой момент он знал из-за чего или почему произошла ошибка, и оперативно устранил её.

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на языке C#, проектировании баз данных и реализации их в СУБД.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Шилдт Г. Полный справочник по SQL.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2004. – 752 с.: ил.

2 Раттц Д. LINQ язык интегрированных запросов в C# 2008 для профессионалов. Пер. с англ. – М.: Вильямс, 2008. – 645с.: ил.

3 Хернандес М., Вьескас Д. SQL-запросы. Практическое руководство.: Пер. с англ. – М.: Лори, 2003. – 473 с.: ил.

4 Коннолли Т., Бегг К., Базы данных. Проектирование, реализация и сопровождение. Теория и практика.: Пер. с англ. – М.: Вильямс, 2003. – 1500 с.: ил.

5 Jennings R., Professional ADO.NET 3.5 with LINQ and the Entity Framework.: – New York.: Wrox, 2009. – 560 с.: ил.

ПРИЛОЖЕНИЕ Б

(обязательное)

СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

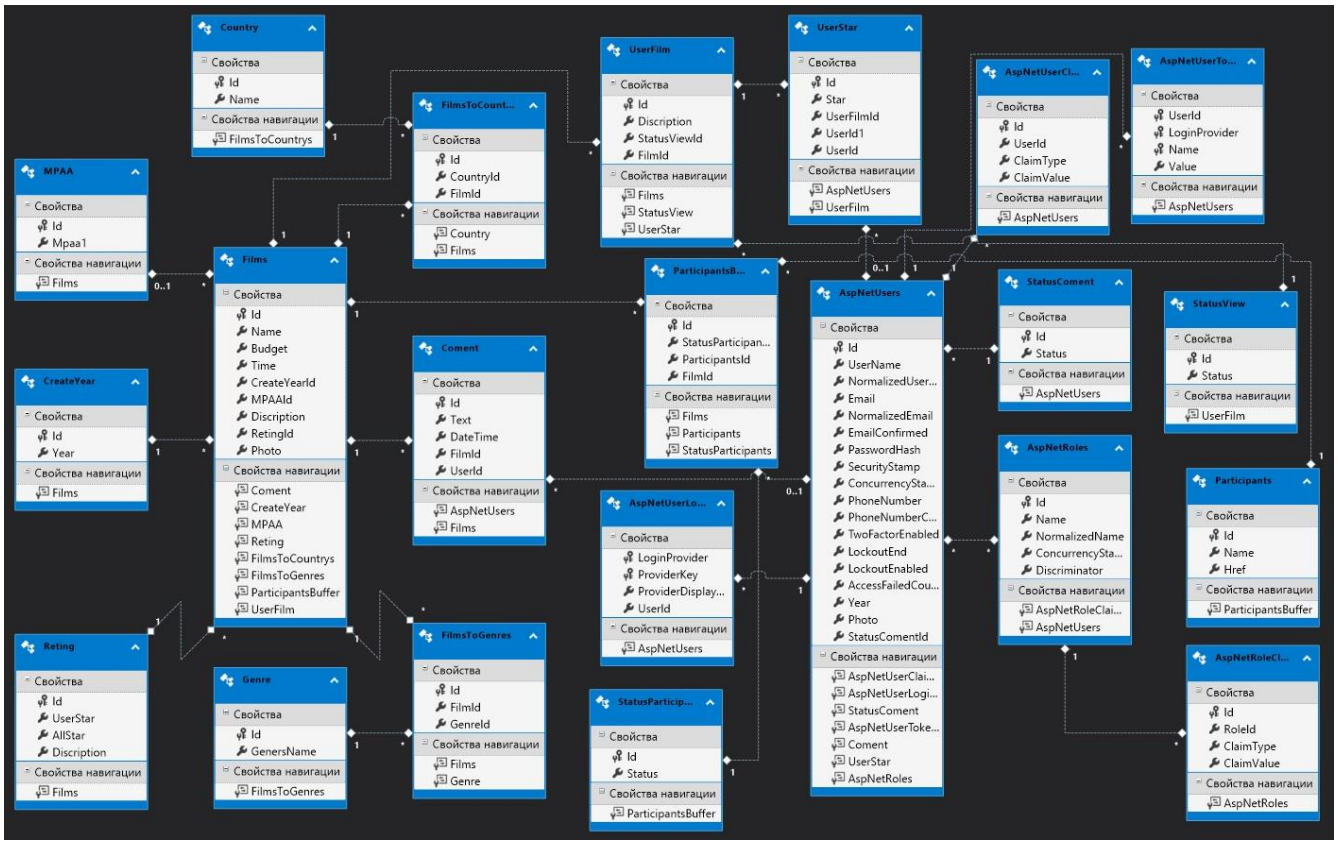

Рисунок Б.1 – Схема реляционной базы данных

ПРИЛОЖЕНИЕ В

(обязательное)

ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ

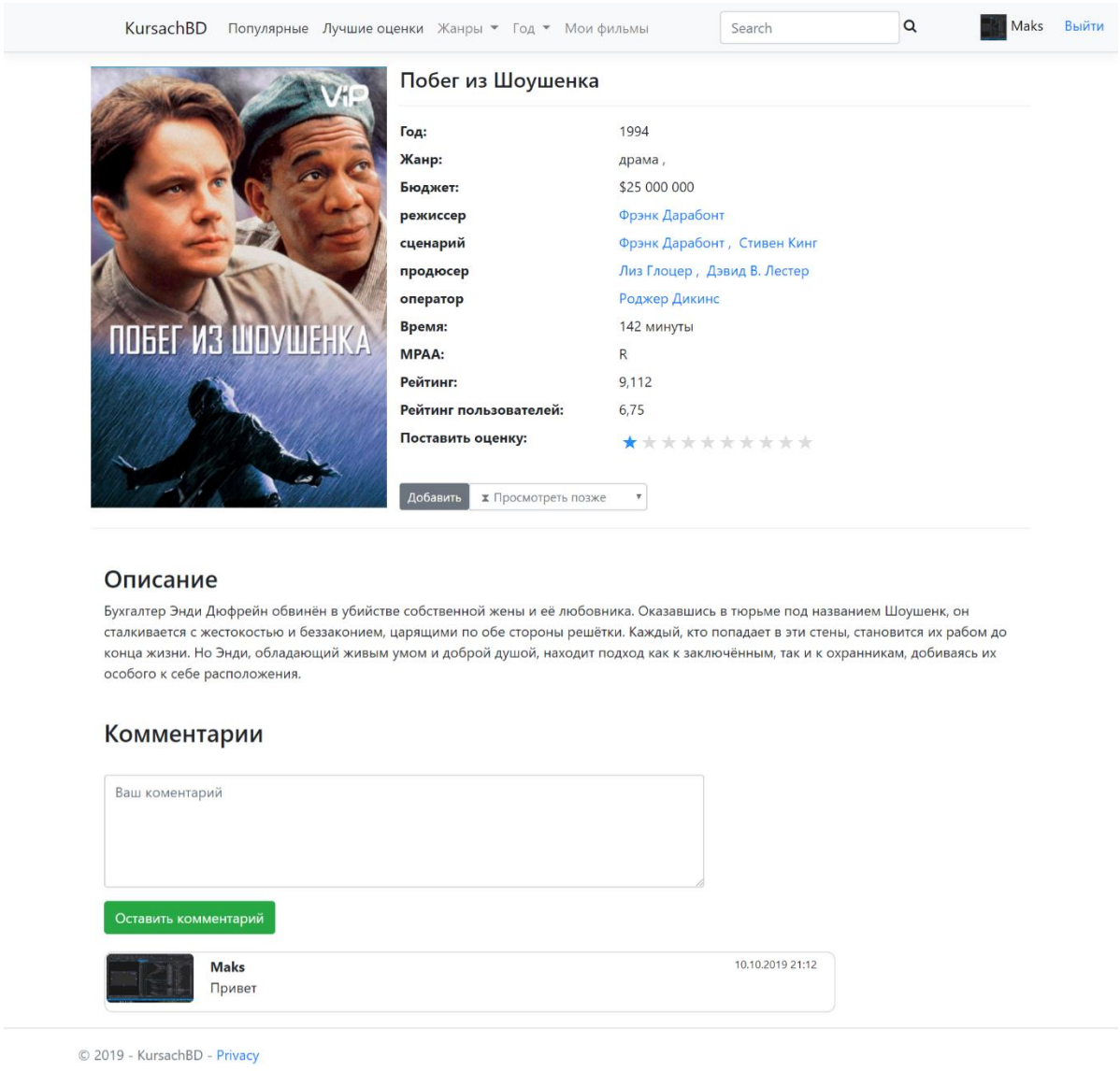

Рисунок В.1 – Полная информация о фильме

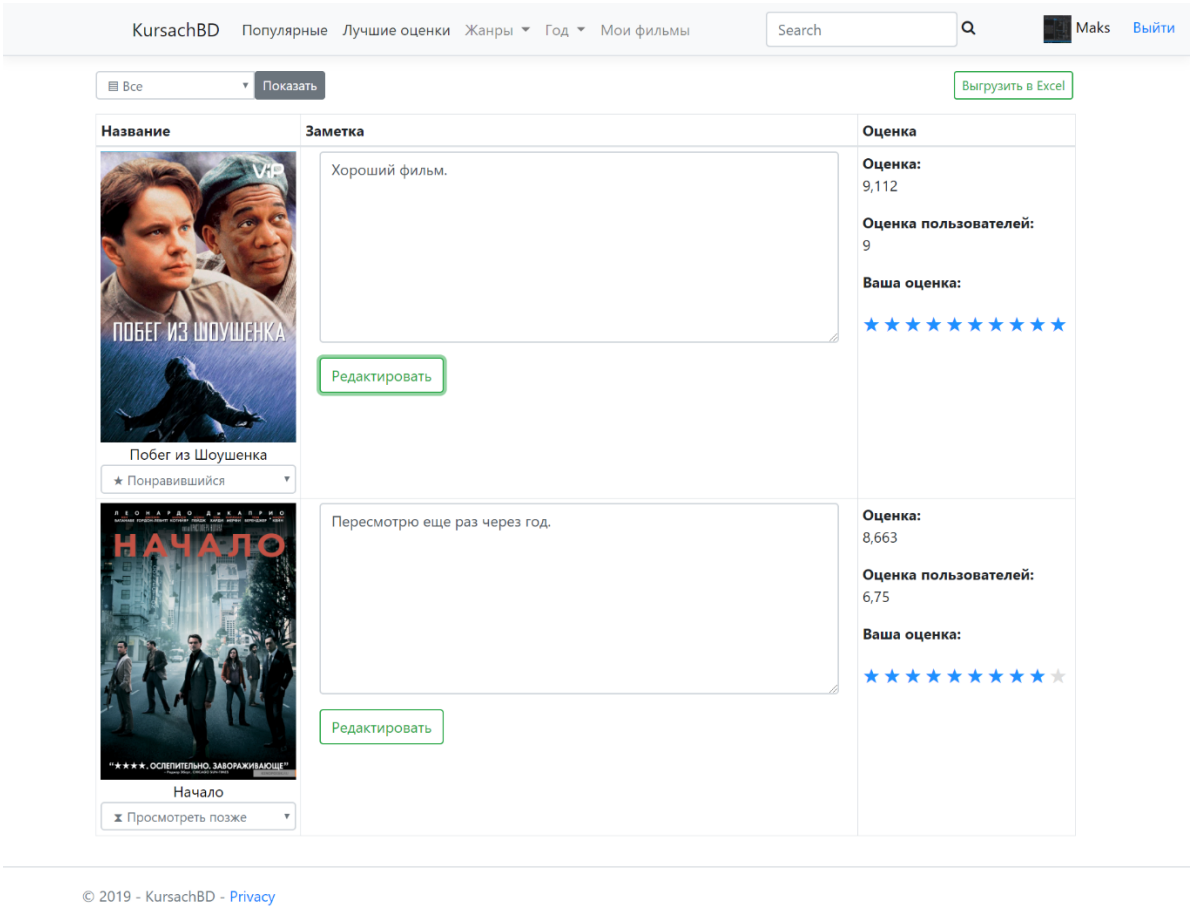

Рисунок В.2 – Личный кабинет пользователя

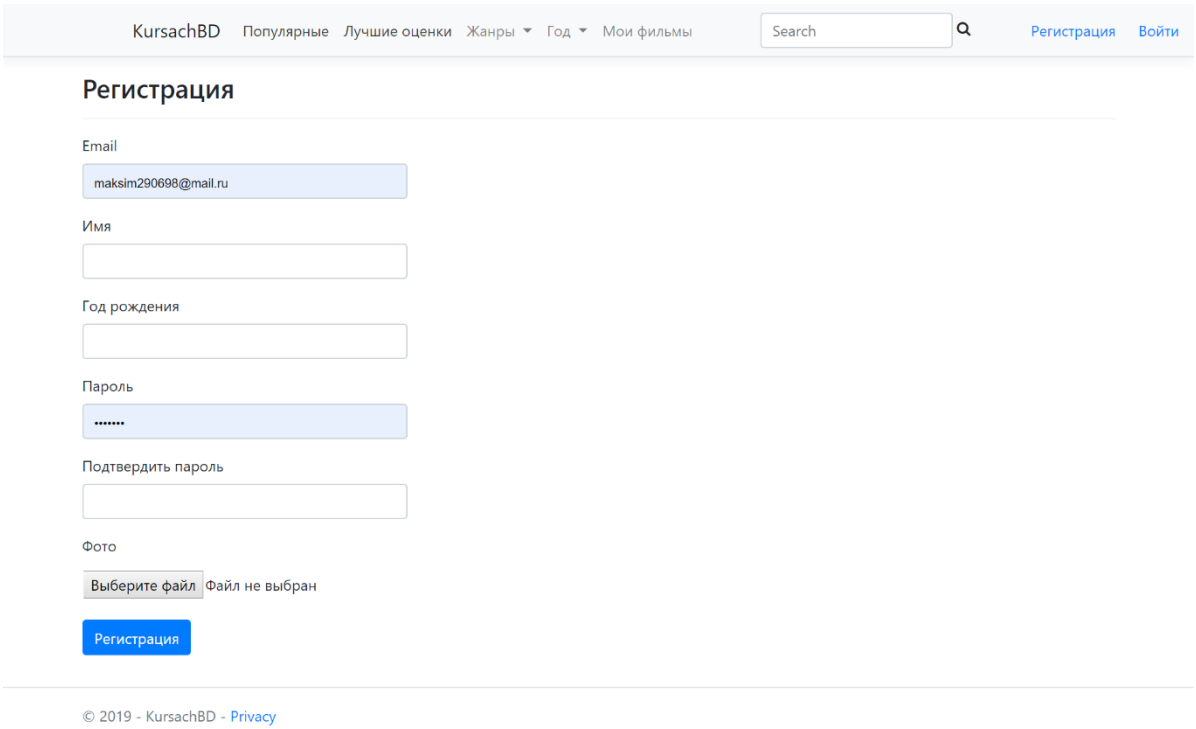

Рисунок В.3 – Форма регистрации

ПРИЛОЖЕНИЕ Г

(обязательное)

ЛИСТИНГ ПРОГРАММЫ

Листинг программы находится на диске.