Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту на тему:
Поиск фильмов по параметрам и просмотр информации о них

Студент _____ Полякова К. А.
(Подпись, дата)

Руководитель курсового проекта _____ Филиппов М. В.
(Подпись, дата)

Москва 2020

Содержание

Введение **3**

1 Аналитический раздел **4**
1.1 Формализация задачи . . . 4
1.2 Общие сведения о базах данных и средствах управления базами данных . . . 4
1.3 Типы БД . . . 4
1.3.1 Иерархическая модель баз данных . . . 4
1.3.2 Сетевая модель баз данных . . . 5
1.3.3 Реляционная модель . . . 5
1.3.4 Сравнение моделей . . . 6
1.4 СУБД . . . 6
1.5 Framework . . . 7
1.6 Вывод . . . 7

2 Конструкторский раздел **8**
2.1 Проектирование диаграмм web-приложения . . . 8
2.2 Проектирование таблиц базы данных . . . 10
2.3 Проектирование системы изменения данных . . . 11
2.3 Проектирование регистрации и аутентификации пользователя . . . 11
2.4 Вывод . . . 11

3 Технологическая часть **12**
3.1 Выбор инструментов разработки . . . 12
3.2 Реализация хранения данных . . . 12
3.3 Frontend-разработка . . . 13
3.4 Интерфейс приложения . . . 15
3.5 Вывод . . . 21

Заключение **22**

Список используемой литературы **23**

Введение

Еще пару десятков лет назад, чтобы посмотреть какой-либо фильм люди посещали кинотеатры или смотреть телевизор. Но тогда выбор был очень ограниченный - приходилось смотреть только то, что шло в кинотеатрах или показывали по телевизору.

Современный же человек не имеет таких ограничений. С появлением Интернета огромное количество различных фильмов, сериалов, мультфильмов доступны каждому прямо из дома. Необходимо заметить, что и количество кинофильмов увеличилось в несколько раз. Как же не запутаться в таком многообразии и выбрать именно то, что нравится? Как раз для этого необходимы приложения для поиска по различным параметрам и для просмотра необходимой информации, чтобы найти определенный фильм.

Целью данной курсовой работы является создание клиент–серверного приложения «ХочуПосмотреть!», которое предоставляет возможность просмотра информации о различных фильмах. Также пользователь может добавить любые фильмы в Избранное для того, чтобы посмотреть их позже или пересмотреть. В приложении можно осуществить поиск по нескольким параметрам: по названию, по году премьеры, по стране или по жанру.

Актуальность разработки состоит в том, что с помощью такого приложения можно с легкостью найти информацию о фильме или добавить фильм в Избранное, чтобы не забыть его посмотреть.

Для выполнения цели необходимо выполнить следующие задачи:

- формализовать задачу в виде определения необходимого функционала;
- провести анализ существующих средств управления базами данных (СУБД);
- спроектировать базу данных, необходимую для хранения и структурирования данных;
- реализовать спроектированную базу данных с использованием выбранной СУБД;
- реализовать приложение для взаимодействия с реализованной базой данных.

1 Аналитический раздел

В данном разделе будут рассмотрены общие сведения о базах данных и средствах управления базами данных, типы баз данных и используемые framework'и.

1.1 Формализация задачи

В соответствии с техническим заданием на курсовой проект пользователям необходима система авторизации и регистрации для предоставления индивидуальной информации.

Также должна быть предусмотрена возможность поиска и просмотра информации о фильмах, а также добавление в Избранное и удаление фильмов из Избранного.

1.2 Общие сведения о базах данных и средствах управления базами данных

База данных представляет собой совокупность определенным образом организованных данных, которые хранятся в памяти вычислительной системы и отображают состояние объектов и их взаимосвязи в рассматриваемой предметной области.
Под системой управления базами данных (СУБД) понимается совокупность программных и языковых средств, предназначенных для создания и обработки БД.

1.3 Типы БД

Модель данных определяет логическую структуру БД и то, каким образом данные будут храниться, организовываться и обрабатываться.

Существует 3 типа моделей организации данных:

- иерархическая модель БД;
- сетевая модель БД;
- реляционная модель БД.

1.3.1 Иерархическая модель баз данных

Иерархическая модель БД представляет собой древовидную (иерархическую) структуру, состоящую из объектов (данных) различных уровней. Каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок.

Иерархической базой данных является файловая система, состоящая из корневого каталога, в котором имеется иерархия подкаталогов и файлов.

Связи записей реализуются в виде физических указателей с одной записи на другую. Основной недостаток иерархической структуры – невозможность реализовать отношения "многие-ко-многим а также ситуации, в которых запись имеет несколько предков. [1]

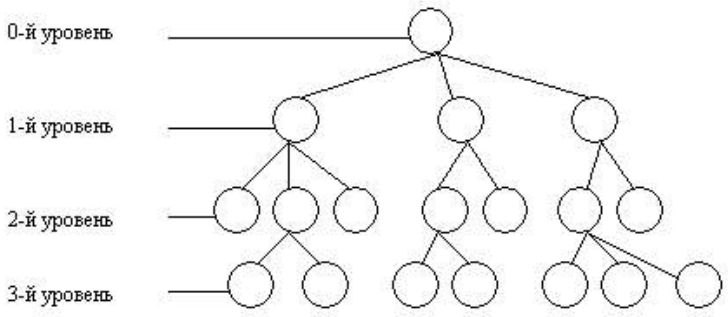

Рис. 1: Иерархическая модель баз данных

1.3.2 Сетевая модель баз данных

Сетевая модель данных является расширением иерархического подхода. Разница между иерархической моделью данных и сетевой заключается в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре у потомка может быть любое число предков. Записи в такой модели связаны списками с указателями.

Примером сетевой СУБД является IDMS (интегрированная система управления данными) от компании Computer Associates international Inc.

Популярность сетевой модели совпала с популярностью иерархической модели. Некоторые данные намного естественнее моделировать с несколькими предками для одного дочернего элемента. Сетевая модель позволяла моделировать отношения «многие-ко-многим».

И хотя эта модель широко применялась на практике, она так и не стала доминантной по двум основным причинам. Во-первых, компания IBM решила не отказываться от иерархической модели в расширениях для своих продуктов, таких как IMS и DL/I. Во-вторых, через некоторое время ее сменила реляционная модель, предлагавшая более высокоуровневый, декларативный интерфейс. [1]

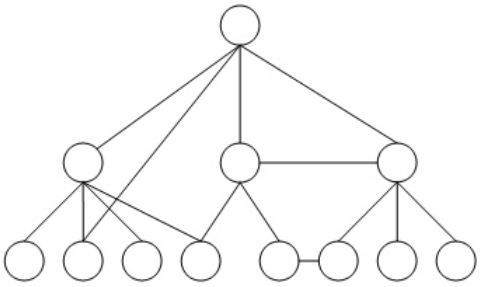

Рис. 2: Сетевая модель БД

1.3.3 Реляционная модель

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. Объекты и их отношения представлены таблицами.

В реляционных моделях нет необходимости просматривать все указатели, что облегчает выполнение запросов на выборку информации по сравнению с сетевыми и иерархическими БД. Это одна из основных причин, почему реляционная модель оказалась более удобна.

Рис. 3: Пример реляционной БД

Распространенные реляционные СУБД: MySql, PostgreSql, Access, Oracle, DB2, MS-SQL Server, SQLite. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все элементы в одном столбце имеют одинаковый тип;

- каждый столбец имеет уникальное имя;
- одинаковые строки (записи, кортежи) в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Каждое поле содержит одну характеристику объекта предметной области. В записи собраны сведения об одном экземпляре этого объекта.

Некоторые поля могут быть определены как ключевые. Это значит, что для ускорения поиска конкретных значений будет использоваться индексация. Когда поля двух различных таблиц получают данные из одного набора, можно использовать оператор JOIN для выбора связанных записей двух таблиц, сопоставив значения полей. Такие действия можно расширить до объединения нескольких полей в нескольких таблицах. Поскольку отношения здесь определяются только временем поиска, реляционные базы данных классифицируются как динамические системы.

1.3.4 Сравнение моделей

Иерархическая модель данных поддерживает отношения типа «один-к-одному» или «один-ко-многим». Она позволяет быстро получать данные, но не отличается гибкостью. Иногда роль элемента (родителя или потомка) неясна и не подходит для иерархической модели.

Вторая, сетевая модель данных, имеет более гибкую структуру, чем иерархическая, и поддерживает отношение «многие ко многим». Но быстро становится слишком сложной и неудобной для управления.

Третья модель – реляционная – более гибкая, чем иерархическая и проще для управления, чем сетевая. Реляционная модель сегодня используется чаще всего, так как имеет множество преимуществ, таких как:

- простота использования;
- гибкость;
- независимость данных;
- безопасность;
- простота практического применения;
- слияние данных;
- целостность данных.

В связи с этим далее будет рассматриваться реляционная модель. Теперь необходимо рассмотреть систему управления такой моделью.

1.4 СУБД

Самых популярные системы управления реляционными базами данных:

- MySQL;
- PostgreSQL;
- SQLite.

В данном проекте будет рассмотрена СУБД SQLite.

Это компактная встраиваемая СУБД. Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, компонующуюся с программой, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу.

Однако SqLite популярна скорее в случаях, когда не требуется выносить базу данных на отдельную машину и данные требуется хранить в рамках одной операционной системы. Будучи файловой БД, она предоставляет отличный набор инструментов для более простой (в сравнении с серверными БД) обработки любых видов данных. [2]

Преимущества:

- **Файловая.** Вся БД хранится в одном файле, что облегчает перемещение.
- **Стандартизированная.** SQLite использует SQL, некоторые функции не используются (RIGHT OUTER JOIN или FOR EACH STATEMENT);
- **Отсутствие пользовательского.** Продвинутые БД предоставляют пользователям возможность управлять связями в таблицах в соответствии с привилегиями, но у SQLite такой функции нет.
- **Невозможность дополнительной настройки.** SQLite нельзя сделать более производительной, путем изменения настроек.

1.5 Framework

SQLite совместима со множеством фреймворков, которые содержат в себе требуемые методы обращения к БД.

В качестве web-framework был выбран Django, который предоставляет все необходимые инструменты для создания подобного проекта, так как предоставляет возможность для написания как frontend, так и backend для полноценного запуска приложения.

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC [3]. Проект поддерживается организацией Django Software Foundation.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других. Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных. [5]

1.6 Вывод

В результате проведенного анализа в качестве модели данных была выбрана реляционная модель, в качестве средства управления базами данных – SQLite.

Таким образом, подобрав необходимый набор инструментов для реализации web-приложения, можно приступить к проектированию решения поставленной задачи.

2 Конструкторский раздел

В соответствии с техническим заданием и аналитическим разделом должно быть получено полноценное приложение для взаимодействия с базой данных.

2.1 Проектирование диаграмм web-приложения

На рисунке 4 представлена Use Case диаграмма приложения. Можно заметить, что на диаграмме два актора: администратор и пользователь. Администратор взаимодействует с базой данных, добавляет и удаляет записи.

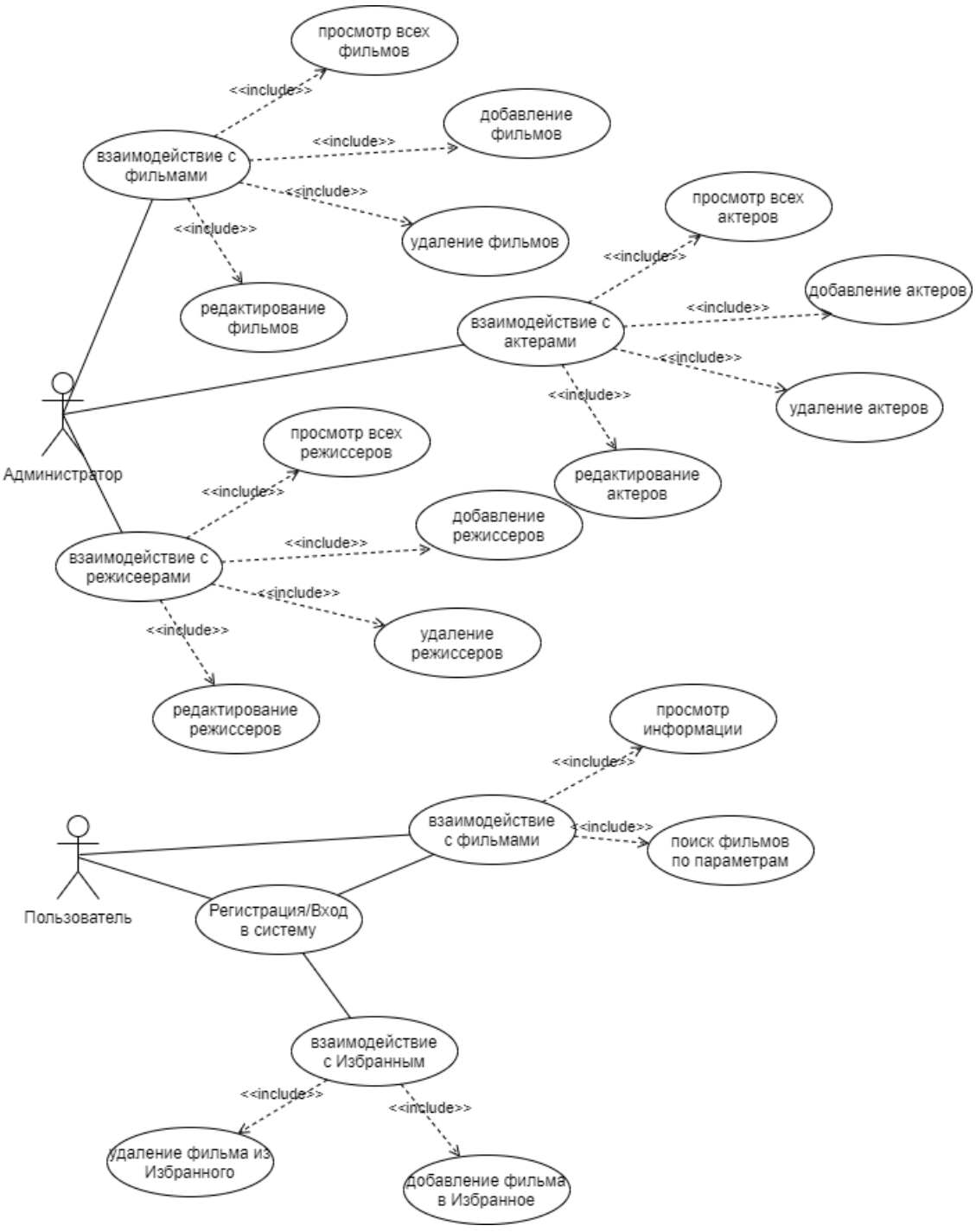

Рис. 4: Use Case диаграмма

На рисунке 5 продемонстрирована ER модель приложения. Можно рассмотреть, какие сущности выделены, какая связь между данными ними и какие атрибуты присутствуют у каждой сущности.

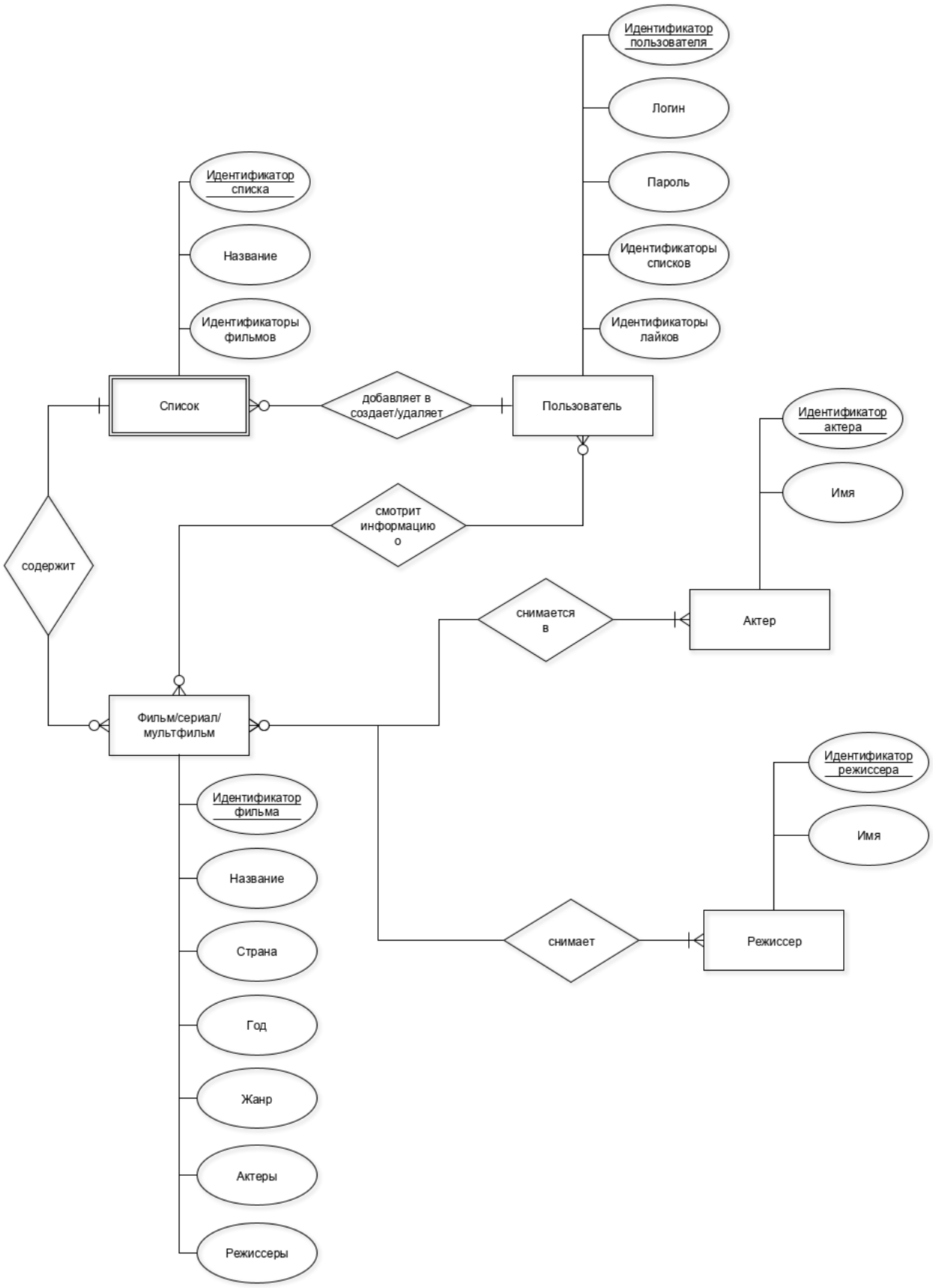

Рис. 5: Er модель

2.2 Проектирование таблиц базы данных

База данных приложения «ХочуПосмотреть!» состоит из следующих таблиц:

- таблица пользователей сайта **User**;
- таблица актеров **Actor**;
- таблица режиссеров **Director**;
- таблица фильмов **Film**;
- таблица избранных фильмов **List**;

Таблица User

Позволяет однозначно идентифицировать пользователя сайта, реализовать авторизацию пользователя. Имеет связь «один-ко-многим» с таблицей List.

Таблица содержит следующие поля:

- username - символьное поле, имя пользователя;
- email - символьное поле, адрес электронной почты клиента;
- password hash - символьное поле, хэш пароля пользователя;

Таблица Actor

Хранит данные о актере. Связана с таблицей фильмов Film «многие-ко-многим».

Таблица содержит следующие поля:

- actor_id - целочисленное поле, идентификатор актера;
- name - символьное поле, имя актера;

Таблица Director

Хранит данные о режиссере. Связана с таблицей фильмов Film «многие-ко-многим».

Таблица содержит следующие поля:

- actor_id - целочисленное поле, идентификатор режиссера;
- name - символьное поле, имя режиссера;

Таблица Film

В данной таблице хранятся данные о фильмах.

Таблица содержит следующие поля:

- id – целочисленное поле, идентификатор фильма;
- name – символьное поле, название фильма;
- description – текстовое поле, описание фильма;
- country - символьное поле, название страны;
- genre - целочисленное поле, жанр фильма;
- actors – поле, в котором хранятся id актеров, которые снимаются в этом фильме;
- directors – поле, в котором хранятся id режиссеров, которые снимали этот фильм;

Таблица List

Таблица, содержащая фильмы, которые находятся в избранном у пользователя. С помощью связи «один-к-одному» связана с таблицей пользователей User.

Таблица содержит следующие поля:

- id – целочисленное поле, идентификатор списка избранного;
- user_id – символьное поле, username пользователя;
- films – поле, в котором содержатся id фильмов, которые входят в данный список избранного;

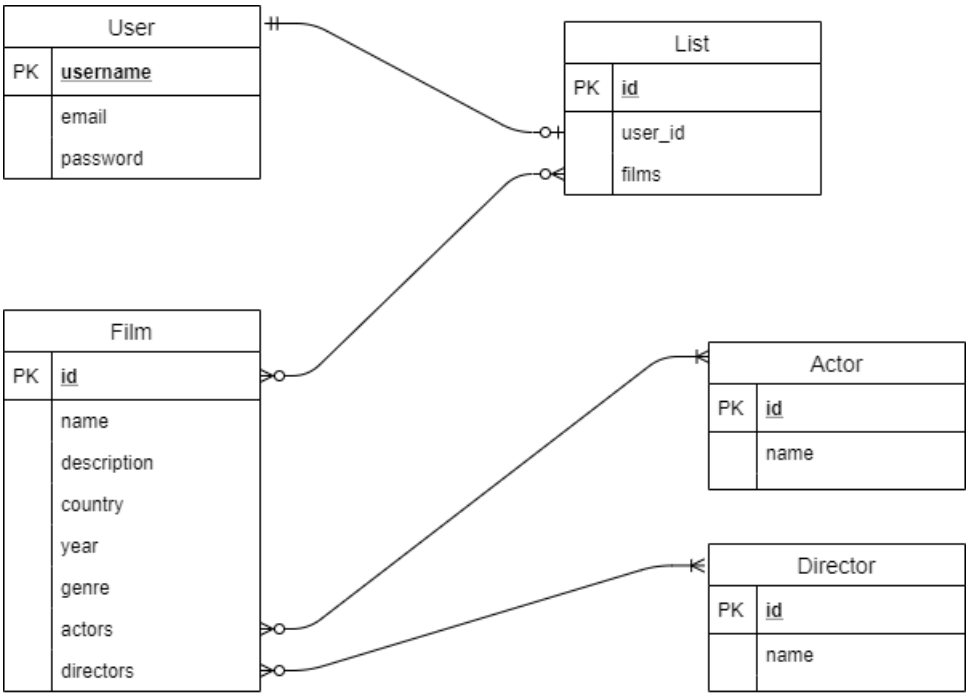

Рис. 6: Диаграмма БД

2.3 Проектирование системы изменения данных

В Django каждую таблицу представляет класс. Он отображает информацию о данных сущности. Он содержит поля и поведение данных. С помощью стандартных методов осуществляются операции добавления, удаления и обновления записей.

2.3 Проектирование регистрации и аутентификации пользователя

Регистрация пользователя в приложении является добавлением в базу данных (в таблицу User) записи, содержащей необходимую информацию для аутентификации. Для этого пользователь вводит соответствующие данные в поля регистрационной формы.

Framework Django предоставляет собой набор данных базовых инструментов для реализации web-приложения. В этот функционал включена реализация аутентификации пользователя.

2.4 Вывод

Была разработана модель приложения, которое дает возможность просматривать информацию фильмов, искать по базе данных, а так же реализовывать регистрацию, аутентификацию и авторизацию пользователей.

3 Технологическая часть

После проектирования структуры поставленной задачи, требуется реализовать набор функций, необходимый для создания web-приложения, а также конкретизировать полный список инструментов, используемых для запуска приложения.

3.1 Выбор инструментов разработки

В ходе реализации были использованы следующие технологии и средства:

- язык программирования Python;
- СУБД SQLite;
- библиотека Django.

Такой набор инструментов был выбран, потому что для каждого из элементов предусмотрено взаимодействие с другими. Также данные инструменты полностью выполняют задачи, необходимые для реализации проекта.

Использован компьютер с операционной системой Windows 10.

3.2 Реализация хранения данных

Для работы с базой данных требуется объявить классы таблиц БД.

В Django существует пользовательская модель User, которая и использовалась в данном курсовом проекте. Поэтому не было необходимости в написании собственного класса пользователя.

Листинг 1: Класс «Актер»

```
1 class Actor(models.Model):
2   name = models.CharField(max_length = 150, unique = True)
3
4   def __str__(self):
5     return self.name
6
7   def save(self, *args, **kwargs):
8     print("Actor is saved!")
9     super().save(*args, **kwargs)
```

Листинг 2: Класс «Режиссер»

```
1 class Director(models.Model):
2   name = models.CharField(max_length = 150, unique = True)
3
4   def __str__(self):
5     return self.name
6
7   def save(self, *args, **kwargs):
8     print("Director is saved!")
9     super().save(*args, **kwargs)
```

Листинг 3: Класс «Фильм»

```
class Film(models.Model):
    name = models.CharField(max_length = 100)
    description = models.TextField()
    country = models.CharField(max_length = 100)
    year = models.PositiveSmallIntegerField()
    genre = models.IntegerField(choices = GENRE)
    actors = models.ManyToManyField(Actor)
    directors = models.ManyToManyField(Director)

    def __str__(self):
        return self.name

    def save(self, *args, **kwargs):
        print("Film is saved!")
        super().save(*args, **kwargs)

    def getactors(self):
        return self.actors

    def getderecors(self):
        return self.directors
```

Листинг 4: Класс «Список»

```
class List(models.Model):
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    films = models.ManyToManyField(Film)

    def addtolist(self, film_id):
        self.films.add(film_id)
        self.save()

    def deletefromlist(self, film_id):
        self.films.remove(film_id)
        self.save()

    def save(self, *args, **kwargs):
        print("List is saved!")
        super().save(*args, **kwargs)
```

3.3 Frontend-разработка

Пользовательский интерфейс при разработке web-приложения представляет из себя полноценную верстку проекта. Для этого использовалась технология Bootstrap.

Bootstrap - это инструментарий с открытым исходным кодом для разработки web-приложений с помощью HTML, CSS и JS. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. С помощью него настроен дизайн сайта. [4]

Django предоставляет инструмент шаблонизатора, который дает возможность вносить динамические данные в html с backend. С помощью шаблонизатора есть возможность проверять данные, изменяя элементы страницы в зависимости от результата проверки.

При рендеринге шаблона переменные в двойных фигурных скобках будут заменяться на вычисленные значения.

Продемонстрируем действие шаблонизатора на примере шаблона вывода результатов поиска. Шаблон проверяет с помощтю шаблонизатора {% if object_list %}, есть ли результат поиска, если есть, то с помощью шаблонизатора {% for film in object_list %} выводит на страницу фильмы. Если же результата нет, то пользователь увидит надпись "Ничего не найдено".

Листинг 5: Шаблон search_results.html

```
{% extends "index.html" %}
{% load staticfiles %}
<title>{% block title %}Films:{% endblock %}</title>
{% block main %}
<h2>Search results:</h2>
<div class="results">
  <ul>
  {% if object_list %}
    {% for film in object_list %}
    <li>
      <a href="../film{{ film.id }}">{{ film.name }}</a>
    </li>
    {% endfor %}
  {% else %}
    <h3>Nothing is found(</h3>
  {% endif %}
  </ul>
</div>
{% endblock %}
```

Все шаблоны используют базовый шаблон {% extends "index.html"%} для избежания дублирования кода.

Листинг 6: Шаблон search_results.html

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>{% block title %}New in cinemaworld:{% endblock %}</title>
  {% block style %}<link type="text/css" href="{% static 'style.css' %}" rel="stylesheet" >{% endblock %}
  <link href="https://fonts.googleapis.com/css2?family=Open+Sans" rel="stylesheet">
</head>
<body>
  <header>
    <div class="headerlogo">IWnannaSeeIt!</div>
    <nav>
      <div class="topnav" id="myTopNav">
      <a href="../about">ABOUT</a>
      <a href="../search">SEARCH</a>
      <a href="../app">HOME</a>
      {% if user.is_authenticated %}
      <a href="">{{ user.username }}</a>
      <a href="../saved">SAVED</a>
      <a href="{%url 'logout' %}">LOGOUT</a>
      {% else %}
      <a href="{% url 'login' %}">LOGIN</a>
      <a href="{% url 'register' %}">REGISTER</a>
      {% endif %}
      </div>
    </nav>
  </header>
  <div class="main">
    {% block main %}
    {% endblock %}
  </div>
</body>
</html>
```

3.4 Интерфейс приложения

Незарегистрированные пользователи могут просматривать информацию о фильмах и совершать поиск по различным параметрам. У зарегистрированного пользователя, помимо перечисленных функций есть также возможность добавлять понравившиеся фильмы в избранное.

На рисунке 7 прлдемонстрирована форма регистрации пользователя, а на рисунке 8 - форма авторизации.

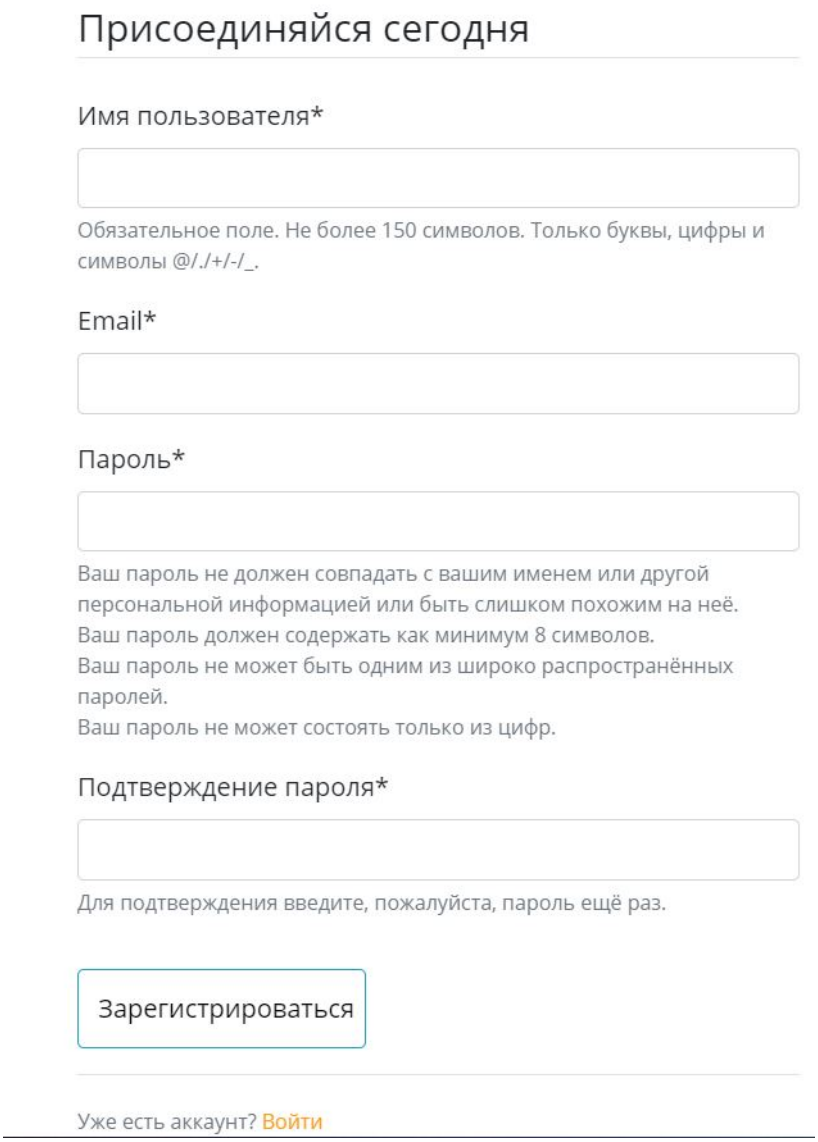

Рис. 7: Страница регистрации

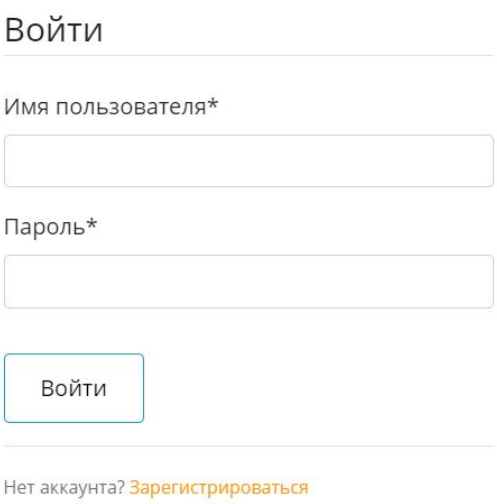

Рис. 8: Страница авторизации

На рисунках 9 и 10 продемонстрирована главная страница приложения, где можно увидеть новинки кино и ссылки на эти фильмы для просмотра информации о них. Единственное отличие между данными рисунками, это то, что на 9 главная страница незарегистрированного пользователя, можно заметить ссылки на авторизацию и регистрацию в правом верхне углу. На рисунке 10 пользователь авторизирован, поэтому перечисленных ранее ссылок нет, вместо этого можно увидеть имя пользователя, ссылку на Избранное и кпопку выхода из учетной записи.

Рис. 9: Главная страница незарегистрированного пользователя

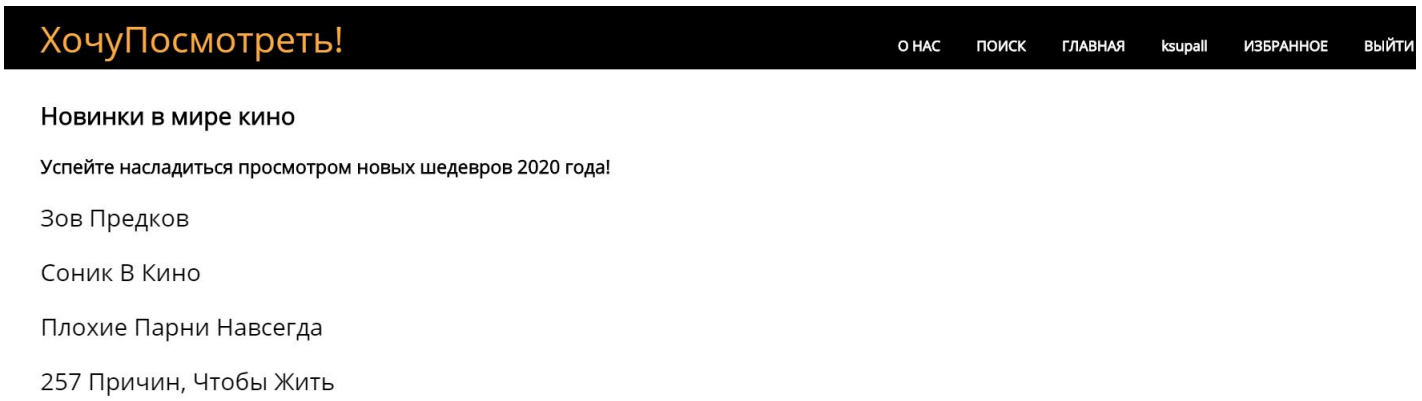

Рис. 10: Главная страница зарегистрированного пользователя

На рисунке 11 приведен пример, когда раздел "Избранное"пуст. Шаблонизатор, о котором говорилось выше, выводит сообщение "Нет фильмов!".

Рис. 11: Раздел "Избранное"пуст

На рисунке 12 продемонстрирована форма поиска. Данная функция доступна как зарегистрированным пользователям, так и незарегистрированным. Форма состоит из нескольких текстовых полей, а так же радиокнопок для выбора жанра. Поиск можно выполнить как по одному параметру, например, по жанру, так и по нескольким параметрам сразу. Если же никакие параметры введены не будут, то поиск выдаст все фильмы, которые находятся в базе данных. Для очистки полей рядом с кнопкой "Искать"расположена кнопка "Очистить".

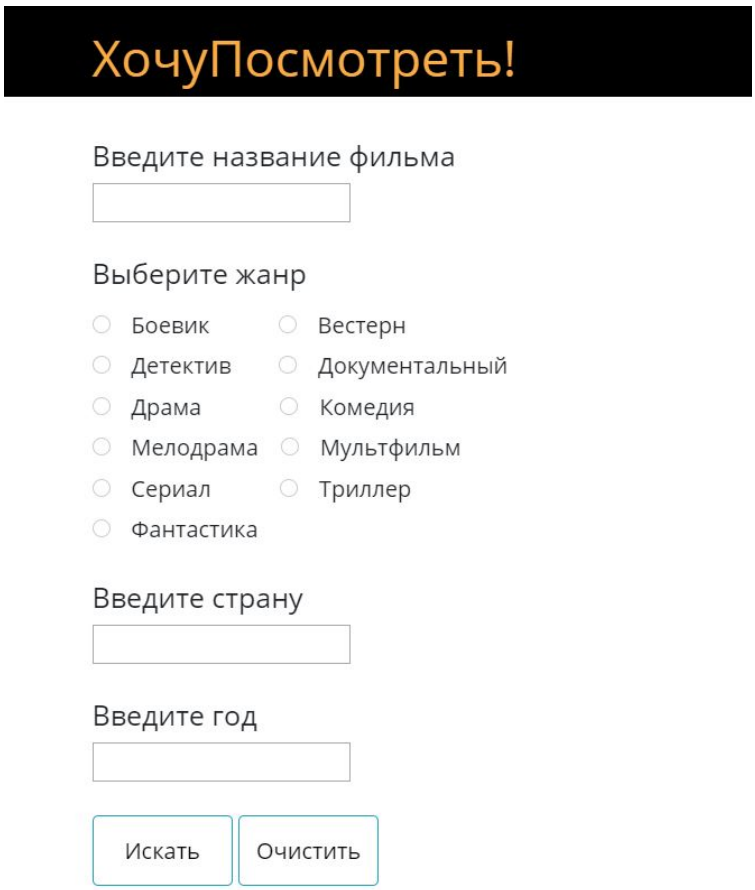

Рис. 12: Форма поиска

На рисунке 13 представлен результат поиска. В качестве результата можно увидеть список фильмов, которые были подобраны по некоторым параметрам. Список состоит из ссылок, поэтому после поиска можно посмотреть более подробную информацию о каждом фильме.

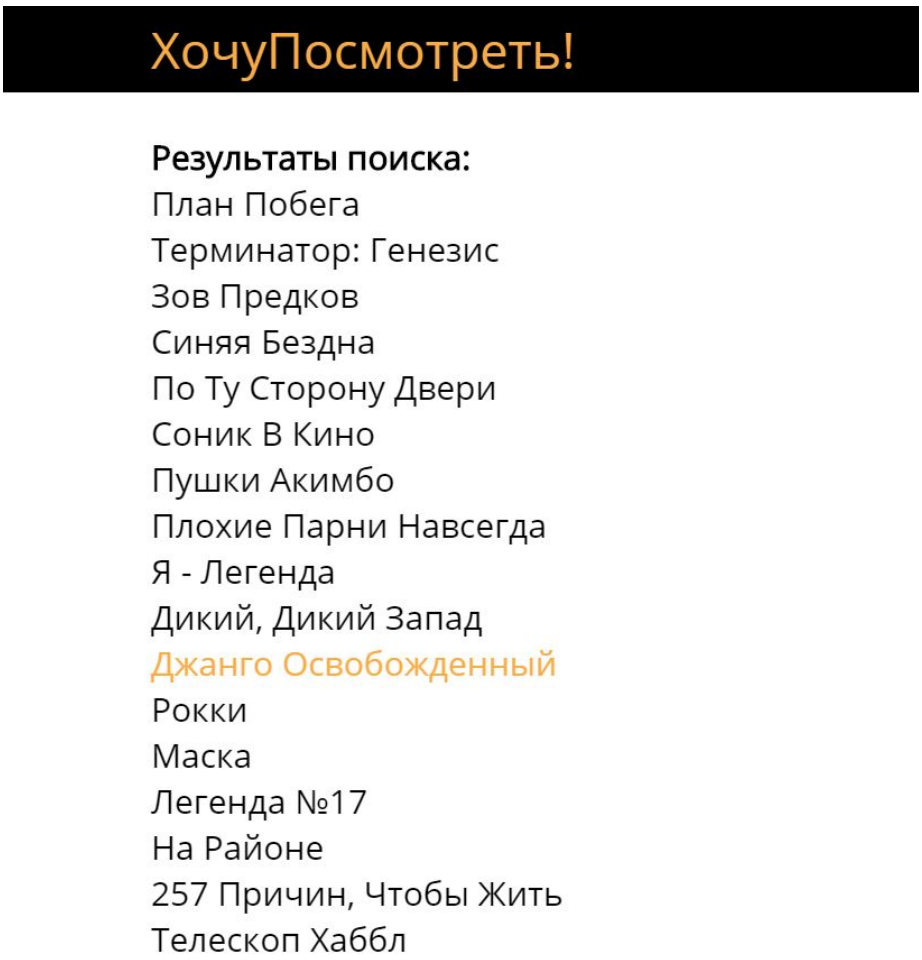

Рис. 13: Результат поиска

На рисунке 14 представлена страница фильма. Здесь находится вся информация о фильме: название, описание, страна, год премьеры, жанр, режиссер и актеры, которые в нем снимались. Ниже данной информации можно увидеть кнопку "Добавить она выполняет добавление данного фильма в избранное. Если этот фильм уже есть в избранном, то вместо кпопки "Добавить"будет кпопка "Удалить". Страница с фильмом, который уже находится в Избранном, продемонстрирована на рисунке 15. Однако данная функция добавления в Избранное доступна только зарегистрированным пользователям, поэтому у незарегистрированных пользователей кнопок не будет вовсе.

Рис. 14: Страница фильма с кнопкой "Добавить"

Рис. 15: Страница фильма с кнопкой "Удалить"

3.5 Вывод

В данной части был выбран набор инструментов был для реализации поставленной задачи. Также были приведены листинги классов для формирования таблиц базы данных. В данной части был рассмотрен интерфейс приложения и основные его функции.

Заключение

В результате проделанной работы:

- были продуманы функции, которые должно было решать приложение;
- проведен анализ инструментов, необходимых для проектирования и реализации задачи, в результате которого был выбраны такие инструменты, как SQLite, Flask;
- разработана структура базы данных, состоящая их нескольких сущностей;
- с помощью выбранных инструментов был реализован web-интерфейс, обладающий возможностью регистрировать пользователей, изменять состояние и местоположение книг, добавлять и удалять книги, а также осуществлять поиск по ключевым словам.

Список литературы

[1] Интернет технологии [Электронный ресурс] URL: https://www.internet-technologies.ru/articles/modeli-...upravleniya-bazami-dannyh.html (Дата обращеня: 28.05.2020)

[2] Документация Sqlite [Электронный ресурс] URL: https://www.sqlite.org (Дата обращеня: 28.05.2020)

[3] MVC [Электронный ресурс] URL: https://tproger.ru/articles/mvc/ (Дата обращеня: 28.05.2020)

[4] Bootstrap [Электронный ресурс] URL: https://bootstrap-4.ru (Дата обращеня: 19.09.2019)

[5] Документация Django [Электронный ресурс] URL: https://docs.djangoproject.com/en/3.0/ (Дата обращеня: 28.05.2020)