УДК 004

Разработка Telegram бота с использованием базы данных для парсинга новостей с сайта

Эрдман Александр Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрена реализация Telegram бота с использованием базы данных для сбора и передачи информации из интернета пользователю. Бот написан на языке программирование Python с использованием библиотек aiogram, sqlite3 и СУБД SQLite. Результатом исследования является Telegram бот с подключённой базой данных для парсинга новостей сайта.
Ключевые слова: бот, Python, SQLite, sqlite3, база данных

Development of a Telegram bot using a database for parsing news from the site

Erdman Alexander Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article discusses the implementation of a Telegram bot using a database to collect and transmit information from the Internet to the user. The bot is written in the Python programming language using the aiogram, sqlite3 and SQLite DBMS libraries. The result of the study will be a Telegram bot with a connected database for parsing site news.
Keywords: bot, Python, sqlite, sqlite3, database

1 Введение

1.1 Актуальность

Необходимость использования ботов в различных сферах деятельности человека постоянно растёт. Часто представители бизнеса для первичного общения с клиентом используют ботов, которые собирают начальную необходимую информацию, которая в дальнейшем передаются в другие отделы, где происходит последующая обработка. Также помимо бизнеса, данные типы ботов используют различные сервисы, которые предоставляют пользователю на основе его предпочтений информацию. Для ботов такого «собирательного» характера в подавляющем большинстве случаев используется база данных, так как в ней удобнее хранить и обрабатывать информацию. Реализация ботов, которые выполняют различные рассылки, приобретает всё большую актуальность. На основе этого, сегодня для каждого разработчика ботов важно уметь подключать базу данных к своим

проектам. О процессе подключения базы данных к боту, а также создание явного примера практичности её использования для бота парсера будет изложено в данной статье.

1.2 Обзор исследований

Н.А. Зонов описал реализацию чат-бота с возможностью сохранения данных пользователей в базу данных SQL [1]. А.Г. Навасардян, Н.А. Зубач, Н.А. Хромова и А.В. Некрасов рассмотрели процесс создание базы данных для телеграмм бота, для удобного добавления, удаления и поиска необходимой информации [2]. Ю.В. Гаврикова и Р.Р. Исламгулов разработали систему управления БД в виде чат-бота [3]. В.А. Кожевников, О.Ю. Сабинин и Ю.Е. Шац рассмотрели вопрос использования современных мессенджеров и их возможностей администраторами баз данных [4]. Н.С. Тарасова и Н.Ю. Сергеева описали работу чат-бота, основанного на алгоритме ответов в базе данных [5].

1.3 Цель исследования

Цель. исследования является создание бота-парсера с поддержкой базы данных для мессенджера Telegram.

2 Материалы и методы

Бот написан на языке программирования Python с использованием библиотек aiogram, sqlite3 и BeautifulSoup. В качестве СУБД использовался SQlite, в качестве IDE был выбран PyCharm.

3 Результаты и обсуждения

Для начала создания бота, необходимо заранее установить библиотеки aiogram, sqlite3 и BeautifulSoup (библиотека для парсинга), а также получить токен бота в телеграмме. Главная идея бота состоит в том, чтобы пользователь мог подписаться на него и получать вовремя интересующие его уведомления о новостях с конкретного сайта – новость отправляется пользователю сразу, как только она была опубликована на сайте. Наравне с ботом существуют обычные группы по определённой тематике, но реализация бота парсера является более лучше идей, чем создание группы, так как боту можно добавлять различные функции, которые нельзя реализовать в обычной группе мессенджера.

Первой и основной функцией бота будет являться подписка на рассылку. Для реализации данной функции необходимо сперва создать базу данных с помощью СУБД SQLite. В базе данных будет хранится список пользователей. Столбцы в таблице следующие: id – уникальный идентификатор, который необходим каждой записи в таблице; второй столбец – это user_id, в котором будут хранится id пользователей в Telegram (тип данных VARCHAR(255)); третий столбец – status, который является индикатором подписки пользователя (тип данных BOOLEAN, который имеет в виду, что пользователь подписан (True) или не подписан (FALSE)) (рис. 1).

UserInfo Table name: DBForBot WITHOUT ROWID STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	id	INTEGER								NULL
2	user_id	VARCHAR (255)								NULL
3	status	BOOLEAN								TRUE

Рисунок 1. База данных пользователей

Для работы с базой данных используется отдельный файл sql.py (рис. 2). В данном файле прописываются функции открытия и закрытия соединения с базой данных:

```
def __init__(self, database):
    """Подключение к БД и сохранение курсора соединения"""
    self.connection = sqlite3.connect(database)
    self.cursor = self.connection.cursor()

def close(self):
    """Закрытие соединения с БД"""
    self.connection.close()
```

Между данными функциями реализуются четыре метода для работы с базой данных:

1. get_subscriptions – возвращает список всех подписчиков бота;
2. subscriber_exists – проверяет, есть ли пользователь в базе данных;
3. add_subscriber – добавляет пользователя в базу данных;
4. update_subscription – обновляет столбец «status» подписки;

```
def __init__(self, database):
    self.connection = sqlite3.connect(database)
    self.cursor = self.connection.cursor()
def get_subscriptions(self, status = True):
    with self.connection:
        return self.cursor.execute("SELECT * FROM `UserInfo` WHERE `status` = ?", (status,)).fetchall()
def subscriber_exists(self, user_id):
    with self.connection:
        result = self.cursor.execute('SELECT * FROM `UserInfo` WHERE `user_id` = ?', (user_id,)).fetchall()
        return bool(len(result))
def add_subscriber(self, user_id, status = True):
    with self.connection:
        return self.cursor.execute("INSERT INTO `UserInfo` (`user_id`, `status`) VALUES(?,?)", (user_id,status))
def update_subscription(self, user_id, status):
    with self.connection:
        return self.cursor.execute("UPDATE `UserInfo` SET `status` = ? WHERE `user_id` = ?", (status, user_id))
def close(self):
    self.connection.close()
```

Рисунок 2. Функционал для работы с базой данных

Далее создаётся файл конфигурации config.py, в который заносится токен бота.

После реализации функционала с базой данных и добавлением токена, следует реализация функционала непосредственно бота в файле bot.py. В

данном файле задаётся уровень логов, для просмотра информации о работе бота, и инициализация бота с указанием на его токен в файле config.py:

```
#Логирование
logging.basicConfig(level=logging.INFO)
#Инициализация бота
bot = Bot(token=config.API_TOKEN)
dp = Dispatcher(bot)
```

Для работы с базой данных подключается файл с функционалом работы базы данных с помощью команды import, а также его инициализация:

```
db = SQLighter('UserInfo.db')
```

Для того, чтобы бот мог добавлять или убирать подписки пользователей создаются две функции – subscribe() и unsubscribe() (рис. 3). В обеих функциях используются команды из подключённого файла sql.py, а именно subscriber_exists для проверки наличия пользователя в базе данных, и если их нет, то выполняется команда на добавление add_subscriber (для функции subscribe()) и функция на отключение подписки у имеющихся пользователей update_subscription с параметром False. Пользователи с отключённой подпиской не удаляются из базы данных, так как в будущем могут вернуться, поэтому меняется только статус подписки. Для функции subscribe() используется метод на обновление базы данных update_subscription и изменение статуса на True в том случае, если пользователь уже был в базе данных с отключённой подпиской и решил вернуться. По такому же принципу работает функция unsubscribe(), только update_subscription принимает значение False.

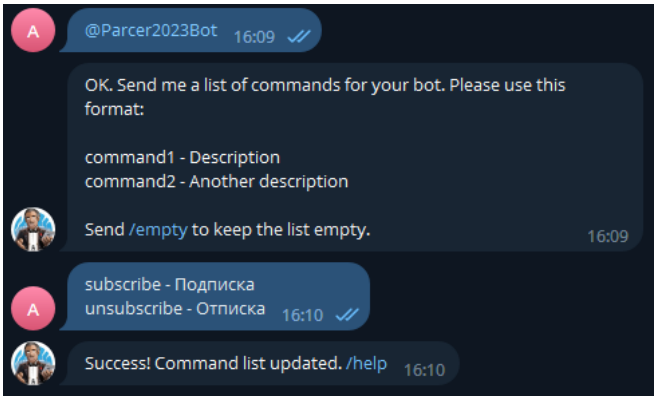
```
@dp.message_handler(commands=['subscribe'])
async def subscribe(message: types.Message):
    if(not db.subscriber_exists(message.from_user.id)):
        db.add_subscriber(message.from_user.id)
    else:
        db.update_subscription(message.from_user.id, True)

    await message.answer("Вы успешно подписались на рассылку!")

@dp.message_handler(commands=['unsubscribe'])
async def unsubscribe(message: types.Message):
    if(not db.subscriber_exists(message.from_user.id)):
        db.add_subscriber(message.from_user.id, False)
        await message.answer("Вы не были подписаны.")
    else:
        db.update_subscription(message.from_user.id, False)
        await message.answer("Вы успешно отписаны от новостей.")
```

Рисунок 3. Функции реализации подписок и отписок ботом

Для назначения команд подписки и отписки боту, необходимо использовать средства Telegram, а именно задать команды subscribe и unsubscribe с помощью @BotFather (рис. 4). Производится проверка работоспособности команд (рис. 5).



Риснуок 4. Определение команд бота

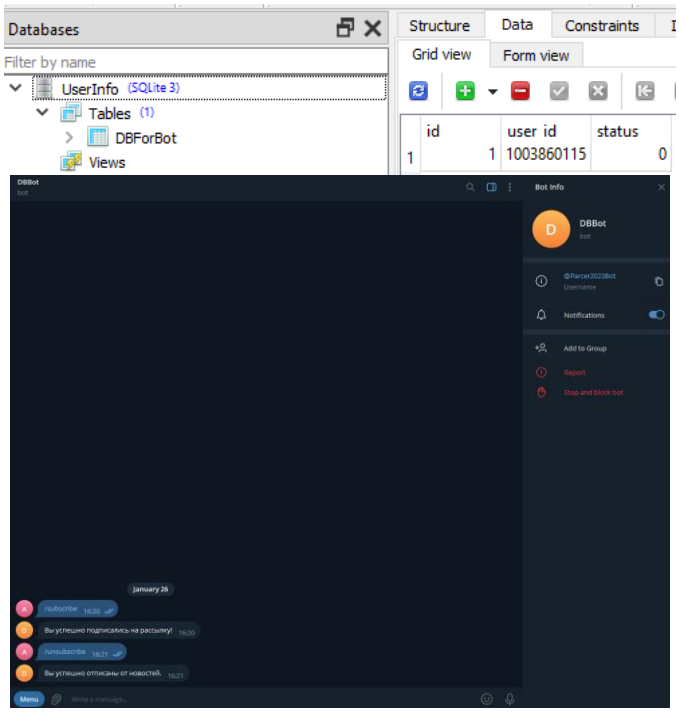
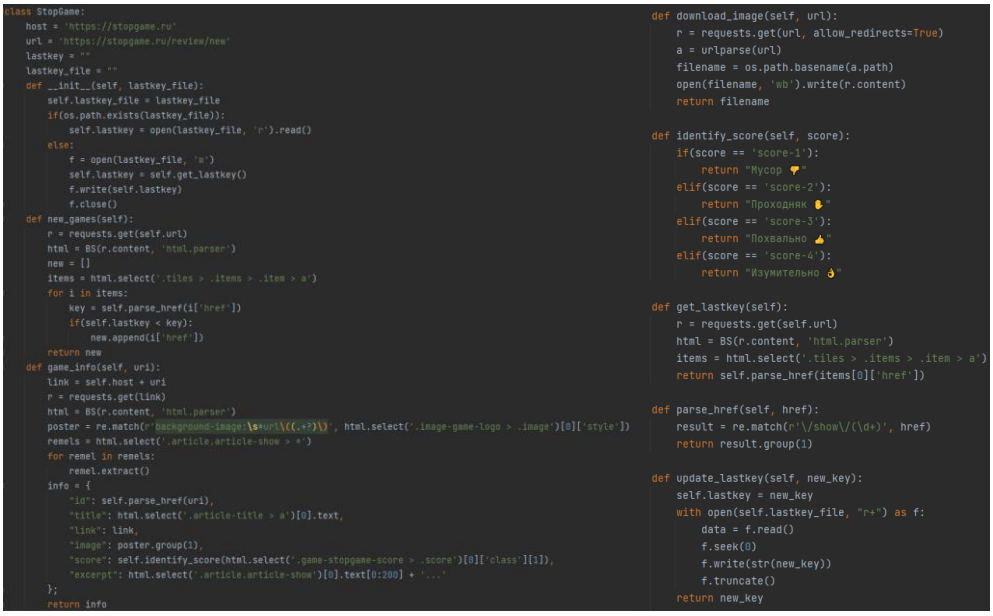


Рисунок 5. Успешное выполнение команд бота и добавление в базу данных записей

Проверка бота завершена – система подписок и отписок работает. Чтобы осуществить подписку на что либо, бот должен получить информацию с сайта. Для скрипта парсинга будет использоваться отдельный файл stopgame.py, в честь сайта, с которого будет браться информация. В данном файле идёт импорт библиотеки BeautifulSoup. Сам функционал файла выполняет проверку на наличие новых новостей на сайте – в случае если нет новостей, то ничего не происходит, а в случае, если имеются новости, то происходит рассылка. Для определения новости используется сравнение идентификатора(ключа) последней новости с настоящей, в случае если они

совпадают, то ничего не происходит, а если ключи разные, то происходит рассылка новости/записи. Первым делом происходит объявление хоста и непосредственный адрес страницы, с которой будут браться записи. В случае если, запись является новостью, то есть не совпадает с предыдущей, то используется функция new_game() для получения названия записи. Функция game_info получает изображение записи, а также текст записи. После получения изображение, оно скачивается при помощи функции download_image. Функция identify_score преобразует полученные оценки темы на сайте в нужный формат. Функция get_lastkey запоминает ключ нынешней отправленной записи. Функция parse_href отвечает за сохранение тега записи (рис. 6).



```
class StopGame:
    host = 'https://stopgame.ru'
    url = 'https://stopgame.ru/review/new'
    lastkey = ""
    lastkey_file = ""
    def __init__(self, lastkey_file):
        self.lastkey_file = lastkey_file
        if(os.path.exists(lastkey_file)):
            self.lastkey = open(lastkey_file, 'r').read()
        else:
            f = open(lastkey_file, 'w')
            self.lastkey = self.get_lastkey()
            f.write(self.lastkey)
            f.close()
    def new_games(self):
        r = requests.get(self.url)
        html = BS(r.content, 'html.parser')
        new = []
        items = html.select('.tiles > .items > .item > a')
        for i in items:
            key = self.parse_href(i['href'])
            if(self.lastkey < key):
                new.append(i['href'])
        return new
    def game_info(self, uri):
        link = self.host + uri
        r = requests.get(link)
        html = BS(r.content, 'html.parser')
        poster = re.match(r"background-image:\s*url\((.+?)\)", html.select('.image-game-logo > .image')[0]['style'])
        remels = html.select('.article.article-show > *')
        for remel in remels:
            remel.extract()
        info = {
            "id": self.parse_href(uri),
            "title": html.select('.article-title > a')[0].text,
            "link": link,
            "image": poster.group(1),
            "score": self.identify_score(html.select('.game-stopgame-score > .score')[0]['class'][1]),
            "excerpt": html.select('.article.article-show')[0].text[0:200] + '...'
        };
        return info
    def download_image(self, url):
        r = requests.get(url, allow_redirects=True)
        a = urlparse(url)
        filename = os.path.basename(a.path)
        open(filename, 'wb').write(r.content)
        return filename

    def identify_score(self, score):
        if(score == 'score-1'):
            return "Мусор 👎"
        elif(score == 'score-2'):
            return "Проходняк ✋"
        elif(score == 'score-3'):
            return "Похвально 👍"
        elif(score == 'score-4'):
            return "Изумительно 👌"

    def get_lastkey(self):
        r = requests.get(self.url)
        html = BS(r.content, 'html.parser')
        items = html.select('.tiles > .items > .item > a')
        return self.parse_href(items[0]['href'])

    def parse_href(self, href):
        result = re.match(r'\/show\/(\d+)', href)
        return result.group(1)

    def update_lastkey(self, new_key):
        self.lastkey = new_key
        with open(self.lastkey_file, "r+") as f:
            data = f.read()
            f.seek(0)
            f.write(str(new_key))
            f.truncate()
        return new_key
```

Рисунок 6. Реализация методов парсинга сайта

Для того, чтобы бот мог проверять новости и отсылать в определённое время записи, нужно добавить несколько команд в фал bot.py. Первым действием будет добавление команд из stopgame.py с помощью import. Далее происходит инициализация парсера с помощью команды:

```
sg = StopGame('lastkey.txt')
```

Для теста время, через которое будет проверяться новость, будет равно 10 секундам:

```
dp.loop.create_task(scheduled(10))
```

На практике данное время выставляется в минутах или часах – в зависимости от ресурса и скорости его публикации записей. Проверка и рассылка записей осуществляется при помощи функции scheduled() (рис. 7). В данной функции используются метод проверки новой записи, который в

случае обнаружение новой записи, начинает получать необходимую информацию со страницы (команда nfo = sg.game_info(ng)). С помощью команды subscriptions = db.get_subscriptions() происходит получение списка подписчиков для отправки. С помощью команды bot.send происходит отправка информации подписчикам. В конце обновляется ключ новости с помощью метода sg.update_lastkey().

```python
async def scheduled(wait_for):
    while True:
        await asyncio.sleep(wait_for)

        new_games = sg.new_games()
        if(new_games):
            new_games.reverse()

            for ng in new_games:
                nfo = sg.game_info(ng)
                subscriptions = db.get_subscriptions()
                with open(sg.download_image(nfo['image']), 'rb') as photo:
                    for s in subscriptions:
                        await bot.send_photo(
                            s[1],
                            photo,
                            caption = nfo['title'] + "\n" + "Оценка: " + nfo['score'] + "\n" + nfo['excerpt'] + "\n\n" + nfo['link'],
                            disable_notification = True
                        )

                sg.update_lastkey(nfo['id'])
```

Рисунок 7. Функция рассылки новостей ботом

После можно проверять работу бота (рис. 8).

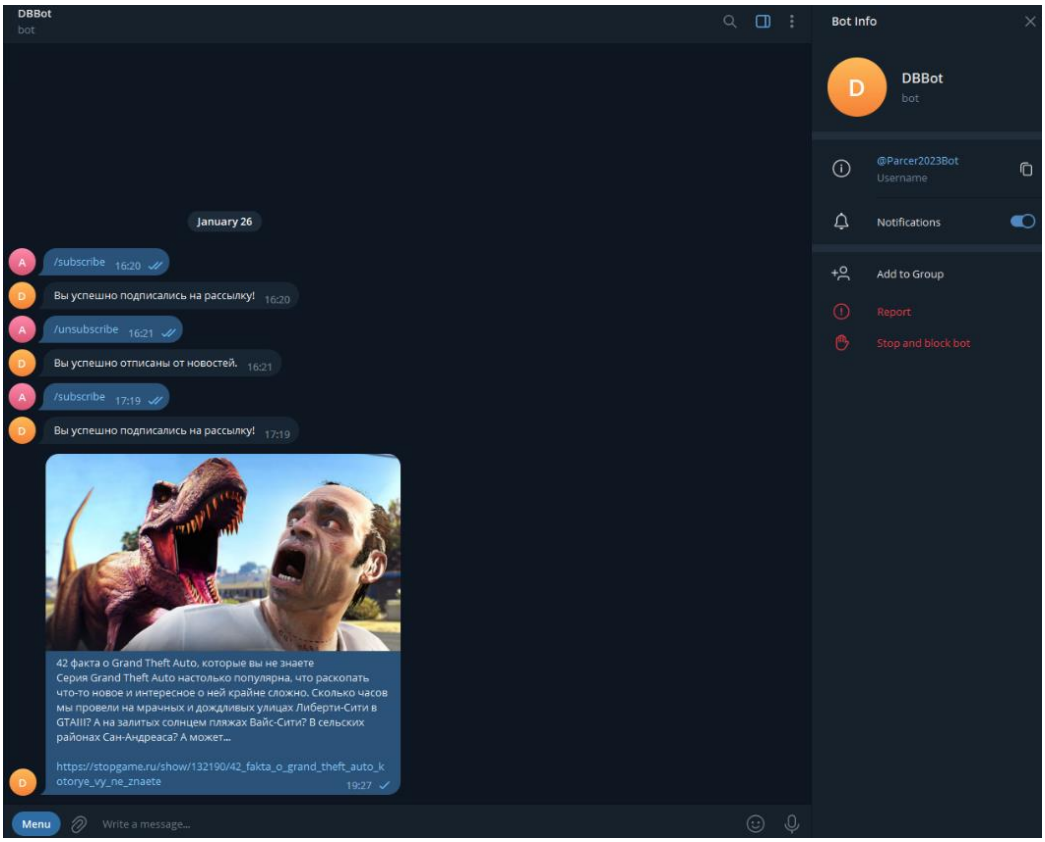


Рисунок 8. Работа Telegram бота

Выводы

Таким образом был разработан Telegram бот парсер записей сайта с подключённой базой данных, где хранятся данные о подписке пользователей.

Библиографический список

1. Зонов Н.А. Разработка чат-бота на python с возможностью сохранения данных пользователей в базу данных SQL // Постулат. 2021. № 1. С. 63.
2. Навасардян А.Г., Зубач Н.А., Хромова Н.А., Некрасов А.В. Разработка базы данных для телеграмм-бота // В сборнике: Технологии 2022: основные проблемы и направления развития. Сборник статей II Международной научно-практической конференции. Пенза, 2022. С. 67-69.
3. Гаврикова Ю.В., Исламгулов Р.Р. Система управления базами данных в рамках самостоятельной работы студента в виде чат-бота // В сборнике: Интеграция науки и образования в вузах нефтегазового профиля - 2018. материалы Международной научно-методической конференции. 2018. С. 65-68.
4. Кожевников В.А., Сабинин О.Ю., Шац Ю.Е. Современные мессенджеры в качестве помощника администратора базы данных // ScienceRise. 2018. Т. 6. С. 32-36.
5. Тарасова Н.С., Сергеева Н.Ю. Использование чат-ботов в повседневной жизни // Вестник современных исследований. 2017. № 12-1 (15). С. 195-197.