Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 Информатика и вычислительная техника
Профиль: Программное обеспечение средств вычислительной техники и автоматизированных систем
(очная форма обучения)

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

на кафедре прикладной математики и кибернетики
(наименование профильной организации/структурного подразделения СибГУТИ)

Telegram-бот

Выполнил:
студент ФИВТ
гр. ИП-811 _____/ Мироненко К.А /
«05» июня 2021г. (подпись)

Проверил:
Руководитель от СибГУТИ _____/ Приставка П.А. /
«05» июня 2021г. (подпись)

Новосибирск 2021

План-график проведения производственной **практики**

Вид практики

Мироненко Кирилла Андреевича

Фамилия Имя Отчество студента

факультета Информатика и вычислительная техника, 3 курса, гр. ИП-811

Направление: 09.03.01 – Информатика и вычислительная техника

Код – Наименование направления (специальности)

Профиль: Программное обеспечение средств вычислительной техники и автоматизированных систем

Место прохождения практики Кафедра прикладной математики и кибернетики

Объем практики: **216/6** часов/ЗЕ

Вид практики ***производственная***

Тип практики Производственная практика - Практика по получению профессиональных умений и опыта профессиональной деятельности

Срок практики с "01" февраля 2021 г.

по "29" мая 2021 г.

Содержание практики*:

Наименование видов деятельности	Дата (начало – окончание)
1. Общее ознакомление со структурным подразделением предприятия, вводный инструктаж по технике безопасности	01.02.2021–13.02.2021
2. Выдача задания на практику, деление студентов на группы (если необходимо), определение конкретной индивидуальной темы, формирование плана работ	15.02.2021–20.02.2021
3. Работа с библиотечными фондами структурного подразделения или предприятия, сбор и анализ материалов по теме практики	22.02.2021–20.03.2021
4. Выполнение работ в соответствии с составленным планом: – Поиск примерных эскизов – Изучение основных принципов работы telegtam-ботов – Установка python, настройка IDE, «разворачивание» виртуальной среды – Разработка и написание алгоритма	22.03.2021 – 22.05.2021
5. Анализ полученных результатов и произведенной работы Составление отчета по практике, защита отчета	24.05.2021–29.05.2021

*В соответствии с программой практики

Руководитель от СибГУТИ _____/ Приставка П.А./

«_____» _____ 2021г. (подпись)

ЗАДАНИЕ НА ПРАКТИКУ

Разработать telegram-бота, позволяющего взаимодействовать с «КиноПоиском»

Язык программирования – Python

Среда разработки – PyCharm IDEA.

ВВЕДЕНИЕ

Telegram — кроссплатформенный мессенджер с функциями VoIP, позволяющий обмениваться текстовыми, голосовыми и видеосообщениями, стикерами и фотографиями, файлами многих форматов. Также можно совершать видео- и аудиозвонки, организовывать конференции, многопользовательские группы и каналы. Клиентские приложения Telegram доступны для Android, iOS, Windows Phone, Windows, macOS и GNU/Linux. Количество ежемесячных активных пользователей сервиса по состоянию на январь 2021 года составляет около 500 млн человек.

При работе использовались следующие пакеты:

- aiogram — это простой и полностью асинхронный фреймворк для API Telegram Bot, написанный на Python 3.7 с asyncio и aiohttp.
- requests — пакет, цель которого сделать HTTP-запросы более простыми и удобными для человека
- logging — модуль для логирования
- emoji — пакет, позволяющий использовать символьное представление emoji
- telegraph — Оболочка API Telegraph | Telegra.ph
- bs4 — модуль парсинга для парсинга HTML

ОСНОВНАЯ ЧАСТЬ

Перед написанием кода бота, ознакомился с теорией, как работают telegram-bot’ы и что из себя представляют. Далее, взвесив все плюсы и минусы, для взаимодействия с API Telegram был выбран асинхронный фрейворк aiogram.

В начале работы было создано виртуальное окружение, для создания изолированной среды проекта.

Следующим шагом был зарегистрирован бот, для этого нужно было написать @BotFather и следовать пошаговой инструкции. Был получен токен, который был сохранен в переменных окружения.

Далее я зарегистрировался на сайте https://kinopoiskapiunofficial.tech/ и получил ключ для взаимодействия с API данного сайта,т.к. «КиноПоиск» не предоставляет официальное API. Ключ также был сохранен в переменных окружения.

На этом подготовительный этап был окончен.

В последующем логика была разделена на 4(5*) файла:

- main.py — Основная логика бота
- kinopoisk_api.py — Модуль для взаимодействия с кинопоиском
- keyboards.py — Файл, хранящий клавиатуры бота
- requirements.txt — Файл с зависимостями проекта
- *cashe.json — Файл, кеширущий результаты поиска фильмов

ЗАКЛЮЧЕНИЕ

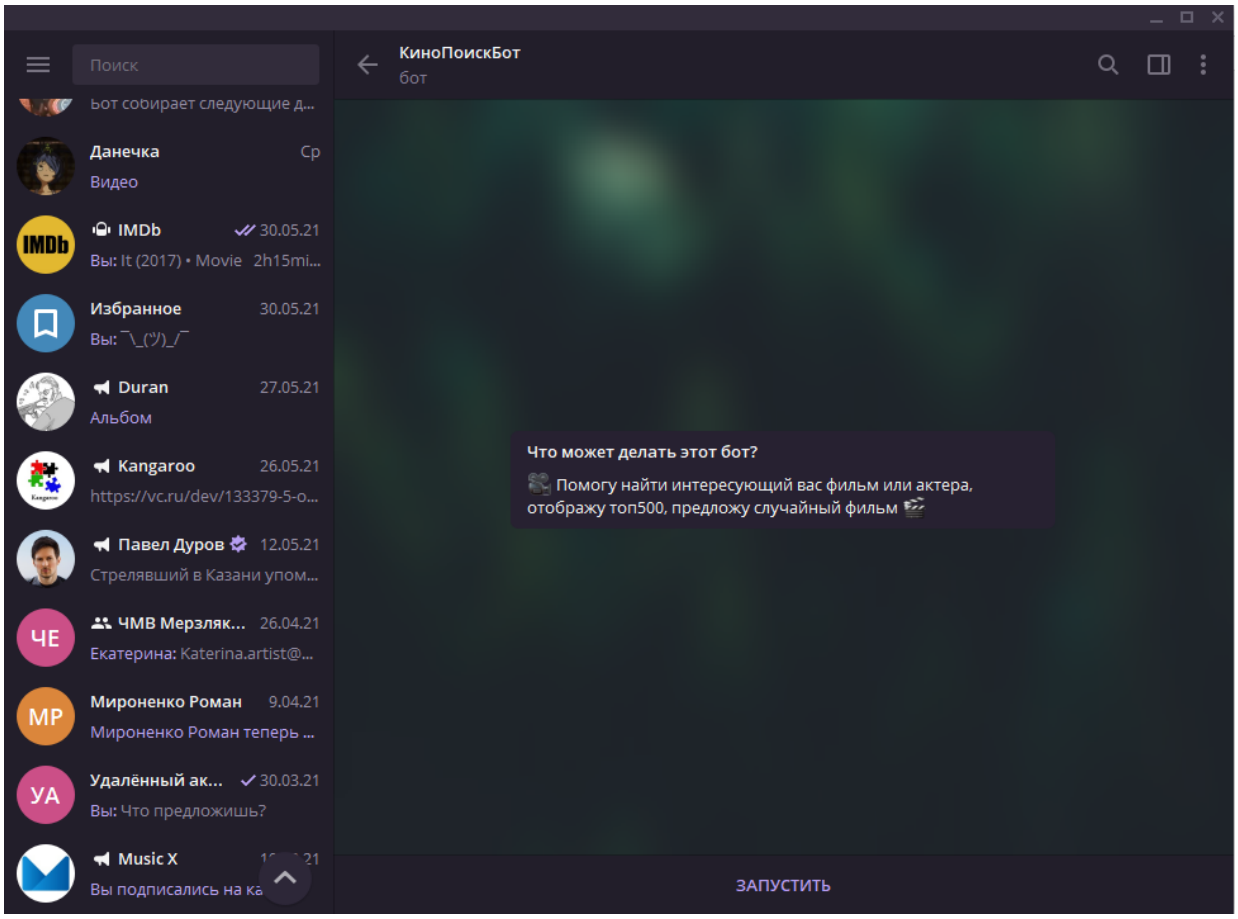
В ходе работы над данным проектом я познакомился с технологиями разработки telegram-ботов, изучил TelegramAPI. Улучшил свои навыки написания программ на языке python, базово освоил асинхронный фреймворк «aiogram», а также изучил некоторые другие пакеты.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

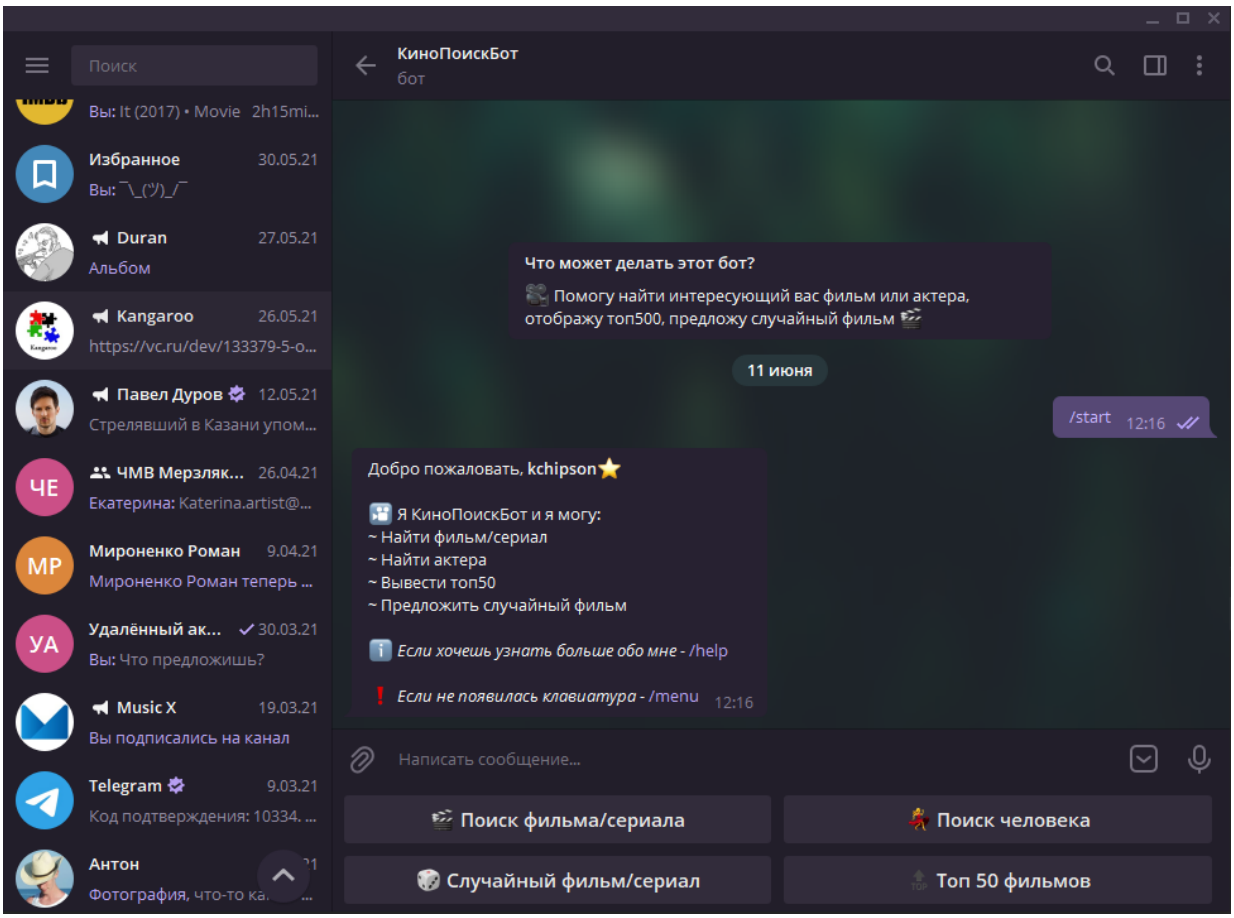
1. Python 3.9.5 documentation [Электронный ресурс] URL: https://docs.python.org/3/
2. Telegram Bot API [Электронный ресурс] URL: https://core.telegram.org/bots/api
3. aiogram’s documentation [Электронный ресурс] URL: https://docs.aiogram.dev/en/latest/
4. Kinopoisk Api Unofficial [Электронный ресурс] URL: https://kinopoiskapiunofficial.tech/

ПРИЛОЖЕНИЯ

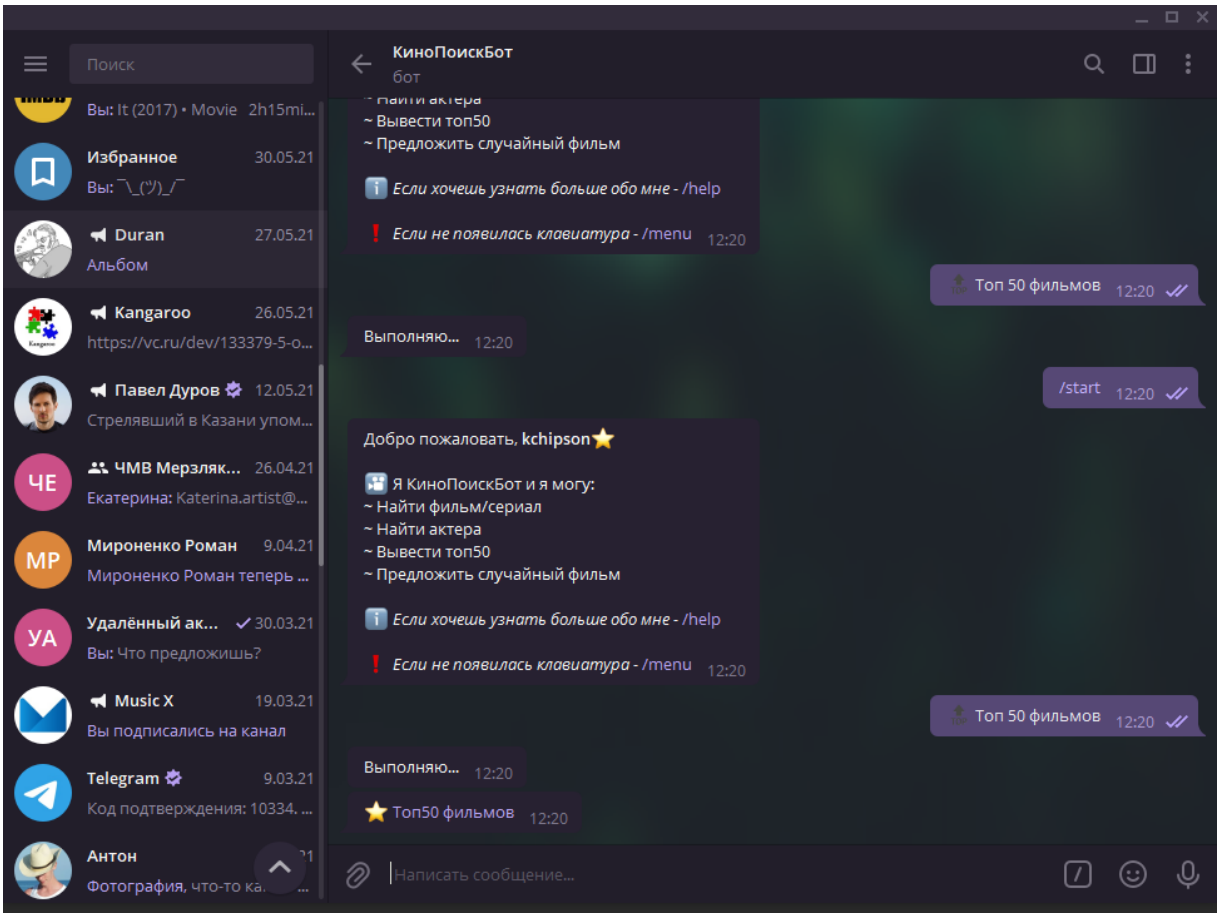
Пример работы



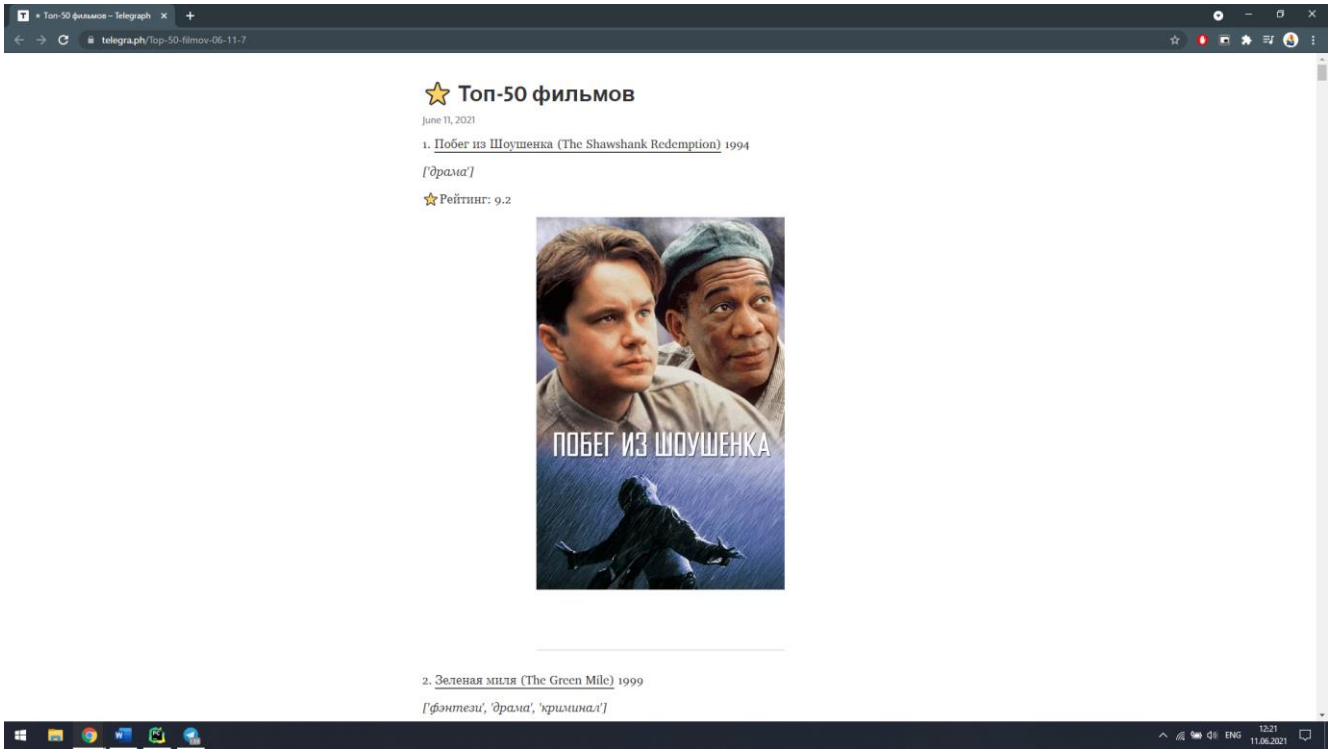
(Рис 1. Начало работы бота)



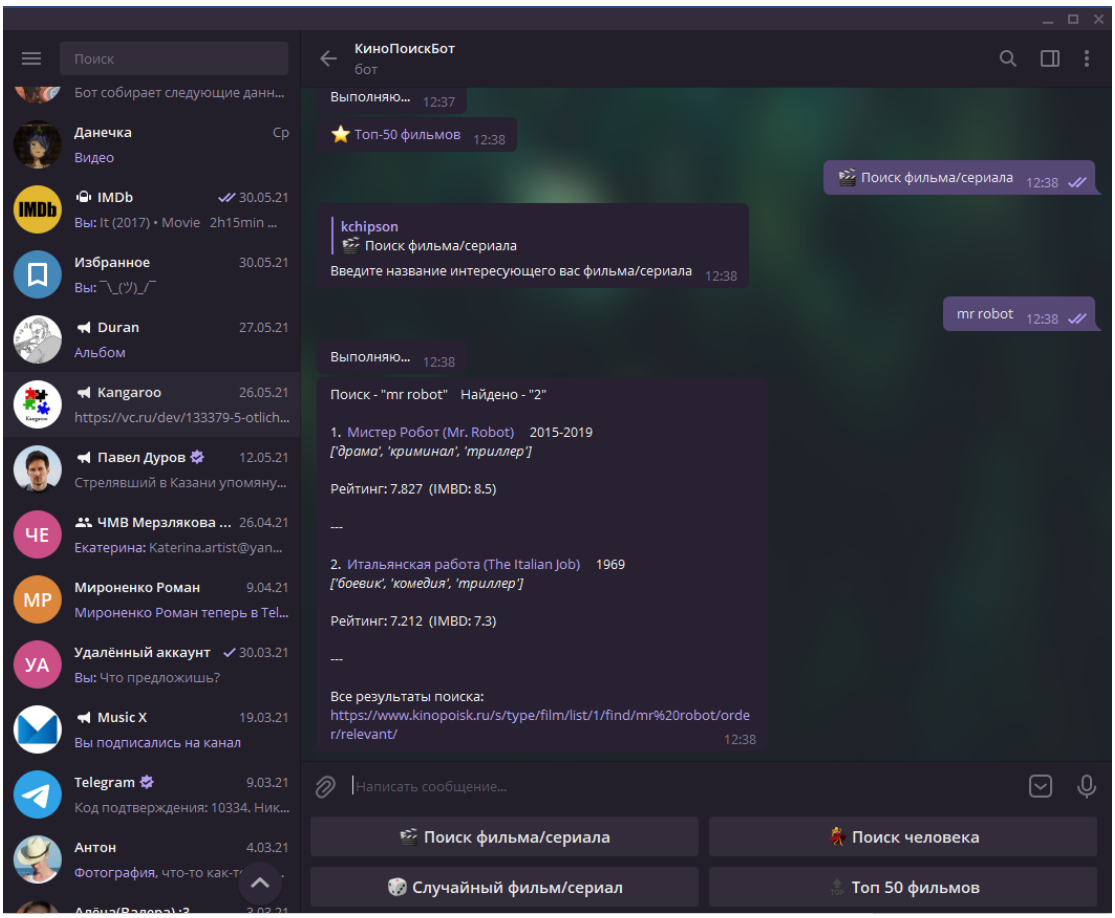
(Рис 2. Приветственное сообщение)



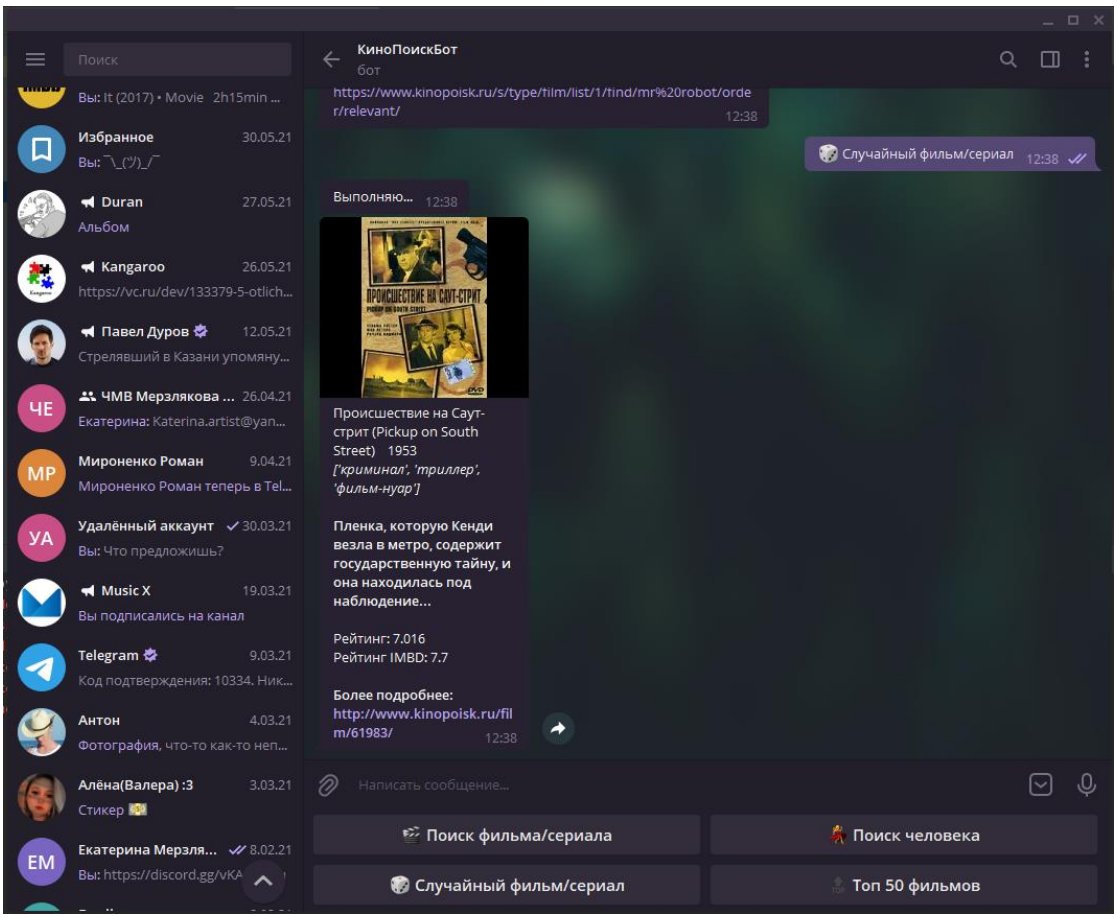
(Рис 3.1. Топ50)



(Рис 3.2. Топ50)



(Рис 4. Поиск фильма)



(Рис 5. Случайный фильм)

Листинг

requirements.txt

```
python-dotenv~=0.17.1
aiogram~=2.13
emoji~=1.2.0
requests~=2.25.1
beautifulsoup4~=4.9.3
```

main.py

```
# -*- coding: utf-8 -*-
import asyncio
import os
from dotenv import load_dotenv, find_dotenv
import logging
from aiogram import Bot, Dispatcher, executor, types
from aiogram.dispatcher import FSMContext
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher.filters.state import State, StatesGroup
import emoji
import keyboards
import kinopoisk_api
from telegraph import Telegraph

# Логирование
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s - %(name)s - %(message)s")

# Токен  из переменной окружения
load_dotenv(find_dotenv())
bot_token = os.environ.get("TELEGRAM_TOKEN")
if not bot_token:
    exit("Error: Переменная \"TELEGRAM_TOKEN\" не найдена в переменных окружения")
api_key = os.environ.get("API_KEY")
if not api_key:
    exit("Error: Переменная \"API_KEY\" не найдена в переменных окружения")


class DataInput(StatesGroup):
    searchFilm = State()
    seacrhHuman = State()


# Объект бота
bot = Bot(token=bot_token, parse_mode=types.ParseMode.MARKDOWN_V2)

# Диспетчер для бота
dp = Dispatcher(bot, storage=MemoryStorage())

# Api
kinopoisk = kinopoisk_api.KinopoiskApi(token=api_key)


# Хэндлер на команду /start
@dp.message_handler(commands=["start"], state="*")
async def cmd_start(message: types.Message, state: FSMContext):
```

```
    await state.finish()
    me = await bot.get_me()
    await message.answer(f"Добро пожаловать, *{message.from_user.full_name}{emoji.emojize(':star:')}*\n"
                         f"\n"
                         f"{emoji.emojize(':cinema:')} Я {me.full_name} и я могу:\n"
                         f"\~ Найти фильм/сериал\n"
                         f"\~ Найти актера\n"
                         f"\~ Вывести топ50\n"
                         f"\~ Предложить случайный фильм\n"
                         f"\n"
                         f"{emoji.emojize(':information:')} _Если хочешь узнать больше обо мне_ \- /help\n"
                         f"\n"
                         f"{emoji.emojize(':red_exclamation_mark:')} _Если не появилась клавиатура_ \- /menu\n",
                         reply_markup=keyboards.mainKeyboard)


# Хэндлер на команду /menu
@dp.message_handler(commands=['menu'], state="*")
async def cmd_menu(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer("Основное меню", reply_markup=keyboards.mainKeyboard)


# Хэндлер на команду /help
@dp.message_handler(commands=['help'], state="*")
async def cmd_help(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer("<i>Список доступных команд:</i>\n\n"
                         "<b>/start</b> - Перезапуск бота\n"
                         "<b>/film</b> - Поиск фильма\n"
                         "<b>/human</b> - Поиск человека\n"
                         "<b>/random</b> - Случайный фильм\n"
                         "<b>/top50</b> - Топ-50 лучших фильмов\n"
                         "<b>/cancel</b> - Отменить текущее действие\n"
                         "<b>/menu</b> - Отобразить клавиатуру\n"
                         "<b>/hide</b> - Скрыть клавиатуру\n"
                         "<b>/help</b> - Помощь",
                         parse_mode=types.ParseMode.HTML)


# Хэндлер на команду /hide
@dp.message_handler(commands=['hide'], state="*")
async def cmd_cancel(message: types.Message, state: FSMContext):
    await message.answer("Клавиатура скрыта", reply_markup=keyboards.hideKeyboard)


# Хэндлер на команду /cancel
@dp.message_handler(commands=['cancel'], state=[DataInput.seacrhHuman, DataInput.searchFilm])
@dp.message_handler(lambda message: message.text == keyboards.cancelKey, state=[DataInput.seacrhHuman,
DataInput.searchFilm])
async def cmd_cancel(message: types.Message, state: FSMContext):
    await state.finish()
    await message.answer("Действие отменено", reply_markup=keyboards.mainKeyboard)


@dp.callback_query_handler(text="cancel_button", state=[DataInput.seacrhHuman, DataInput.searchFilm])
async def callbacks_cancel(call: types.CallbackQuery, state: FSMContext):
```

```
    await state.finish()
    await call.message.delete_reply_markup()
    await call.message.answer("Действие отменено", reply_markup=keyboards.mainKeyboard)


# ------------


# Хэндлер на команду /film
@dp.message_handler(commands=['film'], state="*")
@dp.message_handler(lambda message: message.text == keyboards.MainKey.searchFilm)
async def cmd_film(message: types.Message, state: FSMContext):
    await state.finish()
    await message.reply(f"Введите название интересующего вас фильма/сериала", reply_markup=keyboards.cancelKeyboard)
    await DataInput.searchFilm.set()


# Хэндлер на команду /human
@dp.message_handler(commands=['human'], state="*")
@dp.message_handler(lambda message: message.text == keyboards.MainKey.searchActor)
async def cmd_human(message: types.Message, state: FSMContext):
    await state.finish()
    await message.reply(f"Введите имя интересующего вас человека", reply_markup=keyboards.cancelKeyboard)
    await DataInput.seacrhHuman.set()


# Хэндлер на команду /random
@dp.message_handler(commands=['random'], state="*")
@dp.message_handler(lambda message: message.text == keyboards.MainKey.randomFilm)
async def cmd_random(message: types.Message, state: FSMContext):

    await message.answer("Выполняю...", reply_markup=keyboards.hideKeyboard, parse_mode=types.ParseMode.HTML)
    film = await kinopoisk.get_random_film()

    text = ""
    if film.ru_name:
        text += f"{film.ru_name} ({film.name})"
    else:
        text += f"{film.name}"
    text += f"    {film.year}\n" if film.year else "\n"
    text += f"<i>{film.genres}</i>\n" if film.genres else ""
    text += f"\n<b>{film.description[:100]}...</b>\n" if film.description else ""

    text += f"\nРейтинг: {film.kp_rate}\n" if film.kp_rate else ""
    text += f"Рейтинг IMBD: {film.imdb_rate}\n" if film.imdb_rate else ""
    text += f"\n<b>Более подробнее: {film.url}</b>" if film.url else ""
    await state.finish()
    await bot.send_photo(message.from_user.id, film.poster, text, reply_markup=keyboards.mainKeyboard, parse_mode=types.ParseMode.HTML)


# Хэндлер на команду /top50
@dp.message_handler(lambda message: message.text == keyboards.MainKey.top500Film)
@dp.message_handler(commands=['top50'], state="*")
```

```
async def cmd_top(message: types.Message, state: FSMContext):
    await message.answer("Выполняю...", reply_markup=keyboards.hideKeyboard, parse_mode=types.ParseMode.HTML)
    films = await kinopoisk.top250_films()
    html = ''
    for i in enumerate(films[:50]):
        index = i[0] + 1
        data = i[1]
        star = ':star:'
        html += f"<p>{index}. <a href='{data.url}'>{f'{data.ru_name} ({data.name})' if data.ru_name else f'{data.name}'}</a>" \
                f"{f'  {data.year}<br>' if data.year else ''}" \
                f"{f'<i>{data.genres}</i><br>' if data.genres else ''}" \
                f"{f'{emoji.emojize(star)}Рейтинг: {data.kp_rate}<br>' if data.kp_rate else ''}" \
                f"<a href='{data.url}'><img src='{data.poster_preview}' height='100' alt='lorem'></a></p> <hr>"
    telegraph = Telegraph()
    telegraph.create_account(short_name=(await bot.get_me()).full_name)
    response = telegraph.create_page(
        f"{emoji.emojize(':star:')} Топ-50 фильмов",
        html_content=html
    )
    await state.finish()
    await message.answer(f"[{emoji.emojize(':star:')} Топ\-50 фильмов](https://telegra.ph/{response['path']})",
                         parse_mode=types.ParseMode.MARKDOWN_V2,
                         disable_web_page_preview=True,
                         reply_markup=keyboards.mainKeyboard
                         )


# ------------


# Поиск фильма
@dp.message_handler(state=DataInput.searchFilm)
async def film(message: types.Message, state: FSMContext):
    await message.answer("Выполняю...", reply_markup=keyboards.hideKeyboard, parse_mode=types.ParseMode.HTML)
    films = await kinopoisk.search_film(message.text)

    if films["result"] is None:
        await message.answer(f"{emoji.emojize(':warning:')}ERROR 404\nНичего не найдено\n Попробуйте поискать иначе",
                             reply_markup=keyboards.cancelKeyboard)
    else:
        text = ""
        text += f"Поиск - \"{films['query']}\"    Найдено - \"{films['numResults']}\"\n\n"
        for i in enumerate(films["result"]):
            index = i[0] + 1
            film = i[1]
            text += f"{index}.  <a href='{film.url}'>"
            if film.ru_name:
                text += f"{film.ru_name} ({film.name})"
            else:
                text += f"{film.name}"
            text += f"</a>     {film.year}\n" if film.year else "\n"
            text += f"<i>{film.genres}</i>\n" if film.genres else ""
            text += f"\nРейтинг: {film.kp_rate}" if film.kp_rate else ""
```

```
            text += f"  (IMBD: {film.imdb_rate})\n" if film.imdb_rate else ""
            text += f"\n---\n\n"
        text += f'Все результаты поиска: {films['resultUrl']}"
        await state.finish()
        await message.answer(text,
                             reply_markup=keyboards.mainKeyboard,
                             parse_mode=types.ParseMode.HTML,
                             disable_web_page_preview=True)


# Поиск человека
@dp.message_handler(state=DataInput.seacrhHuman)
async def human(message: types.Message, state: FSMContext):
    await message.answer("Выполняю...", reply_markup=keyboards.hideKeyboard, parse_mode=types.ParseMode.HTML)
    persons = await kinopoisk.search_person(message.text)

    if persons["result"] is None:
        await message.answer(f"{emoji.emojize(':warning:')}ERROR 404\nНикого не найдено\n Попробуйте поискать иначе",
                             reply_markup=keyboards.cancelKeyboard)
    else:
        text = ""
        text += f'Поиск - \"{persons['query']}\"    Найдено - \"{persons['numResults']}\"\n\n"
        for i in enumerate(persons["result"]):
            index = i[0] + 1
            person = i[1]
            text += f"{index}.  <a href='{person.url}'>"
            text += f"{emoji.emojize(':man_dancing:')}" if person.sex == 'M' else f"{emoji.emojize(':woman_dancing:')}"
            text += f"{person.ru_name} ({person.name})" if person.ru_name else f"{person.name}"
            text += f"</a>\n"
            text += f"{person.birthday}-"if person.birthday else ""
            text += f"{person.death}" if person.death else ""
            text += f"  ({person.age})\n" if person.age else "\n"
            text += f"<i>{person.profession}</i>\n" if person.profession else ""
            text += f"\n---\n\n"
        text += f'Все результаты поиска: {persons['resultUrl']}"
        await state.finish()
        await message.answer(text,
                             reply_markup=keyboards.mainKeyboard,
                             parse_mode=types.ParseMode.HTML,
                             disable_web_page_preview=True)


# ------------


@dp.message_handler(content_types=types.ContentType.ANY)
async def unknown_message(msg: types.Message):
    await msg.reply(f"Я не знаю, что с этим делать {emoji.emojize(':upside-down_face:')}\n"
                    f"\n"
                    f"_Список доступных команд_ \- /help", reply_markup=keyboards.mainKeyboard)


async def set_commands(bot: Bot):
    commands = [
```

```
        types.BotCommand(command="/help", description="Помощь"),
        types.BotCommand(command="/start", description="Перезапуск бота"),
        types.BotCommand(command="/film", description="Поиск фильма"),
        types.BotCommand(command="/human", description="Поиск человека"),
        types.BotCommand(command="/random", description="Случайный фильм"),
        types.BotCommand(command="/top50", description="Топ-50 лучших фильмов"),
        types.BotCommand(command="/cancel", description="Отменить текущее действие"),
        types.BotCommand(command="/menu", description="Отобразить клавиатуру"),
        types.BotCommand(command="/hide", description="Скрыть клавиатуру")
    ]
    await bot.set_my_commands(commands)


if __name__ == "__main__":
    # Запуск бота
    executor.start_polling(dp, skip_updates=True)
```

keyboards.py

```
from aiogram.types import ReplyKeyboardRemove, ReplyKeyboardMarkup, KeyboardButton, InlineKeyboard-
Markup, \
    InlineKeyboardButton
import emoji


class MainKey:
    searchFilm = f"{emoji.emojize(':clapper_board:')} Поиск фильма/сериала"
    searchActor = f"{emoji.emojize(':woman_dancing:')} Поиск человека"
    randomFilm = f"{emoji.emojize(':game_die:')} Случайный фильм/сериал"
    top500Film = f"{emoji.emojize(':TOP_arrow:')} Топ 50 фильмов"


mainKeyboard = ReplyKeyboardMarkup(resize_keyboard=True)
mainKeyboard.add(*[MainKey.searchFilm, MainKey.searchActor])
mainKeyboard.add(*[MainKey.randomFilm, MainKey.top500Film])


# ------------


cancelKey = f"{emoji.emojize(':stop_sign:')} Отменить действие"
cancelKeyboard = ReplyKeyboardMarkup(resize_keyboard=True)
cancelKeyboard.add(cancelKey)

cancelKeyboardInline = InlineKeyboardMarkup()
cancelKeyboardInline.add(InlineKeyboardButton(text=cancelKey, callback_data="cancel_button"))


# ------------


hideKeyboard = ReplyKeyboardRemove()
```

kinopoisk_api.py

```
import os
import random
```

```
import xml.etree.ElementTree as xml
import requests
import json
import re
import logging
from bs4 import BeautifulSoup
from urllib.parse import quote


class Cache:
    def __init__(self):
        self.PATH = os.path.dirname(os.path.abspath(__file__))

    def load(self) -> dict:
        try:
            with open(self.PATH + '/cache.json', 'r') as f:
                return json.loads(f.read())
        except FileNotFoundError:
            with open(self.PATH + '/cache.json', 'w') as f:
                return {}

    def write(self, cache: dict, indent: int = 4):
        with open(self.PATH + '/cache.json', 'w') as f:
            return json.dump(cache, f, indent=indent)


class Person:
    def __init__(self, data: dict):
        self._id = data['personId']
        self.name = data['nameRu'] if data['nameEn'] is None else data['nameEn']
        self.ru_name = data['nameRu']
        self.sex = data['sex']
        self.birthday = data['birthday']
        self.death = data['death']
        self.age = data['age']
        self.growth = data['growth']
        self.birthplace = data['birthplace']
        self.deathplace = data['deathplace']
        self.profession = data['profession']
        self.facts = data['facts']
        self.poster = data['posterUrl']
        self.url = data['webUrl']



class Film:
    def __init__(self, data: dict):
        self._id = data['filmId']
        self.name = data['nameRu'] if data['nameEn'] is None else data['nameEn']
        self.ru_name = data['nameRu']
        self.year = data['year']
        self.duration = data['filmLength']
        self.slogan = data['slogan']
        self.description = data['description']
        self.genres = [genre['genre'] for genre in data['genres']]
        self.countries = [country['country'] for country in data['countries']]
        self.age_rating = data['ratingAgeLimits']
```

```
        self.kp_rate = data['kp_rate']
        self.imdb_rate = data['imdb_rate']
        self.url = data['webUrl']
        self.premiere = data['premiereWorld']
        self.poster = data['posterUrl']
        self.poster_preview = data['posterUrlPreview']


class KinopoiskApi:
    def __init__(self, token, secret=None):
        self.token = token
        self.headers = {"X-API-KEY": self.token}
        self.API = 'https://kinopoiskapiunofficial.tech/api/'

    async def get_film(self, film_id):
        api_version = 'v2.1/'
        cache = Cache().load()
        rate_request = requests.get(f'https://rating.kinopoisk.ru/{film_id}.xml')
        if rate_request.status_code == 404:
            return None
        try:
            kp_rate = xml.fromstring(rate_request.text)[0].text
        except IndexError:
            kp_rate = None
        try:
            imdb_rate = xml.fromstring(rate_request.text)[1].text
        except IndexError:
            imdb_rate = None

        if str(film_id) in cache:
            data = cache[str(film_id)]
            logging.info('Фильм был в кеше')
        else:
            request = requests.get(self.API + api_version + 'films/' + str(film_id), headers=self.headers)
            if request.status_code == 404:
                return None
            data = json.loads(request.text)['data']
            logging.info('Фильма не было в кеше')

        data['kp_rate'] = kp_rate
        data['imdb_rate'] = imdb_rate
        cache[str(film_id)] = data
        Cache().write(cache)
        return Film(data)

    async def get_person(self, person_id):
        api_version = 'v1/'

        request = requests.get(self.API + api_version + 'staff/' + str(person_id), headers=self.headers)
        if request.status_code == 404:
            return None
        data = json.loads(request.text)

        return Person(data)

    async def get_random_film(self):
        chance = None
```

```
        while chance is None:
            chance = await self.get_film(random.randint(1, 1450000))
        return chance

    async def top250_films(self):
        api_version = 'v2.1/'
        request = requests.get(self.API + api_version + 'films/top?type=TOP_250_BEST_FILMS', headers=self.headers)
        if request.status_code == 404:
            return None
        pages = json.loads(request.text)["pagesCount"]
        output = []
        for i in range(pages):
            request = requests.get(self.API + api_version + f'films/top?type=TOP_250_BEST_FILMS&page={i + 1}',
                                   headers=self.headers)
            request_json = json.loads(request.text)

            for film in request_json["films"]:
                film["slogan"] = None
                film["description"] = None
                film["ratingAgeLimits"] = None
                film["kp_rate"] = film["rating"]
                film["imdb_rate"] = None
                film["webUrl"] = f"http://www.kinopoisk.ru/film/{film['filmId']}"
                film["premiereWorld"] = None
                output.append(Film(film))
        return output

    async def search_person(self, query):
            url = f"https://www.kinopoisk.ru/s/type/people/list/1/find/{quote(query)}/order/relevant/"

            request = requests.get(url)
            soup = BeautifulSoup(request.text, 'html.parser')

            # search_results = soup.find('span', attrs={'class': re.compile('search_results_topText')}).text.split()
            # query = search_results[1]
            # num_results = search_results[-1]
            search_results = soup.find('span', attrs={'class': re.compile('search_results_topText')})
            query = search_results.b.text
            num_results = search_results.text.split()[-1]
            if num_results == '0':
                return {'query': query, 'numResults': num_results, 'result': None, 'resultUrl': url}

            results = soup.findAll('div', attrs={'class': re.compile('element')})
            results10 = results[:10]
            result = []
            for i in results10:
                result.append(await self.get_person(i.p.a["data-id"]))

            return {'query': query, 'numResults': num_results, 'result': result, 'resultUrl': url}

    async def search_film(self, query):
            url = f"https://www.kinopoisk.ru/s/type/film/list/1/find/{quote(query)}/order/relevant/"

            request = requests.get(url)
            soup = BeautifulSoup(request.text, 'html.parser')
```

```
# search_results = soup.find('span', attrs={'class': re.compile('search_results_topText')}).text.split()
# query = search_results[1]
# num_results = search_results[-1]
search_results = soup.find('span', attrs={'class': re.compile('search_results_topText')})
query = search_results.b.text
num_results = search_results.text.split()[-1]
if num_results == '0':
    return {'query': query, 'numResults': num_results, 'result': None, 'resultUrl': url}

results = soup.findAll('div', attrs={'class': re.compile('element')})
results10 = results[:10]
result = []
for i in results10:
    result.append(await self.get_film(i.p.a["data-id"]))

return {'query': query, 'numResults': num_results, 'result': result, 'resultUrl': url}
```

Отзыв о работе студента

_____

(ФИО студента)

Уровень освоения компетенций

_____

(ФИО студента)

Компетенции	Уровень сформированности компетенций
ОК-6 – способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия	
ОК-7 – способностью к самоорганизации и самообразованию	
ПК-2 - способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	

отметка о зачете _____

Руководитель практики от СибГУТИ:

_____ _____ _____

Должность руководителя подпись ФИО руководителя

"___" _____ 20____ г.