

Разработка чат-бота на Python с возможностью сохранения данных пользователей в базу данных SQL

Зонов Николай Александрович

*Приамурский государственный университет имени Шолом-Алейхема
Студент*

Научный руководитель:

Лучанинов Дмитрий Васильевич

*Приамурский государственный университет имени Шолом-Алейхема
Старший преподаватель*

Аннотация

В работе рассматривается задача по проектированию и разработке чат-бота в мессенджере telegram, с возможностью сохранения данных пользователей в базу данных SQL. Реализация проекта выполняется на языке Python. Описан алгоритм создания чат-бота telegram на языке Python.

Ключевые слова: Чат-бот, мессенджер, telegram, база данных, SQL, разработка.

Development of a chatbot in Python with the ability to save user data to SQL database

Zonov Nikolay Alexandrovich

*Sholom-Aleichem Priamursky State University
Student*

Scientific adviser:

Luchaninov Dmitry Vasilievich

*Sholom-Aleichem Priamursky State University
Senior lector*

Abstract

The work considers the task of designing and developing a chat bot in the telegram messenger, with the ability to save user data to a SQL database. The project is implemented in Python. The algorithm for creating a telegram chat bot in Python is described.

Keywords: Chatbot, messenger, telegram, database, SQL, development.

1. Введение

1.1 Актуальность

В настоящее время чат-боты как современные инструменты коммуникаций стали широко использоваться во многих сферах

жизнедеятельности человека с различной целью, такие как предоставление пользователю нужной информации или сбор данных, отправляя запрограммированный ответ на пользовательское сообщение или выбранную функцию.

В силу того, что контакт с чат-ботами является не только интересным занятием, но и полезным для человека, с точки зрения получения новой информации и решения потребительских задач, интернет-аудитория приняла их появление очень доброжелательно.

Многие организации в свою очередь увидели коммерческую выгоду в интересе аудитории к чат-ботам и стали разрабатывать собственные, используя их для привлечения клиентов, оказание консультаций по интересующих вопросам или товарам, выяснение потребностей клиентов, формирование лояльности к компании, вовлечение потребителя в рекламную коммуникацию, что повышает ее эффективность, автоматическая рассылка рекламы и многое другое.

Чат-бот становится мощным инструментом продвижения как для крупных брендов, так и для новичков.

Таким образом, исходя из современного уровня развития коммуникационных средств, можно сделать вывод о перспективности использования чат-ботов не только в бытовой практике, в популярных мессенджерах, но и в сферах экономики, образования, предоставления услуг и многих других областях деятельности человека.

1.3 Цель исследования

Целью данного исследования является создание функционала сохранения информации от пользователей чат-бота в базу данных, с помощью программного языка Python.

2. Инструменты разработки

Для написания кода использовался Atom – это многофункциональный текстовый редактор, с приятным интерфейсом и возможностью установки огромного количества расширений, позволяющих приспособить его практически к любому языку программирования. Однако для более полноценной работы – валидации кода, проверки синтаксиса, базовой отладки и сниплетов, как и в большинстве редакторов кода, необходимо подключение дополнительных пакетов, но их установка не составляет труда.

Для Python рекомендуются следующие расширения:

1. Script – запускает код на Python.
2. Linter и Linter Python PEP8 – отображение синтаксических ошибок.
3. Atom Python Debugger – отладка и тестирование кода.

Библиотека pyTelegramBotAPI (telebot) на Python. Необходима для взаимодействия с Telegram Bot API (представляет из себя HTTP-интерфейс для работы с ботами в Telegram.)

Стандартная библиотека `sqlite3` на Python от версии 2.5, для создания базы данных SQLite без необходимости скачивания дополнительных инструментов.

3. Проектирование

С использованием перечисленных инструментов, необходимо разработать чат-бота для Telegram с проверкой созданной ранее базы данных, при запуске файла с кодом и созданием базы, если она отсутствует.

Структура базы данных будет состоять из двух таблиц, `messages` куда будут сохраняться все сообщения от зарегистрированных пользователей и таблица `users` для регистрации новых пользователей. Также необходимо реализовать вывод сообщения если пользователь был зарегистрирован ранее.

Предусмотреть возможность администратора чат-бота просмотреть сообщения от любого зарегистрированного пользователя.

3. Разработка

Предварительно необходимо зарегистрировать бота в мессенджере Telegram и получить токен (ключ) для работы с Telegram API. Регистрация происходит с помощью бота `@BotFather`

Токен — это секретный ключ-идентификатор бота, используется при обращении к Telegram API для идентификации бота.

В поисковой строке необходимо вести `@BotFather` и выбрать вариант поиска с соответствующим именем. После выбора бота и нажатия команды `/start`, бот покажет полный список команд для регистрации, настройки и удаления бота (Рис. 2).

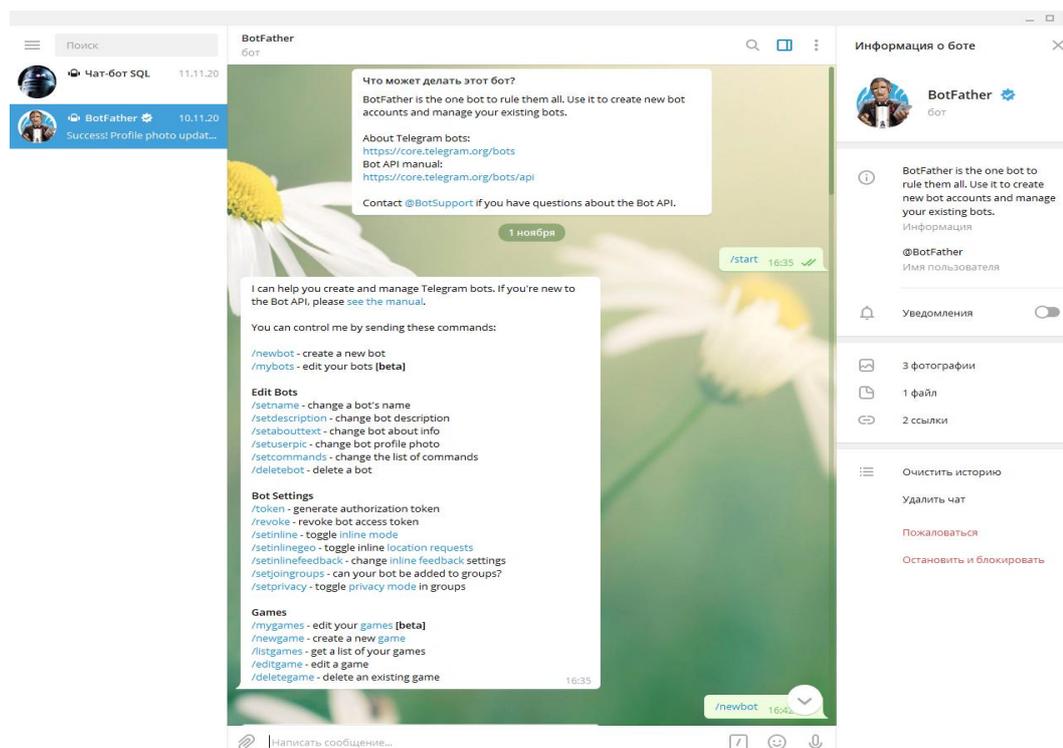


Рисунок 2. Список доступных команд

Далее нам понадобится указать название бота, задать адрес ссылки, описание, установить картинку бота и получить токен.

Для создания бота введите в чат с BotFather команду /newbot. Пользователя попросят ввести название для нового бота. Имя будет отображаться в заголовке и в информации о боте.

После того, как задано имя бота, нужно указать его сокращенное название для ссылок. Оно должно обязательно содержать приставку «bot» на конце. Например: «Test_Bot».

Если все выполнено верно, бот будет зарегистрирован в Телеграм и BotFather выдаст токен следующего формате:

```
API:764645301:AAGdRMMi_bF67lCkJjA0DKQNOwoATJQMWXk
```

Пример выполнения команд (Рис. 3).

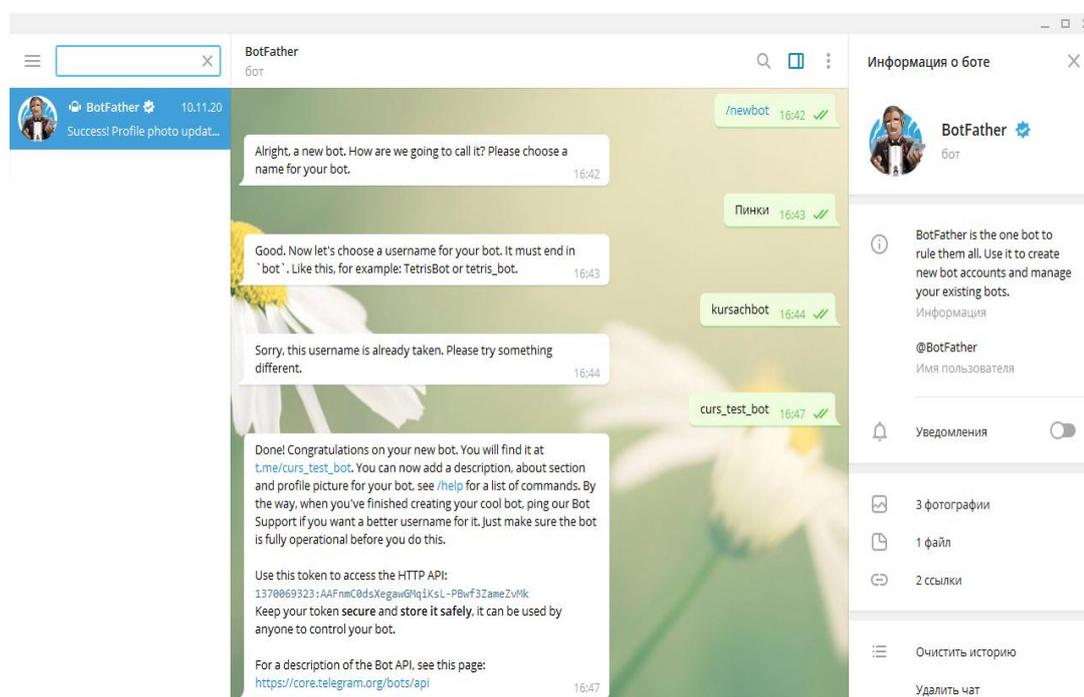


Рисунок 3. Пример выполнения команд

Этот токен используется для авторизации бота и отправки запросов к Bot API.

Как итог, бот зарегистрирован и получен токен. Далее в BotFather можно установить фото профиля, задать описание и приветствие. Токен также необходим для подключения Телеграм бота с его программной частью.

Подготовка к реализации проекта включает в себя установку и настройку необходимых программ и библиотек, а также получение токена.

Скачиваем с официального сайта установочный файл Python актуальной версии. При установке обязательно ставим отметку Add Python «version» to PATH, для дальнейшей установки библиотек через консоль cmd командой pip.

Пример окна установки (Рис. 7).

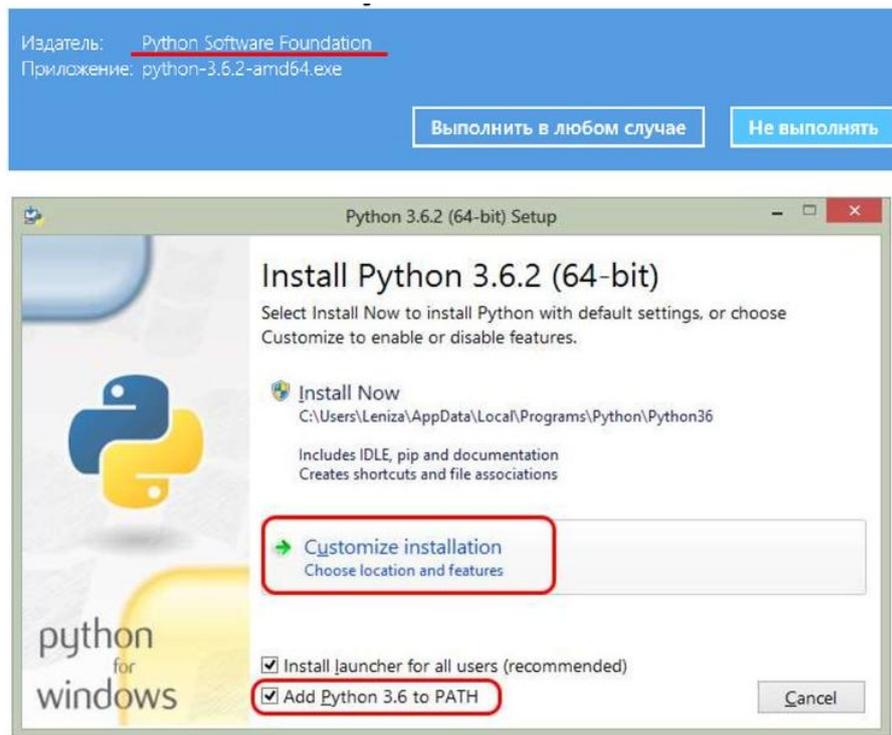


Рисунок 7. Окно установки

Для регистрации, настройки бота и получение токена в BotFather, необходимо выполнить следующие команды (Рис. 8).

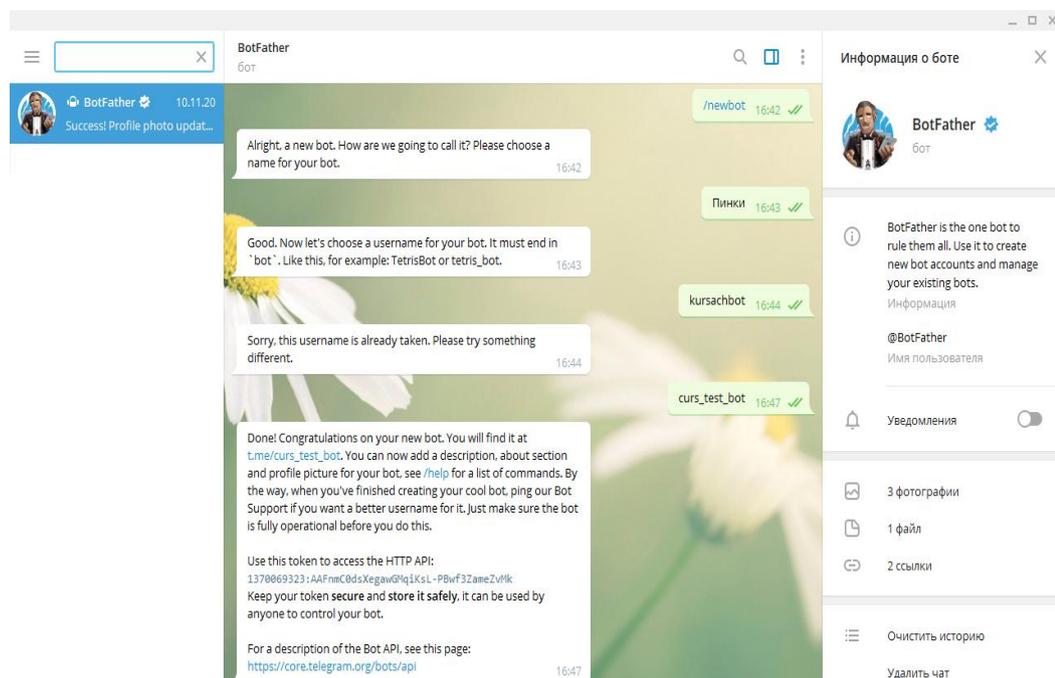


Рисунок 8. Команды для регистрации бота

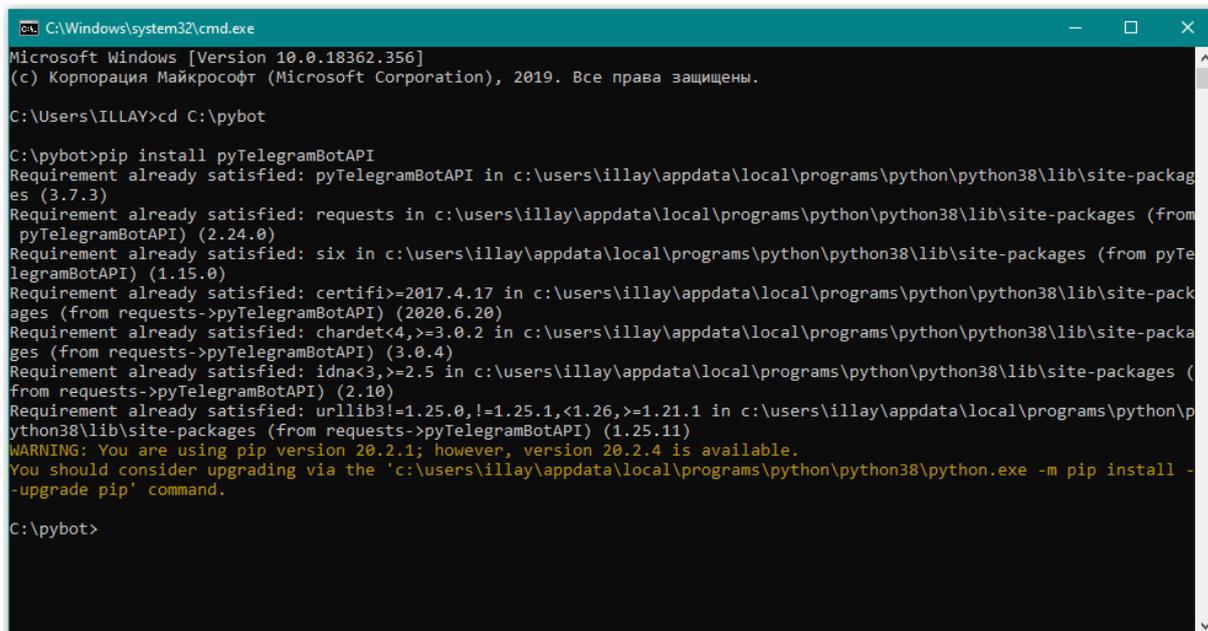
После получения токена приступаем к коду, первым шагом будет подключение библиотек и токена.

Библиотека `sqlite3` является стандартной для Python, начиная с версии 2.5, для ее интеграции достаточно указать в коде:

```
import sqlite3
```

Библиотека `telebot` устанавливается через консоль `cmd`, для этого, через `cmd` заходим в корневую папку с файлом кода, с помощью команды `cd` и прописываем команду `pip install pyTelegramBotAPI` в консоль, установка пройдет автоматически.

Пример установки библиотеки через `cmd` (Рис. 9).



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.356]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

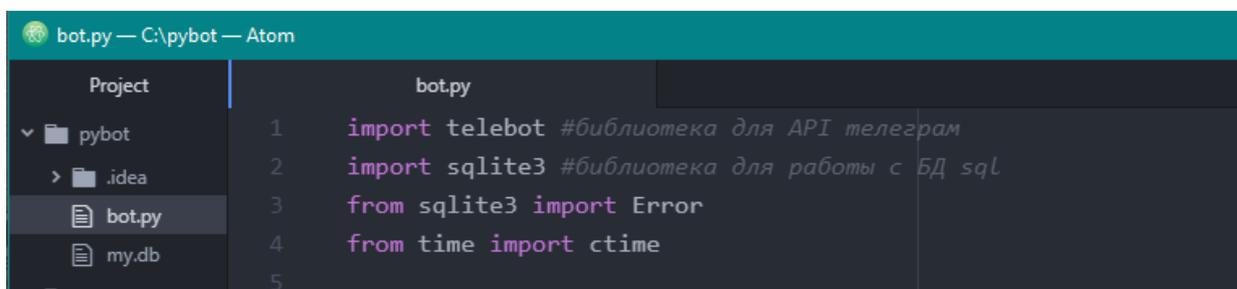
C:\Users\ILLAY>cd C:\pybot

C:\pybot>pip install pyTelegramBotAPI
Requirement already satisfied: pyTelegramBotAPI in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (3.7.3)
Requirement already satisfied: requests in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from pyTelegramBotAPI) (2.24.0)
Requirement already satisfied: six in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from pyTelegramBotAPI) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from requests->pyTelegramBotAPI) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from requests->pyTelegramBotAPI) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from requests->pyTelegramBotAPI) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 in c:\users\illay\appdata\local\programs\python\python38\lib\site-packages (from requests->pyTelegramBotAPI) (1.25.11)
WARNING: You are using pip version 20.2.1; however, version 20.2.4 is available.
You should consider upgrading via the 'c:\users\illay\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.

C:\pybot>
```

Рисунок 9. Пример установки библиотеки через `cmd`

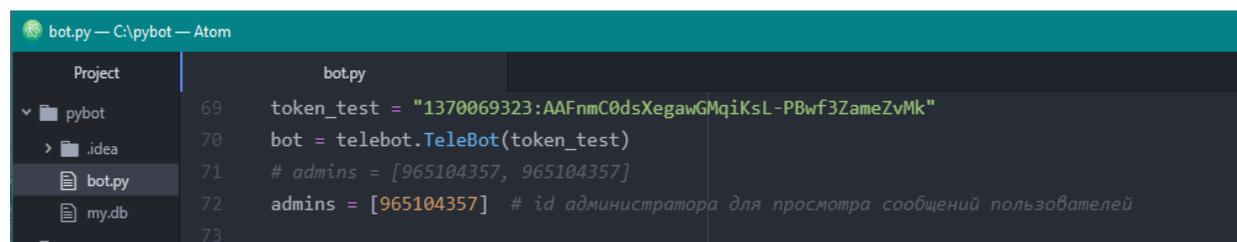
В файле кода указываем необходимые библиотеки (Рис. 10).



```
bot.py — C:\pybot — Atom
Project: bot.py
pybot
  .idea
  bot.py
  my.db
1  import telebot #библиотека для API телеграм
2  import sqlite3 #библиотека для работы с БД sql
3  from sqlite3 import Error
4  from time import ctime
5
```

Рисунок 10. Пример подключения библиотек в коде

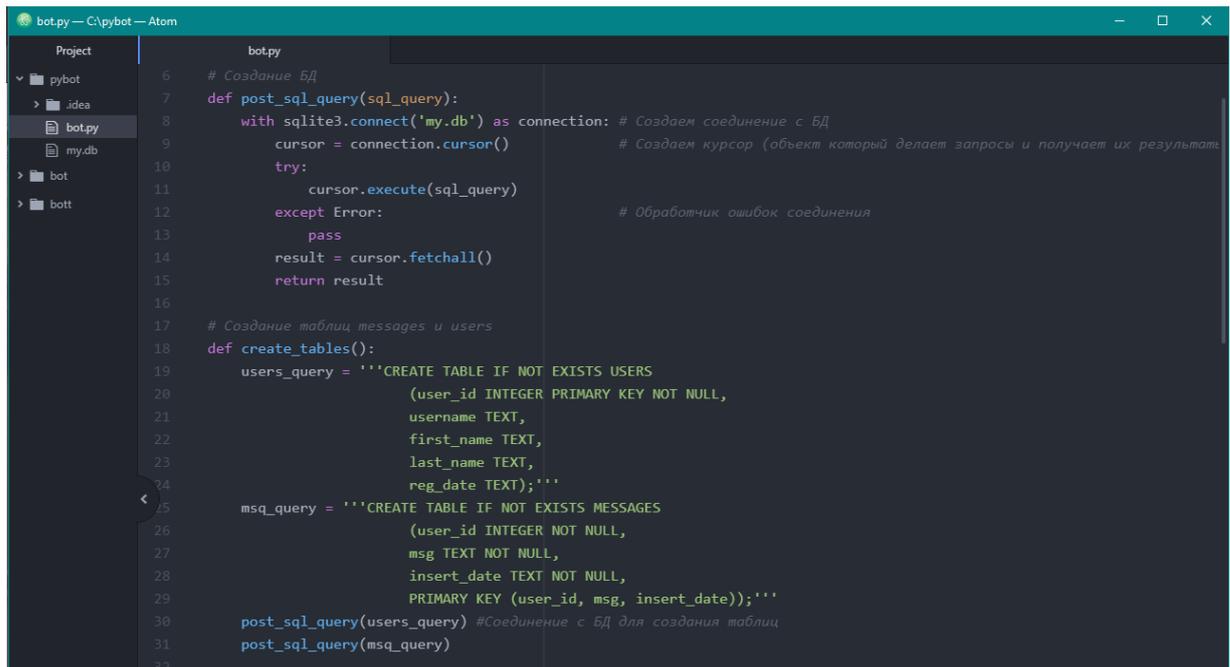
Прописываем в файле токен полученный от BotFather и указываем `id` администратора (Рис. 11).



```
bot.py — C:\pybot — Atom
Project: bot.py
pybot
  .idea
  bot.py
  my.db
69 token_test = "1370069323:AAFnmC0dsXegawGMqiKsL-PBwf3ZameZvMk"
70 bot = telebot.TeleBot(token_test)
71 # admins = [965104357, 965104357]
72 admins = [965104357] # id администратора для просмотра сообщений пользователей
73
```

Рисунок 11. Пример подключения токена

Функции для подключения базы данных с таблицами или создания новой БД если она отсутствует в корневой папке (Рис. 12).

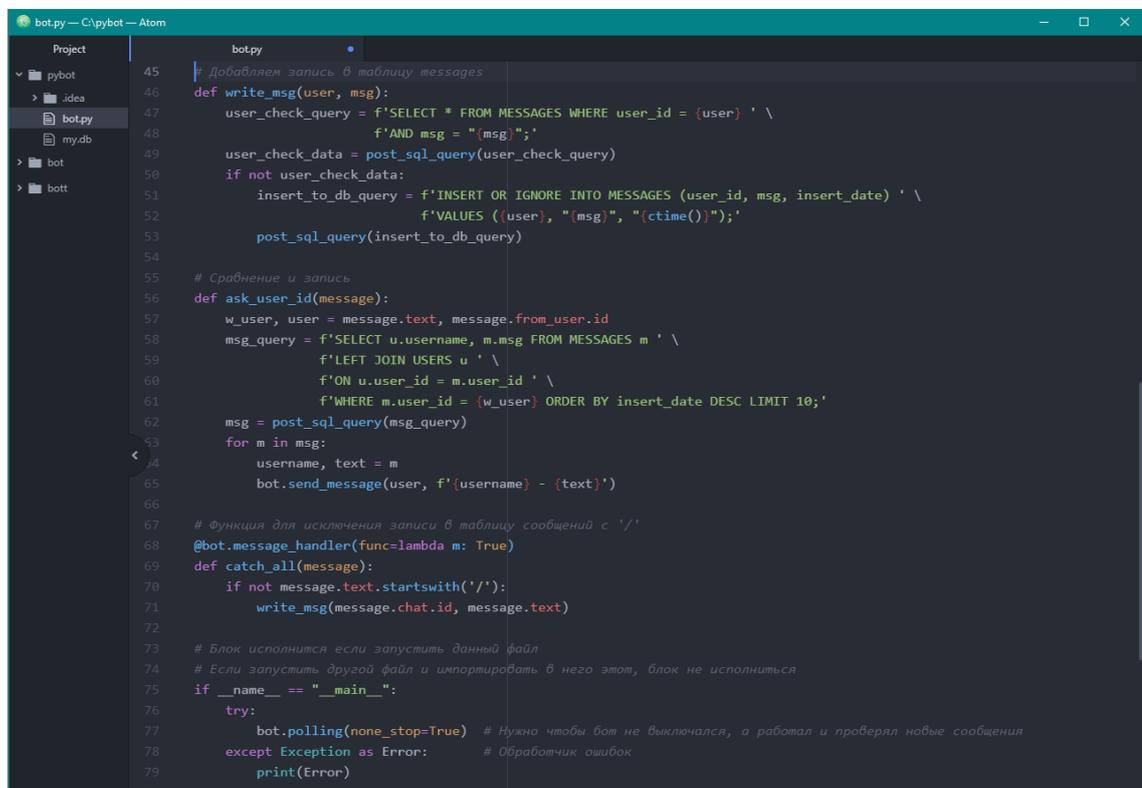


```

bot.py
6 # Создание БД
7 def post_sql_query(sql_query):
8     with sqlite3.connect('my.db') as connection: # Создаем соединение с БД
9         cursor = connection.cursor() # Создаем курсор (объект который делает запросы и получает их результат)
10        try:
11            cursor.execute(sql_query)
12        except Error: # Обработчик ошибок соединения
13            pass
14        result = cursor.fetchall()
15        return result
16
17 # Создание таблиц messages и users
18 def create_tables():
19     users_query = '''CREATE TABLE IF NOT EXISTS USERS
20                    (user_id INTEGER PRIMARY KEY NOT NULL,
21                     username TEXT,
22                     first_name TEXT,
23                     last_name TEXT,
24                     reg_date TEXT);'''
25     msq_query = '''CREATE TABLE IF NOT EXISTS MESSAGES
26                  (user_id INTEGER NOT NULL,
27                   msg TEXT NOT NULL,
28                   insert_date TEXT NOT NULL,
29                   PRIMARY KEY (user_id, msg, insert_date));'''
30     post_sql_query(users_query) #Соединение с БД для создания таблиц
31     post_sql_query(msq_query)
32
  
```

Рисунок 12. Пример функции для подключения БД

Функции для обработки сообщений пользователя, исключения записи команд и обработчики ошибок (Рис. 13).

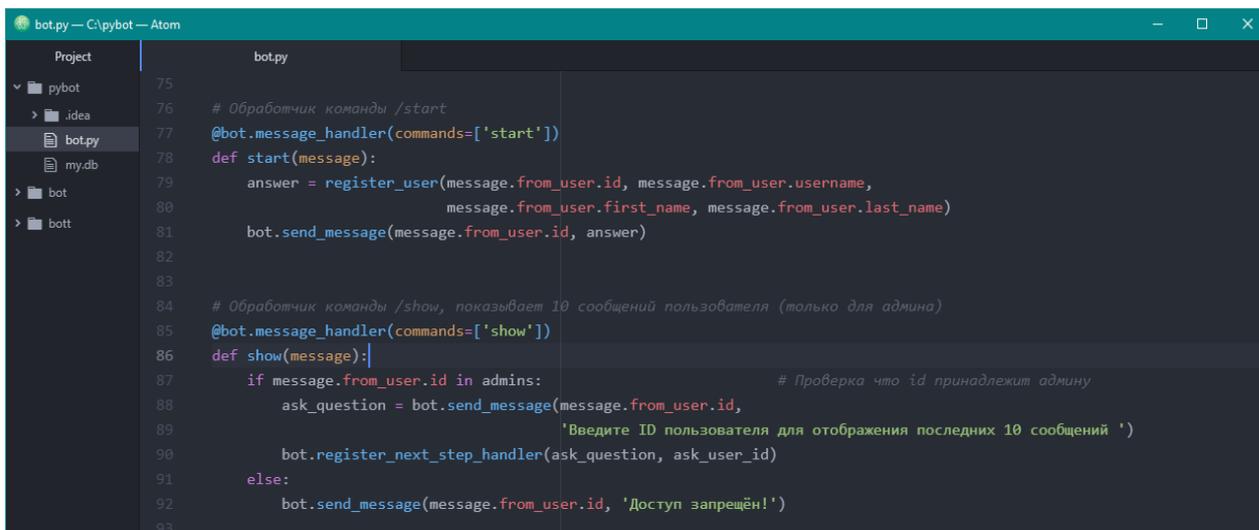


```

bot.py
45 # Добавляем запись в таблицу messages
46 def write_msg(user, msg):
47     user_check_query = f'SELECT * FROM MESSAGES WHERE user_id = {user} ' \
48                       f'AND msg = "{msg}";'
49     user_check_data = post_sql_query(user_check_query)
50     if not user_check_data:
51         insert_to_db_query = f'INSERT OR IGNORE INTO MESSAGES (user_id, msg, insert_date) ' \
52                             f'VALUES ({user}, "{msg}", "{ctime()}");'
53         post_sql_query(insert_to_db_query)
54
55 # Сравнение и запись
56 def ask_user_id(message):
57     w_user, user = message.text, message.from_user.id
58     msg_query = f'SELECT u.username, m.msg FROM MESSAGES m ' \
59               f'LEFT JOIN USERS u ' \
60               f'ON u.user_id = m.user_id ' \
61               f'WHERE m.user_id = {w_user} ORDER BY insert_date DESC LIMIT 10;\'
62     msg = post_sql_query(msg_query)
63     for m in msg:
64         username, text = m
65         bot.send_message(user, f'{username} - {text}')
66
67 # Функция для исключения записи в таблицу сообщений с '/'
68 @bot.message_handler(func=lambda m: True)
69 def catch_all(message):
70     if not message.text.startswith('/'):
71         write_msg(message.chat.id, message.text)
72
73 # Блок исполнится если запустить данный файл
74 # Если запустить другой файл и импортировать в него этот, блок не исполнится
75 if __name__ == "__main__":
76     try:
77         bot.polling(none_stop=True) # Нужно чтобы бот не выключался, а работал и проверял новые сообщения
78     except Exception as Error: # Обработчик ошибок
79         print(Error)
80
  
```

Рисунок 13. Пример функции для обработки сообщений

Обработчики команд регистрации и вывода сообщений «/start» и «/show» (Рис. 14).



```
bot.py — C:\pybot — Atom
Project
  pybot
  .idea
  bot.py
  my.db
  bot
  bott

75
76 # Обработчик команды /start
77 @bot.message_handler(commands=['start'])
78 def start(message):
79     answer = register_user(message.from_user.id, message.from_user.username,
80                           message.from_user.first_name, message.from_user.last_name)
81     bot.send_message(message.from_user.id, answer)
82
83
84 # Обработчик команды /show, показывает 10 сообщений пользователя (только для админа)
85 @bot.message_handler(commands=['show'])
86 def show(message):
87     if message.from_user.id in admins: # Проверка что id принадлежит админу
88         ask_question = bot.send_message(message.from_user.id,
89                                         'Введите ID пользователя для отображения последних 10 сообщений ')
90         bot.register_next_step_handler(ask_question, ask_user_id)
91     else:
92         bot.send_message(message.from_user.id, 'Доступ запрещён!')
93
```

Рисунок 14. Пример обработчиков команд

После завершения кода, запустим файл и протестируем бота:

При запуске код подключается к базе данных в корневой папке, если файл БД отсутствует, создается новая база.

До запуска файла (Рис. 15).

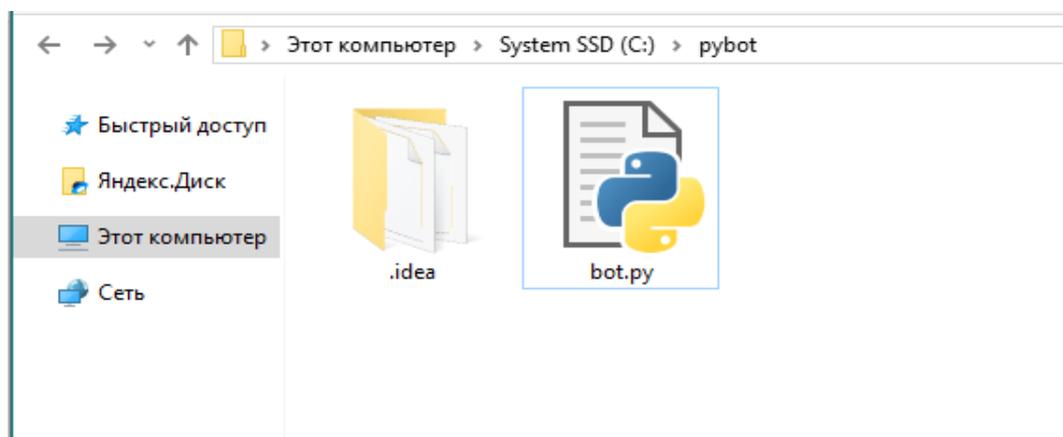


Рисунок 15. Корневая папка до запуска исполнительного файла

После запуска файла (Рис. 16).

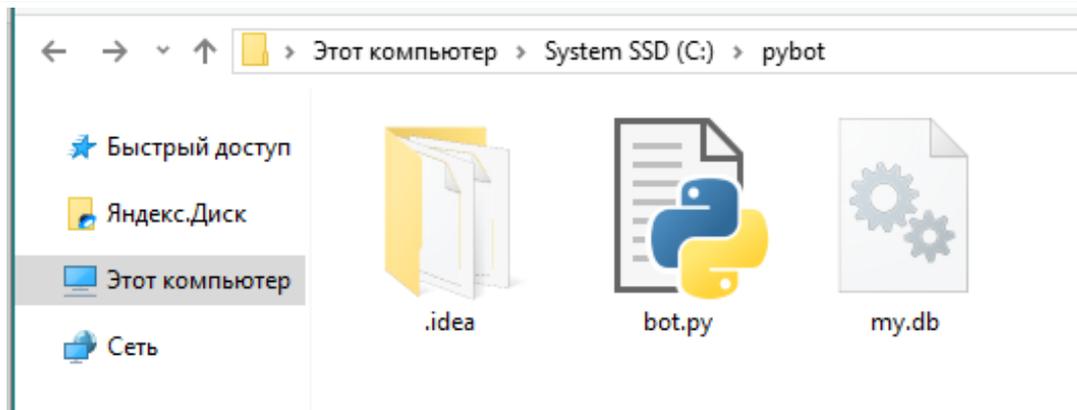


Рисунок 16. Корневая папка до запуска исполнительного файла

Как видно из скриншотов, база данных создается, откроем файл и посмотрим структуру таблицы users (Рис. 17) и messages (Рис. 18).

Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1 user_id	INTEGER	🔑				☹️		NULL
2 username	TEXT							NULL
3 first_name	TEXT							NULL
4 last_name	TEXT							NULL
5 reg_date	TEXT							NULL

Рисунок 17. Структура таблицы users

Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1 user_id	INTEGER	🔑				☹️		NULL
2 msg	TEXT	🔑				☹️		NULL
3 insert_date	TEXT	🔑				☹️		NULL

Рисунок 18. Структура таблицы messages

Создались таблицы для записи сообщений messages и регистрации пользователей users, а также первичные ключи для полей id.

Протестируем бота в мессенджере Telegram. В поисковой строке Telegram введем название @curs_test_bot и выберем бота с соответствующим именем.

Нажимаем кнопку «Запустить» или вручную вводим команду /start.

Бот приветствует нас с выводом никнейма пользователя в Telegram (Рис. 19).

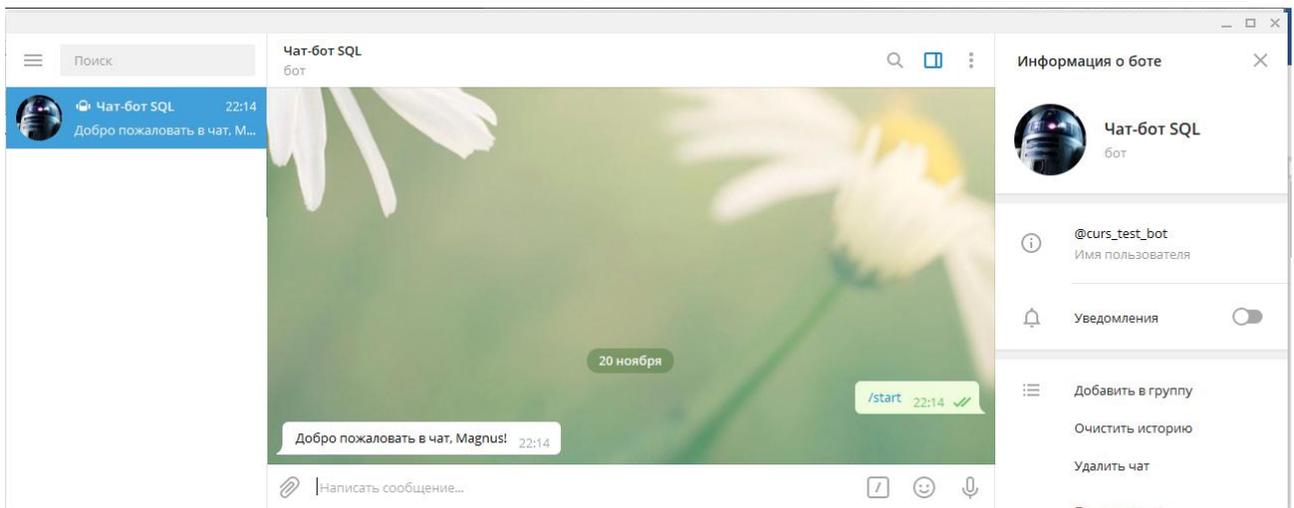


Рисунок 19. Приветствие бота. Регистрация

Проверяем таблицу users в базе данных (Рис. 20).

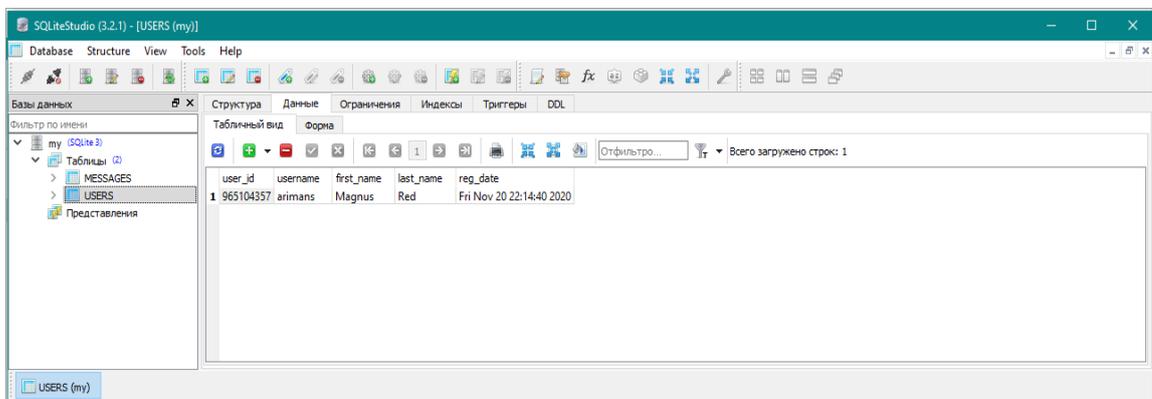


Рисунок 20. Запись в таблицу users

На скриншоте можно наблюдать, что пользователь зарегистрирован и присутствует в таблице.

При попытке снова зарегистрироваться, бот выдаст сообщение (Рис. 21).

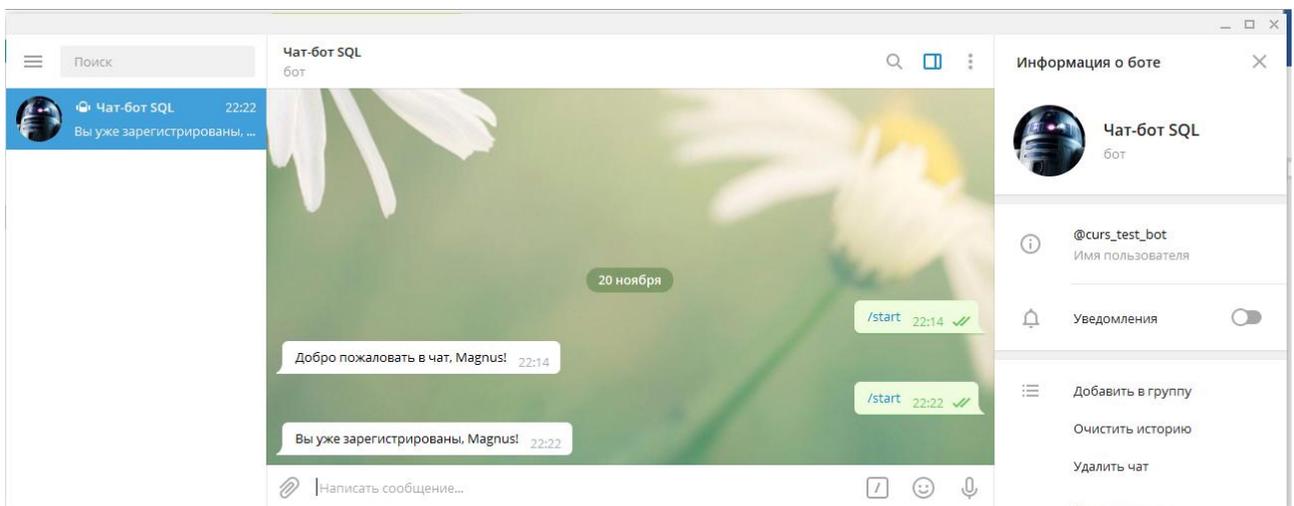


Рисунок 21. Ответ на повторную регистрацию

Напишем сообщения боту для тестирования записи в таблицу messages (Рис. 22).

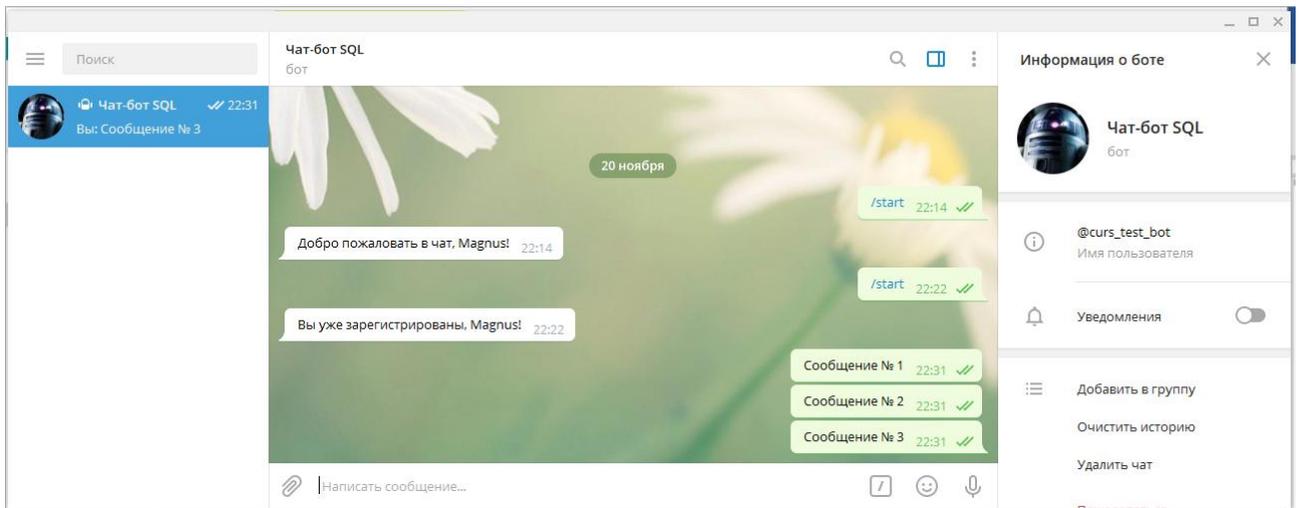


Рисунок 22. Ввод тестовых сообщений

Проверим запись в базе данных (Рис. 23).

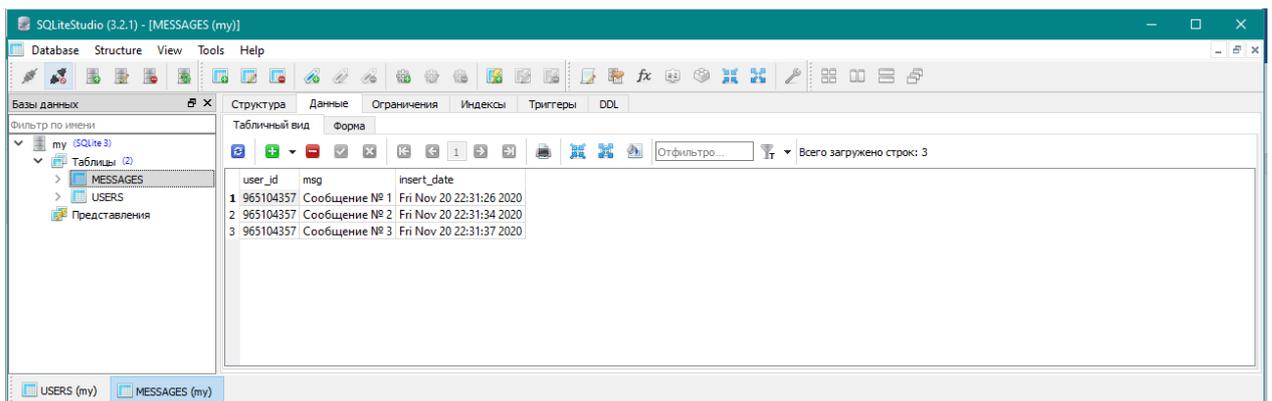


Рисунок 23. Запись тестовых сообщений в таблицу messages

Как видно из скриншотов, сообщения записались в таблицу.

Проверим функцию вывода сообщений пользователя для администратора.

В меню команд выбираем «\show» (Рис. 24).

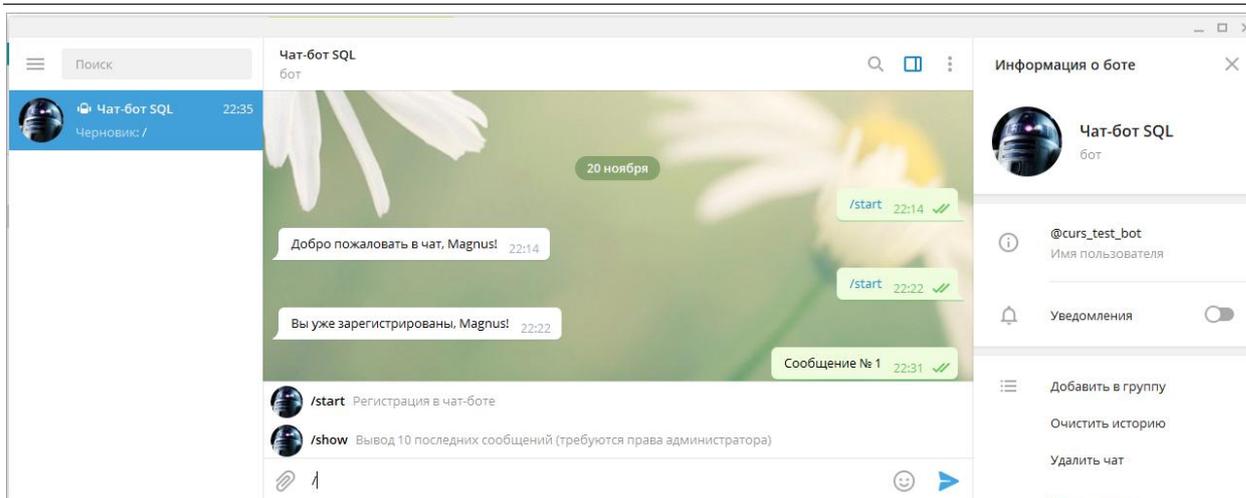


Рисунок 24. Пример интерфейса команд

Бот запросит id зарегистрированного пользователя из таблицы users и выведет последние 10 сообщений данного пользователя.

Пример выполнения команд (Рис. 25).

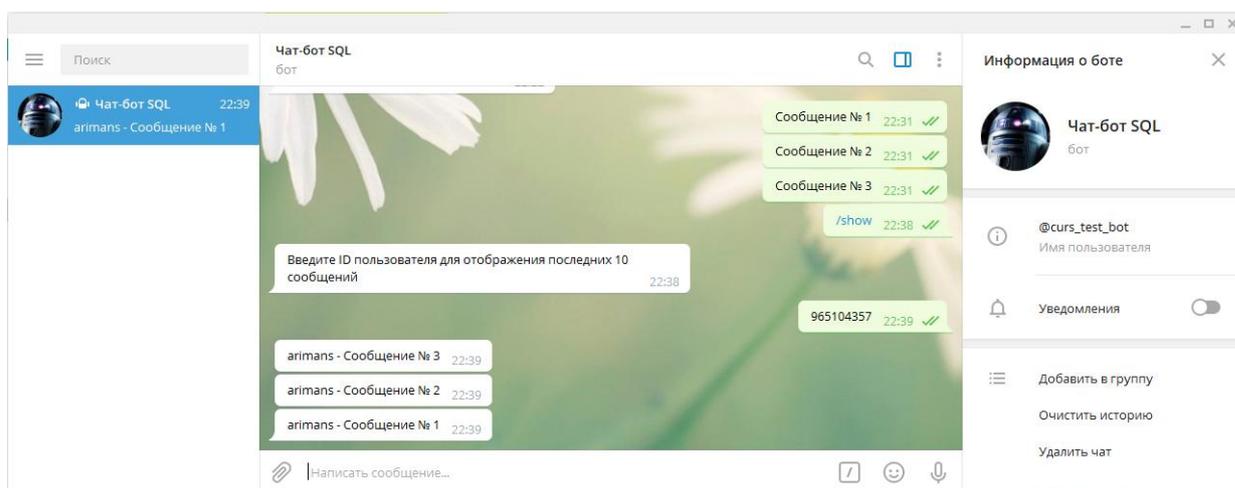


Рисунок 25. Вывод сообщений пользователя

В результате, мы протестировали весь функционал и убедились в работоспособности всех функций.

4. Вывод

В процессе выполнения данного проекта были решены задачи архивирования сообщений в базу данных SQL, регистрация пользователей, а также реализован функционал администрирования.

Библиографический список

1. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение. Рид Групп. 2012.
2. Дейт К. Введение в системы баз данных. М. Вильямс, 2006.
3. Документация. Боты: информация для разработчиков URL:

- <https://tlgrm.ru/docs/bots>.
4. Джанарсанам С. Разработка чат-ботов и разговорных интерфейсов. М.: ДМК Пресс, 2019.
 5. Руководство DB-API 2.0 interface for SQLite databases URL: <https://docs.python.org/3/library/sqlite3.html>.
 6. Справочная документация по Bot API URL: <https://tlgrm.ru/docs/bots/api>.
 7. Telegram-бот на python с помощью библиотеки telebot // Статья пользователя под ником kasad0r от 01.08.2018 <https://habr.com/ru/post/418905/>