**РУКОВОДСТВО ПРОГРАММИСТА**

**Telegram chat-bot**

**«MVBOT»**

**Листов 15**

2022

**АННОТАЦИЯ**

В данном программном документе приведено руководство программиста по работе с чат-ботом «MVBOT».

В разделе «Назначение и условия применения программы» указаны назначение и функции, выполняемые чат-ботом, условия, необходимые для его работы.

В разделе «Характеристика сайта» приведено описание основных

характеристик и особенностей.

1. Назначение и условия применения программы

Целью проекта было создание приложения, специализирующегося на поиске и подборе фильмов и киноплощадок. С помощью данной программы пользователь может выбрать удобный для себя способ просмотра фильма: при посещении оффлайн кинотеатра, находящегося в городе, либо же онлайн просмотр фильма через специальные киноплощадки в Интернете.

Поскольку именно через Интернет потенциальные любители фильмов будут искать информацию о том, какой посмотреть фильм или на какой из доступных сеансов пойти в городе Кирове, был разработан чат-бот, где люди могут просматривать актуальную афишу и информацию, связанную с кинематографом.

В современном мире в сети интернет довольно много киноплощадок, которые предоставляют услуги просмотра фильмов, именно поэтому существует проблема быстрого выбора фильма.

Для решения проблемы быстрого поиска интересующего контента был разработан телеграмм-бот для подбора нужной площадки для просмотра фильма. Также бот обладает функцией подбора рандомного фильма, что также облегчает выбор фильма для просмотра. Телеграмм-бот является универсальным и может быть использован людьми разных возрастных групп, интересов и конфессий.

1. Характеристики и работа программы

Требования к логике чат-бота, которые были выполнены в ходе разработки:

1. Доступ к чат-боту открыт для любого человека, имеющий доступ в интернет и располагающий аккаунтом на платформе «Telegram».
2. Получение актуальной информации об афишах в кинотеатрах города Кирова.
3. Составление свода правил о нашей деятельности.
4. Отправка записей в базу данных «SQLite», для хранений и последующей модерации.
5. Взаимодействие с API Кинопоиска, посредством внедрения его в проект и такие возможности, как: нахождение фильма, актера, отзывов о фильме.
6. Фильтрация и отправка пользователю информации в удобочитаемом виде.

Для того, чтобы развернуть сайт требуется:

1. Скачать и установить клиент системы контроля версий Git по следующей ссылке: <https://git-scm.com/downloads>
2. Скачать и установить язык программирования «Python» по следующей ссылке: <https://www.python.org/>
3. После того, как была установлена данная программная среда, следует установить один из редакторов кода, к примеру, PyCharm: <https://www.jetbrains.com/ru-ru/pycharm/>

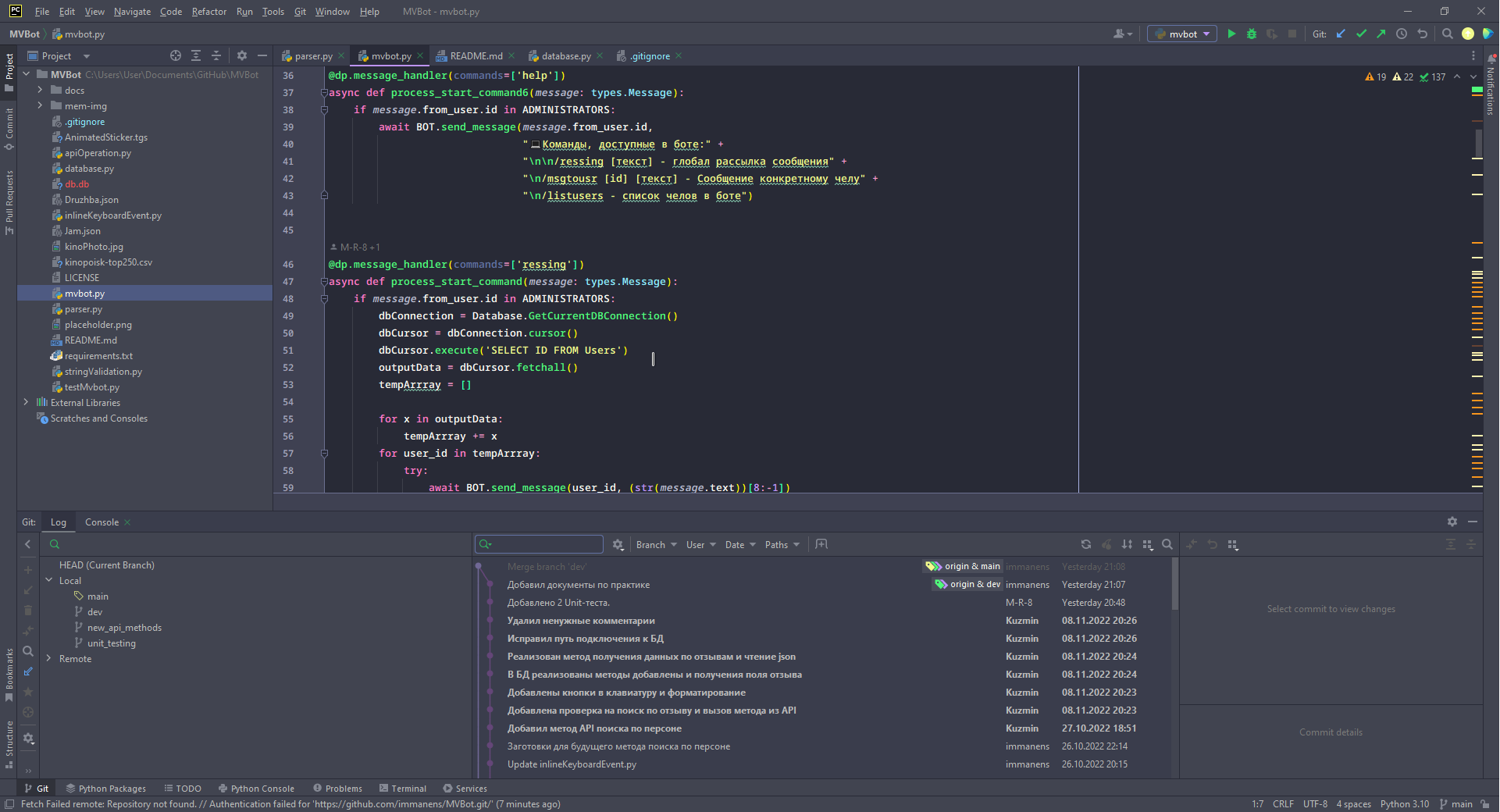


Рисунок 1 – Открытый проект «MVBOT»

На рисунке 2 представлена файловая архитектура форума.

1. Запустите файл mvbot.py из IDE или используете терминал: ‘python3 mvbot.py’

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Файловая архитектура чат-бота

Для работы чат-бота необходимы все файлы, представленные на рисунке 2, их можно разделить на следующие группы:

1. Главный файл – mvbot, откуда происходит запуск всего проекта
2. Виртуальная клавиатура-панель
3. Парсер, собирающий информацию об актуальных афишах кинотеатров
4. Unit-тесты
5. Взаимодействия с API Кинопоиска.
6. Взаимодействиме с базой данных
7. Валидация данных

При создании чат-бота использовались языки Python для разработки и база даи база данных «SQLite».

ЛИСТИНГ ПРОГРАММЫ

**«mvbot.py»**

from aiogram import Bot, types  
from aiogram.contrib.fsm\_storage.memory import MemoryStorage  
from aiogram.dispatcher import Dispatcher  
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton  
from aiogram.utils import executor  
from database import Database  
from apiOperation import ApiOperation  
from datetime import date as dt  
from inlineKeyboardEvent import InlineKeyboardEvent  
  
# region Constants  
  
TOKEN = "5149320687:AAEuCMs7xshTAJYgGjBRLgW1yTRPULpsiG0"  
API\_TOKEN = "9tdyx0v-kj946kd-j7ea9bb-rm2yrfw"  
BOT = Bot(token=TOKEN)  
DATE\_PARSE = dt(2022, 3, 29)  
DB = Database('db.db')  
ADMINISTRATORS = [514133808]  
  
# endregion  
  
# region Variables  
  
storage = MemoryStorage()  
dp = Dispatcher(BOT, storage=storage)  
Actualization\_InProcess = False  
JamDays = [] # Кол-во фильмов у каждой даты  
JamDate = [] # Лист с датами  
Jam = [] # Json в памяти  
  
# endregion  
  
# region Commands  
  
  
@dp.message\_handler(commands=['help'])  
async def process\_start\_command6(*message*: types.Message):  
 if *message*.from\_user.id in ADMINISTRATORS:  
 await BOT.send\_message(*message*.from\_user.id,  
 "💻Команды, доступные в боте:" +  
 "\n\n/ressing [текст] - глобал рассылка сообщения" +  
 "\n/msgtousr [id] [текст] - Сообщение конкретному челу" +  
 "\n/listusers - список челов в боте")  
  
  
@dp.message\_handler(commands=['ressing'])  
async def process\_start\_command(*message*: types.Message):  
 if *message*.from\_user.id in ADMINISTRATORS:  
 dbConnection = Database.GetCurrentDBConnection()  
 dbCursor = dbConnection.cursor()  
 dbCursor.execute('SELECT ID FROM Users')  
 outputData = dbCursor.fetchall()  
 tempArrray = []  
  
 for x in outputData:  
 tempArrray += x  
 for user\_id in tempArrray:  
 try:  
 await BOT.send\_message(user\_id, (str(*message*.text))[8:-1])  
 except:  
 print(str(user\_id) + "заблокировал бота(")  
 else:  
 return  
  
  
@dp.message\_handler(commands=['msgtousr'])  
async def process\_start\_command(*message*: types.Message):  
 if *message*.from\_user.id in ADMINISTRATORS:  
 arrayMsg = *message*.text.split(" ")  
 try:  
 await BOT.send\_message(int(arrayMsg[1]), (str(*message*.text))[10 + len(arrayMsg[1]):])  
 await BOT.send\_message(*message*.from\_user.id, "Отправлено!")  
 except:  
 await BOT.send\_message(*message*.from\_user.id, "Пользователь не найден!")  
 else:  
 return  
  
  
@dp.message\_handler(commands=['listusers'])  
async def process\_start\_command3(*message*: types.Message):  
 if *message*.from\_user.id in ADMINISTRATORS:  
 dbConnection = Database.GetCurrentDBConnection()  
 dbCursor = dbConnection.cursor()  
 dbCursor.execute('SELECT first\_name FROM Users')  
 first\_name = dbCursor.fetchall()  
 dbCursor.execute('SELECT Username FROM Users')  
 username = dbCursor.fetchall()  
 dbCursor.execute('SELECT ID FROM Users')  
 id = dbCursor.fetchall()  
 first\_nameArray = []  
 usernameArray = []  
 idArray = []  
 targetText = "💻Список пользователей бота:\n\n"  
 iteration = 0  
  
 for x in first\_name:  
 first\_nameArray += x  
 for x in username:  
 usernameArray += x  
 for x in id:  
 idArray += x  
  
 for x in usernameArray:  
 if len(targetText) > 4095:  
 tempo = targetText[-(len(targetText) - 4095):-1]  
 targetText = targetText[:-(len(targetText) - 4095)]  
 await BOT.send\_message(*message*.from\_user.id, targetText)  
 targetText = tempo  
 targetText += "\n#" + str(iteration) + " ID - " + str(idArray[iteration]) + ", " + str(  
 first\_nameArray[iteration]) + ", ссылка на профиль - @" + str(x) + "\n"  
 iteration += 1  
 else:  
 targetText += "\n#" + str(iteration) + " ID - " + str(idArray[iteration]) + ", " + str(  
 first\_nameArray[iteration]) + ", ссылка на профиль - @" + str(x) + "\n"  
 iteration += 1  
 targetText += "\n👥Общее число пользователей - " + str(iteration)  
  
 await BOT.send\_message(*message*.from\_user.id, targetText)  
  
  
@dp.message\_handler(commands=['start'])  
async def process\_start\_command(*message*: types.Message):  
 DB.add\_id\_user(*message*.from\_user.id, *message*.from\_user.first\_name, *message*.from\_user.last\_name,  
 *message*.from\_user.username, 0, "False", "False", 0, "False", "none", "False", "False", "False")  
  
 if Actualization\_InProcess:  
 await BOT.answer\_callback\_query(  
 *message*.chat.id,  
 text='♻️В настоящий момент бот актуализирует данные по киношкам.\nПогоди немного🙂', show\_alert=True)  
 return  
  
 button\_hx = KeyboardButton('🍿Просмотр афиш')  
 button\_h1 = KeyboardButton('💻Просмотр кино онлайн')  
 button\_h2 = KeyboardButton('🎰Рандомный фильм🎰')  
 button\_h4 = KeyboardButton('📜Правила')  
 button\_h5 = KeyboardButton('💡О нас')  
 greet\_kb = ReplyKeyboardMarkup(resize\_keyboard=True, one\_time\_keyboard=False).add(button\_hx, button\_h1, button\_h2,  
 button\_h4, button\_h5)  
 sti = open('AnimatedSticker.tgs', 'rb')  
 await BOT.send\_sticker(*message*.chat.id, sti)  
 await *message*.reply(  
 "Велком, {0.first\_name}!\nСервис предоставляет вам список сеансов, которые идут в кинотеатрах города Киров, а так же то, что идет в инете:)🍿".format(  
 *message*.from\_user), parse\_mode='html', reply\_markup=greet\_kb)  
  
  
# endregion  
  
# region Обработка текста  
  
@dp.message\_handler(content\_types=['text'])  
async def process\_command\_1(*message*: types.Message):  
 if *message*.chat.type == 'private':  
 test = str(DB.check\_ACTIVE\_TAB(*message*.from\_user.id))[2:-3]  
 try:  
 #if Actualization\_InProcess:  
 # await BOT.answer\_callback\_query(  
 # message.id,  
 # text='♻️В настоящий момент бот актуализирует данные по киношкам.\nПогоди немного🙂', show\_alert=True)  
 # return  
  
 if (str(DB.check\_findByIdBool(*message*.from\_user.id))[2:-3]) == 'True':  
 await ApiOperation.FindFilmById(DB, BOT, *message*, API\_TOKEN)  
 return  
  
 elif (str(DB.check\_findByNameBool(*message*.from\_user.id))[2:-3]) == 'True':  
 await ApiOperation.FindFilmByName(DB, BOT, *message*, API\_TOKEN)  
 return  
  
 elif (str(DB.check\_findPersonByName(*message*.from\_user.id))[2:-3]) == 'True':  
 await ApiOperation.FindPersonByName(DB, BOT, *message*, API\_TOKEN)  
 return  
  
 elif (str(DB.check\_findReviewById(*message*.from\_user.id))[2:-3]) == 'True':  
 await ApiOperation.FindReviewById(DB, BOT, *message*, API\_TOKEN)  
 return  
  
 if (str(DB.check\_ACTIVE\_TAB(*message*.from\_user.id))[2:-3]) == 'True':  
 await BOT.send\_message(*message*.from\_user.id,  
 'У тебя уже есть активное окно, закончи с ним, чтобы открыть новое💡')  
 return  
  
 if *message*.text == '🍿Просмотр афиш':  
 await InlineKeyboardEvent.ViewCinemaPosters(BOT, DB, *message*)  
 DB.add\_ACTIVE\_TAB('True', *message*.from\_user.id)  
 return  
  
 elif *message*.text == '💻Просмотр кино онлайн':  
 await InlineKeyboardEvent.ViewOnlineCinema(BOT, *message*)  
 DB.add\_ACTIVE\_TAB('True', *message*.from\_user.id)  
 return  
  
 elif *message*.text == '🎰Рандомный фильм🎰':  
 await InlineKeyboardEvent.SendRandomFilm(BOT, DB, *message*)  
 return  
  
 elif *message*.text == '📜Правила':  
 await InlineKeyboardEvent.SendRules(BOT, *message*)  
 DB.add\_ACTIVE\_TAB('False', *message*.from\_user.id)  
 return  
  
 elif *message*.text == '💡О нас':  
 await InlineKeyboardEvent.AboutUs(BOT, DB, *message*)  
 return  
  
 else:  
 DB.add\_ACTIVE\_TAB('False', *message*.from\_user.id)  
 await BOT.send\_message(*message*.from\_user.id, 'Пользуйся кнопками, я же бот, а не собеседник😊')  
 return  
 except:  
 pass  
 else:  
 pass  
  
  
# endregion  
  
# region Обработка кнопок  
  
@dp.callback\_query\_handler(lambda *c*: c.data)  
async def process\_callback\_kb1btn1(*callback\_query*: types.CallbackQuery):  
 global dt, DATE\_PARSE, Jam, JamDate, JamDays, Actualization\_InProcess  
 try:  
 #if (DATE\_PARSE < dt.today()):  
 # DATE\_PARSE = dt.today()  
 # #(jam, jamDate, jamDays) = Parser.Actualization\_Data(Jam, JamDate, JamDays)  
 # return  
 if (Actualization\_InProcess):  
 await BOT.answer\_callback\_query(  
 *callback\_query*.id,  
 text='♻️В настоящий момент бот актуализирует данные по киношкам.\nПогоди немного🙂', show\_alert=True)  
 return  
  
 code = *callback\_query*.data  
 if code == 'exit' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 DB.add\_ACTIVE\_TAB('False', *callback\_query*.message.chat.id)  
 try:  
 await BOT.edit\_message\_text(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id, text="Жду твоих указаний👾",  
 reply\_markup=None)  
 except:  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 return  
  
 elif code == "exitOne" and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 DB.add\_Panel(0, *callback\_query*.from\_user.id)  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 return  
  
 elif code == "smenaPanelLeft" and (  
 str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.ScrollFrirendPanel(BOT, DB, *callback\_query*, "Left", Jam, "smenaPanelLeft", "smenaPanelRight", "smenaCinema")  
 return  
  
 elif code == "smenaPanelRight" and (  
 str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.ScrollFrirendPanel(BOT, DB, *callback\_query*, "Right", Jam, "smenaPanelLeft", "smenaPanelRight", "smenaCinema")  
 return  
  
 elif code == "goCinemaKirow" and (  
 str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.GetCinemaKirow(BOT, DB, *callback\_query*)  
 return  
  
 elif code == 'payCinema' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.GetPayCinemaCallback(BOT, DB, *callback\_query*)  
 return  
  
 elif code == 'findById' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 await BOT.send\_message(*callback\_query*.message.chat.id, "Введите идентификатор фильма!")  
 DB.add\_findByIdBool('True', *callback\_query*.from\_user.id)  
 return  
  
 elif code == 'findByName' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 await BOT.send\_message(*callback\_query*.message.chat.id, "Введите название фильма")  
 DB.add\_findByNameBool('True', *callback\_query*.from\_user.id)  
 return  
  
 elif code == 'findPersonByName' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 await BOT.send\_message(*callback\_query*.message.chat.id, "Введите название персоны")  
 DB.add\_findPersonByName('True', *callback\_query*.from\_user.id)  
 return  
  
 elif code == 'findReviewById' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await BOT.delete\_message(chat\_id=*callback\_query*.message.chat.id,  
 message\_id=*callback\_query*.message.message\_id)  
 await BOT.send\_message(*callback\_query*.message.chat.id, "Введите идентификатор фильма, отзыв на который хотите прочитать! =)")  
 DB.add\_findReviewById('True', *callback\_query*.from\_user.id)  
 return  
  
 elif code == 'goBack' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.GoBack(BOT, *callback\_query*)  
 return  
  
 elif code == 'freeCinema' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.GetFreeCinema(BOT, DB, *callback\_query*)  
 return  
  
 elif code == 'smenaCinema' and (str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False':  
 await InlineKeyboardEvent.GetSmenaCinema(BOT, DB, JamDate, *callback\_query*)  
 return  
 if ((str(DB.check\_ACTIVE\_TAB(*callback\_query*.from\_user.id))[2:-3]) == 'True') and (  
 str(DB.check\_findByIdBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findByNameBool(*callback\_query*.from\_user.id))[2:-3]) == 'False' and (  
 str(DB.check\_findPersonByName(*callback\_query*.from\_user.id))[2:-3] == 'False'):  
 await InlineKeyboardEvent.GetjamFilmsPanel(  
 BOT, DB, JamDate, Jam, *callback\_query*, code)  
 except Exception as e:  
 print(str(e))  
  
# endregion  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 executor.start\_polling(dp, skip\_updates=True)