

# Содержание

<b>Введение</b>	<b>6</b>
<b>1 Аналитический раздел</b>	<b>7</b>
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 Существующие решения . . . . .	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 Вывод . . . . .	13
<b>2 Конструкторский раздел</b>	<b>14</b>

2.1	Сценарии пользователей . . . . .	14
2.1.1	Гость . . . . .	14
2.1.2	Авторизованный пользователь . . . . .	15
2.1.3	Администратор . . . . .	16
2.2	Проектирование базы данных . . . . .	18
2.2.1	Формализация сущностей системы . . . . .	18
2.3	Ролевая модель . . . . .	19
2.3.1	Гость . . . . .	19
2.3.2	Пользователь . . . . .	20
2.3.3	Администратор . . . . .	20
2.3.4	Функции . . . . .	20
2.4	Технологический стек . . . . .	21
2.5	Вывод . . . . .	22
<b>3</b>	<b>Технологический раздел</b>	<b>23</b>
3.1	Выбор СУБД . . . . .	23
3.2	Средства реализации поставленной задачи . . . . .	24
3.3	Создание базы данных . . . . .	25
3.4	Создание таблиц . . . . .	26
3.5	Создание пользователей системы . . . . .	29
3.6	Функции . . . . .	30

3.7	Разработка компонентов . . . . .	31
3.7.1	Компонент доступа к данным . . . . .	31
3.7.2	Компонент бизнес-логики . . . . .	33
3.8	Интерфейс приложения . . . . .	35
3.9	Вывод . . . . .	38
	<b>Заключение</b>	<b>39</b>
	<b>Литература</b>	<b>40</b>
	<b>Приложение А. Презентация.</b>	<b>41</b>

## Введение

В наше время тяжело представить свою жизнь 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 пользователей 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, 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 Существующие решения**

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 существует очень много сервисов для просмотра фильмов: «Кинопоиск», «Ivi», «Netflix» и другие. На них можно смотреть фильмы и сериалы, каждый из них имеет некую систему рекомендаций. У того же «Кинопоиск» есть раздел рекомендаций на своем сайте. Рекомендации основаны на оценках фильмов пользователем. Ниже на Рисунке 1 продемонстрирован внешний вид данного раздела.

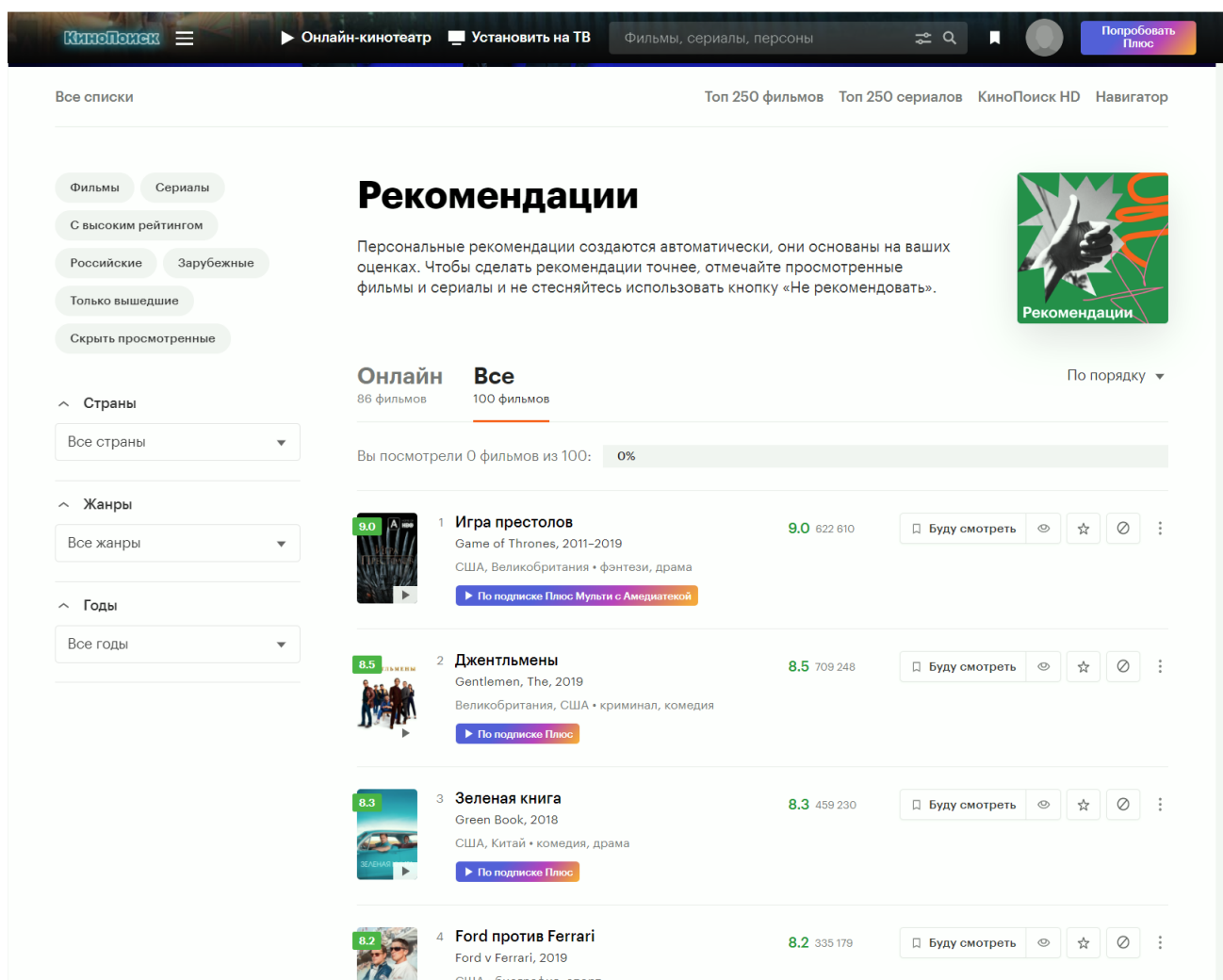


Рисунок 1 – Раздел рекомендаций сервиса «Кинопоиск».

### 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».

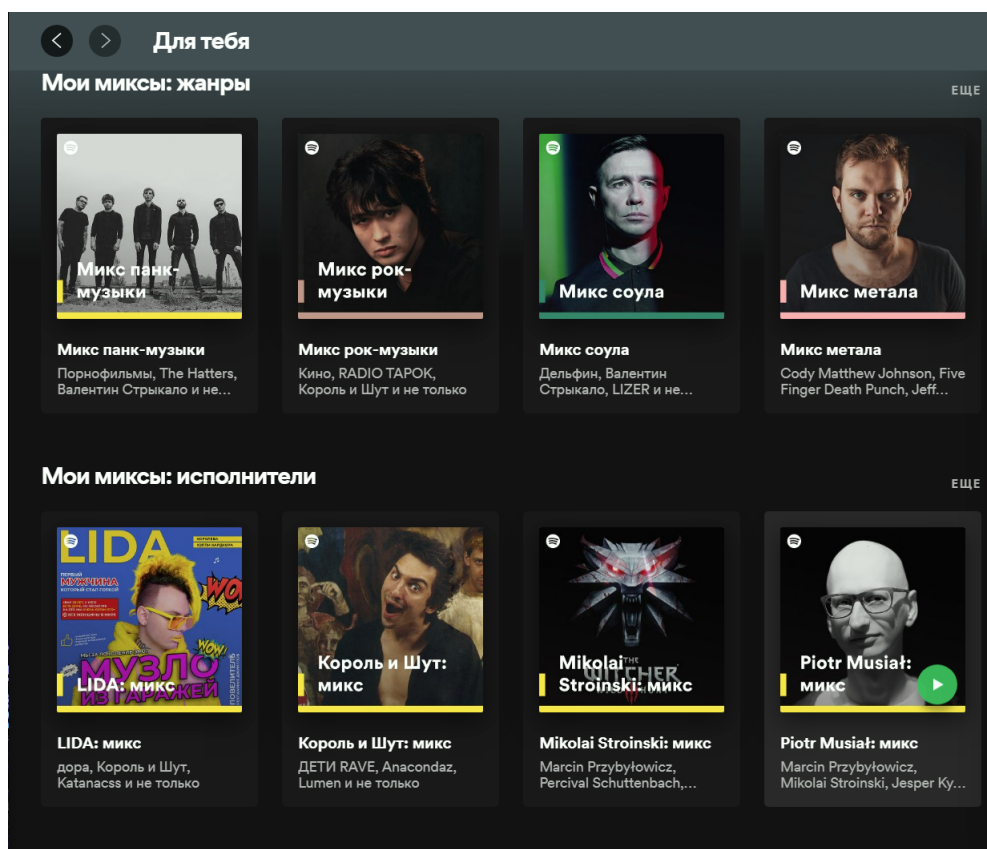


Рисунок 2 – Раздел рекомендаций сервиса «Spotify».



### **1.3.3 Вывод**

Просмотрев популярные медиа-сервисы, приходит понимание, что система рекомендаций играет большую роль для современного пользователя.

## **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. Основной недостаток иерархической структуры базы данных — невозможность реализовать отношения «много-ко-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 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

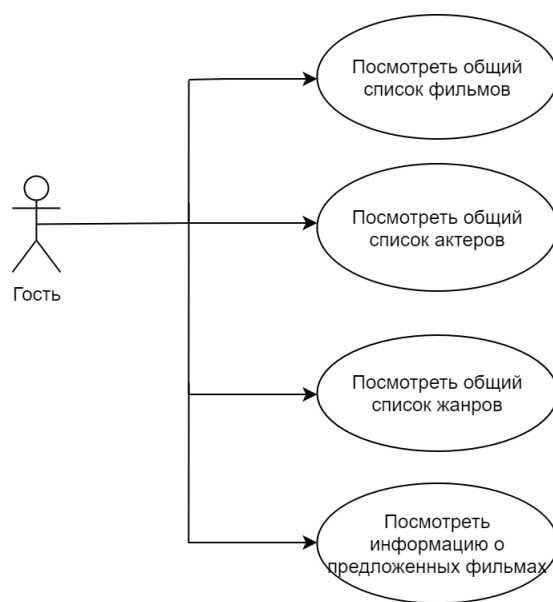
В данном разделе будут рассмотрены сценарии пользователей, спроектирована база данных, описана ролевая модель и спроектировано приложение.

### 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.



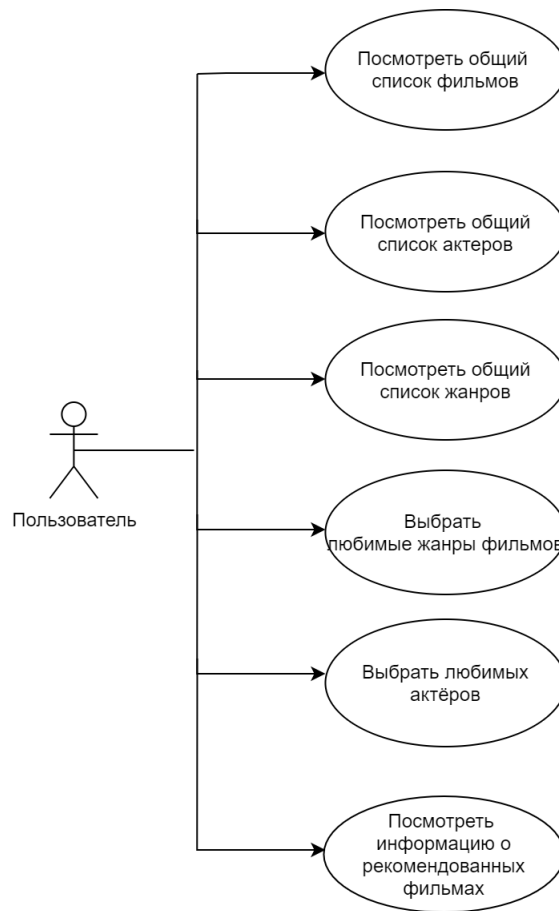
**Рисунок 3** – Сценарии для гостя.

### **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.

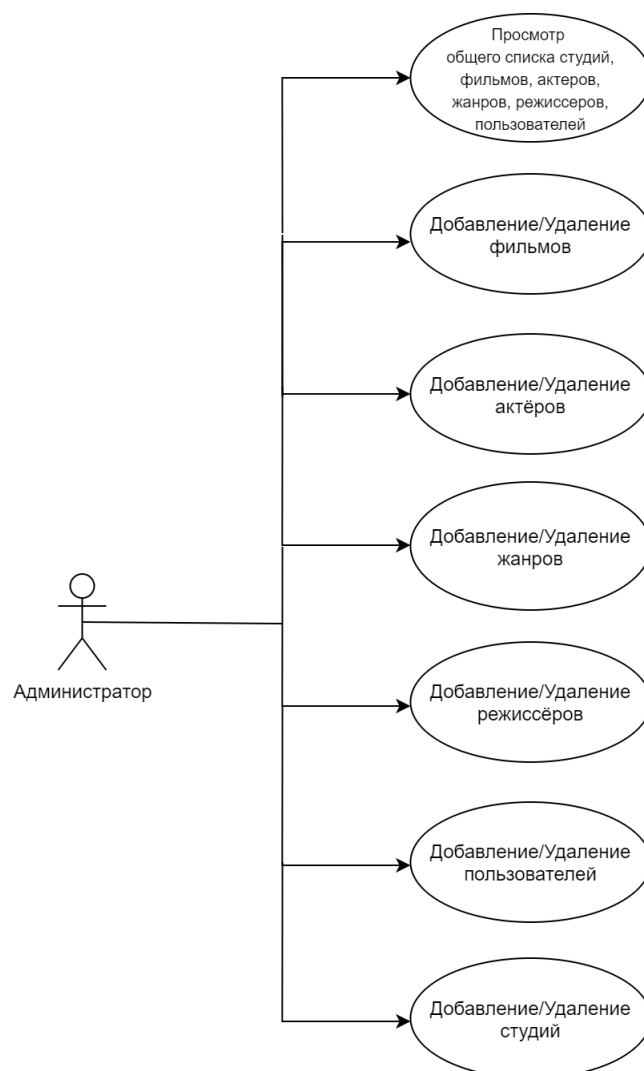
### 2.1.3 Администратор

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 Проектирование базы данных

### 2.2.1 Формализация сущностей системы

Необходимо выделить сущности предметной области и построить ER-диаграмму.

На рисунке 6 представлена ER-диаграмма системы.

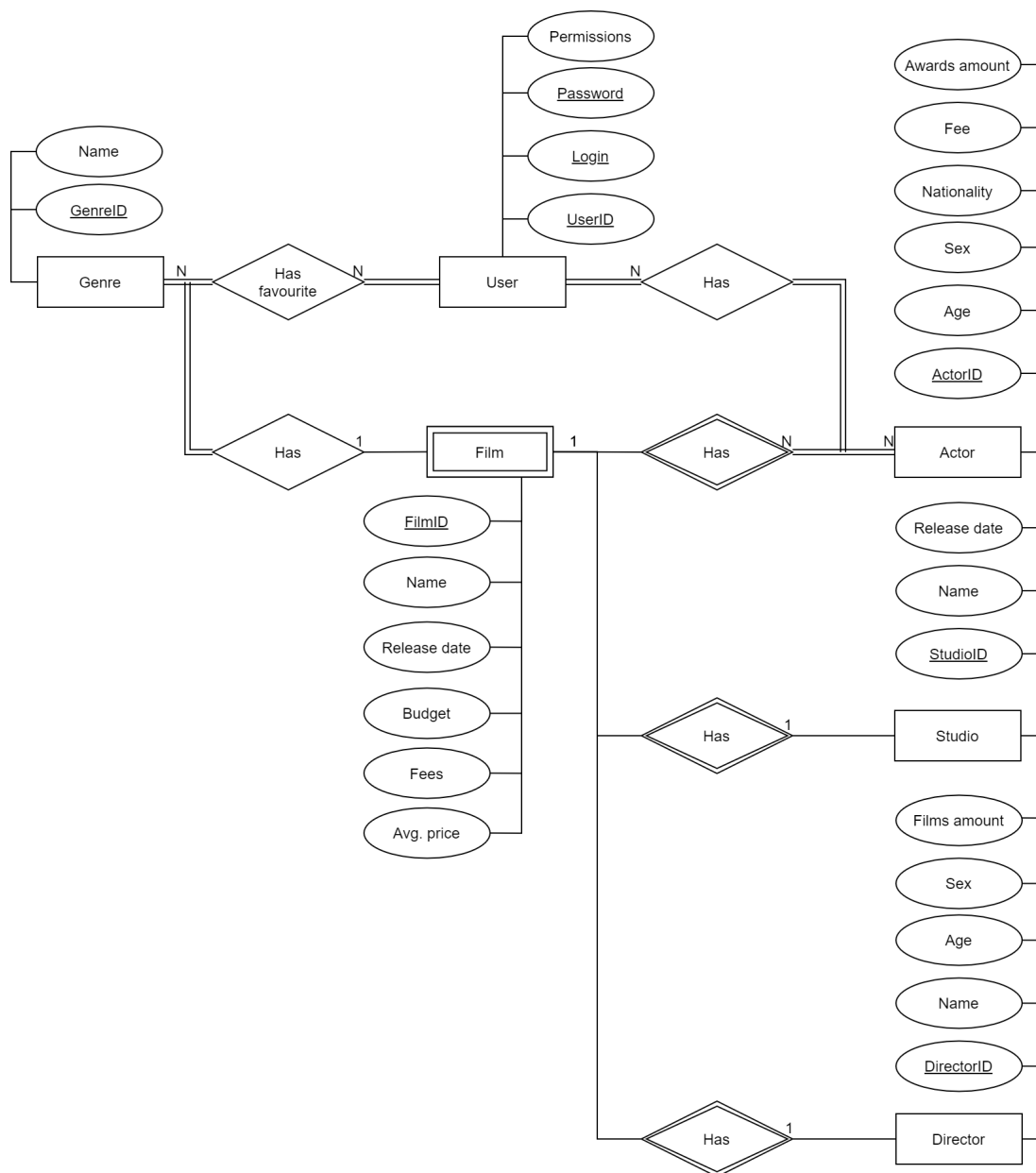


Рисунок 6 – ER-диаграмма системы.



### **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 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.

### **2.3.1 Гость**

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

### **2.3.2 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`.

### **2.3.3 Администратор**

Пользователь с ролью `admin` обладает всеми правами.

### **2.3.4 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. На Рисунке 7 представлен алгоритм этой функции.



**Рисунок 7** – Алгоритм функции.

## 2.4 Технологический стек

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 Выбор СУБД

СУБД — комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Для разработки системы была выбрана PostgreSQL по следующим причинам:

- 1) Надежность. Надёжность СУБД PostgreSQL проверена и доказана. Она обеспечивается соответствием принципам ACID (атомарность, изолированность, непротиворечивость, сохранность данных), многоверсионностью, наличием Write Ahead Logging (WAL) — общепринятого механизма протоколирования всех существующих транзакций. Сюда же стоит отнести и возможность восстановления базы данных Point in Time Recovery (PITR), репликацию, поддержку целостности данных на уровне схемы.
- 2) Производительность. В СУБД PostgreSQL она основана на применении индексов, наличии гибкой системы блокировок и интеллектуального планировщика запросов, использовании системы управления буферами памяти и кэширования. Не стоит забывать и про отличную масштабируемость при конкурентной работе.
- 3) Расширяемость. Для СУБД PostgreSQL это означает, что пользова-

тель может настроить систему посредством определения новых функций, типов, языков, агрегатов, индексов и операторов. А объектная ориентированность СУБД PostgreSQL даёт возможность переносить логику приложения на уровень базы данных, а это, в свою очередь, заметно упрощает разработку клиентов, ведь вся бизнес-логика находится в БД. При этом функции в Postgres однозначно определяются названием, типами и числом аргументов.

- 4) Поддержка SQL. PostgreSQL поддерживает схемы, подзапросы, внешние связи, правила, курсоры, наследование таблиц, триггеры.
- 5) Поддержка многочисленных типов данных. СУБД PostgreSQL поддерживает численные (целые, денежные, с фиксированной/плавающей точкой), булевы, символьные, составные, сетевые типы данных, а также перечисление, типы «дата/время», геометрические примитивы, массивы, XML- и JSON-данные. При этом можно создавать свои типы данных[2].

### **3.2 Средства реализации поставленной задачи**

Для данного проекта в качестве СУБД была выбрана PostgreSQL[3], так как данная система выигрывает по многим параметрам описанным выше.

В качестве языка программирования был выбран язык C#[4], 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»[5], так как:

- 1) имеет удобный интерфейс для написания отладки кода;
- 2) обеспечивает бесплатный доступ для студентов;
- 3) позволяет работать с Windows Forms[6].

Для работы с базой данных был выбран Entity Framework[7], так как EF Core поддерживает запросы LINQ, отслеживание изменений, обновления и миграции схемы и работает с многими базами данных, включая PostgreSQL.

### **3.3 Создание базы данных**

Требуется построить диаграмму БД по выделенным сущностям. На Рисунке 8 представлена диаграмма БД.

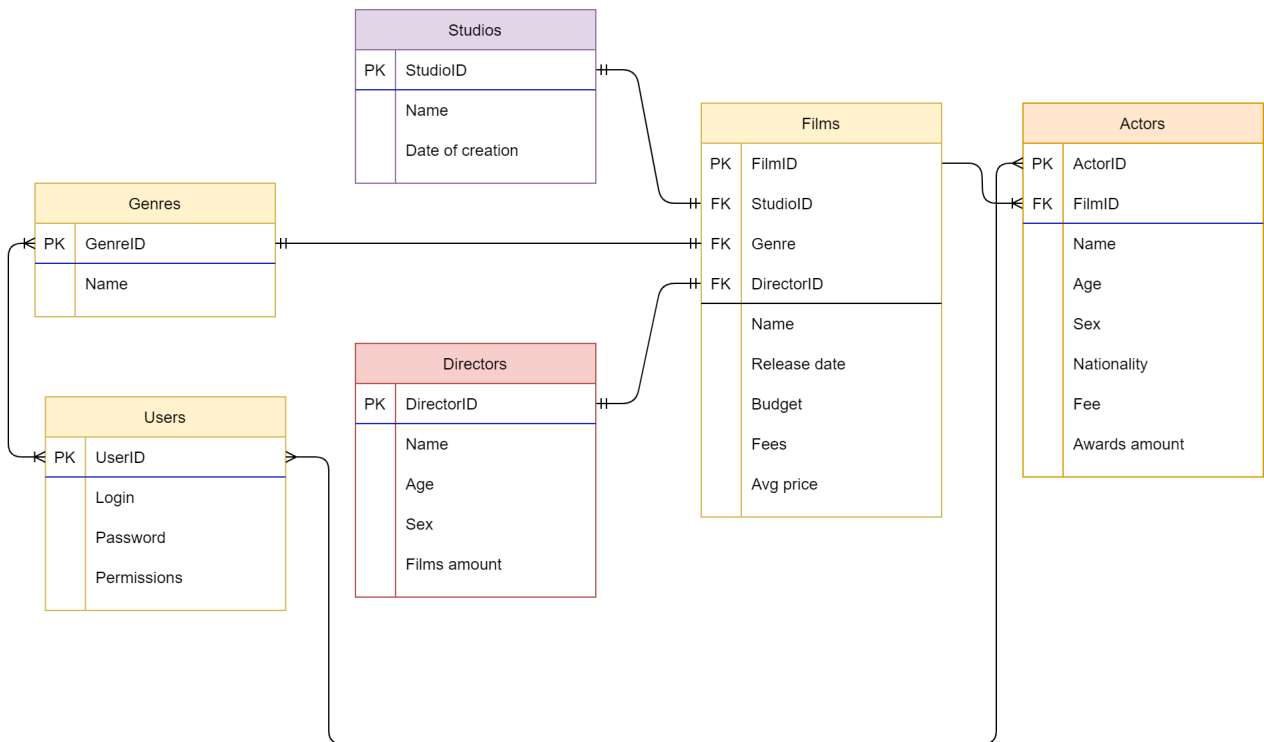


Рисунок 8 – Диаграмма БД.

### 3.4 Создание таблиц

Таблица studios.

Листинг 1 – Создание таблицы studios.

```

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  );
  
```



## Таблица directors.

### Листинг 2 – Создание таблицы directors.

```
0 create table if not exists directors(  
1     director_id int primary key,  
2     director_name varchar(40) not null,  
3     age int check(age > 23 and age < 71),  
4     sex varchar (6),  
5     films_amount int check(films_amount > 0 and films_amount < 10)  
6 );
```

## Таблица genres.

### Листинг 3 – Создание таблицы genres.

```
0 create table if not exists genres(  
1     genre_id int primary key,  
2     genre_name varchar(40)  
3 );
```

## Таблица films.

### Листинг 4 – Создание таблицы films.

```
0 create table if not exists films(  
1     film_id int primary key,  
2     studio_id int references studios(studio_id),  
3     genre_id int references genres(genre_id),  
4     director_id int references directors(director_id),  
5     film_name varchar(40) not null,  
6     release_date date,  
7     budget int check(budget >= 20000 and budget <= 2500000000),  
8     fees int check(fees >= 20000 and fees <= 10000000000),  
9     avg_price int check(avg_price >= 120 and avg_price <= 500)  
10 );
```

**Таблица actors.**

**Листинг 5** – Создание таблицы actors.

```
0  create table if not exists users(  
1      user_id int primary key,  
2      login varchar(40) not null,  
3      password varchar(40) not null,  
4      permissions int  
5  );
```

**Связочные таблицы user\_genres и user\_actors.** Для сохранения любимых жанров и актеров существуют эти таблицы.

**Листинг 6** – Создание связочных таблицы.

```
0  create table if not exists user_genres(  
1      user_id int references users(user_id),  
2      genre_id int references genres(genre_id)  
3  );  
4  
5  create table if not exists user_actors(  
6      user_id int references users(user_id),  
7      actor_id int references actors(actor_id)  
8  );
```

### 3.5 Создание пользователей системы

Для того, чтобы обеспечить безопасность доступа к данным необходимо создать пользователей с соответствующими правами.

#### Гость

**Листинг 7** – Создание пользователя guest.

```
0 create role guest with login password 'guest';  
1 grant select on films, actors, genres, users to guest;
```

#### Авторизованный пользователь

**Листинг 8** – Создание пользователя common\_user.

```
0 create role common_user with login password 'user';  
1 grant select on films, actors, genres, users to common_user;  
2 grant all privileges on user_genres, user_actors to common_user;
```

#### Менеджер

**Листинг 9** – Создание пользователя admin.

```
0 create role admin with login password 'admin';  
1 grant all privileges on films, actors, genres, users, studios,  
directors, user\_genres, user\_actors to admin;
```

### 3.6 Функции

В Листинге 10 представлена реализация функции `get_recommended_films`.

**Листинг 10** – Реализация функции `get_recommended_films`.

```
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 $$
11     select film_name, release_date, budget, fees, avg_price
12     from users join user_actors
13     on $1 = user_actors.user_id
14     join actors on actors.actor_id = user_actors.actor_id
15     join films on films.film_id = actors.film_id
16     union
17     select film_name, release_date, budget, fees, avg_price
18     from users join user_genres
19     on $1 = user_genres.user_id
20     join genres on genres.genre_id = user_genres.genre_id
21     join films on films.genre_id = genres.genre_id
22 $$;
```

Данная функция нужна для получения информации о рекомендованных фильмах. Типы выходных параметров:

- 1) `film_name` - `varchar(40)`, так как хранит название фильма;
- 2) `release_date` - `date`, так как хранит дату выхода фильма;

- 3) budget - int, так как хранит бюджет фильма;
- 4) fees - int, так как хранит сумму сборов;
- 5) avg\_price - int, так как хранит среднюю цену просмотра фильма.

### **3.7 Разработка компонентов**

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

#### **3.7.1 Компонент доступа к данным**

На Рисунке 9 представлена UML-диаграмма компонента доступа к данным.

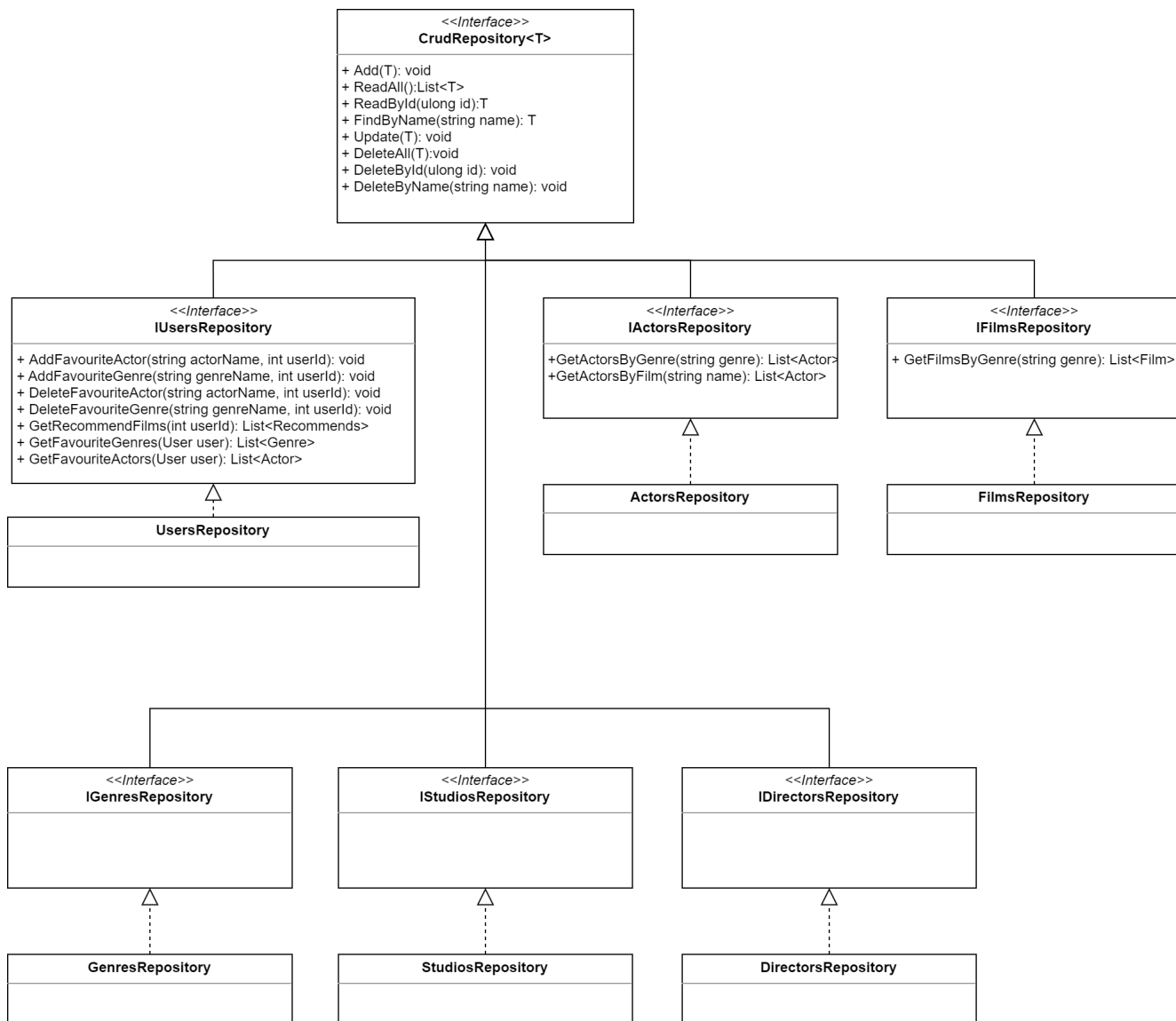


Рисунок 9 – Компонент доступа к данным.

### **3.7.2 Компонент бизнес-логики**

На Рисунке 10 представлена UML-диаграмма компонента бизнес-логики.

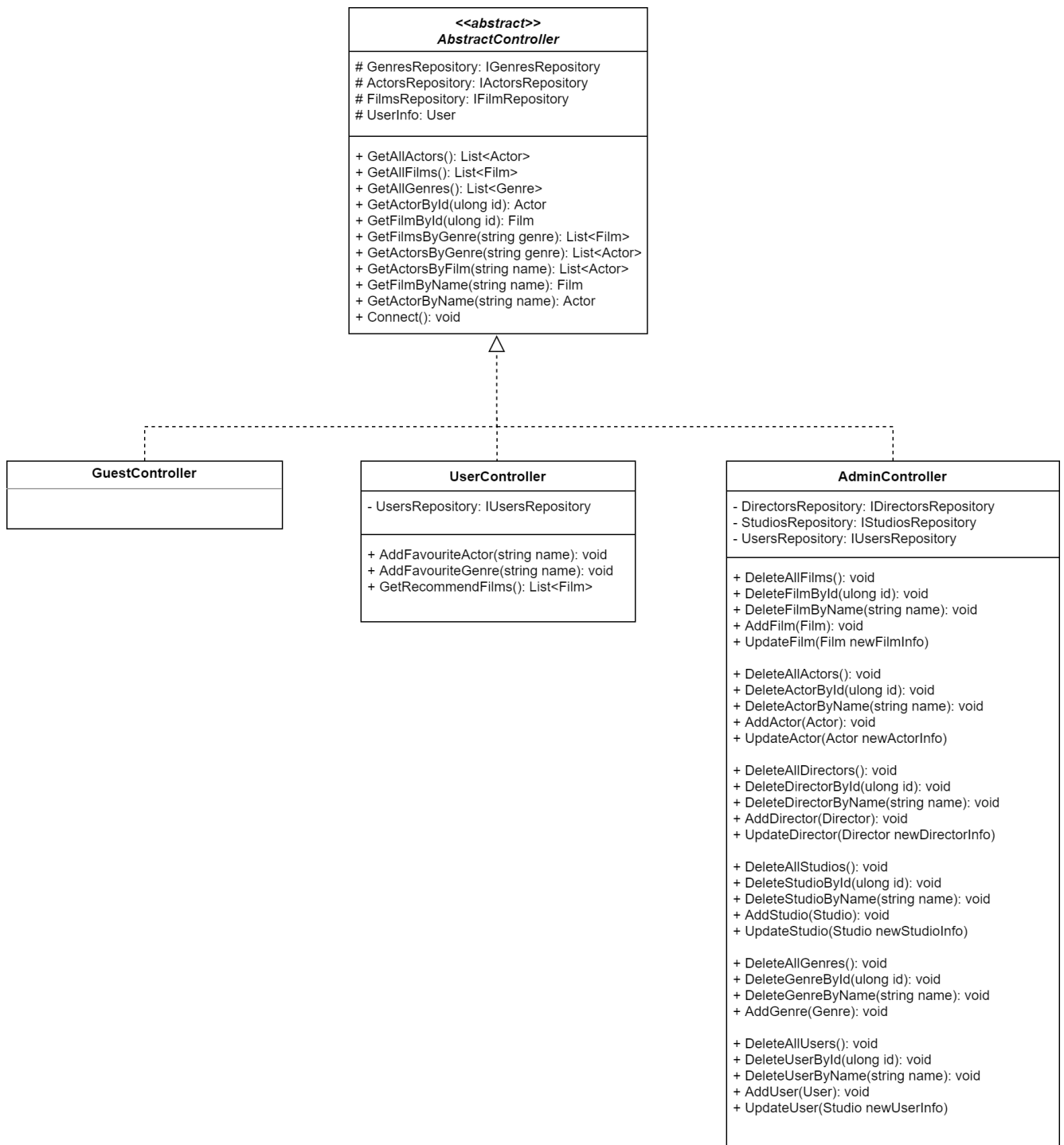


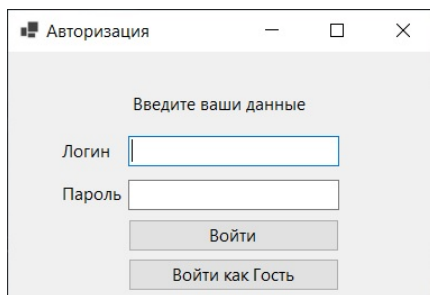
Рисунок 10 – Компонент бизнес-логики.



### 3.8 Интерфейс приложения

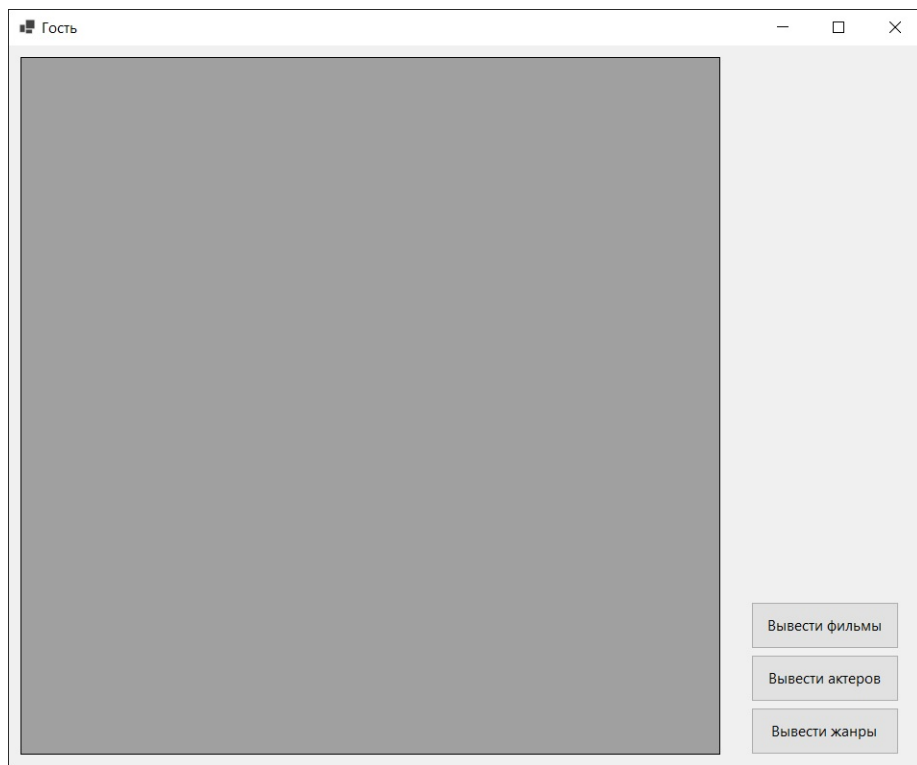
На Рисунках ниже показан интерфейс авторизации и интерфейсы пользователей (гость, авторизованный пользователь, администратор).

На Рисунке 11 изображено окно авторизации. Можно увидеть поля для ввода логина и пароля. Ниже изображена кнопка для входа. Так же есть возможность не вводить данные в поля и зайти как гость, для этого есть кнопка ниже "Войти".



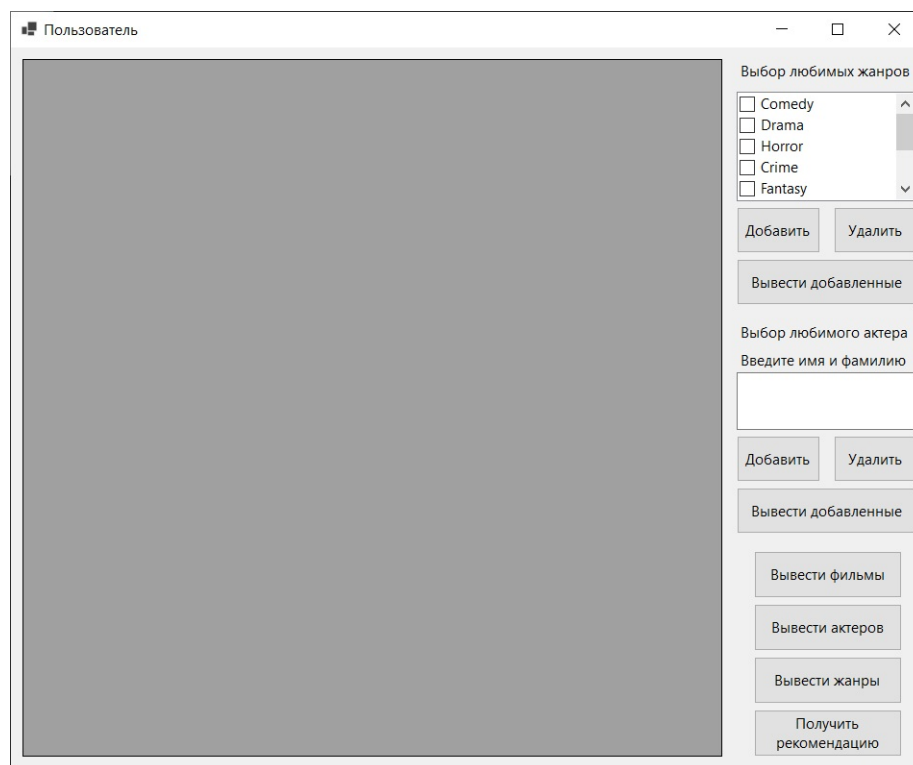
**Рисунок 11** – Окно авторизации.

На Рисунке 12 изображено окно гостя. Можно увидеть окно вывода информации слева и 3 кнопки. Кнопки предназначены для вывода информации о фильмах, актерах, жанрах соответственно.



**Рисунок 12** – Окно гостя.

На Рисунке 13 изображено окно авторизованного пользователя. Помимо составляющих, что были на окне гостя, здесь присутствует функционал добавления/удаления/вывода любимых жанров и актеров.



**Рисунок 13** – Окно авторизованного пользователя.

На Рисунке 14 изображено окно администратора. Слева под окном вывода находятся кнопки для показа информации о фильмах, актерах, режиссерах, студиях, жанрах, пользователях соответственно. Справа находится функционал для добавления и удаления фильмов, актеров, режиссеров, студий, жанров, пользователей соответственно.

The screenshot shows an 'Admin' window with a large grey area on the left and several form sections on the right. At the bottom left, there are six buttons: 'Вывести фильмы', 'Вывести актеров', 'Вывести режиссеров', 'Вывести студии', 'Вывести жанры', and 'Вывести пользователей'.

**Фильмы (Movies):**

- Form fields: Film ID, Studio ID, Genre ID, Director ID, Название (Title), Бюджет (Budget), Сборы (Box Office), Ср. цена (Avg. Price).
- Buttons: 'Добавить фильм в базу' (Add movie to database), 'Удалить фильм из базы' (Delete movie from database).

**Актеры (Actors):**

- Form fields: Actor ID, Film ID, Имя и фам. (Name and surname), Возраст (Age), Пол (Gender), Нация (Nationality), Гонорар (Honorarium), Наград (Awards).
- Buttons: 'Добавить актера в базу' (Add actor to database), 'Удалить актера из базы' (Delete actor from database).

**Режиссеры (Directors):**

- Form fields: Director ID, Имя и фам. (Name and surname), Возраст (Age), Пол (Gender), 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):**

- Form fields: Studio ID, Название (Title).
- Buttons: 'Добавить студию в базу' (Add studio to database), 'Удалить студию из базы' (Delete studio from database).

**Жанры (Genres):**

- Form fields: Genre ID, Название (Title).
- Buttons: 'Добавить жанр в базу' (Add genre to database), 'Удалить жанр из базы' (Delete genre from database).

**Пользователи (Users):**

- Form fields: User ID, Login, Пароль (Password), Права(0-2) (Rights(0-2)).
- Buttons: 'Добавить пользователя в базу' (Add user to database), 'Удалить пользователя из базы' (Delete user from database).

**Рисунок 14** – Окно администратора.

### 3.9 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 пользователи 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 пользователя 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://ru.bmstu.wiki/> (дата обращения: 20.05.2021).
- [2] СУБД PostgreSQL. Особенности и архитектура Postgres : [Электронный ресурс] URL: <https://otus.ru/nest/post/1584/> (дата обращения: 12.06.2021).
- [3] PostgreSQL : Документация [Электронный ресурс] URL: <https://postgrespro.ru/docs/postgresql> (дата обращения: 20.05.2021).
- [4] Документация по C# [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 20.05.2021).
- [5] Документация по семейству продуктов Visual Studio [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019> (дата обращения: 20.05.2021).
- [6] Windows Forms [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-5.0> (дата обращения: 20.05.2021).
- [7] Документация по Entity Framework [Электронный ресурс] URL: <https://docs.microsoft.com/ru-ru/ef/> (дата обращения: 20.05.2021).

## Приложение А. Презентация.