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

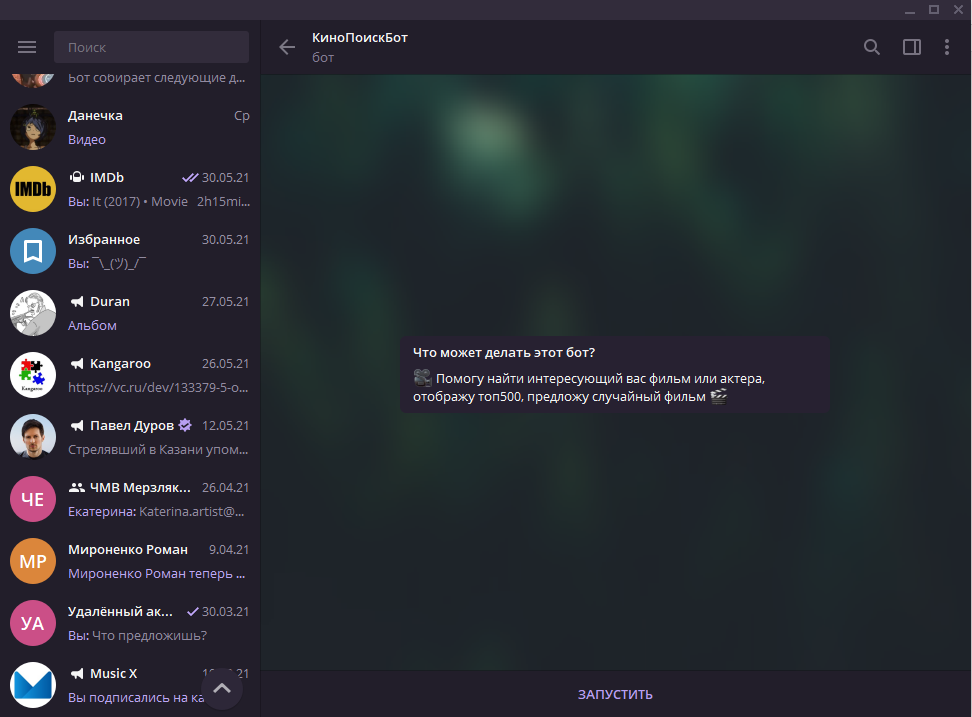
В ходе работы над данным проектом я познакомился с технологиями разработки telegram-ботов, изучил TelegramAPI. Улучшил свои навыки написания программ на языке python, базово освоил асинхронный фреймворк «aiogram», а также изучил некоторые другие пакеты.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

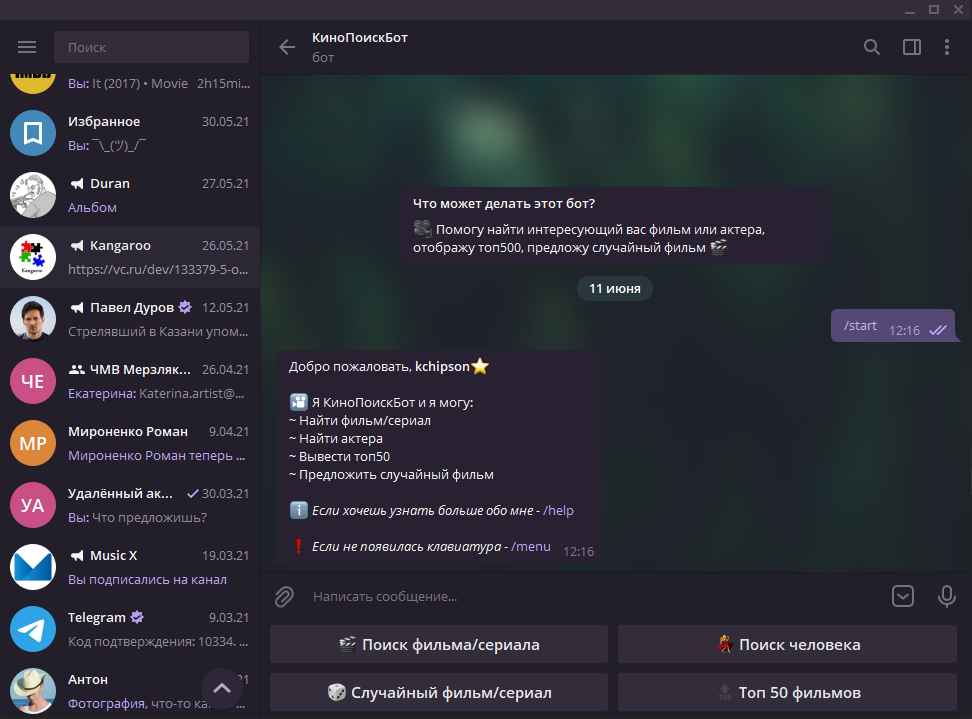
1. Python 3.9.5 documentation [Электронный ресурс] URL: <https://docs.python.org/3/>
2. Telegram Bot API [Электронный ресурс] URL: <https://core.telegram.org/bots/api>
3. aiogram’s documentation [Электронный ресурс] URL: <https://docs.aiogram.dev/en/latest/>
4. Kinopoisk Api Unofficial [Электронный ресурс]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

# ПРИЛОЖЕНИЯ

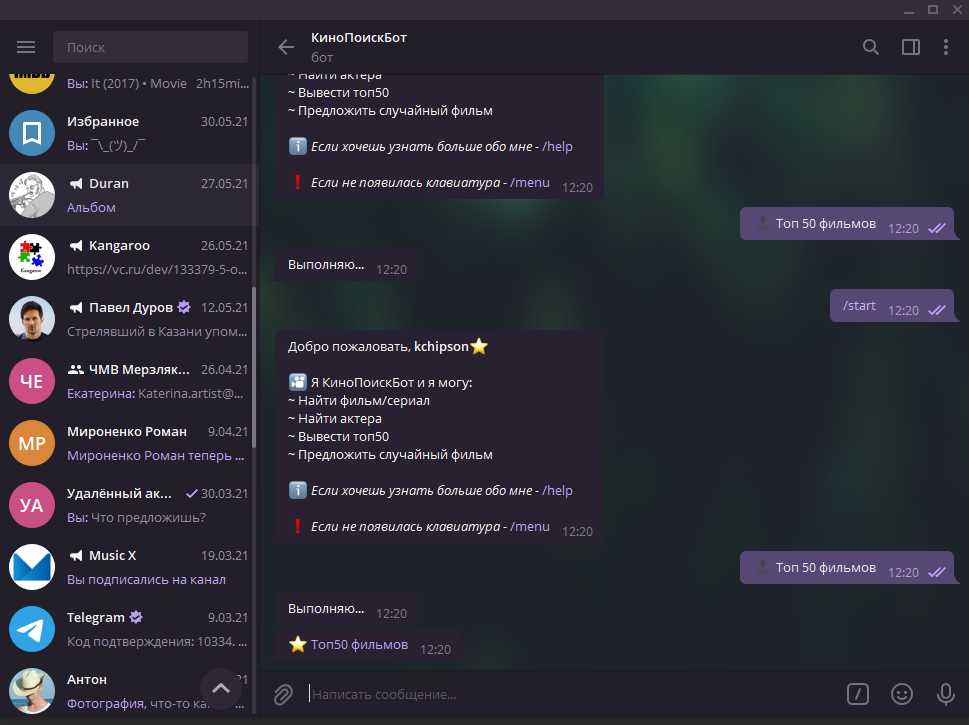
## Пример работы



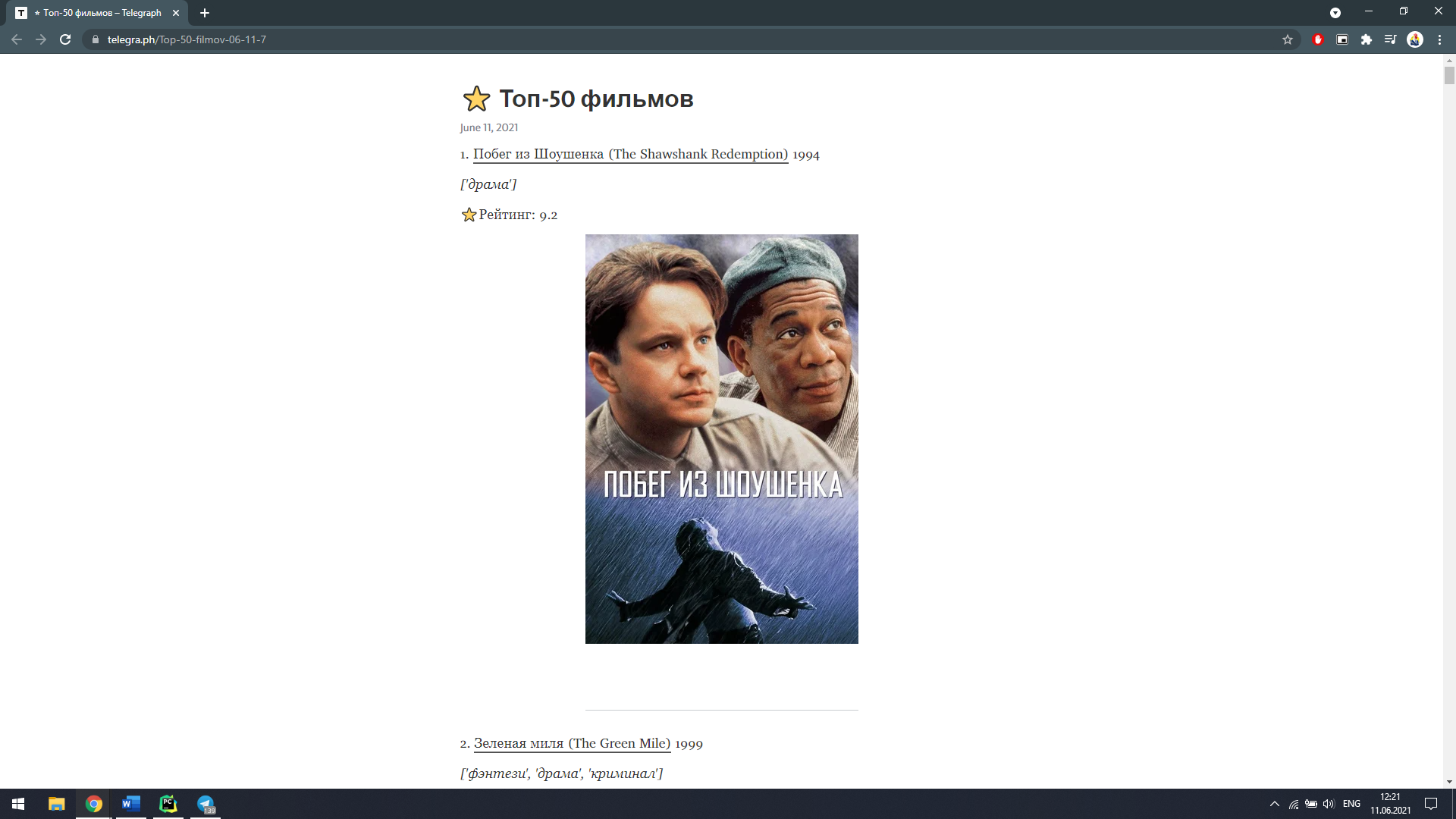
(Рис 1. Начало работы бота)



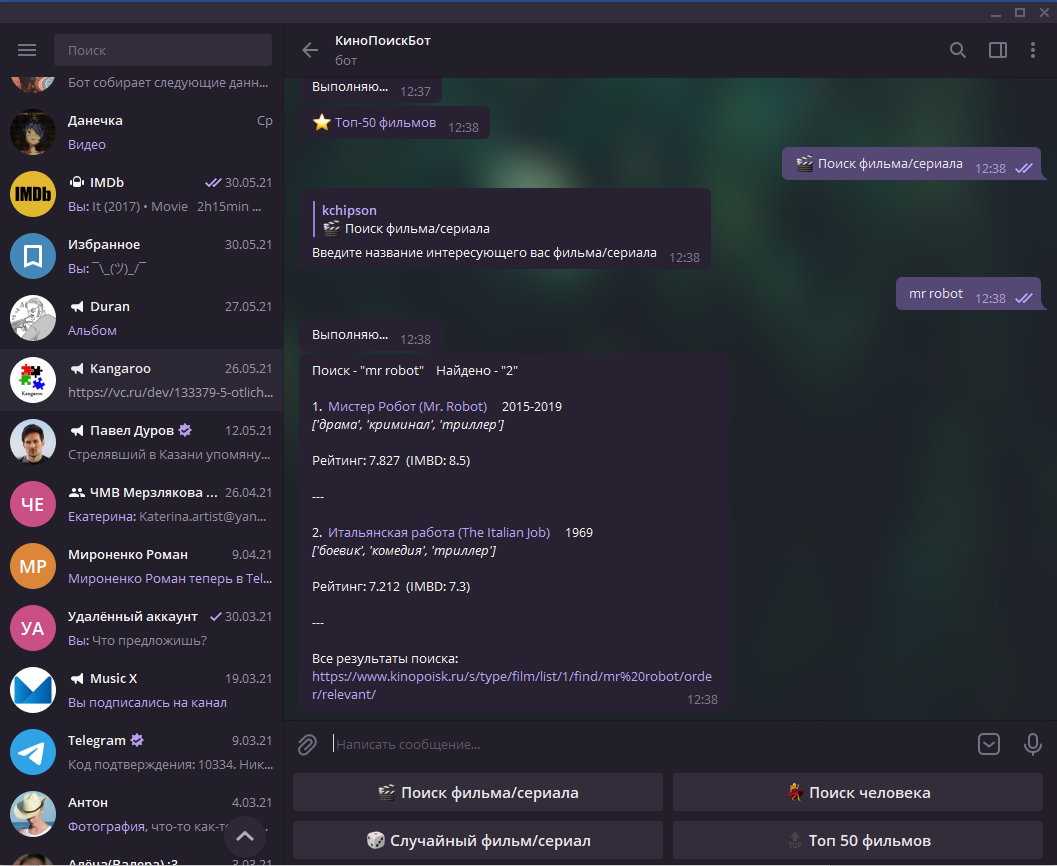
(Рис 2. Приветственное сообщение)



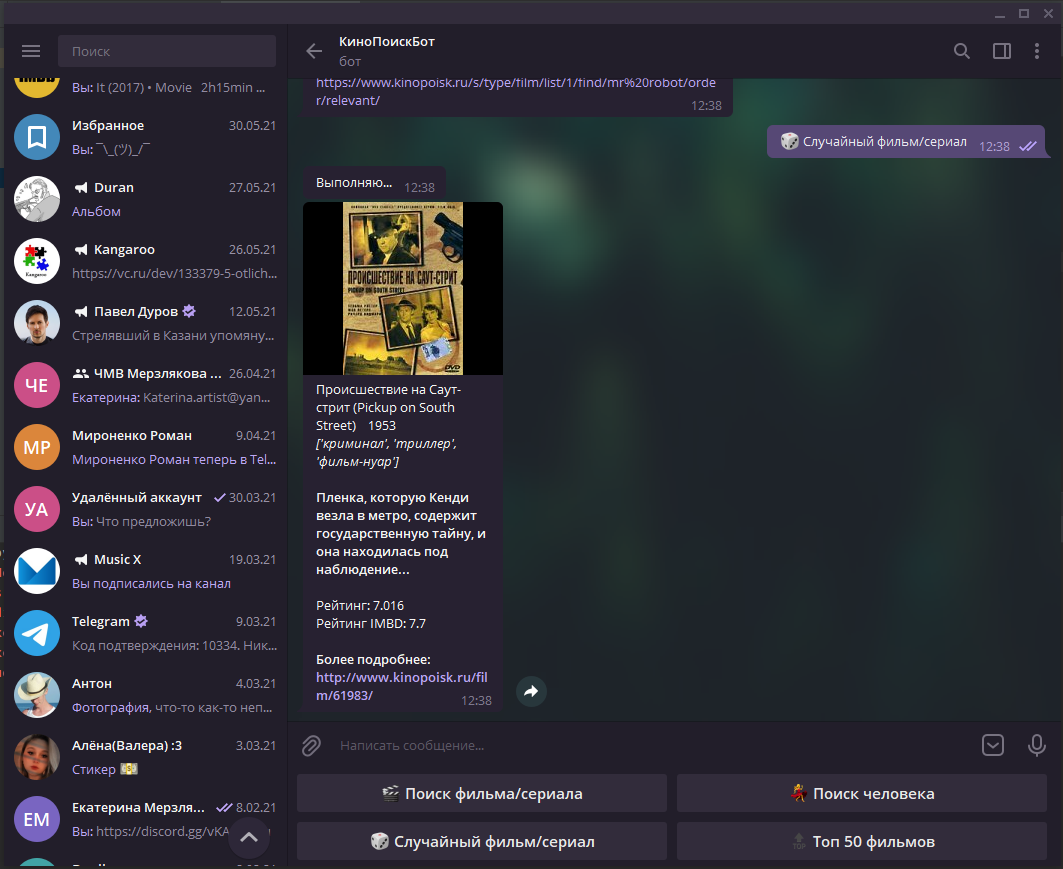
(Рис 3.1. Топ50)



(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, InlineKeyboardMarkup, \

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:')} Cлучайный 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.