Задачи:

1. разработка регулярного выражения для поиска в произвольном тексте команд чат-боту и параметров выполнения команд;
2. реализация требуемых действий чат-бота в предварительной заготовке программы на языке Python;
3. разработка заготовки самого чат-бота, принимающего сообщения от пользователя и отправляющего сообщения пользователю;
4. компоновка всех перечисленных выше этапов в рамках чат-бота.

При разработке чат бота, производящего напоминания пользователю были выявлены следующие ключевые проблемы:

Эффективное хранение сообщений – необходимо максимально экономить пространство для хранения картинок и видео – желательно не хранить ничего вовсе. Для хранения данных использовалась база данных SQLite. Отдельные не табличные данные хранятся в обычных текстовых файлах. Фотографии и иной контент бот не хранит вовсе – при необходимости отправки чего либо в ответ, оно отправляет id и тип контента в ответ пользователю, следовательно нет необходимости проводить мониторинг сохранённого материала.

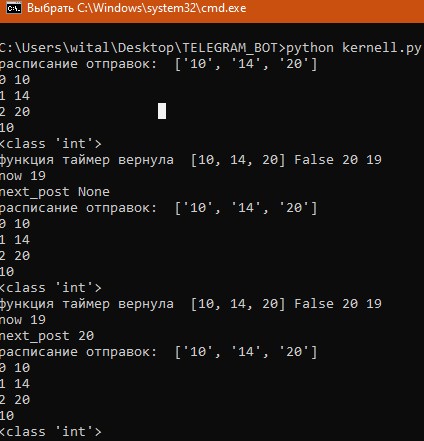
Регулярный опрос серверов телеграма (polling) на наличие новых сообщений от пользователей должен быть прерван на рассылку сохраненных уведомлений, поскольку телеграм не может справиться с двумя подключенными к одному аккаунту машинами. Для решения этой задачи была разработана система из двух процессов – один непосредственно бот, взаимодействующий с пользователем, второй – программа, производящая рассылку сообщений. Очень важно было сделать это именно таким образом, поскольку требуется надежное завершение одного соединения прежде чем начать второе.

База данных SQLite однопоточная – надежный подход, не предполагающий большой нагрузки и нескольких соединений. Как получилось обойти это ограничение? Файл управления базой данных отделен от основной программы. Содержит необходимые для работы функции – каждая функция создает новое подключение и разрывает его после необходимых операций – вызов этих функций осуществляется через посредника, осуществляющего предобработку данных – рассылочный и обычный бот не взаимодействуют с базой данных напрямую.

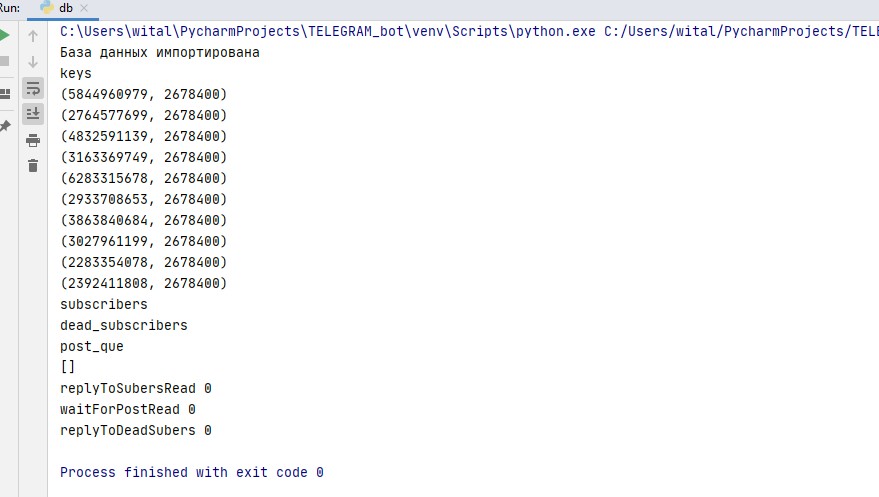
Помимо указанных в задании функций, бот имеет два типа аккаунтов – обычный, если вы хотите подписаться на уведомления и административный – вы можете создавать ключи для подписчиков на разный промежуток времени и получать их в текстовом формате, получать статистику, делать рассылку, делать отложенные сообщения в рассылке, банить обычные аккаунты, изменять время запланированных рассылок, удалять запланированные сообщения. Посредством бота администратор может общаться с подписчиками

Представлены скриншоты работы в случае, со стороны администратора

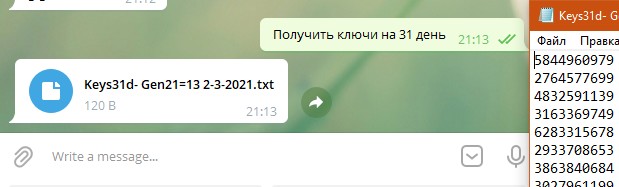
Лог программы при запуске:



Лог базы данных, когда её вызывают:



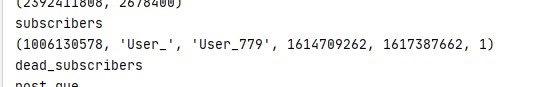
Запрос на ключи от аккаунта администратора:



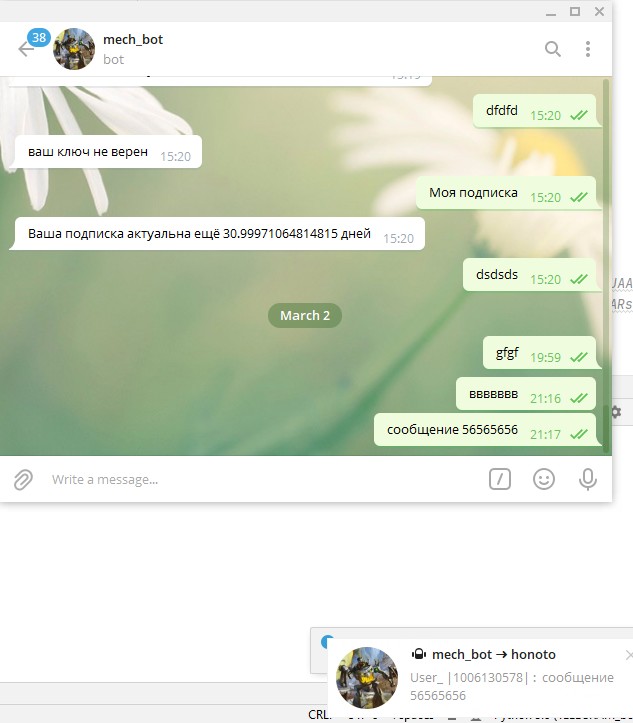
Оформление подписки:



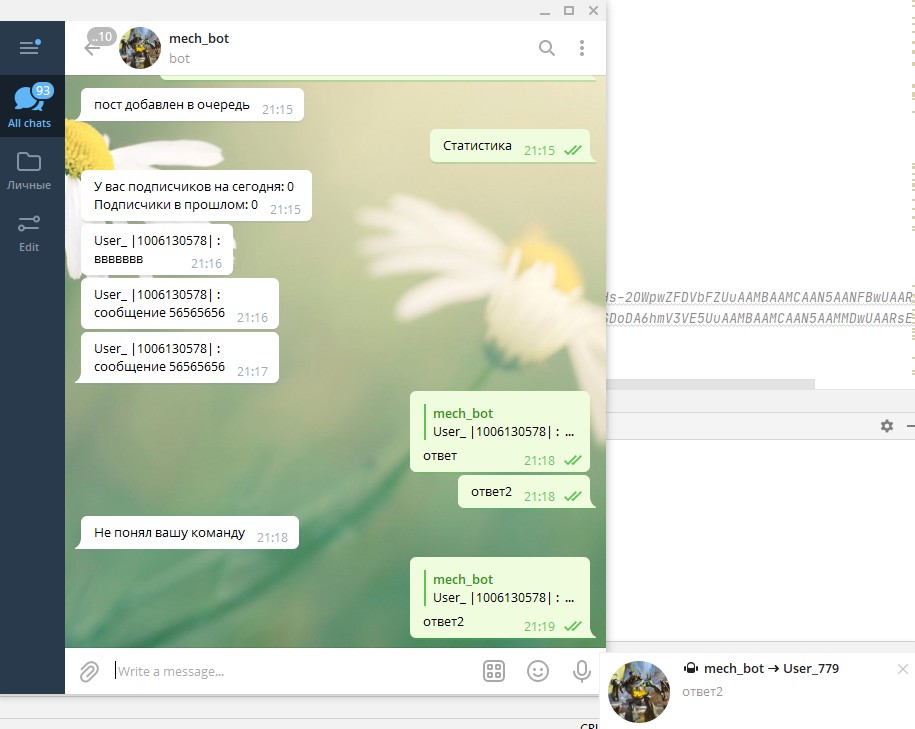
Подписчик в логе:



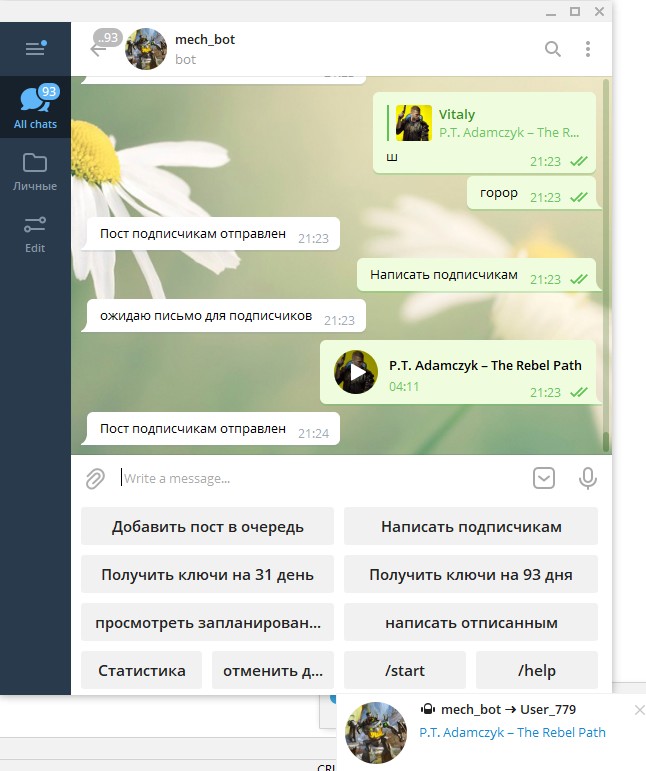
Отправка сообщения администратору через бота:



Ответ пользователю из аккаунта администратора:



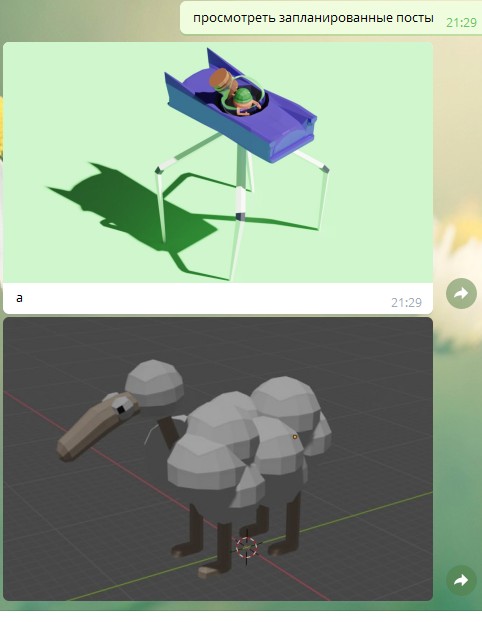
Рассылка сообщений подписчикам:



Добавление поста в отложенную рассылку:



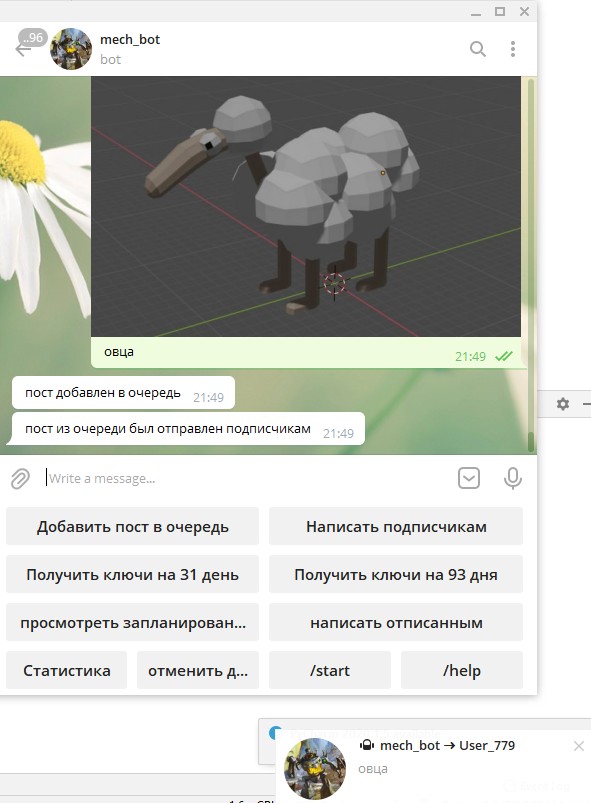
Проверка сохранения постов:

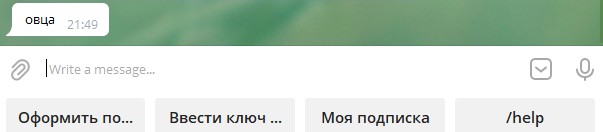


Удаление одного из них:

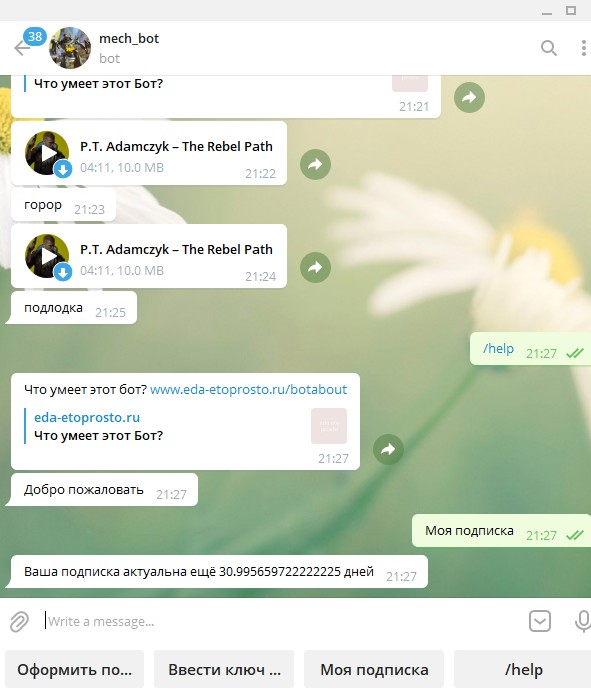


Рассылка запланированного сообщения пользователю:





Проверка подписчиком времени своей подписки:



Код программы, исполняющей работу основного бота

def runnn():  
 import time  
 import re  
 bot = telebot.TeleBot(constants.token)  
 listOfSubs = functions.getUsers()  
 listOfDeadSubs = functions.getDeadUsers()  
 KeyWaitUsers1= functions.waitUsersGet()  
 KeyWaitUsers =[]  
 waitForPost = functions.waitForPostRead()  
 replyToSubers = functions.replyToSubersRead()  
 replyToDeadSubers = functions.replyToDeadSubersRead()  
  
 for each in KeyWaitUsers1:*# это просто перевод значений из строк базы данных внужные* KeyWaitUsers.append(each[0])  
 if waitForPost == **"True"**:  
 waitForPost = True  
 else:  
 waitForPost = False  
 if replyToSubers == **"True"**:  
 replyToSubers = True  
 else:  
 replyToSubers = False  
 if replyToDeadSubers == **"True"**:  
 replyToDeadSubers = True  
 else:  
 replyToDeadSubers = False  
  
  
 def MakeValue(what, value):  
 nonlocal waitForPost  
 nonlocal replyToSubers  
 nonlocal replyToDeadSubers  
 if what == **"waitForPost"**:  
 waitForPost = value  
 if what == **"replyToSubers"**:  
 replyToSubers = value  
 if what == **"replyToDeadSubers"**:  
 replyToDeadSubers = value  
  
  
  
 def globaler():*# функция, которая решает проблему с недоступностью к переменных снаруж функции* return waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs  
  
  
 def updator(lisst):  
 file = open(**"raspisanie\_otpravok.txt"**, **"w"**)  
 file.write(str(lisst))  
 file.close()  
  
 def log(message, answer):  
 from datetime import datetime  
 print(datetime.now())  
 print(**"сообщение от {0}{1}.id = {2}**\n **Текст= {3}"**.format(message.from\_user.first\_name,  
 message.from\_user.last\_name, str(message.from\_user.id),  
 message.text))  
 print(answer)  
  
 @bot.message\_handler(commands=[**'start'**])  
 def handle\_start(message):  
 if int(message.chat.id) == int(constants.owner):  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, False)  
 user\_markup.row(**"Добавить пост в очередь"**, **"Написать подписчикам"**)  
 user\_markup.row(**"Получить ключи на 31 день"**, **"Получить ключи на 93 дня"**)  
 user\_markup.row(**"просмотреть запланированные посты"**, **"написать отписанным"**)  
 user\_markup.row(**"Статистика"**, **"отменить действие"**, **"/start"**, **"/help"**)  
 bot.send\_message(message.from\_user.id, **"Добро пожаловать"**, reply\_markup=user\_markup)  
 pass  
 else:  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, False)  
 user\_markup.row(**"Оформить подписку"**, **"Ввести ключ подписки"**, **"Моя подписка"**, **"/help"**)  
 bot.send\_message(message.from\_user.id, **"Добро пожаловать"**, reply\_markup=user\_markup)  
  
 @bot.message\_handler(commands=[**'stop'**])  
 def handle\_stop(message):  
 hide\_markup = telebot.types.ReplyKeyboardRemove  
 bot.send\_message(message.chat.id, **"Удачи!"**)  
  
 @bot.message\_handler(commands=[**'help'**])  
 def handle\_help(message):  
  
 if int(message.chat.id) != int(constants.owner):  
 bot.send\_message(message.chat.id, **"""Что умеет этот бот? www.eda-etoprosto.ru/botabout"""**)  
 user\_markup = telebot.types.ReplyKeyboardMarkup(True, False)  
 user\_markup.row(**"Оформить подписку"**, **"Ввести ключ подписки"**, **"Моя подписка"**, **"/help"**)  
 bot.send\_message(message.from\_user.id, **"Добро пожаловать"**, reply\_markup=user\_markup)  
 elif int(message.chat.id) == int(constants.owner):  
 tt = **"основные команды реализованы с помощью клавиатуры. если у вас нет клавиатуры, можете отправить /start**\n**"** \  
 **"дополнительные команды -** \n\n **"** \  
 **" 'Время' - Если вы захотите поменять расписание запланированных постов. отправьте 'Время' и нужные часы через запятую. Например 'Время 10,14,19'** \n \n **"** \  
 **"'Удалить' - если вам нужно удалить пост из очереди. Отправьте 'Удалить 1', если хотите удалить пост который идет в очереди первым.** \n**"** \  
 **"'Удалить 0', если вы хотите удалить последний пост. 'Удалить -1'-предпоследний и.т.д.**\n**"** \  
 **"для удалиения пользователя отправьте 'Бан 1006130578', где цифры это его айди. их можно найти в сообщении между плками | |**\n**"** bot.send\_message(message.from\_user.id, tt)  
  
 @bot.message\_handler(content\_types=[**"text"**])  
 def messages(message):  
 *# если шлет владелец бота* if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers,listOfDeadSubs = globaler()  
 def zatuchka (a, b):  
 MakeValue(a, b)  
 print(**"made"**, a, b)  
  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_message(idd, message.text)  
 *#блоки трай нужны чтоб выбрать человека которому слать сообщение  
 #функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text))*#если ответ админа на текстовое сообщение* no\_repeat(message,gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption))*#если ответ админа на фото* no\_repeat(message,gg)  
 except Exception as e:  
 print(e)  
 if message.reply\_to\_message == None: *# not reply* if message.text == **"отменить действие"**:  
 functions.waitForPostChange(False)  
 zatuchka(**"waitForPost"**, False)  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"отменено"**)  
 pass  
 elif waitForPost == True:  
 time\_ = time.time()  
 text = message.text  
 photos = None  
 audio = None  
 functions.add\_post(time\_, text, photos, audio)  
 functions.waitForPostChange(False)  
 zatuchka(**"waitForPost"**, False)  
 bot.send\_message(constants.owner, **"пост добавлен в очередь"**)  
  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_message(each[0], message.text)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_message(each[0], message.text)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
  
 elif message.text == **"Статистика"**:  
  
 ls =**"У вас подписчиков на сегодня: "**+str(len(listOfSubs))+**"**\n**"** ls = ls + **"Подписчики в прошлом: "** + str(len(listOfDeadSubs)) + **"**\n**"** + **"**\n**"** for each in listOfSubs:  
 t1 = str(each[1])  
 t2 = str(each[2])  
  
 t4 = int(each[4])-time.time()  
 t5 = str(t4/60/60/24)  
 ls=ls+**"Подписчик "**+t1+**" "**+t2+**" ещё "**+t5+**"дней** \n**"** pass  
 bot.send\_message(constants.owner, ls)  
 pass  
  
 elif message.text.startswith(**"Удалить"**):  
 try:  
 indexes = []  
 pattern = **r"\d+"** indexes = re.findall(pattern, message.text)  
 ok = True  
 for each in indexes:  
 functions.deler(int(each))  
 if ok == True:  
 bot.send\_message(constants.owner, **"готово"**)  
 except:  
 bot.send\_message(constants.owner, **"не получилось удалить пост"**)  
  
 elif message.text.startswith(**"Бан"**):  
  
 pattern = **r"\d+"** aidi = re.search(pattern, message.text)  
 print(aidi)  
 print(type(aidi))  
 try:  
 resul = functions.unsubscribeOOONE(int(aidi[0]))  
 if resul == True:  
 bot.send\_message(constants.owner, **"отписан"**)  
 else:  
  
 bot.send\_message(constants.owner, **"не получилось его удалить"**)  
 except:  
 print(**"smth got wrong with deling"**)  
 bot.send\_message(constants.owner, **"не получилось его удалить"**)  
  
  
  
  
 elif message.text.startswith(**"Время"**):  
 try:  
 times = []  
 pattern = **r"\d+"** times = re.findall(pattern, message.text)  
 ok = True  
 for each in times:  
 if int(each) < 0 or int(each) > 23:  
 print(**"значение "** + str(each) + **' не подходит'**)  
 bot.send\_message(constants.owner, **"значение "** + str(each) + **' не подходит'**)  
 ok = False  
 if ok == True:  
 updator(times)  
 print(**"Новое время "**, times)  
 bot.send\_message(constants.owner, **"Время отправки сообщений обновлено: "** + str(times))  
 except:  
 bot.send\_message(constants.owner, **"не получилось обновить часы"**)  
  
  
 elif message.text == **"просмотреть запланированные посты"**:  
 *# bot.send\_message(constants.owner, "Напишите одним сообщением ваш пост")* posts = functions.get\_posts()  
  
 for each in posts:  
  
 if each[2]!=None:*#для фото  
 # try:* bot.send\_photo(constants.owner,each[2], each[1])  
 *# except Exception as e:  
 # bot.send\_message(constants.owner, "Exception"+str(e))* elif each[2]==None:*#для текст :  
 # try:* bot.send\_message(constants.owner, each[1])  
 *# except Exception as e:  
 # bot.send\_message(constants.owner, "Exception"+str(e))* else:  
 bot.send\_message(constants.owner, **"что-то не получилось "**)  
 pass  
 elif message.text == **"Добавить пост в очередь"**:  
 bot.send\_message(constants.owner, **"Напишите одним сообщением ваш пост"**)  
 functions.waitForPostChange(True)  
 zatuchka(**"waitForPost"**, True)  
 pass  
 elif message.text == **"Написать подписчикам"**:  
 bot.send\_message(constants.owner, **"ожидаю письмо для подписчиков"**)  
 functions.replyToSubersChange(True)  
 zatuchka(**"replyToSubers"**, True)  
  
  
 elif message.text == **"написать отписанным"**:  
 bot.send\_message(constants.owner, **"ожидаю письмо для отписчиков"**)  
 functions.replyToDeadSubersChange(True)  
 zatuchka(**"replyToDeadSubers"**, True)  
  
 elif message.text == **"Получить ключи на 31 день"**:  
 file = functions.createKeys(31)  
 openFile = open(file, **"rb"**)  
 bot.send\_document(constants.owner, openFile)  
 openFile.close()  
 pass  
  
 elif message.text == **"Получить ключи на 93 дня"**:  
 file = functions.createKeys(93)  
 openFile = open(file, **"rb"**)  
 bot.send\_document(constants.owner, openFile)  
 openFile.close()  
 pass  
 elif message.text == **"Статистика"**:  
 bot.send\_message(constants.owner, **"пока не работает"**)  
 pass  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
 *#если шлет обычный юзер* else:  
 if message.text == **"Оформить подписку"**:  
 bot.send\_message(message.chat.id,  
 **'Отлично! Страница, на которой вы сможете оформить подписку** \n**www.eda-etoprosto.ru/bot'**)  
 elif message.text == **"Ввести ключ подписки"**:  
 functions.waitUsersadd(message.chat.id)  
 KeyWaitUsers.append(message.chat.id)  
 *# KeyWaitUsers = True* print(**'user appended'**)  
 bot.send\_message(message.chat.id, **'Ожидаю от вас ключ. Если вы передумаете слать ключ, напишите "нет"'**)  
  
 elif **"нет"** in message.text:  
 try:  
 functions.waitUsersremove(message.chat.id)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 except:  
 pass  
  
  
 elif message.text == **"Моя подписка"**:  
  
  
 listOfSubs = functions.getUsers()  
 actualnali = False  
 for each in listOfSubs:  
 if message.chat.id == each[0]:  
 *# print(each[4])  
 # print(time.time())  
 # print(each[4]-time.time())* t4 = int(each[4])-int(time.time())  
 t5 = str(t4 / 60 / 60 / 24)  
 *# print(t5)* bot.send\_message(message.chat.id,**'Ваша подписка актуальна ещё '**+ t5+**" дней"**)  
 actualnali = True  
 else:  
 bot.send\_message(message.chat.id, **'Ваша подписка не актуальна, однако вы можете её продлить. отправьте заново /start'**)  
 pass  
 if actualnali== False:  
 bot.send\_message(message.chat.id,**'Ваша подписка не актуальна, однако вы можете её продлить. отправьте заново /start'**)  
  
 elif message.chat.id in KeyWaitUsers:  
 try:  
 k = **"ваш ключ не верен"** try:  
 hisKey = (**r'{}'**.format(message.text))  
 pattern2 = **r'\d+'** k = re.search(pattern2, hisKey)[0]  
 repl = functions.activateKey(message.chat.id, message.chat.first\_name, message.chat.username, k)  
 functions.waitUsersremove(message.chat.id)  
 bot.send\_message(message.chat.id, repl)  
  
 except:  
 print(**"человек ввел в качестве ключа "**,message.text)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 *# bot.send\_message(message.chat.id, 'что-то не так. отправьте ключ ещё раз или "нет", если вы передумали')* pass  
  
 else:  
  
  
 bot.send\_message(constants.owner, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + message.text)  
  
 @bot.message\_handler(content\_types=[**"photo"**])  
 def photo(message):  
 *# если шлет владелец бота* if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers,listOfDeadSubs = globaler()  
 def zatuchka (a, b):  
 MakeValue(a, b)  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_photo(idd, message.photo[-1].file\_id, message.caption)  
 *#блоки трай нужны чтоб выбрать человека которому слать сообщение  
 #функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text))*#если ответ админа на текстовое сообщение* no\_repeat(message,gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption))*#если ответ админа на фото* no\_repeat(message,gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 time\_ = time.time()  
 text = message.caption  
 photos = message.photo[-1].file\_id  
 audio = None  
 functions.add\_post(time\_, text, photos, audio)  
 functions.waitForPostChange(False)  
 zatuchka(**"waitForPost"**, False)  
 bot.send\_message(constants.owner, **"пост добавлен в очередь"**)  
  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_photo(each[0], message.photo[-1].file\_id, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_photo(each[0], message.photo[-1].file\_id, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
  
  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
  
  
  
 *#если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 *# bot.send\_message(message.chat.id, 'что-то не так. отправьте ключ ещё раз или "нет", если вы передумали')* pass  
  
 else:  
 bot.send\_photo(constants.owner, message.photo[-1].file\_id, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + str(message.caption))  
  
  
  
  
  
  
  
  
 *# Handles all sent documents and audio files* @bot.message\_handler(content\_types=[**'voice'**])  
 def handle\_docs\_audio(message):  
 *# если шлет владелец бота* if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs = globaler()  
  
 def zatuchka(a, b):  
 MakeValue(a, b)  
  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_voice(idd, message.voice.file\_id, message.caption)  
  
 *# блоки трай нужны чтоб выбрать человека которому слать сообщение  
 # функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text)) *# если ответ админа на текстовое сообщение* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption)) *# если ответ админа на фото* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 bot.send\_message(constants.owner, **"отложенная отправка голосовых не поддерживается"**)  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_photo(each[0], message.voice.file\_id, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_photo(each[0], message.voice.file\_id, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
 *# если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 pass  
  
 else:  
  
 bot.send\_voice(constants.owner, message.voice.file\_id, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** )  
 pass  
  
  
  
 @bot.message\_handler(content\_types=[**'document'**])  
 def handle\_text\_doc(message):  
  
 *# если шлет владелец бота* if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs = globaler()  
 def zatuchka(a, b):  
 MakeValue(a, b)  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_document(idd, message.document.file\_id, message.caption)  
 *# блоки трай нужны чтоб выбрать человека которому слать сообщение  
 # функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text)) *# если ответ админа на текстовое сообщение* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption)) *# если ответ админа на фото* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 bot.send\_message(constants.owner, **"отложенная отправка документов сейчас не поддерживается"**)  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_document(each[0], message.document.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_document(each[0], message.document.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
 *# если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 pass  
 else:  
 bot.send\_document(constants.owner, message.document.file\_id,None, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + str(message.caption))  
  
 pass  
  
  
  
  
  
  
  
 @bot.message\_handler(content\_types=[**'sticker'**])  
 def handle\_text\_doc(message):  
  
  
 *# если шлет владелец бота* if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs = globaler()  
 def zatuchka(a, b):  
 MakeValue(a, b)  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_sticker(idd, message.sticker.file\_id, message.caption)  
 *# блоки трай нужны чтоб выбрать человека которому слать сообщение  
 # функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text)) *# если ответ админа на текстовое сообщение* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption)) *# если ответ админа на фото* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 bot.send\_message(constants.owner, **"отложенная отправка стикеров сейчас не поддерживается"**)  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_sticker(each[0], message.sticker.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_sticker(each[0], message.sticker.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
  
  
  
 *# если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 pass  
 else:  
 bot.send\_sticker(constants.owner,message.sticker.file\_id, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + str(message.caption))  
  
 pass  
  
  
  
  
 @bot.message\_handler(content\_types=[**'audio'**])  
 def handle\_text\_doc(message):  
  
  
 if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs = globaler()  
 def zatuchka(a, b):  
 MakeValue(a, b)  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_audio(idd, message.audio.file\_id, message.caption)  
 *# блоки трай нужны чтоб выбрать человека которому слать сообщение  
 # функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text)) *# если ответ админа на текстовое сообщение* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption)) *# если ответ админа на фото* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 bot.send\_message(constants.owner, **"отложенная отправка стикеров сейчас не поддерживается"**)  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_audio(each[0], message.audio.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_audio(each[0], message.audio.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
  
  
  
 *# если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 pass  
 else:  
 bot.send\_audio(constants.owner,message.audio.file\_id, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + str(message.caption))  
  
 pass  
  
 @bot.message\_handler(content\_types=[**'video'**])  
 def handle\_text\_doc(message):  
  
 if int(message.chat.id) == int(constants.owner):  
 waitForPost, replyToSubers, listOfSubs, replyToDeadSubers, listOfDeadSubs = globaler()  
 def zatuchka(a, b):  
 MakeValue(a, b)  
 if message.reply\_to\_message != None: *# reply* def no\_repeat(message, gg):  
 pattern2 = **r'(?<=\|)(\d+)(\|)'** idd = re.search(pattern2, gg)[0][0:-1]  
 bot.send\_video(idd, message.video.file\_id, message.caption)  
 *# блоки трай нужны чтоб выбрать человека которому слать сообщение  
 # функция ноу рипит непосредственно отправляет текст пользователю. она одна здесь, поскольку мы точно знаем, что пользователь слал текст* try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.text)) *# если ответ админа на текстовое сообщение* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
 try:  
 gg = (**r'{}'**.format(message.reply\_to\_message.caption)) *# если ответ админа на фото* no\_repeat(message, gg)  
 except Exception as e:  
 print(e)  
  
 if message.reply\_to\_message == None: *# not reply* if waitForPost == True:  
 bot.send\_message(constants.owner, **"отложенная отправка стикеров сейчас не поддерживается"**)  
 elif replyToSubers == True:  
 listOfSubs = functions.getUsers()  
 for each in listOfSubs:  
 try:  
 bot.send\_video(each[0], message.video.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToSubersChange(False)  
 zatuchka(**"replyToSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост подписчикам отправлен"**)  
  
 elif replyToDeadSubers == True:  
 listOfDeadSubs = functions.getDeadUsers()  
 for each in listOfDeadSubs:  
 try:  
 bot.send\_video(each[0], message.video.file\_id,None, message.caption)  
 print(each[0])  
 except:  
 continue  
 functions.replyToDeadSubersChange(False)  
 zatuchka(**"replyToDeadSubers"**, False)  
 bot.send\_message(constants.owner, **"Пост отписчикам отправлен"**)  
 else:  
 bot.send\_message(constants.owner, **"Не понял вашу команду"**)  
  
  
  
 *# если шлет обычный юзер* else:  
 if message.chat.id in KeyWaitUsers:  
 try:  
 k = **"я ожидаю от вас ключ"** print(**"человек отправил фото вместо ключа"**)  
 KeyWaitUsers.pop(KeyWaitUsers.index(message.chat.id))  
 bot.send\_message(message.chat.id, k)  
 pass  
 except:  
 pass  
 pass  
 else:  
 bot.send\_video(constants.owner,message.video.file\_id, str(message.chat.first\_name) + **' |'** + str(message.chat.id) + **'| '** + **':** \n**'** + str(message.caption))  
  
 pass  
  
 bot.polling(none\_stop=False, interval=3, timeout=20)  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# listOfSubs = functions.getUsers()  
 # waitForPost = False  
 # replyToSubers = False  
  
 # import urllib.request  
 # import urllib.request  
 # from bs4 import BeautifulSoup* import re  
 *# import os* import time  
 *# from threading import Thread  
 # from multiprocessing import Process  
 # import ast* import telebot  
 import constants  
 import functions  
  
 *# runnn()* while True:  
 try:  
 runnn()  
 time.sleep(3)  
 except:  
 print(**'Main file exception'**)  
 time.sleep(7)

Код программы, осуществляющей рассылку:

import time  
import ast  
import os  
  
  
def updator(lisst):  
 file = open(**"raspisanie\_otpravok.txt"**, **"w"**)  
 file.write(str(lisst))  
 file.close()  
  
  
*# процесс с ботом телеги. тут осуществляется проверка на наличие ошибки соединения, так как она ловится только тут. в main она за ошибку не считается.*def sender(): *# в этой функции мы непосредственно коннектимся к тг и бд. делаем попытку отправить пост, пока не удастся* import telebot  
 import constants  
 import time  
 import functions  
 thePost = None  
  
 listOfSubs1 = functions.getUsers()  
 bot = telebot.TeleBot(constants.token)  
  
 for each in listOfSubs1:  
 try:  
 if each[4] - each[3] < 259200:  
 bot.send\_message(each[0], **"ваша подписка закончится в течении трех дней"**)  
 else:  
 *# print("ok")* pass  
 except:  
 print(**"проблема с отправкой уведомления о заканчивающейся подписке юзеру"**)  
 pass  
  
 done\_flag = False  
 while done\_flag == False:  
 try:  
 thePost = functions.poster()  
 done\_flag = True  
 except:  
 time.sleep(1)  
  
 def sendddd(each, thePost):  
 retry\_limit = 2  
 done\_flag2 = False  
  
 if thePost[2] != None or thePost[2] != **"None"**: *# отправка текстового поста* while done\_flag2 == False:  
 try:  
 bot.send\_message(each[0], thePost[1])  
 done\_flag2 = True  
 print(**"отпралено сообщение пользователю "**, each[0], each[1])  
 except Exception as e:  
 *# print("each",each)* retry\_limit-=1  
 if retry\_limit < 0:  
 print(**'отправка сообщеня пользователю'**, each[0], each[1], **'не удалась.'**)  
 return True  
 break  
 print(**'проблема с отправкой сообщения '**, each, **"**\n**"**, e)  
 time.sleep(1)  
 return True  
 else:  
 while done\_flag2 == False:  
 try:  
 print(thePost[2])  
 print(type(thePost[2]))  
 print(thePost[3])  
 bot.send\_photo(each[0], thePost[2], thePost[1])  
 done\_flag2 = True  
 except Exception as e:  
 *# print("each",each)* retry\_limit -= 1  
 if retry\_limit < 0:  
 print(**'отправка сообщеня пользователю'**, each[0], each[1], **'не удалась.'**)  
 return True  
 break  
 print(**'проблема с отправкой сообщения '**, each, **"**\n**"**, e)  
 time.sleep(1)  
 return True  
  
 if thePost == None:  
 print(**"no post to post"**)  
 bot.send\_message(constants.owner, **"нет поста на этот час для отправки подписчикам"**)  
 pass  
 else:  
 for each in listOfSubs1:  
 *# print("\_\_\_\_\_\_")  
 # print(type(each))  
 # print(each)* try:  
 sendddd(each, thePost)  
 except Exception as e:  
 print(**"sending file exception "**)  
 print(each)  
 print(e)  
 bot.send\_message(constants.owner, **"пост из очереди был отправлен подписчикам"**)  
  
  
*# жутко намудрил, но он точно проверяет время отправки и наличие файла с часами отправки дополнительно*def process\_creator(next\_post, lisst): *# в этой функции мы делаем проверку на то, сейчас время или не время делать пост* from time import sleep  
 global listOfSubs  
 lisst = lisst *# может это не правильно, но для надежности закрепил наличие переменных* next\_post = next\_post  
 start\_time = time.time()  
  
 file = open(**"vrema\_rassylky.txt"**, **"r"**)  
 hh = file.readline()  
 hh2 = int(file.readline())  
 file.close()  
 file = open(**"vrema\_rassylky.txt"**, **"w"**)  
 file.write(str(**"False"** + **"**\n**"** + str(int(hh2))))  
 file.close()  
  
 def timer(next\_time1, lisst):  
 raspis = None *# время постов. дублирует лисст, если он есть* booler = False *# пора делать пост, или нет* tim = time.localtime()  
 hour = tim.tm\_hour *# сейчас час* next\_hour = None *# следующий час из списка* rer = False  
  
 if os.path.exists(**"raspisanie\_otpravok.txt"**) == False:  
 tt = [9, 14, 20]  
 file = open(**"raspisanie\_otpravok.txt"**, **"w"**)  
 file.write(str(tt))  
 file.close()  
 raspis = tt  
 rer = True  
 if os.path.exists(**"raspisanie\_otpravok.txt"**) == True:  
 file = open(**"raspisanie\_otpravok.txt"**, **"r"**)  
 LList = ast.literal\_eval(file.read())  
 print(**"расписание отправок: "**,LList)  
 raspis = LList  
 rer = True  
  
 if rer == False:  
 raspis = lisst  
  
 for index in range(len(raspis)):  
 print(index, raspis[index])  
 raspis[index] = int(raspis[index])  
  
 def next\_time(raspis, hour):  
 if hour in raspis:  
 next\_hour = raspis.index(hour)  
 try:  
 return raspis[next\_hour + 1]  
 except:  
 return raspis[0]  
 for each in range(24):  
 if each < hour:  
 continue  
 if each in raspis:  
 return each  
 if each == 23:  
 return raspis[0]  
 if each == 24:  
 return raspis[0]  
  
 next\_hour = next\_time(raspis, hour)  
 print(raspis[0])  
 print(type(raspis[0]))  
  
 if hour in raspis:  
 booler = True  
 pass  
 print(**"функция таймер вернула "**,raspis, booler, next\_hour, hour)  
 return raspis, booler, next\_hour, hour  
  
 while True:  
  
 raspis1, booler1, next\_hour1, now = timer(next\_post, lisst)  
 lisst = raspis1  
 if booler1 == True: *# если сейчас время для поста* if next\_post == now or next\_post == None: *# если в этот час ещё не было поста или постов не было вообще  
 # post = functions.poster()#вытащили пост  
 # BOT! DO POST* sender()  
  
 *# pr = Process(target=sender, args=())  
 # pr.start()  
 # pr.join()  
 # pr.terminate()  
  
  
  
 # time.sleep(35)* import functions  
 functions.usubscribeR()  
 print(**"\_\_\_\_\_\_\_\_\_post made"**)  
  
 *#  
 # print(start\_time)* end\_time=int(time.time())  
  
 *# print(end\_time)* end\_time = end\_time - start\_time  
 *# print(end\_time)* file = open(**"vrema\_rassylky.txt"**, **"r"**)  
 hh = file.readline()  
 hh2 = int(file.readline())  
 print(**"hhhhhhhh in timer"**,hh,hh2)  
 file.close()  
 if end\_time>int(hh2):  
 file = open(**"vrema\_rassylky.txt"**, **"w"**)  
 file.write(str(**"True"** + **"**\n**"** + str(int(end\_time) +2)))  
 file.close()  
 else:  
 file = open(**"vrema\_rassylky.txt"**, **"w"**)  
 file.write(str(**"True"** + **"**\n**"** + str(int(hh2))))  
 file.close()  
 pass  
  
  
  
 else:  
 print(**"time for post, but no"**)  
 print(**"now"**, now)  
 print(**"next\_post"**, next\_post)  
 next\_post = next\_hour1 *# переключили флажок* sleep(3)  
  
  
def runn():  
 next\_post = None  
 lisst = [9, 15, 20] *# None* process\_creator(next\_post, lisst)  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 while True:  
 try:  
 runn()  
 except:  
 print(**'Main file exception'**)  
 time.sleep(42)

Работа с базой данных:

import sqlite3  
import random  
import time  
  
  
def waitUsersGet():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 users = []  
 for each in myCursor.execute(**f"SELECT \* FROM KeyWaitUsers"**):  
 *# print(each[0])  
 # print(each)* users.append(each)  
 *# print("losts",lostS)* myCursor.close()  
 con.close()  
 return users  
  
  
  
def waitUsersadd(id):  
 print(**"waitUsersadd"**,id)  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"SELECT id\_ FROM KeyWaitUsers WHERE id\_ ='**{id}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO KeyWaitUsers VALUES (?)"**, (id,))  
 con.commit()  
 else:  
 print(**"уже в списке ожидающих"**)  
 *# for each in myCursor.execute("SELECT \* FROM subscribers"):  
 # print(each)* myCursor.close()  
 con.close()  
 pass  
  
  
def waitUsersremove(id):  
 print(**"waitUsersremove"**,id)  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"DELETE FROM KeyWaitUsers WHERE id\_ ='**{id}**' "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 pass  
  
def waitForPostRead():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT booler FROM waitForPost Where id\_ = 0 """**)  
 waiters = myCursor.fetchone()  
 myCursor.close()  
 con.close()  
 return waiters[0]  
 pass  
  
def waitForPostChange(booler):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"UPDATE waitForPost SET booler ='**{booler}**' WHERE id\_ =0 "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 pass  
def replyToSubersRead():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT booler FROM replyToSubers Where id\_ = 0 """**)  
 waiters = myCursor.fetchone()  
 myCursor.close()  
 con.close()  
 return waiters[0]  
 pass  
def replyToSubersChange(booler):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"UPDATE replyToSubers SET booler ='**{booler}**' WHERE id\_ =0 "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 pass  
  
  
  
def replyToDeadSubersRead():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT booler FROM replyToDeadSubers Where id\_ = 0 """**)  
 waiters = myCursor.fetchone()  
 myCursor.close()  
 con.close()  
 return waiters[0]  
 pass  
  
def replyToDeadSubersChange(booler):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"UPDATE replyToDeadSubers SET booler ='**{booler}**' WHERE id\_ =0 "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 pass  
  
  
  
  
  
def add\_sub(id, first\_name,username,start,end,relevance):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
 myCursor.execute(**f"SELECT id FROM subscribers WHERE id ='**{id}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO subscribers VALUES(?,?,?,?,?,?)"**, (id, first\_name, username, start, end, relevance))  
 print(**"подписчик "**, first\_name, **" "**, username, **" добавлен в список подписчиков"**)  
 myCursor.execute(**f"DELETE FROM dead\_subscribers WHERE id ='**{id}**' "**)  
 con.commit()  
 else:  
 print(**"подписка ещё активна"**)  
  
 myCursor.close()  
 con.close()  
  
  
  
def unsubscribeONE(id):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
  
  
 user = None  
 for each in myCursor.execute(**f"SELECT \* FROM subscribers WHERE id ='**{id}**'"**):  
 user=each  
 myCursor.execute(**f"SELECT id FROM dead\_subscribers WHERE id ='**{id}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO dead\_subscribers VALUES(?,?,?,?,?,?)"**, (user))  
 myCursor.execute(**f"DELETE FROM subscribers WHERE id ='**{id}**' "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
  
def unsubscribeREGULAR(now):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
  
 lostS = []  
 for each in myCursor.execute(**f"SELECT \* FROM subscribers WHERE end <='**{now}**'"**):  
 *# print(each[0])  
 # print(each)* lostS.append(each)  
 for rrr in lostS:  
 myCursor.execute(**f"INSERT INTO dead\_subscribers VALUES(?,?,?,?,?,?)"**, (rrr))  
 myCursor.execute(**f"DELETE FROM subscribers WHERE id ='**{rrr[0]}**' "**)  
 con.commit()  
 *# print("losts",lostS)* con.commit()  
 myCursor.close()  
 con.close()  
 return lostS  
  
  
  
def CreateKey(days):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
  
 *# with sql.connect("database.db") as con:  
 # myCursor = con.cursor()* def get\_random\_key():  
 def r():  
 return random.randint(42, 4242)  
 return r() \* r() + r()  
 new\_key = get\_random\_key()  
 myCursor.execute(**f"SELECT \* FROM keys WHERE key ='**{new\_key}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO keys VALUES(?,?)"**, (new\_key, days\*24\*60\*60))*#2678400 = 1 month* con.commit()  
 *# print("ключ",new\_key," добавлен")* pass  
 else:  
 print(**"такой ключ уже есть"**)  
  
 *# myCursor.close()  
 # con.close()* myCursor.close()  
 con.close()  
 return new\_key  
 pass  
  
  
  
  
def CreateKeys(days,howMany):  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 keyList = []  
  
 def get\_random\_key():  
 return random.randint(42, 4242424242) + random.randint(42, 4242424242)  
  
 for each in range(howMany):  
 new\_key = get\_random\_key()  
 myCursor.execute(**f"SELECT \* FROM keys WHERE key ='**{new\_key}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO keys VALUES(?,?)"**, (new\_key, days\*24\*60\*60))*#2678400 = 1 month* keyList.append(new\_key)  
 *# print("ключ",new\_key," добавлен")* pass  
 else:  
 print(**"такой ключ уже есть"**)  
  
 con.commit()  
 myCursor.close()  
 con.close()  
 return keyList  
 pass  
  
  
def ActivateKey(id, first\_name,username,start, key):*#этот модуль надо будет жестко тестить* con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"SELECT time FROM keys WHERE key ='**{key}**'"**)  
 tt = myCursor.fetchone()[0]  
 print(**"tt"**,tt)  
 end = start+tt  
 his\_time = tt/60/60/24  
 print(end)  
 relevance = True  
 myCursor.execute(**f"SELECT \* FROM subscribers WHERE id ='**{id}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"DELETE FROM keys WHERE key ='**{key}**' "**)  
  
  
 myCursor.execute(**f"SELECT id FROM subscribers WHERE id ='**{id}**'"**)  
 if myCursor.fetchone() is None:  
 myCursor.execute(**f"INSERT INTO subscribers VALUES(?,?,?,?,?,?)"**,  
 (id, first\_name, username, start, end, relevance))  
 print(**"подписчик "**, first\_name, **" "**, username, **" добавлен в список подписчиков"**)  
 myCursor.execute(**f"DELETE FROM dead\_subscribers WHERE id ='**{id}**' "**)  
 con.commit()  
  
 else:  
  
 print(**"подписка ещё активна"**)  
  
  
  
 con.commit()  
 else:  
 print(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_else"**)  
 myCursor.execute(**f"SELECT end FROM subscribers WHERE id ='**{id}**'"**)  
 user\_end = myCursor.fetchone()[0]  
  
 print(**"user\_end"**,user\_end)  
  
 *# print("key time ", tt)* new\_end = int(user\_end)+int(tt)  
 his\_time = (new\_end - start)/60/60/24  
 print(**"new\_end"**, new\_end)  
 myCursor.execute(**f"UPDATE subscribers SET end ='**{new\_end}**' WHERE id ='**{id}**'"**)  
  
 myCursor.execute(**f"DELETE FROM keys WHERE key ='**{key}**' "**)  
 con.commit()  
 ss = **'Спасибо, ваш ключ активирован, подписка закончится через '**+ str(his\_time)+**" день"** print(ss)  
 myCursor.close()  
 con.close()  
 return ss  
 pass  
  
  
  
  
  
def getAllSubs():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
 myCursor.execute(**"""SELECT \* from subscribers"""**)  
 items = myCursor.fetchall()  
 myCursor.close()  
 con.close()  
 return items  
 pass  
  
def getAllDEADSubs():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
 myCursor.execute(**"""SELECT \* from dead\_subscribers"""**)  
 items = myCursor.fetchall()  
  
 myCursor.close()  
 con.close()  
 return items  
 pass  
  
  
  
def printall():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
  
 print(**"keys"**)  
 for each in myCursor.execute(**"SELECT \* FROM keys"**):  
 print(each)  
 print(**"subscribers"**)  
 for each in myCursor.execute(**"SELECT \* FROM subscribers"**):  
 print(each)  
 print(**"dead\_subscribers"**)  
 for each in myCursor.execute(**"SELECT \* FROM dead\_subscribers"**):  
 print(each)  
 print(**"post\_que"**)  
 for each in myCursor.execute(**"SELECT \* FROM post\_que"**):  
 print(each)  
 print(waitUsersGet())  
 print(**"replyToSubersRead"**, replyToSubersRead())  
 print(**"waitForPostRead"**, waitForPostRead())  
 print(**"replyToDeadSubers"**, replyToDeadSubersRead())  
  
 myCursor.close()  
 con.close()  
  
  
def getAllPosts():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT \* from post\_que ORDER BY time"""**)  
 posts = myCursor.fetchall()  
 myCursor.close()  
 con.close()  
 *# print(posts)* return posts  
  
def addPost(time\_, text\_,photos,audio):  
 time\_ = int(time\_)  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**f"SELECT \* FROM post\_que WHERE time ='**{time\_}**'"**)  
 if myCursor.fetchone() is None:  
 print(**"myCursor.fetchone()---------------------"**,myCursor.fetchone())  
 myCursor.execute(**f"INSERT INTO post\_que VALUES(?,?,?,?)"**, (time\_, text\_,photos,audio,))  
 con.commit()  
 print(**"пост добавлен**\n**"**)  
 pass  
 else:  
 print(**"такой пост уже есть"**)  
 *# for each in myCursor.execute("SELECT \* FROM post\_que"):  
 # print(each)* myCursor.close()  
 con.close()  
  
def del\_post (number):  
 try:  
 number = int(number)  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT \* FROM post\_que ORDER BY time"""**)  
 post = myCursor.fetchall()  
 print(post)  
 print(**"-----"**)  
 number = int(post[number][0])  
 print(number)  
 myCursor.execute(**f"SELECT \* FROM post\_que WHERE time ='**{number}**'"**)  
 post1 = myCursor.fetchone()  
 myCursor.execute(**f"DELETE FROM post\_que WHERE time ='**{number}**' "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 return post1  
 except:  
 return **"не смог удалить пост. возможно вы указали номер поста, которого нет. Либо список постов пуст"**def DeleteAndGetPost():  
 con = sqlite3.connect(**'teleBot.db'**)  
 myCursor = con.cursor()  
 myCursor.execute(**"""SELECT MIN(time) from post\_que"""**)  
 post = int(myCursor.fetchone()[0])  
 *# print(type(post[0]))* myCursor.execute(**f"SELECT \* FROM post\_que WHERE time ='**{post}**'"**)  
 post1 = myCursor.fetchone()  
 myCursor.execute(**f"DELETE FROM post\_que WHERE time ='**{post}**' "**)  
 con.commit()  
 myCursor.close()  
 con.close()  
 return post1  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
  
  
 *# id = 345373872* id = 1006130578  
 first\_name = **"User\_"** username = **"User\_779"** start = 1600178518  
 end = 1701478518  
 relevance = True  
 time = 1601478519 *# 1601478519* post1 = **"Hi There"** post2 = **"post2"** print(**"База данных импортирована"**)  
 con = sqlite3.connect(**'teleBot.db'**) *# connect us to the database and will let us execute the SQL statements* myCursor = con.cursor() *# cursor is a method of the connection object* myCursor.execute(**"""CREATE TABLE IF NOT EXISTS subscribers (  
 id INTEGER,  
 first\_name TEXT,  
 username TEXT,  
 start INTEGER,  
 end INTEGER,  
 relevance INTEGER  
 )"""**)  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS dead\_subscribers (  
 id INTEGER,  
 first\_name TEXT,  
 username TEXT,  
 start INTEGER,  
 end INTEGER,  
 relevance INTEGER  
 )"""**)  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS post\_que (  
 time INTEGER,  
 text\_ TEXT,  
 photos TEXT,  
 audio TEXT  
 )"""**)  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS keys (  
 key INTEGER,  
 time INTEGER  
 )"""**)  
  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS KeyWaitUsers (  
 id\_ INTEGER  
 )"""**)  
  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS waitForPost (  
 id\_ INTEGER,  
 booler INTEGER  
 )"""**)  
  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS replyToSubers (  
 id\_ INTEGER,  
 booler INTEGER  
 )"""**)  
  
 myCursor.execute(**"""CREATE TABLE IF NOT EXISTS replyToDeadSubers (  
 id\_ INTEGER,  
 booler INTEGER  
 )"""**)  
  
 *# ЭТОТ КОД ЗАПУСТИТЬ ТОЛЬКО ПРИ СОЗДАНИИ БАЗЫ ДАННЫХ!!!!!!  
  
 # myCursor.execute(f"INSERT INTO waitForPost VALUES (?,?)", (0,0))  
 # myCursor.execute(f"INSERT INTO replyToSubers VALUES (?,?)", (0,0))  
 # myCursor.execute(f"INSERT INTO replyToDeadSubers VALUES (?,?)", (0,0))* con.commit()  
 myCursor.close()  
 con.close()  
printall()

Код промежуточного звена, между ботами и базой данных:

import time  
def activateKey(id, first\_name,username,k):  
 not\_done\_flag = False  
 retry\_limit = 3  
 while not\_done\_flag == False:  
 try:  
 import db  
 start = int(time.time())  
 bdreply = db.ActivateKey(id, first\_name, username, start, k)  
 bdreply = bdreply + **"**\n\n**Что умеет этот бот? www.eda-etoprosto.ru/botabout"** db.printall()  
  
 print(**"подписчик добавлен"**)  
 not\_done\_flag = True  
 return bdreply  
 except:  
 print(**'ключ подписчика неверный. ещё попытка'**)  
 if retry\_limit<0:  
 return(**"ваш ключ не верен"**)  
 break  
 retry\_limit-=1  
 time.sleep(1)

def createKey(daes):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 print(**"stuff done"**)  
 k = db.CreateKey(daes)  
 not\_done\_flag = True  
  
 return k  
 except:  
 print(**'stuff not done. Waiting1'**)  
 time.sleep(1)  
  
def get\_posts():  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 print(**"stuff done"**)  
 k = db.getAllPosts()  
 not\_done\_flag = True  
 return k  
 except:  
 print(**'stuff not done. Waiting2'**)  
 time.sleep(1)  
  
  
def createKeys(daes):  
  
  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 k = db.CreateKeys(daes, 10)  
name = **"Keys{}d- Gen{}={} {}-{}-{}.txt"**.format(daes, time.localtime().tm\_hour, time.localtime().tm\_min,  
 time.localtime().tm\_mday, time.localtime().tm\_mon,  
 time.localtime().tm\_year)  
 file = open(name, **"w"**)  
 for each in k:  
 file.write(str(each))  
 file.write(**'**\n**'**)  
 file.close()  
  
 print(**"stuff done"**)  
 not\_done\_flag = True  
  
 return name  
 except:  
 print(**'stuff not done. Waiting3'**)  
 time.sleep(1)  
def unsubscribeOOONE(id):  
 done = False  
 try:  
 import db  
 db.unsubscribeONE(id)  
 done = True  
 except:  
 print(**"deleting got bad"**)  
 return done  
  
  
  
def usubscribeR():  
  
 try:  
 import db  
 unsubscribed = db.unsubscribeREGULAR(int(time.time()))  
 print(**"удалиение старых подписчиков сделано"**)  
  
 return unsubscribed  
 except:  
 print(**'удалиение старых подписчиков НЕ СДЕЛАНО'**)  
def Remind():  
 pass  
  
def deler(digit):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 aa = db.del\_post(digit-1)  
 *# print("got post: ")* not\_done\_flag = True  
 return aa  
 except Exception as e:  
 print(**'could not get post from db'**)  
 print(e)  
 time.sleep(1)  
  
 pass  
  
*# print(deler(1))*def poster():  
 not\_done\_flag = False  
 retry\_limit = 4  
 while not\_done\_flag == False:  
 try:  
 import db  
 aa = db.DeleteAndGetPost()  
 *# print("got post: ")* not\_done\_flag = True  
 return aa  
 except:  
 print(**'could not get post from db'**)  
 retry\_limit -= 1  
 if retry\_limit < 0:  
 print(**'Не получилось взять пост из бд. отбой'**)  
 return None  
 break  
 time.sleep(1)  
  
 pass  
  
def add\_post(time\_, text\_,photos,audio):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.addPost(time\_, text\_, photos, audio)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting4'**)  
 time.sleep(1)  
 pass  
  
  
  
def getUsers():  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 aaa = db.getAllSubs()  
  
 print(**"stuff done"**)  
 not\_done\_flag = True  
  
 return aaa  
 except:  
 print(**'stuff not done. Waiting5'**)  
 time.sleep(1)  
  
def waitUsersGet():  
 users = None  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 users = db.waitUsersGet()  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting6'**)  
 time.sleep(1)  
  
 return users  
def waitUsersadd(id):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.waitUsersadd(id)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting7'**)  
 time.sleep(1)  
  
 pass  
def waitUsersremove(id):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.waitUsersremove(id)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting8'**)  
 time.sleep(1)  
  
 pass  
  
def waitForPostRead():  
 not\_done\_flag = False  
 waiters = None  
 while not\_done\_flag == False:  
 try:  
 import db  
 waiters = db.waitForPostRead()  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting9'**)  
 time.sleep(1)  
  
 return waiters  
 pass  
def waitForPostChange(booler):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.waitForPostChange(booler)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting10'**)  
 time.sleep(1)  
  
 pass  
def replyToSubersRead():  
 not\_done\_flag = False  
 waiters = None  
 while not\_done\_flag == False:  
 try:  
 import db  
 waiters = db.replyToSubersRead()  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting11'**)  
 time.sleep(1)  
  
 return waiters  
 pass  
def replyToSubersChange(booler):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.replyToSubersChange(booler)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting12'**)  
 time.sleep(1)  
  
 pass  
  
  
  
def replyToDeadSubersRead():  
 not\_done\_flag = False  
 waiters = None  
 while not\_done\_flag == False:  
 try:  
 import db  
 waiters = db.replyToDeadSubersRead()  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting13'**)  
 time.sleep(1)  
  
 return waiters  
 pass  
def replyToDeadSubersChange(booler):  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 db.replyToDeadSubersChange(booler)  
 print(**"stuff done"**)  
 not\_done\_flag = True  
 except:  
 print(**'stuff not done. Waiting14'**)  
 time.sleep(1)  
  
 pass  
  
  
def getDeadUsers():  
 not\_done\_flag = False  
 while not\_done\_flag == False:  
 try:  
 import db  
 aaa = db.getAllDEADSubs()  
  
 print(**"stuff done"**)  
 not\_done\_flag = True  
  
 return aaa  
 except:  
 print(**'stuff not done. Waiting15'**)  
 time.sleep(1)  
  
Программа, управляющая рассылкой и обычным ботом:

import asyncio  
import re  
import time  
import telebot  
import time  
import ast  
import os  
import constants  
import telebot  
import constants  
import time  
from multiprocessing import Process  
import subprocess  
import os  
import asyncio  
  
def get\_rasp():  
 if os.path.exists(**"raspisanie\_otpravok.txt"**) == False:  
 vr = [10,14,20]  
 file = open(**"raspisanie\_otpravok.txt"**, **"w"**)  
 file.write(str(vr))  
 file.close()  
 file = open(**"raspisanie\_otpravok.txt"**, **"r"**)  
 file1 = file.read()  
 LList = ast.literal\_eval(file1)  
 file.close()  
 return LList  
  
  
  
def vrema\_rassylky():  
 *# tt = time.localtime().tm\_hour* if os.path.exists(**"vrema\_rassylky.txt"**) == False:  
 vr = **"False"**+**"**\n**"**+str(10)  
 file = open(**"vrema\_rassylky.txt"**, **"w"**)  
 file.write(str(vr))  
 file.close()  
 file = open(**"vrema\_rassylky.txt"**, **"r"**)  
 hh = file.readline()  
 if hh==**"False**\n**"**:  
 hh=False  
 else:  
 hh=True  
 hh2 = int(file.readline())  
 *# print(hh2)* file.close()  
 print(**"hhhhhhhhhhhhhh"**,hh,hh2)  
 return hh,hh2  
  
  
  
if \_\_name\_\_ ==**'\_\_main\_\_'**:  
 prev\_time = 0  
  
 while True:  
 LList = get\_rasp()  
 updated, secs = vrema\_rassylky()  
 prev\_time = time.localtime().tm\_hour  
 print(type(time.localtime().tm\_hour))  
  
  
 if str(time.localtime().tm\_hour) in LList:  
 print(**"делаем рассылку"**)  
 process = subprocess.Popen(**"python timer.py"**)  
  
 if updated == False:  
 print(**"not enough time"**)  
  
 file = open(**"vrema\_rassylky.txt"**, **"w"**)  
 file.write(str(**"False"** + **"**\n**"** + str(secs + 5)))  
 file.close()  
 time.sleep(secs + 10)  
 else:  
 time.sleep(secs + 2)  
 process.terminate()  
 print(**"рассылка окончена. возвращаемся к работе бота. "**)  
  
  
 process2 = subprocess.Popen(**"python mainer.py"**)  
 while prev\_time == time.localtime().tm\_hour:  
 time.sleep(1000)  
  
  
 if str(time.localtime().tm\_hour) in LList:  
 process2.terminate()  
 else:  
 pass  
  
 print(**"работа обычного бота приостановлена для рассылки"**)