Документация Телеграм-бота «СПС Автобот»

Колузанов Филипп Эдуардович

30.08.2024

Документация «СПС Автобот»

Описание

Скрипт доступен по ссылке: https://github.com/KoluzanovFE/sps_telebot

Скрипт Telegram-бота, который взаимодействует с базой данных SQLite для массовой рассылки сообщений в инициализированные чаты.

Внутри группового чата администратор инициализирует бота путем его добавления в беседу и ввода команды \start.

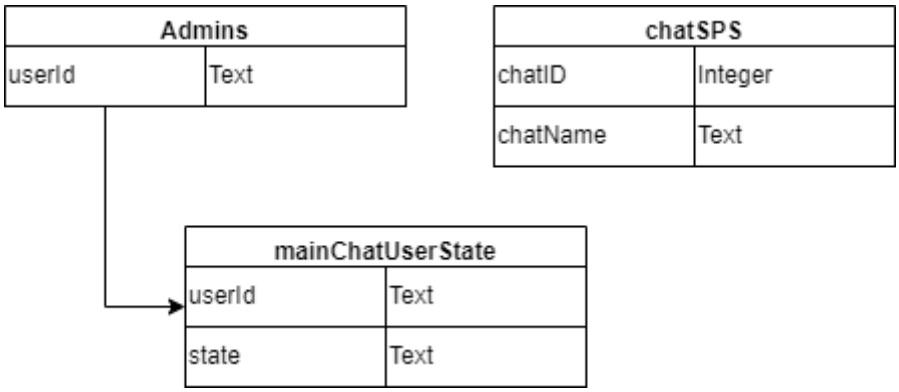
Панель администратора (приватные сообщения с ботом) позволяет с помощью команды:

- \send_to_all - ввести сообщения для отправки во все инициализированные группы.
- \group_list - посмотреть список инициализированных групп.
- \help – вывести справку.

Набор файлов

Файл	**Описание**
main.py	Основной скрипт бота
config.py	Файл с токеном бота
AutoSendBot.db	БД бота

Описание БД



Admins – таблица администраторов

- userId – идентификатор пользователя Telegram

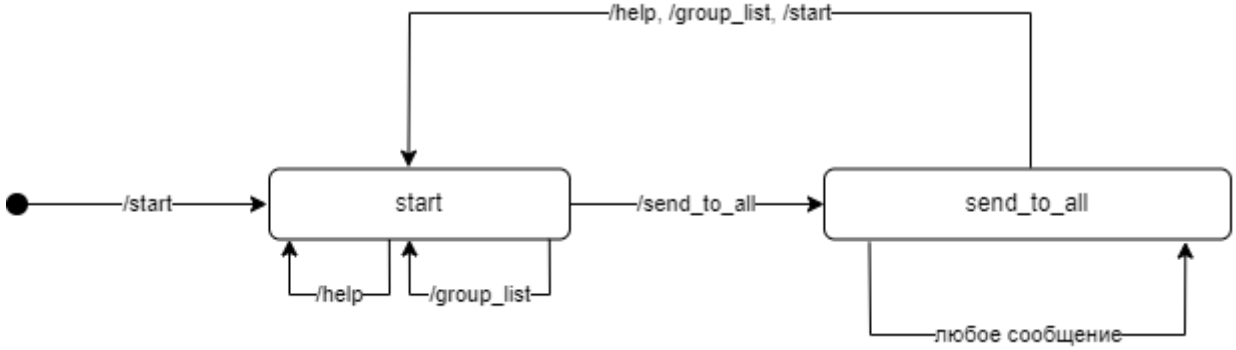
chatSPS – таблица инициализированных чатов

- chatID – идентификатор чата в телеграм
- chatName – название чата

mainChatUserState – хранение состояний пользователей панели администратора

- userId – идентификатор пользователя Telegram (должны быть связаны с таблицей Admins через Primary\Foreign Key, но это сейчас не прописано)
- state – состояние пользователя

Автомат состояний панели администратора



В случае ввода некорректной команды состояние не меняется.

Импортируемые модули

- **sqlite3** Работа с базой данных SQLite.
- **telebot** Используется для создания и управления Telegram-ботом.
- **time** Используется для задержки при отправке сообщений.
- **telebot.types** Работа с типами сообщений и клавиатурами в Telegram.
- **config** Файл конфигурации, содержащий `TOKEN` — токен бота для Telegram.

Инициализация

Создание экземпляра бота с токеном, который используется для аутентификации бота в Telegram.

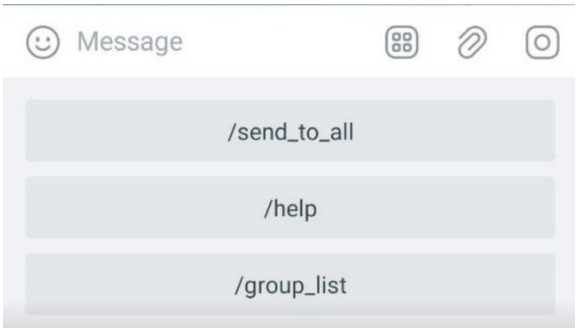
bot = telebot.TeleBot(TOKEN)

Функции

Функция	**Комментарий**
check_admins_rights_by_message(message)	- Проверяет, есть ли у пользователя права администратора по его `username` из message. Подключается к базе данных `AutoSendBot.db` и выполняет SQL-запрос для поиска пользователя в таблице `admins`.
check_admins_rights_by_name(name)	Проверяет, является ли пользователь администратором по имени.
check_group(group_id)	Проверяет, находится ли группа с данным `group_id` в таблице `chatSPS`.
add_group(group_id, chat_title)	Добавляет новую группу в таблицу `chatSPS` с `group_id` и `chat_title`.
get_group_name(chatId)	Получает название группы по ее `chatId`.
get_all_groups()	Получает список всех групп из таблицы `chatSPS`.
delete_group(groupId)	Удаляет группу из таблицы `chatSPS` по `groupId`.
check_state(userId)	Проверяет, есть ли запись состояния для пользователя в таблице `mainChatUserState`.
get_state(userId)	Получает состояние пользователя из таблицы `mainChatUserState`.

update_group_name(groupId, new_name)	Обновляет название группы в таблице `chatSPS` по `groupId`.
update_state(userId, new_state)	Обновляет состояние пользователя в таблице `mainChatUserState`.
add_state(userId, state)	Добавляет состояние пользователя в таблицу `mainChatUserState`.
add_admin(name)	Добавляет нового администратора в таблицу `admins`.
get_all_chats()	Получает все `chatID` из таблицы `chatSPS`.
check_state_send_to_all(message)	Проверяет, установлено ли состояние пользователя на `send_to_all`.

Создание клавиатуры с кнопками для управления ботом




```
markup1 = types.ReplyKeyboardMarkup(resize_keyboard=True)
item1 = types.KeyboardButton("/send_to_all")
item2 = types.KeyboardButton("/help")
item3 = types.KeyboardButton("/group_list")
markup1.add(item1, item2, item3)
```

Обработчики сообщений

Обработчик	**Комментарий**
@bot.message_handler(func=check_admins_rights_by_message, commands=['start'])	В групповом чате команда `/start` добавляет группу в базу данных. В приватном чате команда `/start` открывает администраторам доступ к функционалу бота.
@bot.message_handler(func=lambda x: check_admins_rights_by_message(x) * (x.chat.type == 'group' or x.chat.type == 'supergroup'), commands=['delete'])	Команда `/delete` удаляет группу из базы данных.
@bot.message_handler(func=lambda x: check_admins_rights_by_message(x) * x.chat.type == 'private', commands=['help'])	Команда `/help` предоставляет информацию о том, как использовать бота.
@bot.message_handler(func=lambda x: check_admins_rights_by_message(x) * x.chat.type == 'private', commands=['group_list'])	Команда `/group_list` показывает список всех групп.
@bot.message_handler(func=lambda x: check_admins_rights_by_message(x) * x.chat.type == 'private', commands=['send_to_all'])	Команда `/send_to_all` переводит бота в режим рассылки сообщений.
@bot.message_handler(func=(lambda x: check_admins_rights_by_message(x) * check_state_send_to_all(x) * x.chat.type == 'private'), content_types=['text', 'audio', 'document', 'photo', 'sticker', 'video', 'video_note', 'voice', 'location'])	Отправляет все типы сообщений в группы, подключенные к боту.
@bot.message_handler(func=lambda x: check_admins_rights_by_message(x) * x.chat.type == 'private')	Обрабатывает все неверные команды администраторов.
@bot.message_handler(func=lambda x: x.chat.type == 'private')	Обрабатывает все команды обычных пользователей.

Запуск бота

Бот запускается в бесконечном режиме опроса для обработки входящих сообщений.

```
bot.infinity_polling()
```