

Содержание

Введение	6
1 Аналитический раздел	7
1.1 Постановка задачи	7
1.2 Пользователи системы	7
1.2.1 Гость	7
1.2.2 Авторизованный пользователь	8
1.2.3 Администратор	8
1.3 Анализ существующих решений	8
1.3.1 Кино-сервисы	8
1.3.2 Аудио-сервисы	9
1.3.3 Вывод	11
1.4 Анализ моделей баз данных	11
1.4.1 Иерархическая база данных	11
1.4.2 Сетевая модель базы данных	12
1.4.3 Реляционная модель базы данных	12
1.4.4 Выбор модели данных	13
1.5 Анализ СУБД	13
1.5.1 MySQL	13

1.5.2	Microsoft SQL Server	14
1.5.3	PostgreSQL	15
1.5.4	Oracle	15
1.6	Вывод	16
2	Конструкторский раздел	17
2.1	Сценарии пользователей	17
2.1.1	Гость	17
2.1.2	Авторизованный пользователь	18
2.1.3	Администратор	19
2.2	Ролевая модель	21
2.2.1	Гость	21
2.2.2	Пользователь	21
2.2.3	Администратор	21
2.3	Проектирование базы данных	22
2.3.1	Формализация сущностей системы	22
2.3.2	Функции	23
2.4	Проектирование приложения	23
2.5	Вывод	24
3	Технологический раздел	25

3.1	Средства реализации поставленной задачи	25
3.2	Создание базы данных	26
3.3	Функции	27
3.4	Разработка компонентов	27
3.4.1	Компонент доступа к данным	28
3.4.2	Компонент бизнес-логики	28
3.5	Интерфейс приложения	29
3.6	Вывод	33
	Заключение	34
	Литература	35
	Приложение А. Создание таблиц базы данных.	36
	Приложение Б. Презентация.	37

Введение

В наше время тяжело представить медиапространство без фильмов. Современным зрителям становится все тяжелее и тяжелее подобрать киноленту, что их завлечет и понравится. Следовательно, для желающих посмотреть интересное для них кино требуется нечто, способное порекомендовать удовлетворяющую запросы ленту. Приложение для рекомендации фильмов способно решить данный вопрос.

Целью данной работы является реализация простого в использовании и многофункционального приложения для получения информации о фильмах и их рекомендации пользователю.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) формализовать задание, выделив соответствующих акторов и их функционал;
- 2) провести анализ существующих решений;
- 3) провести анализ СУБД и выбрать наиболее подходящую;
- 4) спроектировать базу данных;
- 5) спроектировать архитектуру приложения;
- 6) разработать приложение.

1 Аналитический раздел

В данном разделе будет поставлена задача, рассмотрены возможные пользователи системы (акторы), модели данных и СУБД.

1.1 Постановка задачи

Разработать программу, предоставляющую интерфейс для получения информации о фильмах, рекомендованных пользователю. Посредством интерфейса нужно обеспечить для пользователя выбор любимых жанров и актёров, доступ к списку рекомендованных фильмов с информацией о них.

1.2 Пользователи системы

В системе существуют следующие виды пользователей:

- 1) гость;
- 2) авторизованный пользователь;
- 3) администратор.

1.2.1 Гость

Гость — это неавторизованный пользователь, обладающий минимальным набором возможностей взаимодействия с системой. Он может авторизоваться, просмотреть общий список фильмов, актёров, жанров с информацией о них.

1.2.2 Авторизованный пользователь

В функционал авторизованного пользователя входит возможность просмотра общего списка фильмов, актеров, жанров с информацией о них. Также есть возможность выбрать любимые жанры и актеров. Помимо этого пользователь может получить список рекомендуемых ему фильмов, основанных на выборе любимых актеров и жанров.

1.2.3 Администратор

Администратор - это пользователь, обладающий возможностью просмотра, добавления, удаления данных, связанных с фильмами актерами, жанрами, студиями, режиссерами, пользователями.

1.3 Анализ существующих решений

В настоящее время существует множество популярных медиа-сервисов, как для фильмов, так и для музыки. Рассмотрим некоторые из них.

1.3.1 Кино-сервисы

Сейчас появилось очень много сервисов для просмотра фильмов: «Кинопоиск», «Ivi», «Netflix» и другие. На них можно смотреть фильмы и сериалы, но система рекомендаций есть на всех из них. Тот же «Кинопоиск» лишь недавно вновь вернул раздел рекомендаций на своем сайте. Рекомендации основаны на оценках фильмов пользователем. Ниже на Рисунке 1 продемонстрирован внешний вид данного раздела.

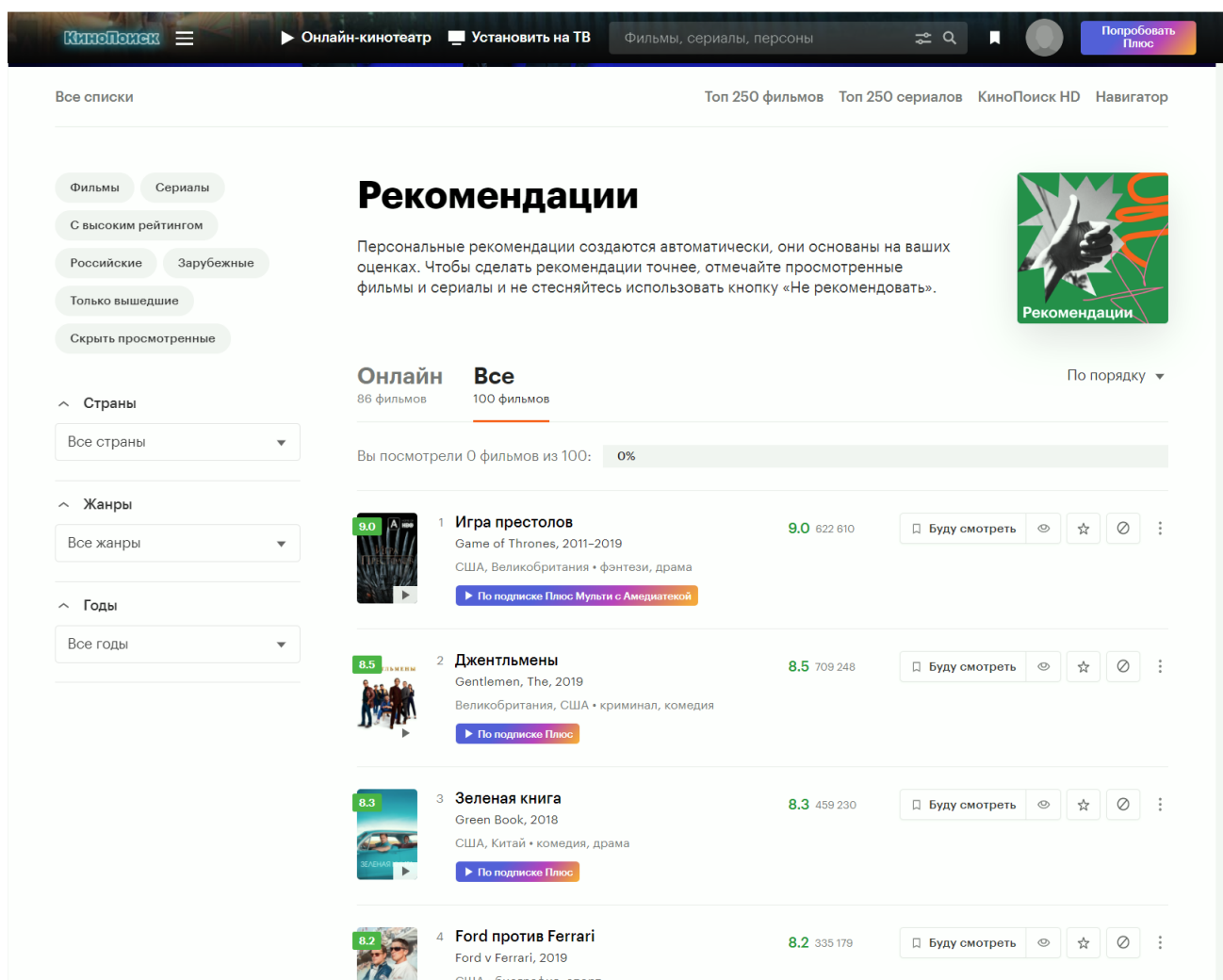


Рисунок 1 – Раздел рекомендаций сервиса «Кинопоиск».

1.3.2 Аудио-сервисы

Помимо распространенных кино-сервисов существует великое множество сервисов по подборке музыки: «Spotify», «Яндекс.Музыка», «Boom» и другие. Данные сервисы предоставляют для прослушивания музыку разных жанров и от разных исполнителей. Однако, каждый из них имеет систему рекомендаций и всячески старается подчеркнуть ее наличие. У аудио-сервисов рекомендации основаны на добавленных песнях, на любимых ис-

полнителях и жанрах, на прослушанном материале. Ниже на Рисунке 2 продемонстрирован внешний вид данного раздела рекомендаций в приложении «Spotify».

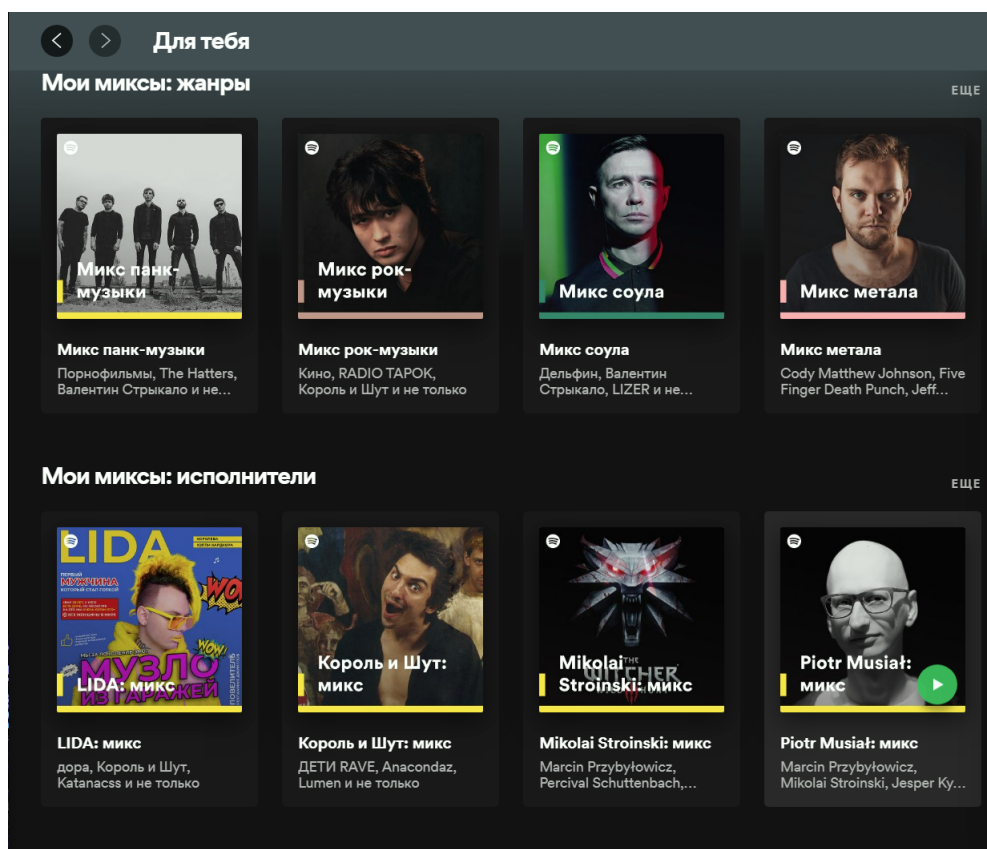


Рисунок 2 – Раздел рекомендаций сервиса «Spotify».

1.3.3 Вывод

Просмотрев популярные медиа-сервисы, приходит понимание, что кино-сервисам не хватает большой и разнообразной системы рекомендаций, основанной на широком спектре интересов пользователя, что присутствует во многих аудио-сервисах.

1.4 Анализ моделей баз данных

Модель базы данных - это тип модели данных, которая определяет логическую структуру базы данных и в корне определяет, каким образом данные могут храниться, организовываться и обрабатываться[1].

1.4.1 Иерархическая база данных

Иерархические базы данных имеют форму деревьев с дугами-связями и узлами-элементами данных. Иерархическая структура предполагала неравноправие между данными – одни жестко подчинены другим[1].

Иерархические базы данных — самая ранняя модель представления сложной структуры данных. Информация в иерархической базе организована по принципу древовидной структуры, в виде отношений «предок-потомок». Каждая запись может иметь не более одной родительской записи и несколько подчиненных. Связи записей реализуются в виде физических указателей с одной записи на другую. Основным недостатком иерархической структуры базы данных — невозможность реализовать отношения «много-многим», а также ситуации, когда запись имеет несколько предков[1].

1.4.2 Сетевая модель базы данных

В сетевых БД наряду с вертикальными реализованы и горизонтальные связи. Однако унаследованы многие недостатки иерархической и главный из них, необходимость четко определять на физическом уровне связи данных и столь же четко следовать этой структуре связей при запросах к базе[1].

Сетевая модель базы данных подразумевает, что у родительского элемента может быть несколько потомков, а у дочернего элемента — несколько предков. Записи в такой модели связаны списками с указателями. Сетевая модель позволяет иметь несколько предков и потомков, формирующих решётчатую структуру.

1.4.3 Реляционная модель базы данных

В реляционной модели, в отличие от иерархической или сетевой, не существует физических отношений. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Для манипуляций с рядами данных существуют специальные операторы.

Реляционные таблицы обладают следующими свойствами:

- 1) все значения атомарны;
- 2) каждый ряд уникален;
- 3) порядок столбцов не важен;
- 4) порядок рядов не важен;

5) у каждого столбца есть своё уникальное имя.

1.4.4 Выбор модели данных

Реляционная модель была выбрана в качестве модели данных. Ее структура данных однозначно определена, не является быстроизменяющейся и данные подчиняются строгим правилам и ограничениям.

1.5 Анализ СУБД

СУБД — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Одними из самых популярных реляционных систем управления базами данных сейчас являются MSSQL, MySQL, PostgreSQL, Oracle.

1.5.1 MySQL

MySQL — реляционная СУБД с открытым исходным кодом, главными плюсами которой являются ее скорость и гибкость, которая обеспечена поддержкой большого количества различных типов таблиц.

Преимущества:

- 1) простота в использовании;
- 2) масштабируемость;
- 3) безопасность;

- 4) обширный функционал;
- 5) скорость.

Недостатки:

- 1) недостаточная надежность;
- 2) низкая скорость разработки.

1.5.2 Microsoft SQL Server

Система позволяет синхронизироваться с другими программными продуктами компании Microsoft, а также обеспечивает надежную защиту данных и простой интерфейс, однако отличается высокой стоимостью лицензии и повышенным потреблением ресурсов.

Преимущества:

- 1) простота в использовании;
- 2) масштабируемость;
- 3) возможность интеграции с другими продуктами Microsoft.

Недостатки:

- 1) высокая стоимость продукта для юридических лиц;
- 2) высокая ресурсоемкость SQL Server.

1.5.3 PostgreSQL

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности.

Преимущества:

- 1) бесплатное ПО с открытым исходным кодом;
- 2) активная поддержка сообщества;
- 3) расширяемость;

Недостатки:

- 1) производительность;

1.5.4 Oracle

Oracle – это объектно-реляционная система управления базами данных.

Преимущества:

- 1) поддержка огромных баз данных;
- 2) быстрая обработка транзакций;
- 3) большой и постоянно развивающийся функционал.

Недостатки:

- 1) высокая стоимость;
- 2) значительные вычислительные ресурсы.

1.6 Вывод

В данном разделе была поставлена задача, рассмотрены возможные пользователи системы, представлены и проанализированы современные решения, проведен анализ модели данных и СУБД.

2 Конструкторский раздел

В данном разделе будут рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и спроектировано приложение.

2.1 Сценарии пользователей

Нужно определить возможности каждого из видов пользователей.

2.1.1 Гость

Гость — это неавторизованный пользователь, обладающий минимальным набором возможностей взаимодействия с системой. Он может авторизоваться, просмотреть общий список фильмов, актеров, жанров с информацией о них. На Рисунке 3 представлена Use-Case-диаграмма для гостя.

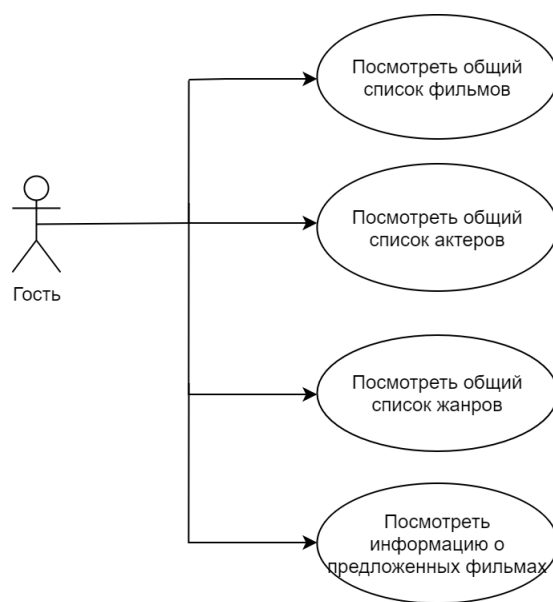


Рисунок 3 – Сценарии для гостя.

2.1.2 Авторизованный пользователь

Авторизованному пользователю доступно:

- 1) просмотр общего списка фильмов;
- 2) просмотр общего списка актеров;
- 3) просмотр общего списка жанров;
- 4) добавление/удаление любимых актеров;
- 5) добавление/удаление любимых жанров;
- 6) просмотр информации о рекомендованных фильмах.

На Рисунке 4 представлена Use-Case-диаграмма для пользователя.

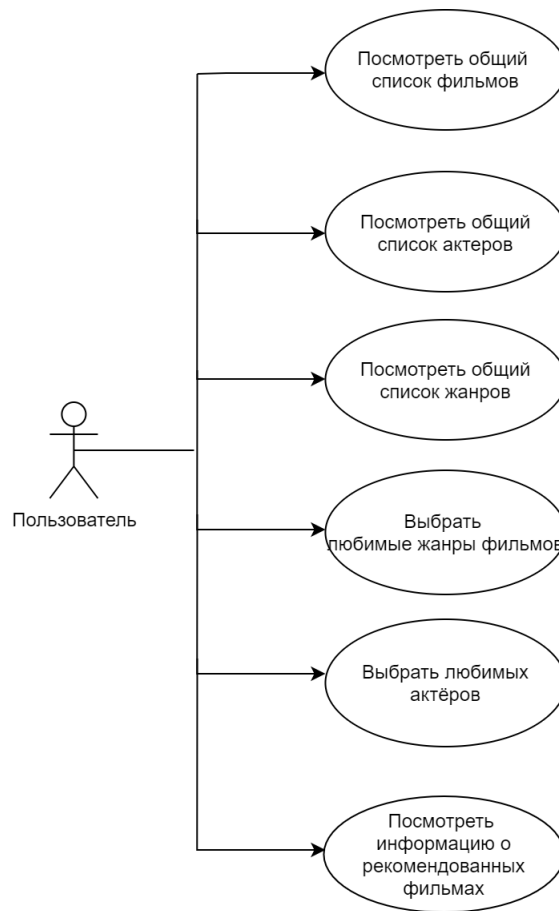


Рисунок 4 – Сценарии для пользователя.

2.1.3 Администратор

Администратору доступно:

- 1) просмотр общего списка студий, фильмов, актеров, жанров, режиссеров, пользователей;
- 2) добавление/удаление фильмов;
- 3) добавление/удаление актёров;
- 4) добавление/удаление жанров;

- 5) добавление/удаление режиссёров;
- 6) добавление/удаление пользователей;
- 7) добавление/удаление студий.

На Рисунке 5 представлена Use-Case-диаграмма для администратора.

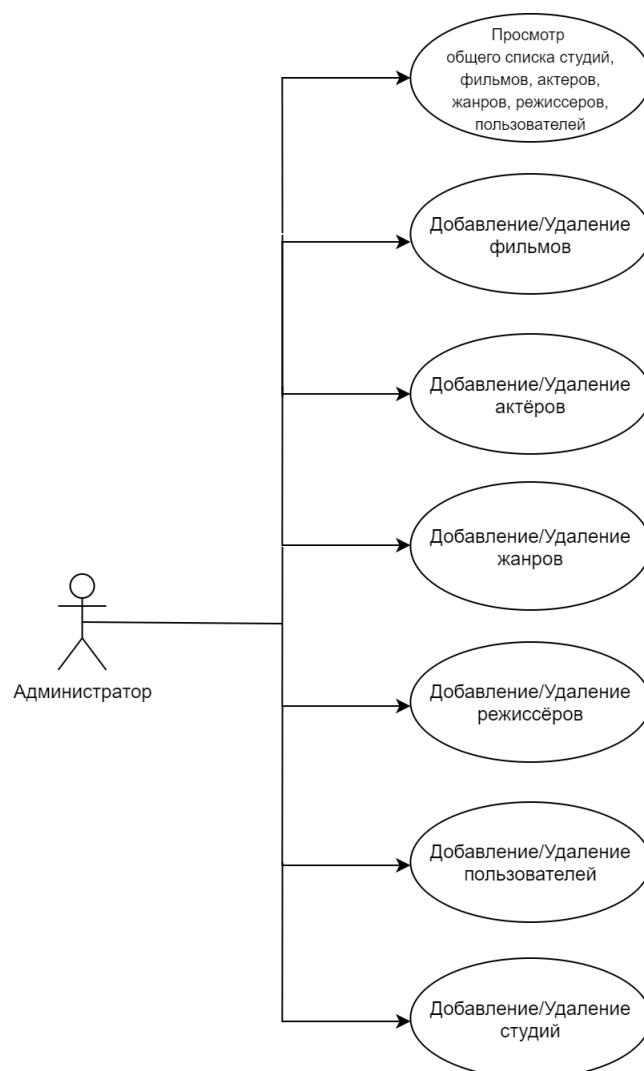


Рисунок 5 – Сценарии для администратора.

2.2 Ролевая модель

На уровне базы данных выделена следующая ролевая модель:

- 1) guest - гость;
- 2) common_user - пользователь;
- 3) admin - администратор.

Использование ролевой модели на уровне базы данных гарантирует безопасность доступа к объектам базы данных.

2.2.1 Гость

Пользователь с ролью guest имеет следующие права: SELECT над таблицами films, actors, genres, users.

2.2.2 Пользователь

Пользователь с ролью common_user имеет следующие права:

- 1) SELECT над всеми таблицами;
- 2) SELECT, UPDATE, DELETE, INSERT над таблицами любимых жанров и актеров пользователя - user_genres, user_actors.

2.2.3 Администратор

Пользователь с ролью admin обладает всеми правами.

2.3 Проектирование базы данных

2.3.1 Формализация сущностей системы

Необходимо выделить сущности предметной области и построить ER-диаграмму.

На рисунке 6 представлена ER-диаграмма системы.

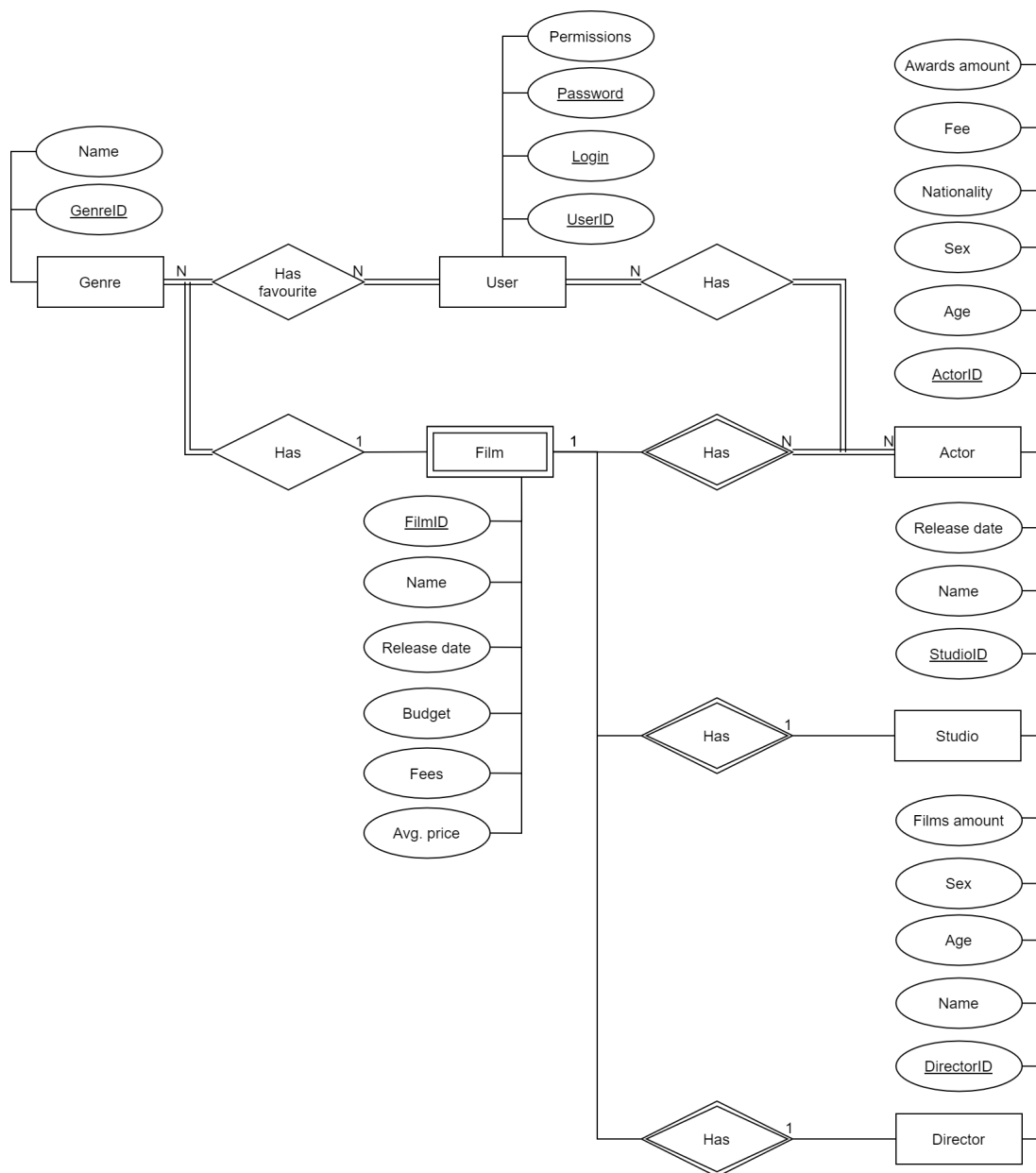


Рисунок 6 – ER-диаграмма системы.

Выделенные сущности:

- 1) Film - таблица, в которой хранится информация о фильмах;
- 2) Actor - таблица, в которой хранится информация об актерах;
- 3) User - таблица, в которой хранится информация о пользователях;
- 4) Genre - таблица, в которой хранится информация о жанрах;
- 5) Studio - таблица, в которой хранится информация о студиях;
- 6) Director - таблица, в которой хранится информация о режиссерах;

2.3.2 Функции

Для того, чтобы пользователь мог получать рекомендации по фильмам, необходимо добавить функцию, которая возвращает список фильмов, которые могут заинтересовать пользователя, и информацию о них.

2.4 Проектирование приложения

В качестве реализации проекта выбрано Desktop-приложение. Оно спроектировано по паттерну MVC. Выделены два компонента - доступа к данным и бизнес-логики. Компонент доступа к данным спроектирован по паттерну «Репозиторий». Для контроля ролей при авторизации пользователя создан класс Connection, в котором обрабатывается конфигурационный файл и возвращается строка подключения.

2.5 Вывод

В данном разделе были рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и спроектировано приложение.

3 Технологический раздел

В данном разделе описаны средства реализации поставленной задачи, создание базы данных и ролевая модель, разработаны компоненты и описан порядок работы.

3.1 Средства реализации поставленной задачи

Для данного проекта в качестве СУБД была выбрана PostgreSQL[2], так как данная система выигрывает по многим параметрам:

- 1) бесплатное ПО с открытым исходным кодом;
- 2) активная поддержка сообщества;
- 3) расширяемость;

В качестве языка программирования был выбран язык C#[3], так как:

- 1) имеются удобные пакеты для работы с PostgreSQL;
- 2) ООП язык программирования, что позволяет использовать наследование, интерфейсы, абстракции и т.п.

В качестве среды разработки была выбрана «Microsoft Visual Studio 2019»[4], так как:

- 1) имеет удобный интерфейс для написания отладки кода;
- 2) обеспечивает бесплатный доступ для студентов;

3) позволяет работать с Windows Forms[5].

Для работы с базой данных был выбран Entity Framework[6], так как EF Core поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы и работает с многими базами данных, включая PostgreSQL.

3.2 Создание базы данных

Требуется построить диаграмму БД по выделенным сущностям. В приложении А приведен листинг создания таблиц БД. На Рисунке 7 представлена диаграмма БД.

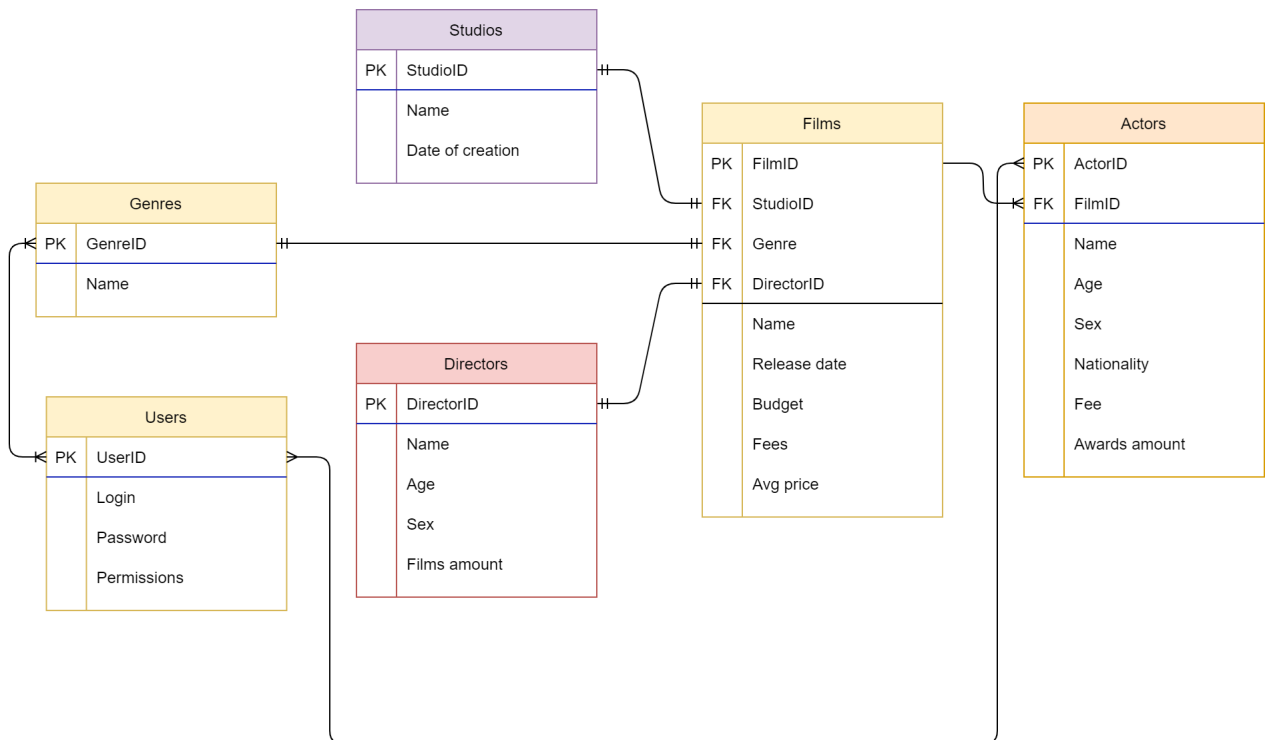


Рисунок 7 – Диаграмма БД.

3.3 Функции

В Листинге 1 представлена реализация функции `get_recommended_films`.

Листинг 1 – Реализация функции `GetPlayers`.

```
0 create or replace function get_recommended_films(int)
1 returns table
2 (
3     film_name varchar(40),
4     release_date date,
5     budget int,
6     fees int,
7     avg_price int
8 )
9 language sql
10 as $$
11     select film_name, release_date, budget, fees, avg_price
12     from users join user_actors
13     on $1 = user_actors.user_id
14     join actors on actors.actor_id = user_actors.actor_id
15     join films on films.film_id = actors.film_id
16     union
17     select film_name, release_date, budget, fees, avg_price
18     from users join user_genres
19     on $1 = user_genres.user_id
20     join genres on genres.genre_id = user_genres.genre_id
21     join films on films.genre_id = genres.genre_id
22 $$;
```

3.4 Разработка компонентов

Приложение спроектировано по паттерну MVC, поэтому следует реализовать компоненты доступа к данным и бизнес-логики.

3.4.1 Компонент доступа к данным

На Рисунке 8 представлена UML-диаграмма компонента доступа к данным.

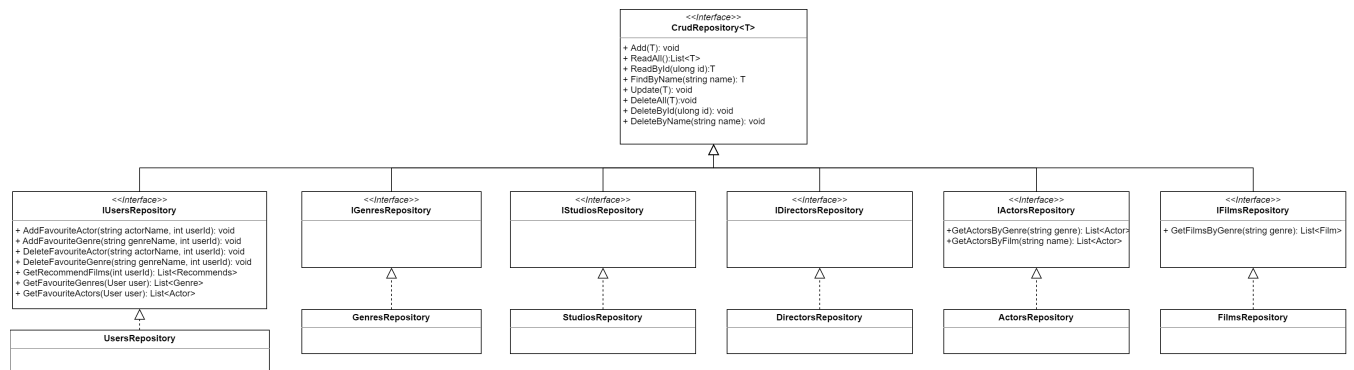


Рисунок 8 – Компонент доступа к данным.

3.4.2 Компонент бизнес-логики

На Рисунке 9 представлена UML-диаграмма компонента бизнес-логики.

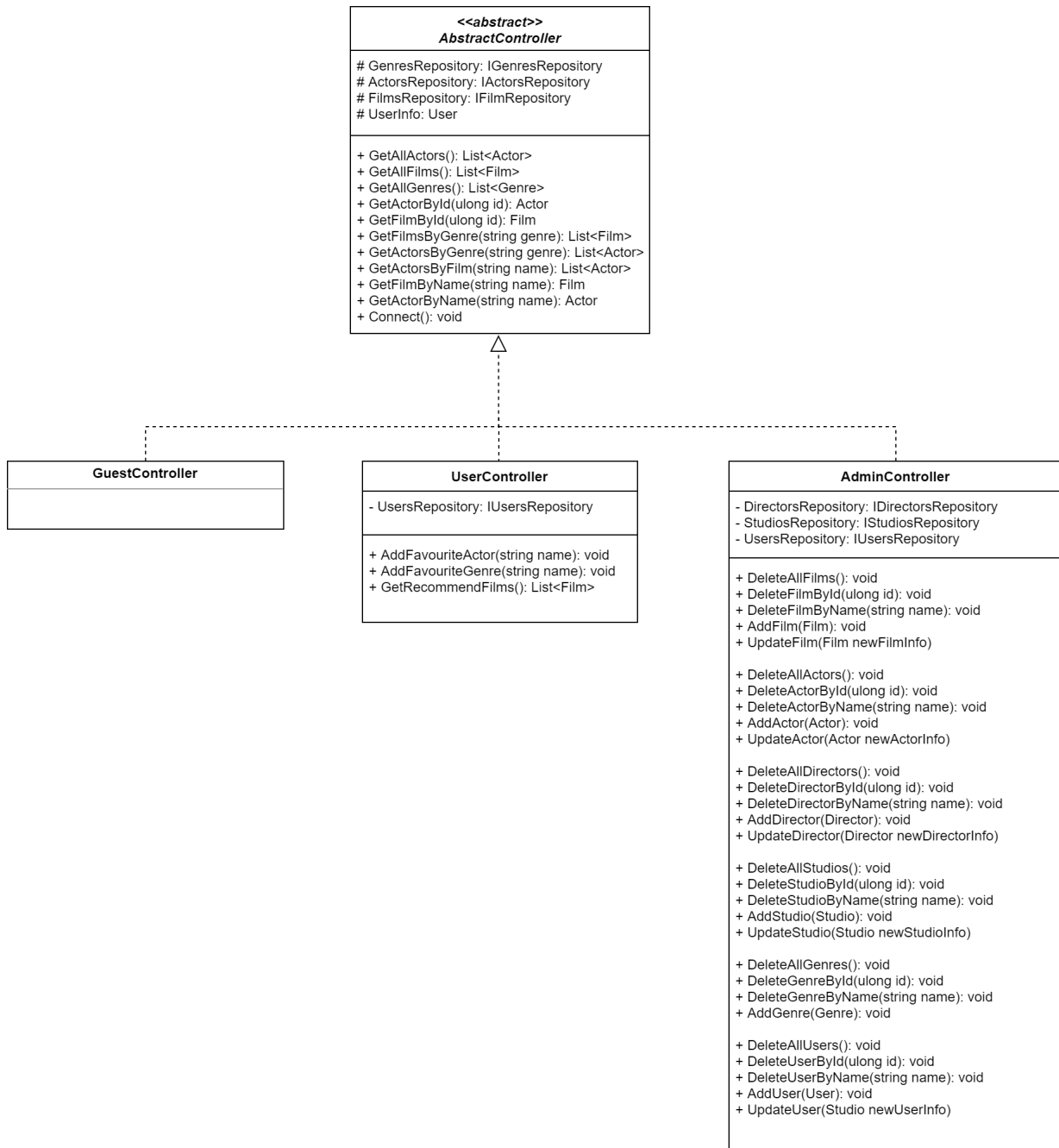
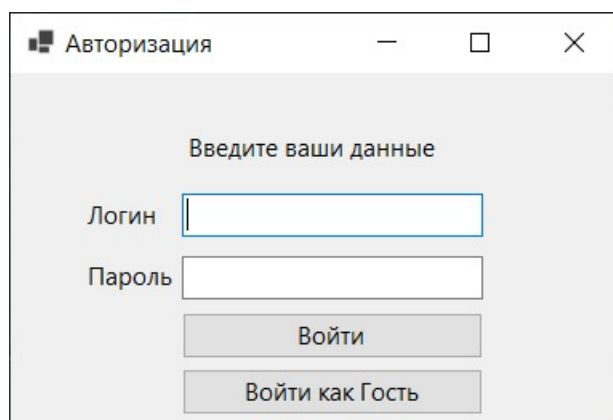


Рисунок 9 – Компонент бизнес-логики.

3.5 Интерфейс приложения

На Рисунках ниже показан интерфейс авторизации и интерфейсы пользователей (гость, авторизованный пользователь, администратор).



Авторизация

Введите ваши данные

Логин

Пароль

The image shows a standard Windows-style dialog box for authentication. The title bar contains the text 'Авторизация' (Authorization) and standard window control icons. The main area has a light gray background. At the top, it says 'Введите ваши данные' (Enter your data). Below this, there are two input fields: 'Логин' (Login) and 'Пароль' (Password). The 'Логин' field is a simple text box, while the 'Пароль' field is a password box with a white background and a light gray border. Below the password field are two buttons: 'Войти' (Log In) and 'Войти как Гость' (Log In as Guest). Both buttons have a light gray background and a thin border.

Рисунок 10 – Окно авторизации.

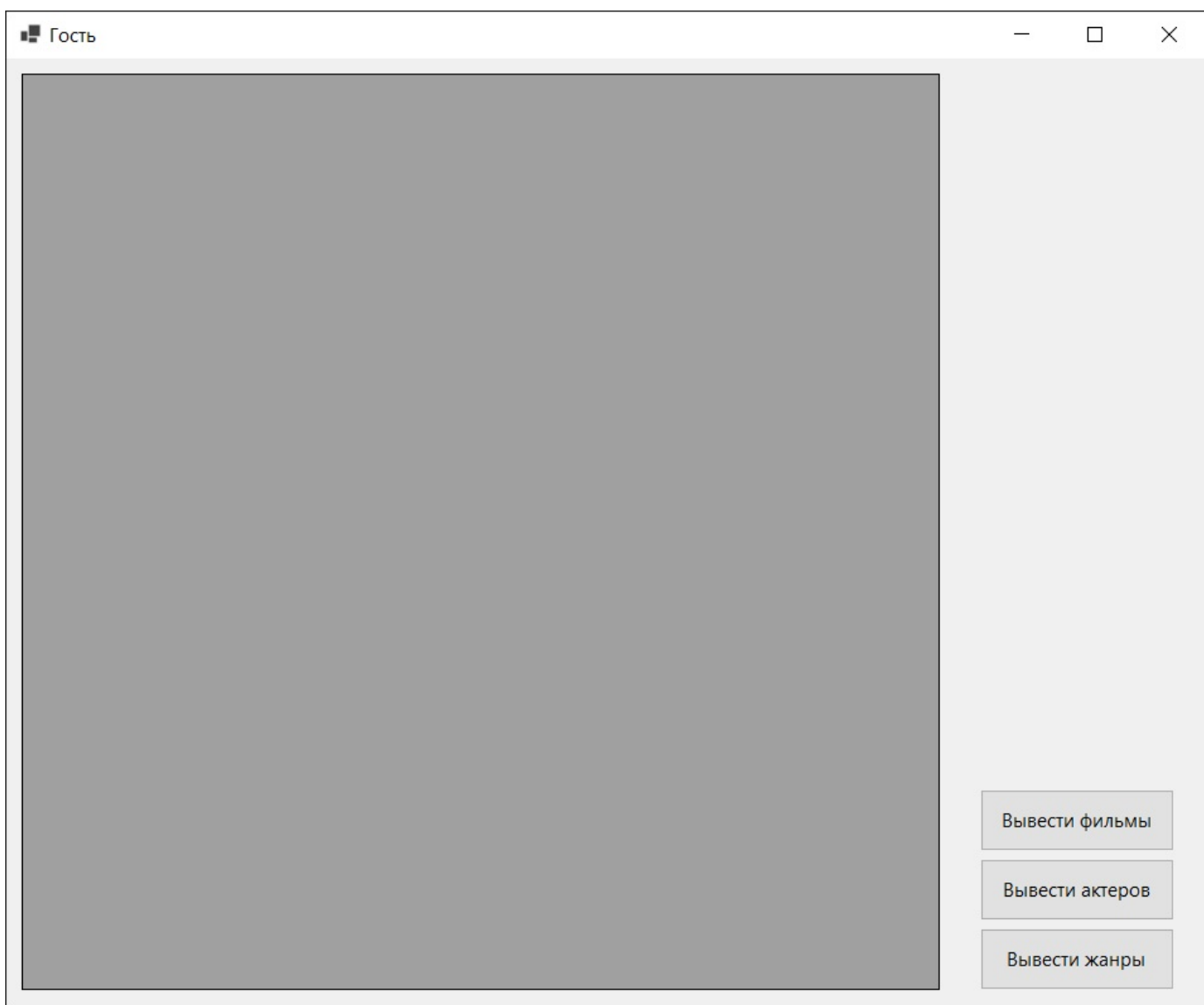


Рисунок 11 – Окно гостя.

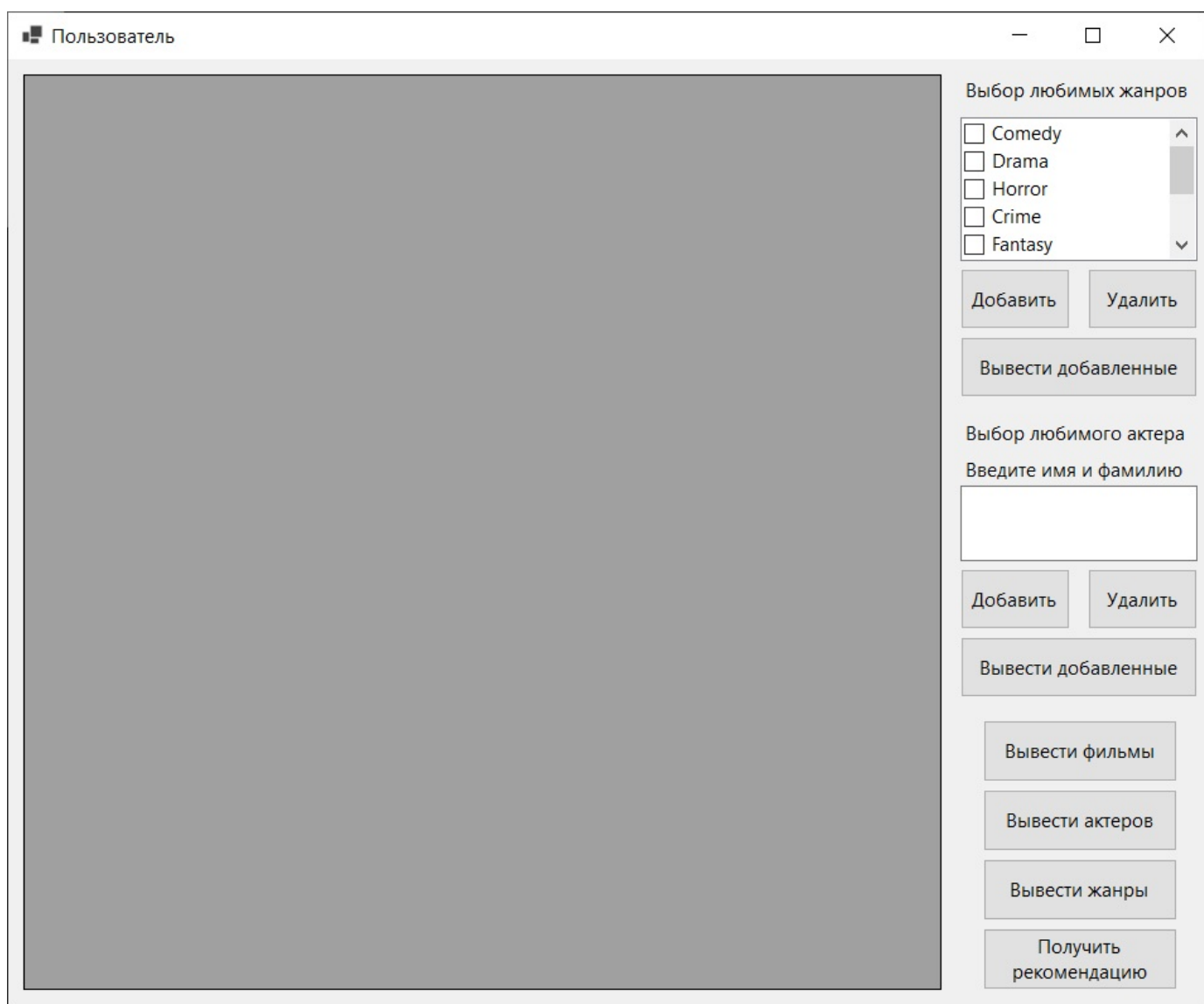


Рисунок 12 – Окно авторизованного пользователя.

The screenshot shows an 'Admin' window with a large grey placeholder on the left and several form sections on the right:

- Фильмы (Movies):** Fields for Film ID, Studio ID, Genre ID, Director ID, Название (Title), Бюджет (Budget), Сборы (Box Office), and Ср. цена (Average Price). Buttons: 'Добавить фильм в базу' (Add movie to database), 'Удалить фильм из базы' (Delete movie from database).
- Актеры (Actors):** Fields for Actor ID, Film ID, Имя и фам. (Name and surname), Возраст (Age), Пол (Gender), Нация (Nationality), Гонорар (Honorarium), and Наград (Awards). Buttons: 'Добавить актера в базу' (Add actor to database), 'Удалить актера из базы' (Delete actor from database).
- Режиссеры (Directors):** Fields for Director ID, Имя и фам. (Name and surname), Возраст (Age), Пол (Gender), and Кол-во фильмов (Number of movies). Buttons: 'Добавить режиссера в базу' (Add director to database), 'Удалить режиссера из базы' (Delete director from database).
- Студии (Studios):** Fields for Studio ID and Название (Title). Buttons: 'Добавить студию в базу' (Add studio to database), 'Удалить студию из базы' (Delete studio from database).
- Жанры (Genres):** Fields for Genre ID and Название (Title). Buttons: 'Добавить жанр в базу' (Add genre to database), 'Удалить жанр из базы' (Delete genre from database).
- Пользователи (Users):** Fields for User ID, Login, Password, and Права(0-2) (Rights(0-2)). Buttons: 'Добавить пользователя в базу' (Add user to database), 'Удалить пользователя из базы' (Delete user from database).

At the bottom, there is a row of buttons: 'Вывести фильмы' (Show movies), 'Вывести актеров' (Show actors), 'Вывести режиссеров' (Show directors), 'Вывести студии' (Show studios), 'Вывести жанры' (Show genres), and 'Вывести пользователей' (Show users).

Рисунок 13 – Окно администратора.

3.6 Вывод

В данном разделе были выбраны средства реализации поставленной задачи, создана база данных и описана ролевая модель на уровне БД, разработаны компоненты и описан порядок работы.

Заключение

Цель курсовой работы достигнута.

В ходе выполнения курсовой работы было формализовано задание, выделены соответствующие акторы и их функционал, проведен анализ и выбор наиболее подходящей для данной задачи СУБД, спроектирована база данных, приложение.

В результате, с использованием языка программирования C# и СУБД PostgreSQL было создано многофункциональное приложение для получения информации о фильмах, рекомендованных пользователю. Получен опыт разработки базы данных и приложения по паттерну MVC.

В дальнейшей перспективе приложение и база данных могут быть масштабированы. Может быть добавлен следующий функционал:

- 1) оценки фильмов пользователями и рекомендации на их основе;
- 2) добавление новой информации о фильмах путем новых полей и сущностей;
- 3) добавление актора с не такими огромными правами, как у администратора, но способного исполнять его основные обязанности.

Список литературы

- [1] Национальная библиотека им. Н. Э. Баумана Bauman National Library : [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql> (дата обращения: 20.05.2021).
- [2] PostgreSQL : Документация [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql> (дата обращения: 20.05.2021).
- [3] Документация по C# [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 20.05.2021).
- [4] Документация по семейству продуктов Visual Studio [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019> (дата обращения: 20.05.2021).
- [5] Windows Forms [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-5.0> (дата обращения: 20.05.2021).
- [6] Документация по Entity Framework [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/ef/> (дата обращения: 20.05.2021).

Приложение А.

Создание таблиц базы данных.

```
0 create table if not exists studios(  
1     studio_id int primary key,  
2     studio_name varchar(40) not null,  
3     date_of_creation date  
4 );  
5  
6 create table if not exists directors(  
7 director_id int primary key,  
8 director_name varchar(40) not null,  
9 age int check(age > 23 and age < 71),  
10 sex varchar (6),  
11 films_amount int check(films_amount > 0 and films_amount < 10)  
12 );  
13  
14 create table if not exists genres(  
15     genre_id int primary key,  
16     genre_name varchar(40)  
17 );  
18  
19 create table if not exists films(  
20     film_id int primary key,  
21     studio_id int references studios(studio_id),  
22     genre_id int references genres(genre_id),  
23     director_id int references directors(director_id),  
24     film_name varchar(40) not null,  
25     release_date date,  
26     budget int check(budget >= 20000 and budget <= 2500000000),  
27     fees int check(fees >= 20000 and fees <= 10000000000),  
28     avg_price int check(avg_price >= 120 and avg_price <= 500)  
29 );  
30  
31 create table if not exists actors(  
32     actor_id int primary key,  
33     actor_name varchar(40) not null,  
34     date_of_birth date,  
35     sex varchar(10) not null,  
36     height int check(height >= 150 and height <= 250),  
37     weight int check(weight >= 50 and weight <= 150),  
38     films_amount int check(films_amount >= 0 and films_amount <= 10),  
39     avg_price int check(avg_price >= 120 and avg_price <= 500)  
40 );
```

```

32     actor_id int primary key,
33     film_id int references films(film_id),
34     actor_name varchar(40) not null,
35     age int check(age > 17 and age < 71),
36     sex varchar(6),
37     nationality varchar(40),
38     fee int check(fee >= 1000 and fee <= 1000000),
39     awards int check(awards >= 0 and awards < 6)
40 );
41
42 create table if not exists users(
43     user_id int primary key,
44     login varchar(40) not null,
45     password varchar(40) not null,
46     permissions int
47 );
48
49 create table if not exists user_genres(
50     user_id int references users(user_id),
51     genre_id int references genres(genre_id)
52 );
53
54 create table if not exists user_actors(
55     user_id int references users(user_id),
56     actor_id int references actors(actor_id)
57 );

```