**РУКОВОДСТВО ПРОГРАММИСТА**

**Телеграм-бот**

**«Cinema Project Club»**

**Листов 13**

2022

**АННОТАЦИЯ**

В данном программном документе приведено руководство программиста по работе с PyCharm, в которой был разработан телеграмм-бот «Cinema Project Club»

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые программой, условия, необходимые для правильной работы бота.

В разделе «Характеристика программы» приведено описание основных характеристик и особенностей программы.

1. Назначение и условия применения программы

В условиях перехода общества в информационную эпоху, все большую и большую актуальность приобретает простота и универсальность, ныне, телеграмм предоставляет практически безграничные возможности в плане потребления и создания контента.

В наше время присутствуют много различных телеграмм-ботов, которые могут подстроиться под вкусы и потребность любых пользователей. Бот которого мы презентуем является же – визуальной новеллой.

Телеграмм-бот “Cinema Project Club” разрабатывался в рамках аналоговых проектов, является проектом для подбора фильма на вечерний досуг, возможности общения. Основная задача продукта – дать пользователю возможность получить фильм основываясь на его предпочтениях, возможность общаться в чате, который модерируется в реальном времени.

В ходе работы надо игрой был использован язык программирования под названием «Python», необходимый для реализации всех основных действий в разработке проекта.

1. Характеристики и работа программы

Требования к реализации функций в боте:

Пользователь может получить случайный фильм из обширной базы данных бота, поставить ему оценку, начать общение в модерируемом чате. В проекте реализована корреляция, при помощи которой не понравившиеся фильмы вам не попадутся, также и похожие на них. Реализован модератор с проверкой сообщений на нежелательные слова.  
В боте должны присутствовать такие функции как:

* Функция «/start», при ее нажатии активируется бот и начинает дальнейшее общение.
* Функция «/info» дает пользователю информацию о работе бота.
* Функция «/recommend» показывает пользователю случайный фильм из базы данных бота.
* Функция «Эхо» позволяет боту отвечать таким же сообщением на сообщение пользователя.
* Функция «Лайк/Нравится», позволяет получить похожий фильм основываясь на предыдущем.
* Функция «Ненравится» позволяет убрать непонравившийся фильм и все максимально похожие на него.
* Функция «IMDB» позволяет пользователю перейти на сайт IMDB.

Для написания бота, в приложении PyCharm был создан файл bot.py с основными функциями.

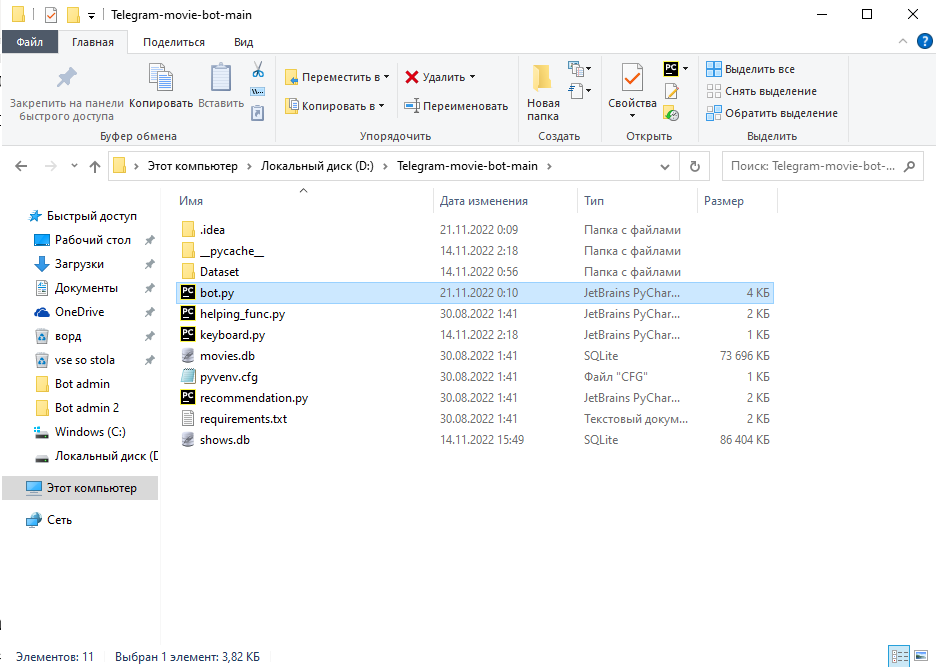


Рисунок 2 – Папка с проектом

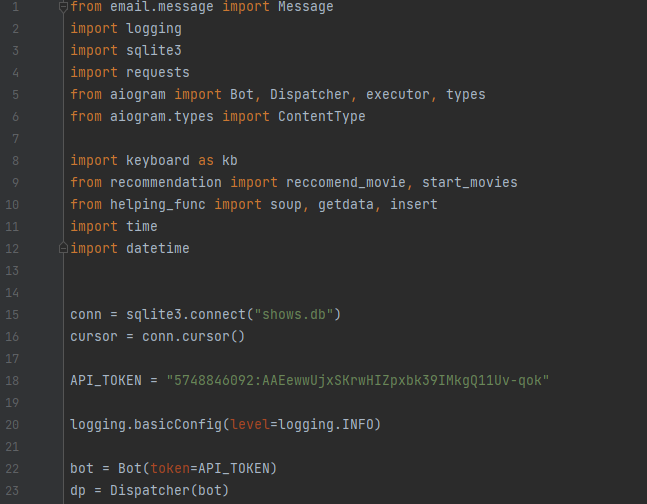


Рисунок 2 – Редактируемый файл bot.py

Мы видим следующее: первые 12 строчек — это импорты библиотек, их функций, также это нужно для того, чтобы сделать код из одного модуля доступным для других. Многие функции питон изначально не поддерживает, поэтому недостающие библиотеки нужно скачать и объявить.

Ниже можно увидеть подключение к базе данных sql и объявление токена бота, с которым программе нужно будет работать

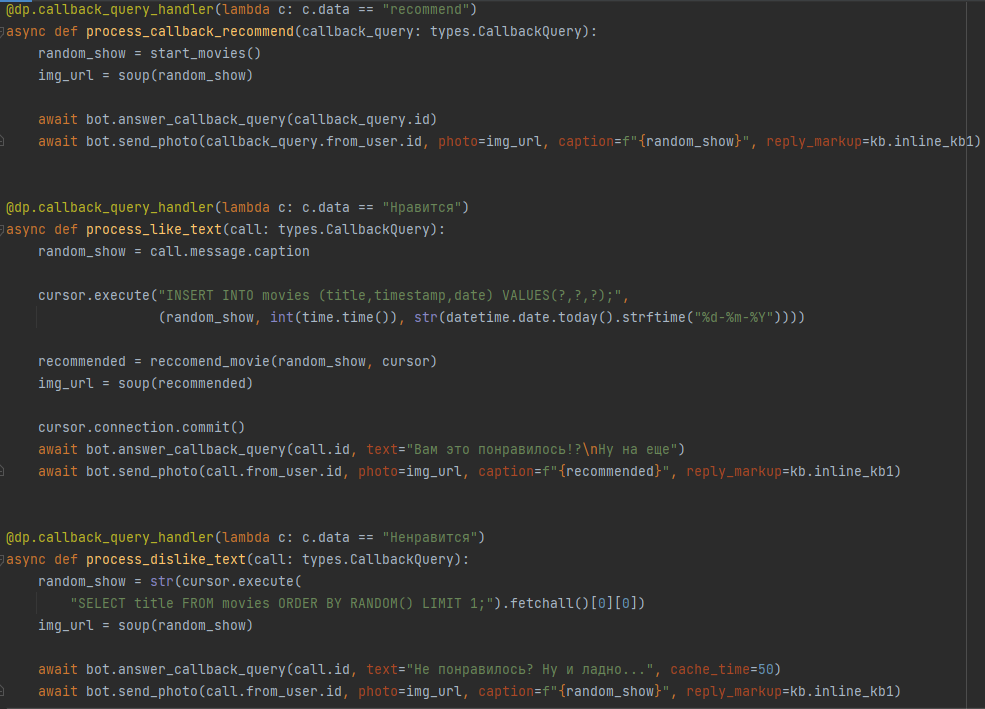
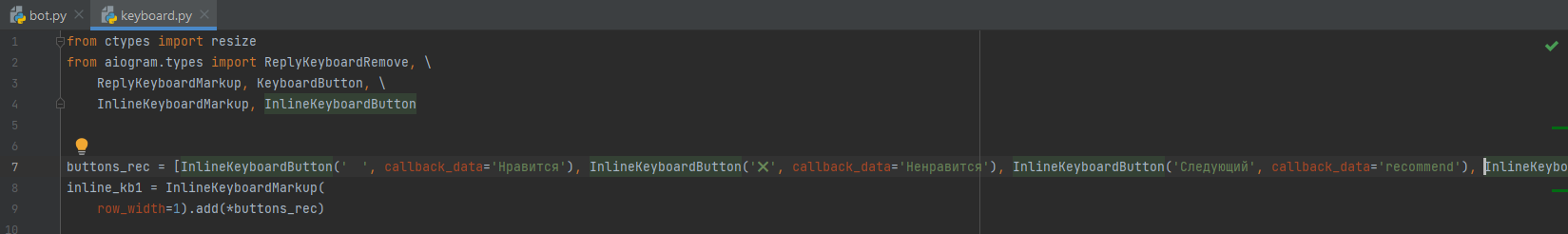


Рисунок 3 – Кнопки запросов

При помощи команд, бот запрашивает у базы данных определённые параметры и выводит их для пользователя в ответ на нажатую кнопку: «Нравится», «Не нравится». Сверху находится основной запрос, ответственный за команду /recommend.

На рисунке 4 показан скрип для работы с кнопками в самом телеграме, здесб же я заменил их на смайлики.

  
Рисунок 4 – Скрипт для работы с кнопками в телеграмм

На рисунке 5 показаны основные обработчики команд у бота. Все команды, которые бот воспринимает, обрабатывает и отвечает на них, прописаны именно здесь. В самом низу находится маленький скрипт для ответа на сообщение, таким же сообщением.

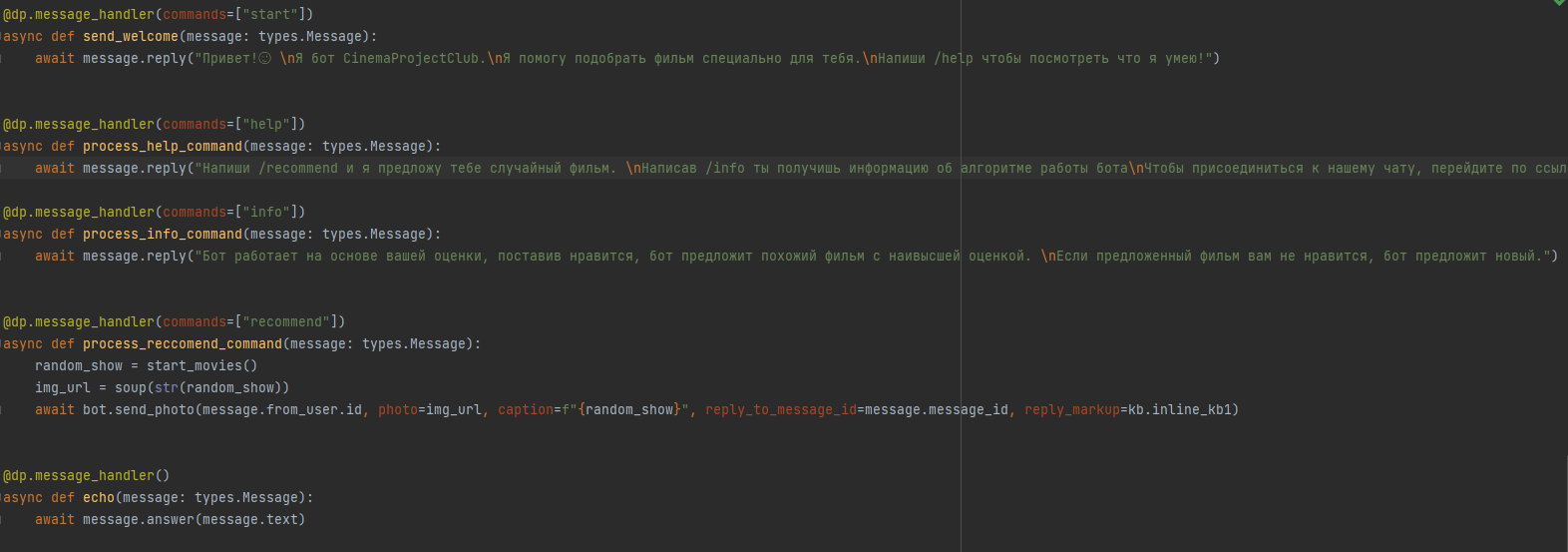


Рисунок 5 – Обработчики команд

ЛИСТИНГ ПРОГРАММЫ

**«Программа работы бота»**

from email.message import Message  
import logging  
import sqlite3  
import requests  
from aiogram import Bot, Dispatcher, executor, types  
from aiogram.types import ContentType  
  
import keyboard as kb  
from recommendation import reccomend\_movie, start\_movies  
from helping\_func import soup, getdata, insert  
import time  
import datetime  
  
  
conn = sqlite3.connect("shows.db")  
cursor = conn.cursor()  
  
API\_TOKEN = "5748846092:AAEewwUjxSKrwHIZpxbk39IMkgQ11Uv-qok"  
  
logging.basicConfig(level=logging.INFO)  
  
bot = Bot(token=API\_TOKEN)  
dp = Dispatcher(bot)  
  
  
def getdata(url):  
 r = requests.get(url)  
 return r.text  
  
  
@dp.callback\_query\_handler(lambda c: c.data == "recommend")  
async def process\_callback\_recommend(callback\_query: types.CallbackQuery):  
 random\_show = start\_movies()  
 img\_url = soup(random\_show)  
  
 await bot.answer\_callback\_query(callback\_query.id)  
 await bot.send\_photo(callback\_query.from\_user.id, photo=img\_url, caption=f"{random\_show}", reply\_markup=kb.inline\_kb1)  
  
  
@dp.callback\_query\_handler(lambda c: c.data == "Нравится")  
async def process\_like\_text(call: types.CallbackQuery):  
 random\_show = call.message.caption  
  
 cursor.execute("INSERT INTO movies (title,timestamp,date) VALUES(?,?,?);",  
 (random\_show, int(time.time()), str(datetime.date.today().strftime("%d-%m-%Y"))))  
  
 recommended = reccomend\_movie(random\_show, cursor)  
 img\_url = soup(recommended)  
  
 cursor.connection.commit()  
 await bot.answer\_callback\_query(call.id, text="Вам это понравилось!?\nНу на еще")  
 await bot.send\_photo(call.from\_user.id, photo=img\_url, caption=f"{recommended}", reply\_markup=kb.inline\_kb1)  
  
  
@dp.callback\_query\_handler(lambda c: c.data == "Ненравится")  
async def process\_dislike\_text(call: types.CallbackQuery):  
 random\_show = str(cursor.execute(  
 "SELECT title FROM movies ORDER BY RANDOM() LIMIT 1;").fetchall()[0][0])  
 img\_url = soup(random\_show)  
  
 await bot.answer\_callback\_query(call.id, text="Не понравилось? Ну и ладно...", cache\_time=50)  
 await bot.send\_photo(call.from\_user.id, photo=img\_url, caption=f"{random\_show}", reply\_markup=kb.inline\_kb1)  
  
  
@dp.message\_handler(commands=["start"])  
async def send\_welcome(message: types.Message):  
 await message.reply("Привет!😉 \nЯ бот CinemaProjectClub.\nЯ помогу подобрать фильм специально для тебя.\nНапиши /help чтобы посмотреть что я умею!")  
  
  
@dp.message\_handler(commands=["help"])  
async def process\_help\_command(message: types.Message):  
 await message.reply("Напиши /recommend и я предложу тебе случайный фильм. \nНаписав /info ты получишь информацию об алгоритме работы бота\nЧтобы присоединиться к нашему чату, перейдите по ссылке: https://t.me/+lT9zCCJ9ZGA1NWYy")  
  
@dp.message\_handler(commands=["info"])  
async def process\_info\_command(message: types.Message):  
 await message.reply("Бот работает на основе вашей оценки, поставив нравится, бот предложит похожий фильм с наивысшей оценкой. \nЕсли предложенный фильм вам не нравится, бот предложит новый.")  
  
  
@dp.message\_handler(commands=["recommend"])  
async def process\_reccomend\_command(message: types.Message):  
 random\_show = start\_movies()  
 img\_url = soup(str(random\_show))  
 await bot.send\_photo(message.from\_user.id, photo=img\_url, caption=f"{random\_show}", reply\_to\_message\_id=message.message\_id, reply\_markup=kb.inline\_kb1)  
  
  
@dp.message\_handler()  
async def echo(message: types.Message):  
 await message.answer(message.text)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 executor.start\_polling(dp, skip\_updates=True)