

**Федеральное государственное автономное образовательное учреждение высшего
образования
“Национальный исследовательский университет “Высшая школа экономики”
Факультет компьютерных наук**

**ОТЧЕТ
по дисциплине
“Управление данными в интеллектуальных системах”**

Выполнили студенты
Антонова Евгения БПМИ209
Наумова Евгения БПМИ209
Копцева Александра БПМИ209

Проверил:

17.12.2022

2022 год

Содержание

1. Описание предметной области
 - 1.1. Цель
 - 1.2. Внешние данные
 - 1.3. Основные сценарии использования
2. Концептуальная модель
 - 2.1. Диаграмма “Сущность-связь”
 - 2.2. Описание сущностей и связей
3. Инфологическая модель
 - 3.1. Диаграмма “Таблица-связь”
 - 3.2. Словарь данных
4. Дatalogическая модель
 - 4.1. Используемая СУБД и диалект SQL
 - 4.2. DDL-скрипты
5. Клиентское приложение
 - 5.1. Архитектура
 - 5.2. Сценарии использования
 - 5.3. Организация доступа к данным
 - 5.4. Интерфейс с пользователем
 - 5.5. Отчеты
6. Заключение
 - 6.1. Объемные характеристики разработки
 - 6.2. Авторский вклад и комментарии по выполнению проекта
7. Источники

1. Описание предметной области

На сегодняшний день существует множество сайтов, где можно посмотреть информацию о фильмах и выбрать подходящие именно Вам, однако в Телеграмм подобных программ в виде ботов не особо много. В тех сервисах, которые существуют на данный момент, доступна лишь часть функционала, который может быть интересен пользователю, а использования всех возможностей необходимо пользоваться сайтом или отдельным приложением.

1.1 Цель

Целью данного проекта было создание удобного бота в Телеграмм, который мог бы обеспечить удобный поиск фильмов по пожеланиям.

1.2 Внешние данные

Для эффективного пользования данным сервисом планируется генерирование исходной базы данных, в которой хранятся некоторые фильмы. Данные были получены из открытых источников и прописаны вручную, из-за чего их не особо много, но это лишь демонстрационный вариант. Далее возможно подключение базы данных больших размеров.

1.3 Основные сценарии использования

Для регистрации в реализованном Телеграмм-боте необходимо перейти в него и отправить команду `/start`, после чего пользователю будет направлена captcha из пяти символов для подтверждения личности. При её корректном введении у пользователя появляется следующий функционал:

- Просмотреть топ-5 фильмов в текущем рейтинге
Для использования данной функции необходимо прописать сообщение `/top5`, после чего в ответ приходит сообщение со списком фильмов, идущих первыми в рейтинге, и соответственно рейтингом для каждого из них.
- Оценить конкретный фильм по 10-балльной шкале
Для запуска данной функции необходимо отправить сообщение вида `/rating <название фильма> <оценка>`. В таком случае рейтинг автоматически пересчитывается в базе данных и изменяется.
- Найти фильмы, в которых снимался интересующий актер
В данном случае пользователь должен послать сообщение вида `/actor <имя актера>`, в ответ пользователю отправляется список подходящих фильмов.
- Найти фильмы, снятые данным режиссером
Для запуска этой функции требуется отослать сообщение вида `/director <имя режиссёра>`. Функция работает аналогично функции `/actor`.
- Найти фильмы по жанру

В таком случае бот ожидает сообщение `/genre <название жанра>` и также отправляет список фильмов соответствующим жанром.

- Добавить интересующий фильм в WishList

Если пользователю понравился какой-то фильм, у него есть возможность отправить сообщение `/add <название фильма>` для добавления его в список любимых / планируемых к просмотру фильмов.

- Просмотреть свой WishList

Данная функция может использоваться с помощью команды `/favourite` для просмотра фильмов, которые пользователь уже отобрал.

- Посмотреть информацию о фильме

Данная функция может использоваться с помощью команды `/info` для просмотра информации о фильме: режиссер, актер, год выпуска, жанр, продолжительность и рейтинг.

- Получить информацию о боте

Для запуска данной функции необходимо отправить команду `/help`

2. Концептуальная модель

2.1 Диаграмма “Сущность-связь”

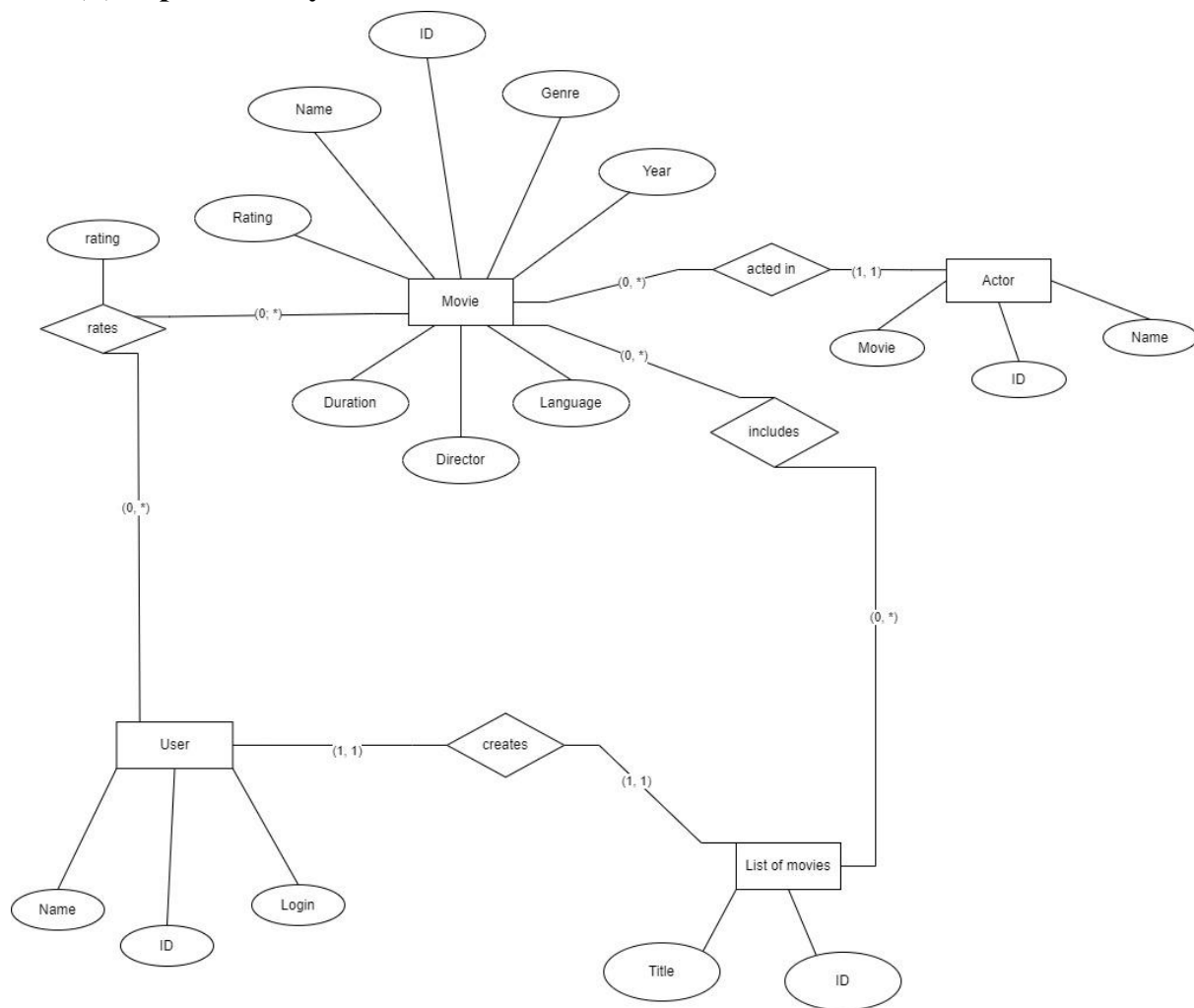


Рисунок 2.1 ER диаграмма в нотации Чена

2.2 Описание сущностей и связей

В нашей модели 4 сущности:

Сущность “User”

Содержит информацию о пользователе: его уникальный идентификатор, логин и имя

Сущность “Movie”

Содержит информацию о фильмах: уникальный идентификатор, название, язык, год выпуска, длительность, жанр, режиссёра и рейтинг

Сущность “Actor”

Содержит информацию об актёре: идентификатор, имя и фильм, где он снимался

Сущность “List of movies”

Содержит идентификатор, название фильмов и уникальный идентификатор пользователя, выбравшего фильм

Связей всего 4:

Связь “Acted in”

Связь один ко многим между актером и фильмом. Данная связь показывает в каком фильме снялся данный актер.

Связь “Includes”

Связь один ко многим между списком фильмов и фильмом.

Связь “Rates”

Связь один ко многим между пользователем и фильмом. Пользователь может оценить неограниченное число фильмов.

Связь “Creates”

Связь один к одному между пользователем и списком фильмов. Каждый пользователь может создать единственный WishList.

3. Инфологическая модель

3.1 Диаграмма “Таблица-связь”

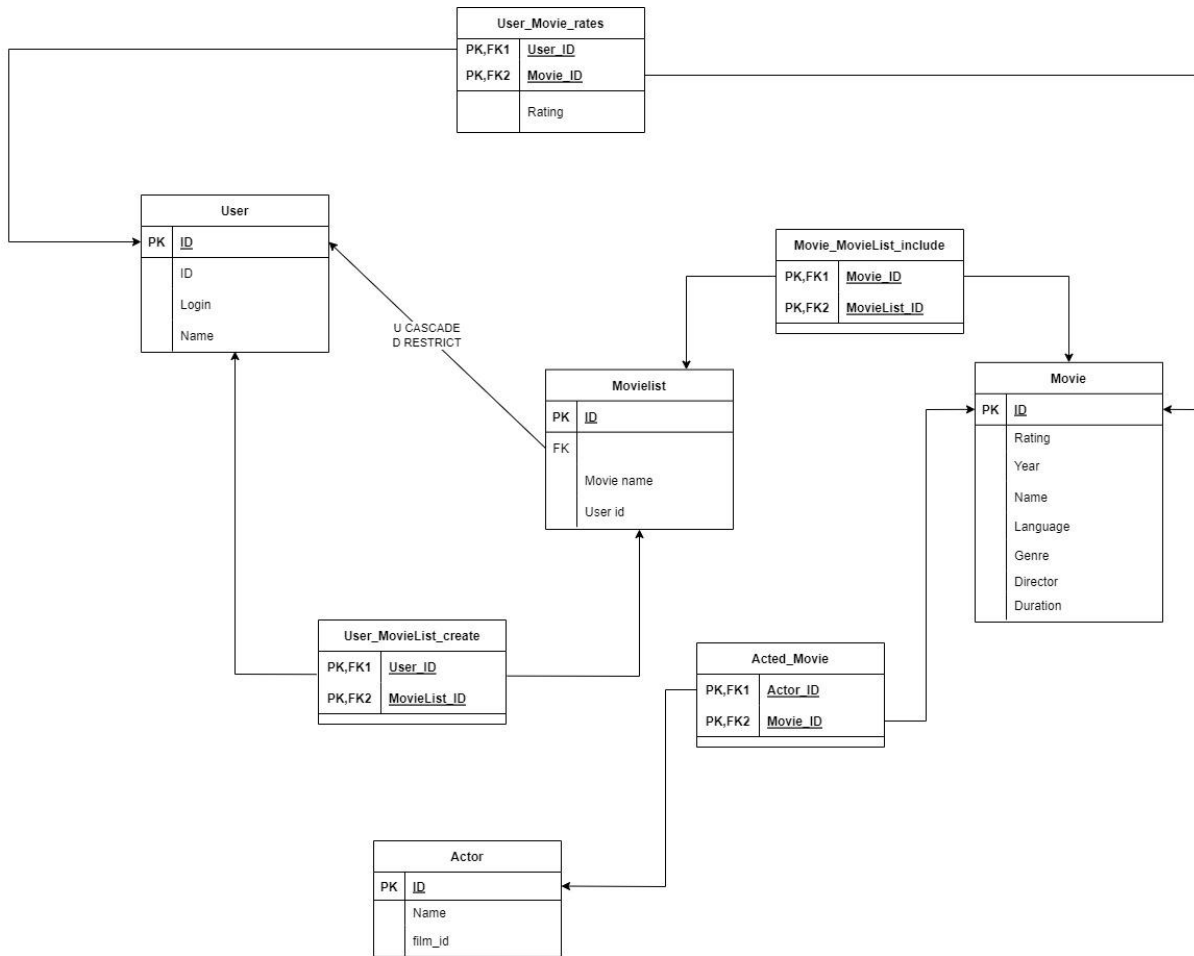


Рисунок 3.1 TR диаграмма

3.2 Словарь данных

- User – аккаунт человека, зарегистрированного на сервисе.
 - id – уникальный идентификатор
 - login – логин пользователя
 - name – имя пользователя
- Actor
 - id – уникальный идентификатор
 - name – имя актера
 - film – название фильма
- Movie
 - id – уникальный идентификатор
 - rating – рейтинг фильма
 - year – год выпуска фильма
 - language – язык оригинала
 - name – название фильма

- duration – длительность фильма
 - director – имя режиссера
 - genre – жанр фильма
- MovieList
 - id – уникальный идентификатор
 - creator_id – идентификатор пользователя, добавившего фильм
 - film_name – название фильма

4. Даталогическая модель

4.1. Используемая СУБД и диалект *SQL*

Использована СУБД PostgreSQL.

Использован диалект MySQL.

4.2 DDL-скрипты

Для создания таблиц был использован следующий скрипт:

```
create table if not exists bot_users (  
    uuid      uuid not null primary key,  
    telegram_id bigint not null,  
    chat_id   bigint not null,  
    first_name text not null,  
    last_name text,  
    tg_username text  
);  
  
create unique index if not exists bot_users_telegram_id  
    on bot_users (telegram_id);  
  
create unique index if not exists bot_users_chat_id  
    on bot_users (chat_id);  
  
  
create table if not exists films (  
    uuid      uuid not null primary key,  
    name      text not null,  
    director text not null,  
    rating    integer,  
    year      integer not null,  
    language text,  
    duration integer,  
    genre     text  
);  
  
create unique index if not exists films_name
```

```
on films (name);
```

```
create table if not exists actors (  
    uuid    uuid not null primary key,  
    actor   text not null,  
    film_name text not null  
        references films (name)  
        on delete cascade  
);  
create index if not exists actors_film_name  
on actors (film_name);
```

```
create table if not exists wish_list (  
    uuid    uuid not null primary key,  
    film_name text not null  
        references films (name)  
        on delete cascade,  
    telegram_id bigint not null  
        references bot_users (telegram_id)  
        on delete cascade  
);  
create index if not exists wish_list_film_name  
on wish_list (film_name);  
create index if not exists wish_list_telegram_id  
on wish_list (telegram_id);
```

Заполнение таблиц с информацией о пользователе и wish-листами происходило в процессе работы бота, а таблицы фильмов и актёров заполнялись вручную с помощью данного скрипта:

```
INSERT INTO films (uuid, name, director, rating, counter, year, language, duration,  
genre) VALUES
```

```
    ('21b607ce-cf9b-4fb2-a0c8-c013fa27da0a', 'Зеленая миля', 'Фрэнк  
Драбонт', 9, 1, '1999', 'English', '189', 'Драма'),
```

('867c9b13-28e4-4cb4-aab6-d8abff9824ac', 'Побег из Шоушенка', 'Фрэнк Дарабонт', 9, 1, '1994', 'English', '142', 'Драма'),

('7a876737-085c-4f4e-b7c9-0003fb32752a', 'Властелин колец: Возвращение короля', 'Питер Джексон', 8, 1, '2003', 'English', '201', 'Фэнтези'),

('de85604d-3ddd-417b-9881-3dae4e741f92', 'Властелин колец: Две крепости', 'Питер Джексон', 8, 1, '2002', 'English', '179', 'Приключения'),

('fe0bb9d8-f2ec-464e-9b15-6fefe94d4100', 'Властелин колец: Братство Кольца', 'Питер Джексон', 8, 1, '2001', 'English', '178', 'Драма'),

('6a704166-2d73-4c81-b000-ef121bbe4844', 'Форрест Гамп', 'Роберт Земекис', 9, 1, '1994', 'English', '142', 'Комедия'),

('38974963-b0bd-4d67-b672-f3c57c2109e8', 'Король Лев', 'Джон Фавро', 7, 1, '1994', 'English', '183', 'Мультфильм'),

('a9c84717-1d4d-45ef-b93d-8600dafb6a67', 'Карты, деньги, два ствола', 'Гай Ричи', 5, 1, '1998', 'English', '195', 'Боевик'),

('0bc8b6ee-6391-429d-954a-fd08754d7fe6', 'Список Шиндлера', 'Стивен Спилберг', 6, 1, '1993', 'English', '195', 'Биография'),

('3fae65fa-b653-4c5d-934d-6e599b7d24e0', 'Начало', 'Кристофер Нолан', 8, 1, '2010', 'English', '148', 'Боевик');

INSERT INTO actors (uuid, actor, film_name) VALUES

('0a3fd8b3-069e-4ea0-b22e-b42bca85a390', 'Киану Ривз', 'Зеленая миля'),

('19550ca1-6d7a-4656-bade-b8309af1d668', 'Леонардо ДиКаприо', 'Начало'),

('284a7093-8598-430e-929a-14e1a75868f2', 'Мадс Миккельсен', 'Властелин колец: Возвращение короля'),

('d0967cb1-1460-4cc5-9fed-eb282919e87c', 'Хью Джекман', 'Властелин колец: Две крепости'),

('8446b032-8d7e-49f2-87de-4cea385e00d4', 'Сирша Ронан', 'Побег из Шоушенка'),

('dccc76b0-b824-4ab7-b9fd-474d00fa1203', 'Мэттью МакКонахи', 'Форрест Гамп'),

('07ab1725-cc3f-410c-ac56-5f8888344da8', 'Хью Джекман', 'Властелин колец: Братство Кольца'),

('72311304-d076-4ca4-9de9-47cd9c44ab58', 'Райан Рейнольдс', 'Список Шиндлера'),

('95c3da9c-9d63-4469-b9b8-babd38459d65', 'Марго Робби', 'Карты, деньги, два ствола'),

('bbb895d0-ff97-4373-b5bd-346a374113eb', 'Том Холланд', 'Король Лев');

5. Клиентское приложение

5.1 Архитектура

Данный проект был реализован в виде телеграмм-бота, язык программирования, который был использован для его реализации - python. Весь написанный код можно увидеть в нашем репозитории: [Github](#).

5.2 Сценарии использования

Основная цель нашего бота заключается в том, чтобы человек мог с легкостью находить фильмы для просмотра по разным параметрам, будь то информация о фильме, режиссер, актер или жанр.

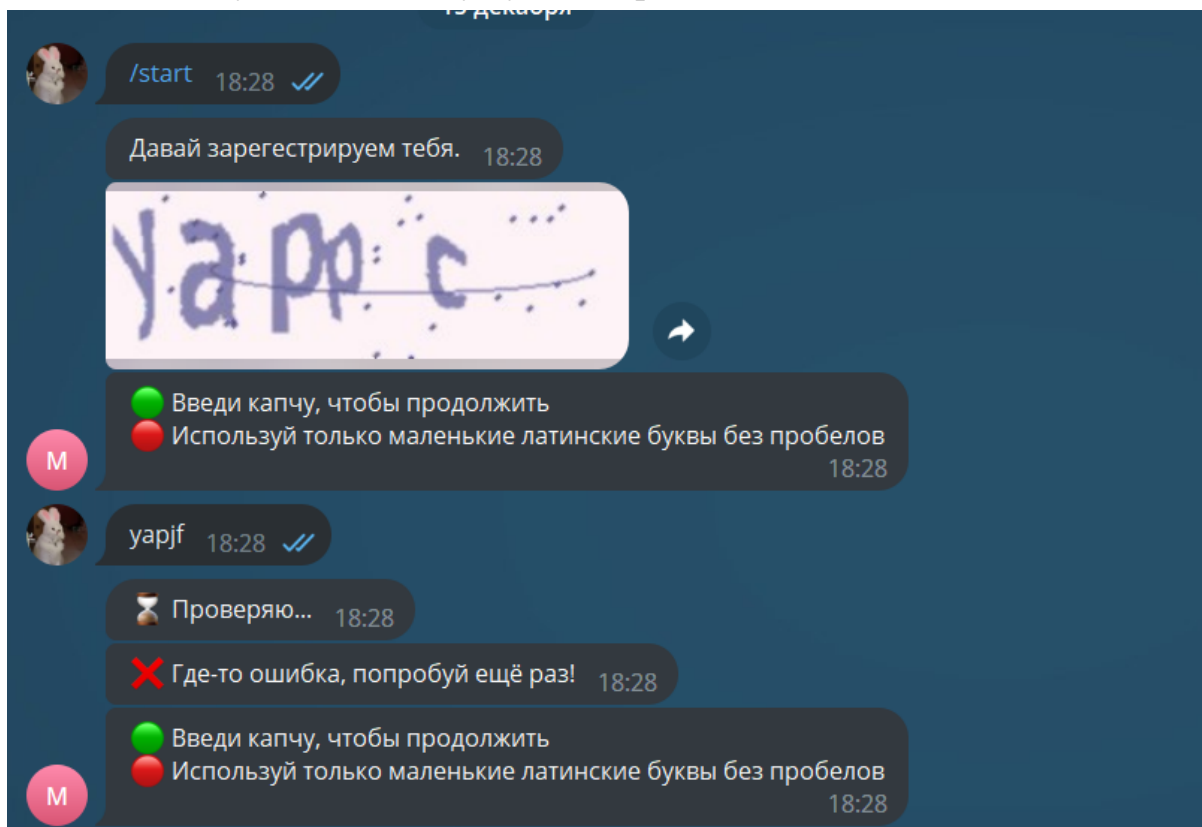
В данном боте пользователю также доступны следующие моменты: он может в любой момент оценить просмотренный фильм, узнать топ лучших фильмов на данный момент или добавить какой-то фильм в избранное. Как пример: после добавления в избранное, пользователь также может убедиться в том, что все сработало с помощью команды /favourite.

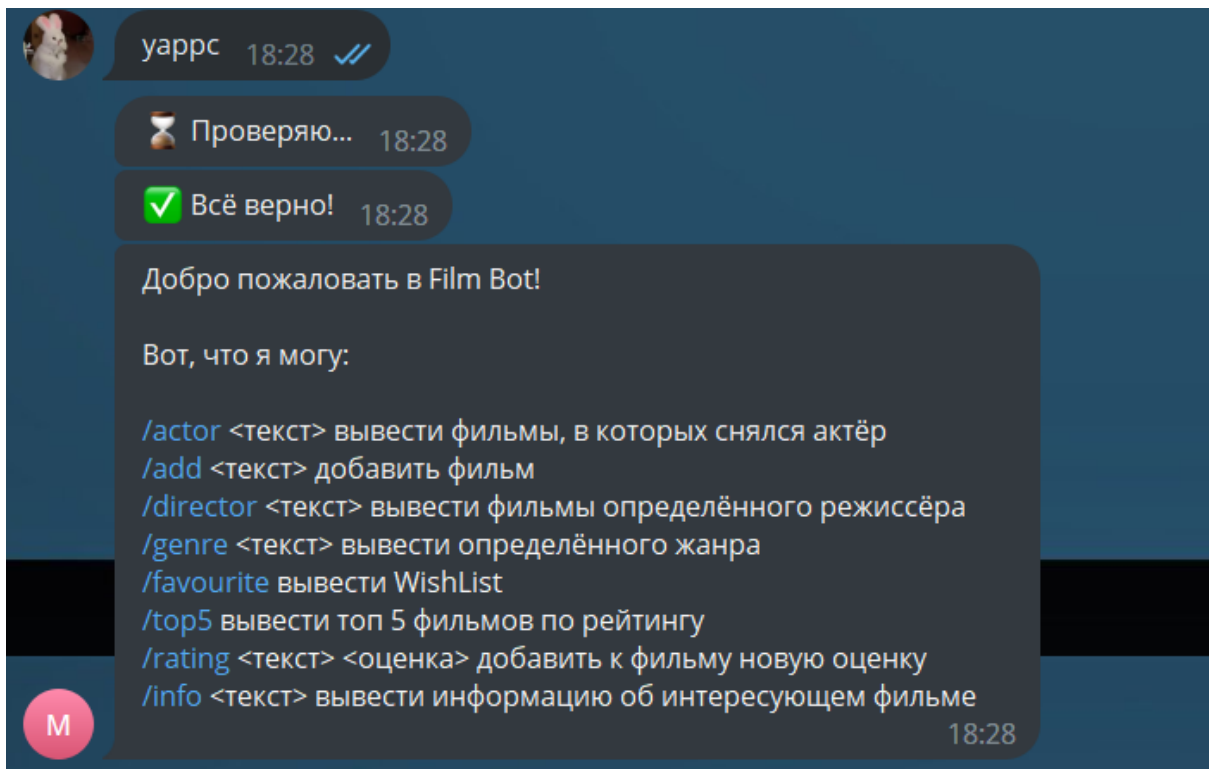
5.3 Организация доступа к данным

Библиотека с помощью которой мы смогли реализовать бота - psycorg2, мы выбрали ее поскольку в ней достаточно функций для реализации нашего проекта. В сумме мы смогли реализовать бота, который поддерживает работу с большим количеством пользователей и несмотря на большое количество пользователей, быстро реагирует на запросы. Изменения в базе данных вносятся с помощью обработки команд пользователя благодаря скриптам на питоне. При всем этом в боте предусмотрена проверка данных и в базу не вносятся некорректные задачи и пользователи.

5.4 Интерфейс с пользователем

Для регистрации пользователю необходимо ввести captcha, которая высылается в ответ на начальную команду /start, после чего он будет добавлен в базу данных и ему будет открыты все возможности бота:





5.5 Отчёты

База данных

Приведем пример того как выглядит база данных для актеров: тут хранится информация об актере и название фильма

| uuid | actor | film_name |
|--------------------------------------|-------------------|-------------------------------------|
| 0a3fd8b3-069e-4ea0-b22e-b42bca85a390 | Киану Ривз | Зеленая миля |
| 19550ca1-6d7a-4656-bade-b8309af1d668 | Леонардо ДиКаприо | Начало |
| 284a7093-8598-430e-929a-14e1a75868f2 | Мадс Миккельсен | Властелин колец: Возвращение короля |
| d0967cb1-1460-4cc5-9fed-eb282919e87c | Хью Джекман | Властелин колец: Две крепости |
| 8446b032-8d7e-49f2-87de-4cea385e00d4 | Сирша Ронан | Побег из Шоушенка |
| dccc76b0-b824-4ab7-b9fd-474d00fa1203 | Мэттью МакКонахи | Форрест Гамп |
| 07ab1725-cc3f-410c-ac56-5f8888344da8 | Хью Джекман | Властелин колец: Братство Кольца |
| 72311304-d076-4ca4-9de9-47cd9c44ab58 | Райан Рейнольдс | Список Шиндлера |
| 95c3da9c-9d63-4469-b9b8-babd38459d65 | Марго Робби | Карты, деньги, два ствола |
| bbb895d0-ff97-4373-b5bd-346a374113eb | Том Холланд | Король Лев |

И для фильмов: здесь хранится название фильма, режиссёр, рейтинг, год выпуска, оригинальный язык, продолжительность фильма и жанр.

| uid | name | director | rating | counter | year | language | duration | genre |
|-----|---|-----------------|--------|---------|------|----------|----------|-------------|
| 1 | 867c9b13... Побег из Шоушенка | Фрэнк Дарабонт | 9 | 1 | 1994 | English | 142 | Драма |
| 2 | 7a876737... Властелин колец: Возвращение короля | Питер Джексон | 8 | 1 | 2003 | English | 201 | Фэнтези |
| 3 | de85604d... Властелин колец: Две крепости | Питер Джексон | 8 | 1 | 2002 | English | 179 | Приключения |
| 4 | fe0bb9d8... Властелин колец: Братство Кольца | Питер Джексон | 8 | 1 | 2001 | English | 178 | Драма |
| 5 | 38974963... Король Лев | Джон Фавро | 7 | 1 | 1994 | English | 183 | Мультфильм |
| 6 | a9c84717... Карты, деньги, два ствола | Гай Ричи | 5 | 1 | 1998 | English | 195 | Боевик |
| 7 | 0bc8b6ee... Список Шиндлера | Стивен Спилберг | 6 | 1 | 1993 | English | 195 | Биография |
| 8 | 3fae65fa... Начало | Кристофер Нолан | 8 | 1 | 2010 | English | 148 | Боевик |
| 9 | 21b607ce... Зеленая миля | Фрэнк Дарабонт | 23 | 3 | 1999 | English | 189 | Драма |
| 10 | 6a704166... Форрест Гамп | Роберт Земекис | 19 | 2 | 1994 | English | 142 | Комедия |

1) Старт работы:

В самом начале пользователю необходимо подтвердить свою личность с помощью captcha далее приведена часть кода, с помощью которого мы это и реализуем.


```

@bot.message_handler(commands=["start"])
def send_welcome(message: telebot.types.Message) -> None:
    _LOGGER.info(f"Start by {message.from_user.id}")

    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():

        (
            bot_users.update(
                {
                    bot_users.first_name: message.from_user.first_name,
                    bot_users.last_name: message.from_user.last_name,
                    bot_users.tg_username: message.from_user.username,
                    bot_users.chat_id: message.chat.id,
                }
            )
            .where(bot_users.telegram_id == message.from_user.id)
            .execute()
        )

        bot.send_message(message.chat.id, welcome.WELCOME_MESSAGE)

    else:
        _LOGGER.info(f"Attempting to register {message.from_user.id}")

        bot.send_message(message.chat.id, "Давай зарегистрируем тебя.")

        send_captcha(message)

```

2) Функционал бота

После регистрации бот выдает пользователю список своих возможностей

```

WELCOME_MESSAGE = """
Добро пожаловать в Film Bot!

Вот, что я могу:

/actor <текст> вывести фильмы, в которых снялся актёр
/add <текст> добавить фильм
/director <текст> вывести фильмы определённого режиссёра
/genre <текст> вывести определённого жанра
/favourite вывести WishList
/top5 вывести топ 5 фильмов по рейтингу
/rating <текст> <оценка> добавить к фильму новую оценку
/info <текст> вывести информацию об интересующем фильме
"""

```

```

@bot.message_handler(commands=["help"])
def send_welcome(message: telebot.types.Message) -> None:
    bot.send_message(message.chat.id, welcome.
WELCOME_MESSAGE)

```

3) Узнать информацию фильме

Как один из вариантов, пользователю предлагается узнать всю информацию о фильме, это реализовано следующим образом:

```

@bot.message_handler(commands=["info"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():
        arguments = message.text.split(" ")

        if len(arguments) < 2:
            bot.send_message(message.chat.id, "Укажите название фильма.")

        elif arguments[1].isalnum():
            film_ = ''
            for i in range(1, len(arguments) - 1):
                film_ += str(arguments[i]) + ' '
            film_ += arguments[len(arguments) - 1]

            bot.send_message(message.chat.id, "Ищу...")

            query: peewee.ModelSelect = films.select().where(
                films.name == film_
            )

            if query:
                reply = f"Вот информация о фильме {film_}:\n"
                for row in query:
                    reply += 'Режиссёр: '
                    reply += f"{row.director}\n"
                    reply += 'Год выпуска: '
                    reply += f"{row.year}\n"
                    reply += 'Язык оригинала: '
                    reply += f"{row.language}\n"
                    reply += 'Продолжительность в минутах: '
                    reply += f"{row.duration}\n"
                    reply += 'Жанр: '
                    reply += f"{row.genre}\n"

                bot.send_message(message.chat.id, reply)

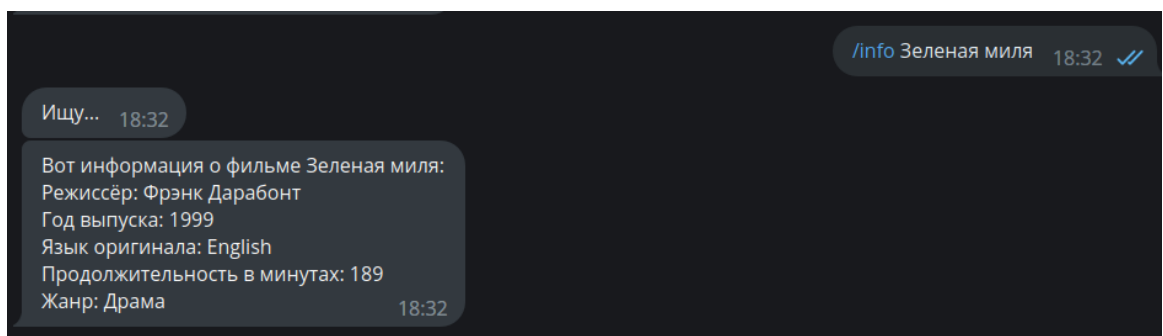
            else:
                bot.send_message(message.chat.id, "Не найдено фильмов с таким названием")

        else:
            bot.send_message(message.chat.id, "Некорректно указан фильм!")

    else:
        bot.send_message(message.chat.id, "Нажми /start, чтобы зарегистрироваться!")

```

И как это выглядит в чате с пользователем:



4) Добавить фильм в избранное:

Пользователь также может создать список с фильмами которые ему понравились. Мы предусмотрели случаи, что пользователь добавит один и тот же фильм несколько раз. Вот как выглядит реализация:

```
@bot.message_handler(commands=["add"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():
        arguments = message.text.split(" ")

        if len(arguments) < 2:
            bot.send_message(message.chat.id, "Укажите название фильма.")

        elif arguments[1].isalnum():
            film_ = ''
            for i in range(1, len(arguments) - 1):
                film_ += str(arguments[i]) + ' '
            film_ += arguments[len(arguments) - 1]

            if (
                wish_list.select()
                .where(
                    wish_list.telegram_id == message.from_user.id, wish_list.film_name == film_
                )
                .exists()
            ):
                bot.send_message(message.chat.id, "Фильм уже в избранном")

            else:
                wish_list.insert(
                    {
                        wish_list.telegram_id: message.from_user.id,
                        wish_list.film_name: film_,
                    }
                ).execute()
                bot.send_message(message.chat.id, "Фильм добавлен.")

        else:
            bot.send_message(message.chat.id, "Некорректное название.")

    else:
        bot.send_message(message.chat.id, "Нажми /start, чтобы зарегистрироваться!")
```

А в чате с пользователем это выглядит так:



5) Посмотреть свой Wishlist

После добавления фильма в список избранных, пользователь может просмотреть данный список, посмотрим как это реализовано:

```
@bot.message_handler(commands=["favourite"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():

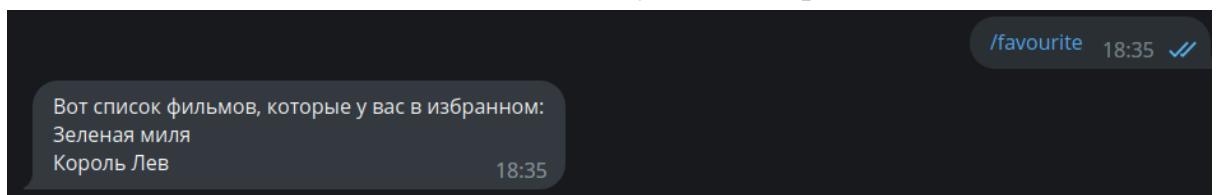
        query: peewee.ModelSelect = wish_list.select().where(
            wish_list.telegram_id == message.from_user.id
        )

        if query:
            reply = f"Вот список фильмов, которые у вас в избранном:\n"
            for row in query:
                print(row.film_name_id)
                reply += f"{row.film_name_id}\n"

            bot.send_message(message.chat.id, reply)

        else:
            bot.send_message(message.chat.id, "Ваш список избранных фильмов пуст!")
```

И в чате с пользователем выглядит следующим образом:



6) Поиск по актеру/режиссеру/жанру

Пользователю также доступен поиск фильма по актеру, режиссеру или жанру. Реализацию покажем на примере поиска по жанру:

```
@bot.message_handler(commands=["genre"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():
        arguments = message.text.split(" ")
        if len(arguments) < 2:
            bot.send_message(message.chat.id, "Укажите жанр")

        elif arguments[1].isalpha():
            bot.send_message(message.chat.id, "Ищу...")

            query: peewee.ModelSelect = films.select().where(
                films.genre == arguments[1]
            )

            if query:
                reply = f"Вот список фильмов в жанре {arguments[1]}:\n"
                for row in query:
                    reply += f"{row.name}\n"

                bot.send_message(message.chat.id, reply)

            else:
                bot.send_message(message.chat.id, "Не найдено фильмов с таким жанром")

        else:
            bot.send_message(message.chat.id, "Некорректно указан жанр!")

    else:
        bot.send_message(message.chat.id, "Нажми /start, чтобы зарегистрироваться!")
```

В чате с пользователем выводится список соответствующих фильмов:



7) Оценка фильма

Также у пользователя есть возможность посмотреть топ 5 лучших фильмов или оценить просмотренные фильмы, в зависимости от этого рейтинг будет меняться.

Реализация поиска 5 лучших фильмов:

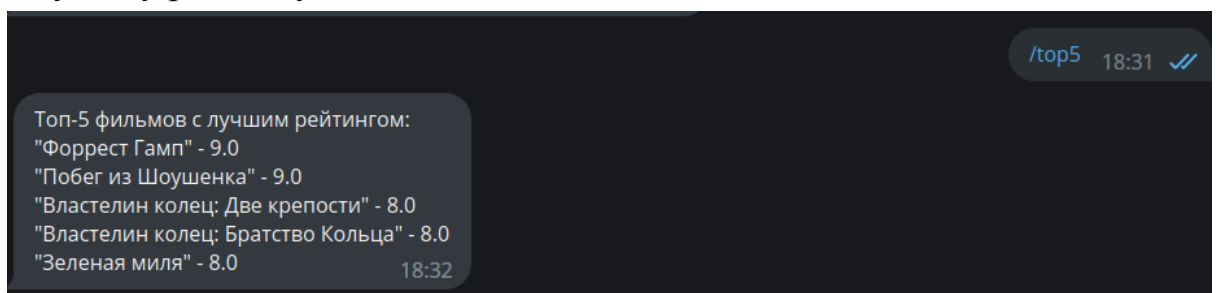
```
@bot.message_handler(commands=["top5"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():
        query = films.select().order_by((films.rating / films.counter).desc()).limit(5)

        if query:
            reply = "Топ-5 фильмов с лучшим рейтингом:\n"

            for row in query:
                reply += f'"{row.name}" - {row.rating / row.counter}\n'

            bot.send_message(message.chat.id, reply)
```

В диалоге с пользователем будет показан топ 5 фильмов по текущему рейтингу:



Реализация оценки фильма:

```
@bot.message_handler(commands=["rating"])
def send_welcome(message: telebot.types.Message) -> None:
    if bot_users.select().where(bot_users.telegram_id == message.from_user.id).exists():
        arguments = message.text.split(" ")

        if len(arguments) < 3:
            bot.send_message(message.chat.id, "Некорректный ввод.")

        elif arguments[1].isalnum():
            film_ = ''
            for i in range(1, len(arguments) - 2):
                film_ += str(arguments[i]) + ' '
            film_ += arguments[len(arguments) - 2]
            score_ = float(arguments[len(arguments) - 1])

            films.update(
                {
                    films.rating: (films.rating + score_),
                    films.counter: films.counter + 1,
                }
            ).where(films.name == film_).execute()
            bot.send_message(message.chat.id, "Фильм оценен.")

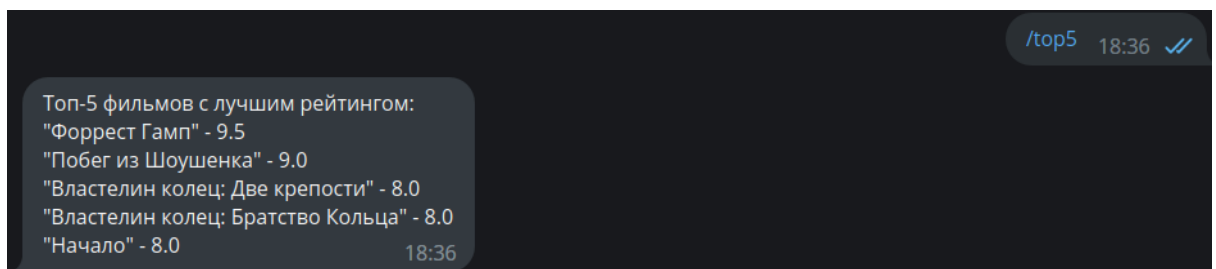
        else:
            bot.send_message(message.chat.id, "Некорректное название.")

    else:
        bot.send_message(message.chat.id, "Нажми /start, чтобы зарегистрироваться!")
```

Если пользователь захочет оценить фильм, то в диалоге это будет выглядеть следующим образом:



Посмотрим как изменился топ 5 лучших фильмов после данной оценки:



6. Заключение

Код проекта находится в репозитории по ссылке [MR](#).

6.1 Объемные характеристики разработки

В результате работы над данным проектом была создана база данных, состоящая из 4 таблиц и 4 связей.

При разработке Телеграмм-бота было написано порядка 400 строк основного кода и около 300 строк подключающихся файлов.

6.2 Авторский вклад и комментарии по выполнению проекта

Выполнением диаграмм и реализацией кода для бота мы занимались совместно.

7. Источники

- Код проекта - https://github.com/unrebbby/movie_recommendation