**Інформаційні мережі та телекомінікації**

**Лабораторна робота №5.**

**Тема:** Інтеграція зовнішніх систем у власний telegram bot.

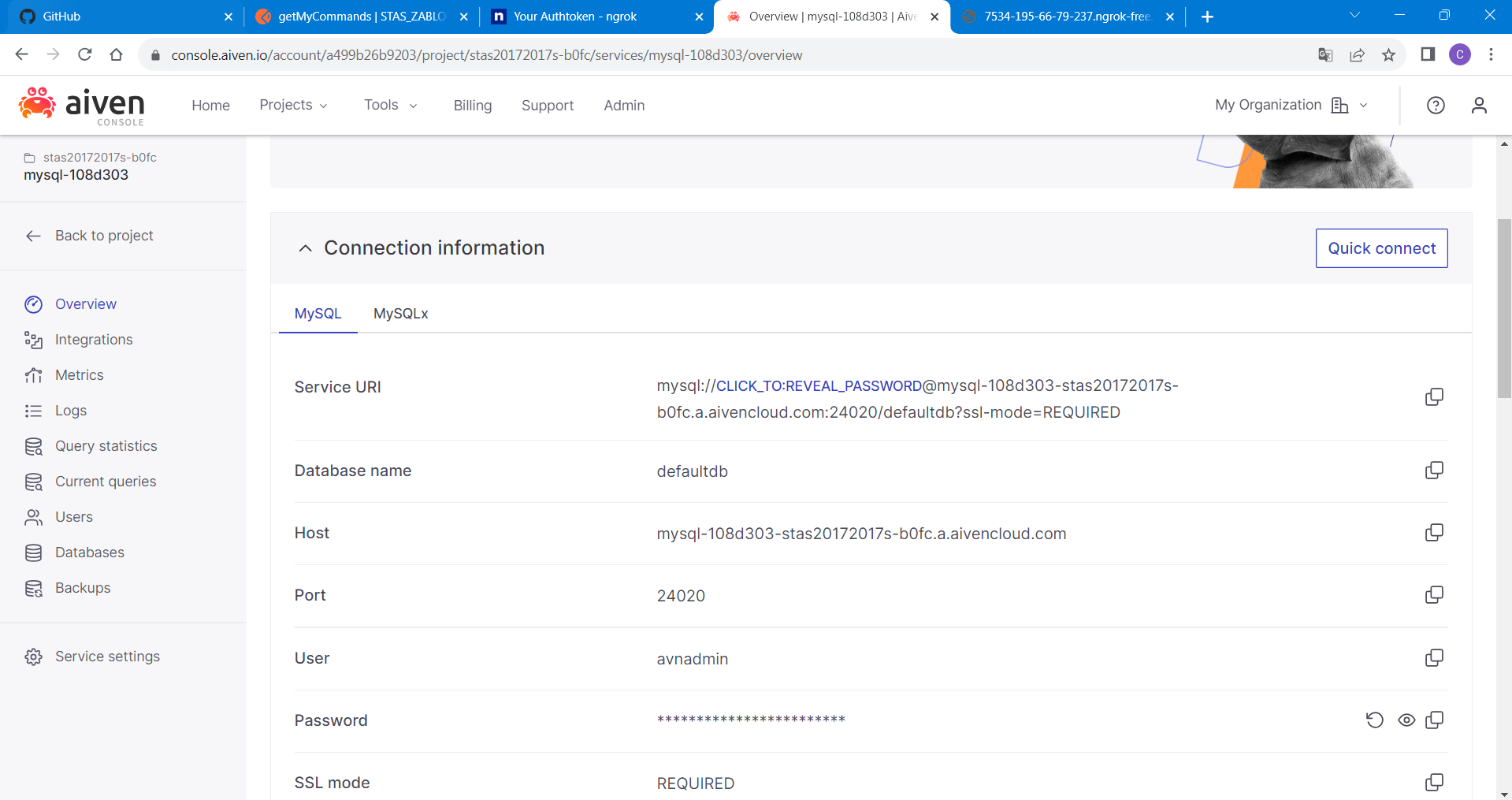
**Мета:** Навчитися використовувати зовнішні системи під час взаємодії з користувачем.

**Хiд роботи**

**І. Виконайте наступні завдання** (у файлі звіту опишіть відповідні процеси):

Продовжимо розробку telegram bot на python. В цій лабораторній підключимо зовнішню базу даних mysql для збереження даних від користувача. Представимо, що нам необхідно створити базу даних користувачів магазину і через бот зробити можливість реєстрації користувачів.

Результат виконання цієї лабораторної закомітьте в гілку lab5 вашого проекту.



**Завдання 1.** Зареєструватися на сайті <https://aiven.io/free-mysql-database> взявши безкоштовний план і створити базу даних shop.

**CREATE** **TABLE** customers (

id **INT** **AUTO\_INCREMENT** **PRIMARY** **KEY**,

telegram\_id **BIGINT**,

phone\_number **VARCHAR**(20),

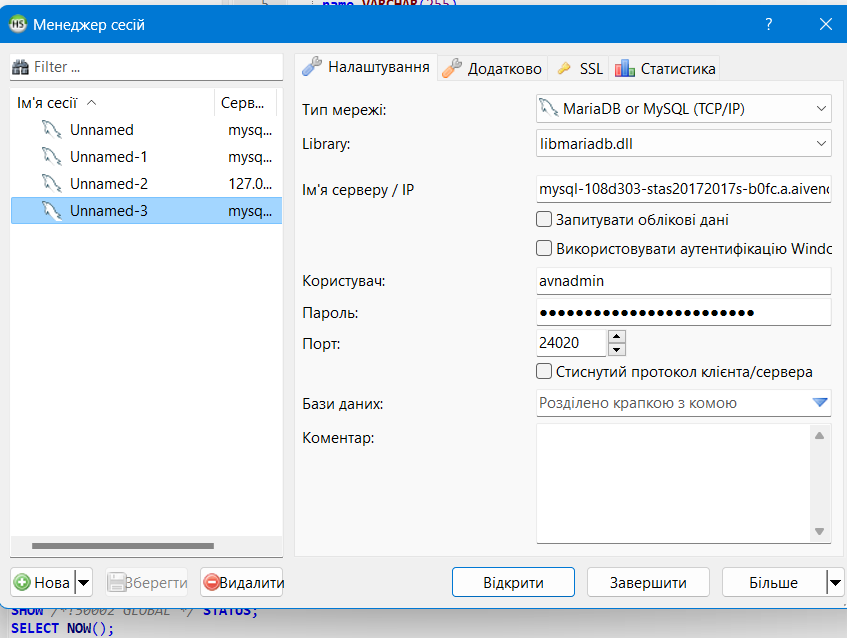
**name** **VARCHAR**(255),

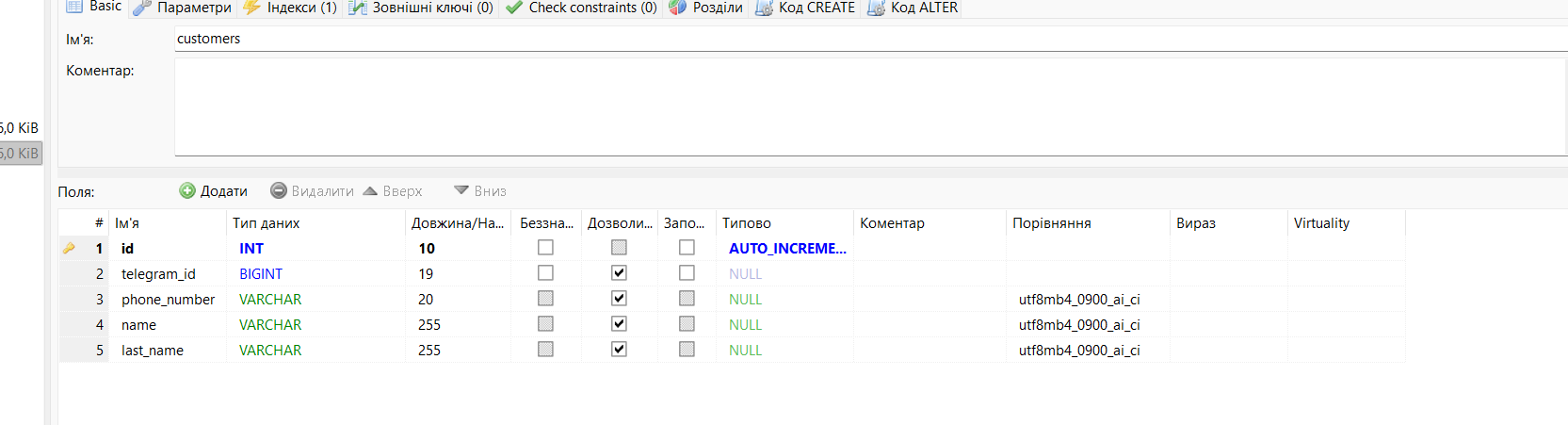
last\_name **VARCHAR**(255)

);

Далі використовуючи sql manager, або microsoft sql management server, або phpMyAdmin або php adminer або напряму використовуючи пакет mysql в системі, підключіться до сервера бази даних (інформація про підключення в пункті меню Overview) та створіть таблицю customers, яка буде містити id(налаштуйте як primary key, autoincrement), Ім’я, Прізвище та номер телефону користувача, а також його telegram id (chat\_id)

Встановіть пакети для роботи з mysql з сторони python





python -m pip install pymysql

python -m pip install cryptography

Додайте в credentials.py нові змінні для роботи з mysql

HOSTDB, DBNAME, PORTDB, USERDB, PASSDB, TIMEOUT

Значення візьміть з вкладки Overview на aiven

Приклад підключення до бази даних і взаємодії з mysql можете переглянути в цьому прикладі:

import pymysql

from myapp.credentials import HOSTDB, DBNAME, PORTDB, USERDB, PASSDB, TIMEOUT

timeout = TIMEOUT

connection = pymysql.connect(

charset="utf8mb4",

connect\_timeout=timeout,

cursorclass=pymysql.cursors.DictCursor,

db=DBNAME,

host=HOSTDB,

password=PASSDB,

read\_timeout=timeout,

port=PORTDB,

user=USERDB,

write\_timeout=timeout,

)

try:

cursor = connection.cursor()

cursor.execute("CREATE TABLE customers (id INTEGER PRIMARY KEY)")

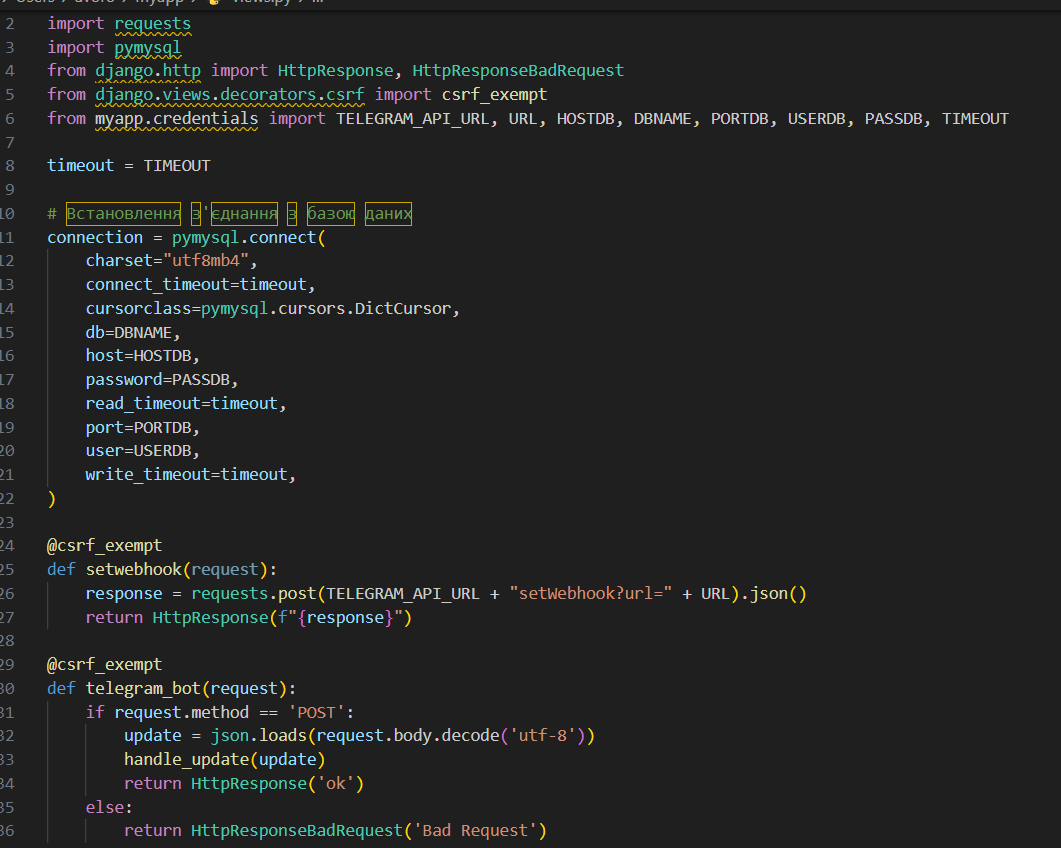
cursor.execute("INSERT INTO customers (id) VALUES (1), (2)")

cursor.execute("SELECT \* FROM customers ")

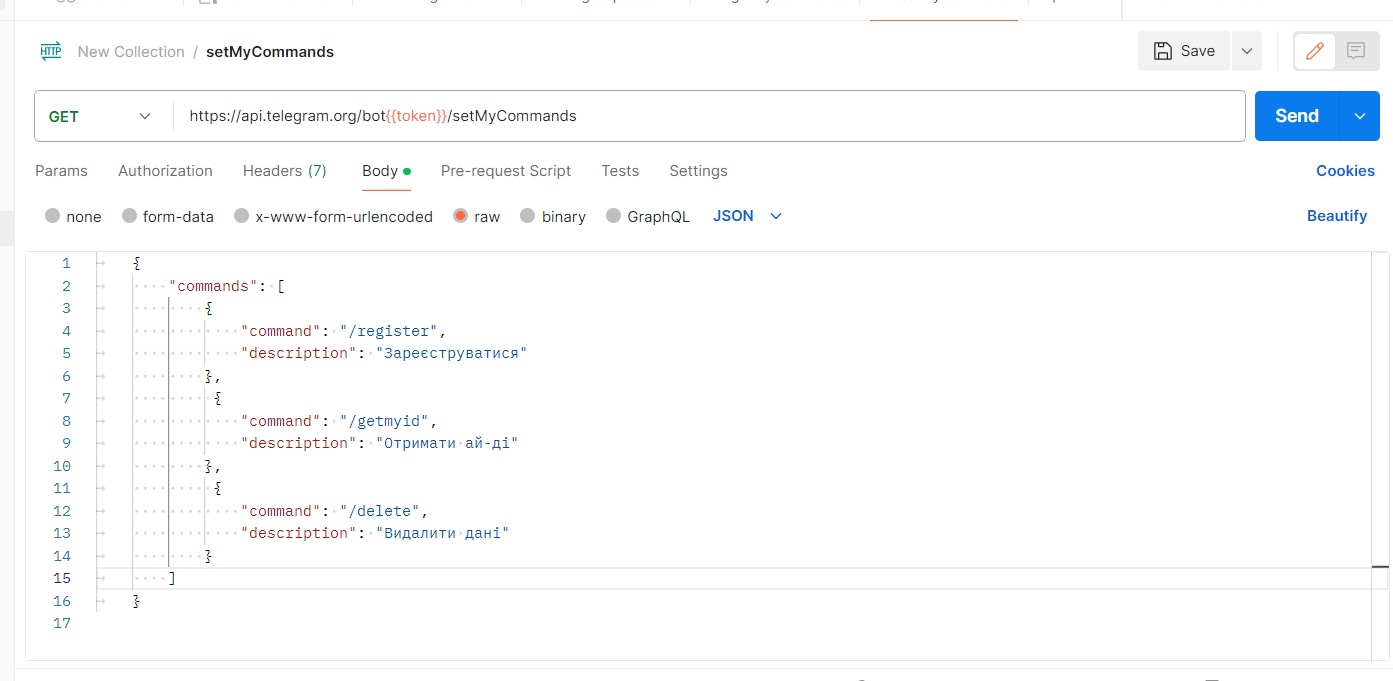
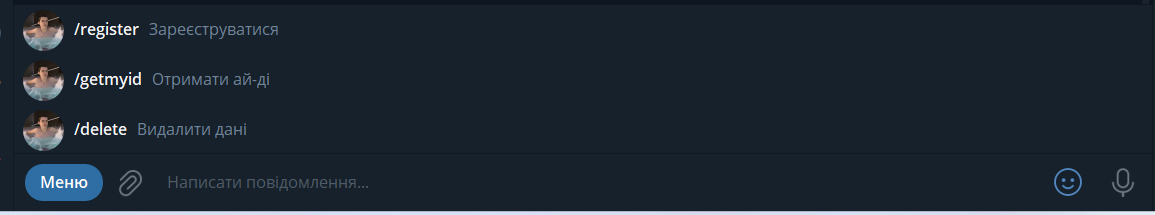
print(cursor.fetchall())

finally:

connection.close()



**Завдання 2.** Створити нову команду /register в меню бота, додати описову інформацію про те, що дана команда виконує реєстрацію нового користувача. Ви це можете зробити двома способами:

* через postman використовуючи /setMyCommands
* або через наш python бот, зареєструвавши новий шлях, аналогічно як ми робити для /setwebhook
* {
* "commands": [
* {
* "command": "/register",
* "description": "Зареєструватися"
* },
* {
* "command": "/getmyid",
* "description": "Отримати ай-ді"
* },
* {
* "command": "/delete",
* "description": "Видалити дані"
* }
* ]
* }
* 
* 

**Завдання 3.** Розробити функціонал команди /register бота таким чином, щоб користувач міг пошарити свої дані для створення нового профілю.

Ось приклад запиту для отримання об’єкту контакту користувача через кнопку контакту через команду /sendMessage:

{

"chat\_id": "ХХХХХХХХХХХХ",

"text": "Please send your contact:",

"reply\_markup": {

"keyboard": [

[

{

"text": "My phone",

"request\_contact": true

}

]

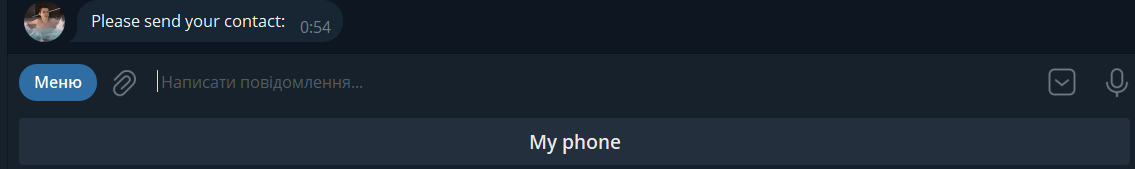
],

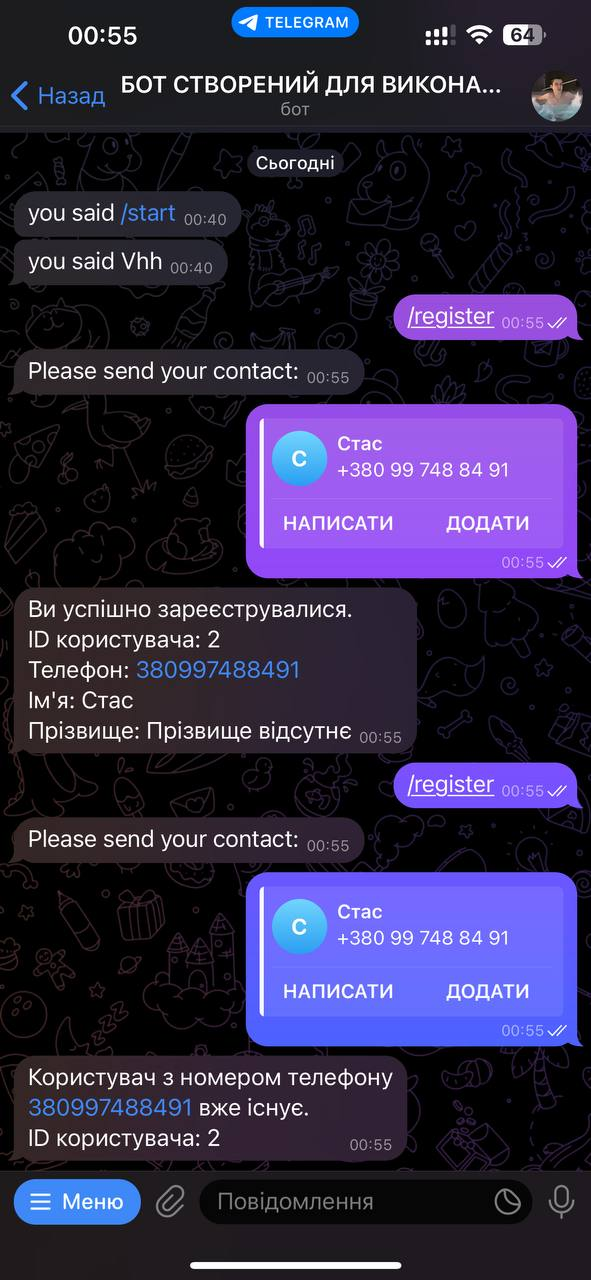
"resize\_keyboard": true,

"one\_time\_keyboard": true

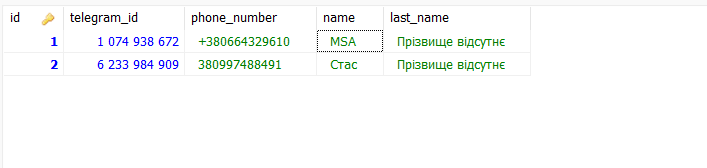
}

}  

**Завдання 4.** Отримавши всі необхідні дані від користувача, перевірити за номером телефону, чи існує такий користувач в базі даних. Якщо існує, то повернути відповідь що користувач існує і його ІД. 

**Завдання 5.** Якщо це новий користувач, зберегти дані в базу даних таблицю customers і повернути користувачу повідомлення про успішну реєстрацію та його ІД.



import json

import requests

import pymysql

from django.http import HttpResponse, HttpResponseBadRequest

from django.views.decorators.csrf import csrf\_exempt

from myapp.credentials import TELEGRAM\_API\_URL, URL, HOSTDB, DBNAME, PORTDB, USERDB, PASSDB, TIMEOUT

timeout = TIMEOUT

# Встановлення з'єднання з базою даних

connection = pymysql.connect(

    charset="utf8mb4",

    connect\_timeout=timeout,

    cursorclass=pymysql.cursors.DictCursor,

    db=DBNAME,

    host=HOSTDB,

    password=PASSDB,

    read\_timeout=timeout,

    port=PORTDB,

    user=USERDB,

    write\_timeout=timeout,

)

@csrf\_exempt

def setwebhook(request):

    response = requests.post(TELEGRAM\_API\_URL + "setWebhook?url=" + URL).json()

    return HttpResponse(f"{response}")

@csrf\_exempt

def telegram\_bot(request):

    if request.method == 'POST':

        update = json.loads(request.body.decode('utf-8'))

        handle\_update(update)

        return HttpResponse('ok')

    else:

        return HttpResponseBadRequest('Bad Request')

def handle\_update(update):

    try:

        chat\_id = update['message']['chat']['id']

        telegram\_id = update['message']['from']['id']

        text = update['message'].get('text', '')

        if text == '/register':

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': 'Please send your contact:',

                'reply\_markup': {

                    'keyboard': [

                        [

                            {

                                'text': 'My phone',

                                'request\_contact': True

                            }

                        ]

                    ],

                    'resize\_keyboard': True,

                    'one\_time\_keyboard': True,

                }

            })

        elif text == '/delete':

            success = delete\_user\_data(telegram\_id)

            if success:

                send\_message("sendMessage", {

                    'chat\_id': chat\_id,

                    'text': 'Your data has been successfully deleted.'

                })

            else:

                send\_message("sendMessage", {

                    'chat\_id': chat\_id,

                    'text': 'Failed to delete your data. Please try again later.'

                })

        elif 'contact' in update['message']:

            contact = update['message']['contact']

            phone\_number = contact.get('phone\_number', 'Номер телефону відсутній')

            name = contact.get('first\_name', 'Ім\'я відсутнє')

            last\_name = contact.get('last\_name', 'Прізвище відсутнє')

            # Перевірка наявності користувача за номером телефону в базі даних

            user\_id = check\_user\_existence(phone\_number)

            if user\_id:

                user\_info = f'Користувач з номером телефону {phone\_number} вже існує.\nID користувача: {user\_id}'

            else:

                user\_id = save\_user\_data(telegram\_id, phone\_number, name, last\_name)

                user\_info = f'Ви успішно зареєструвалися.\nID користувача: {user\_id}\nТелефон: {phone\_number}\nІм\'я: {name}\nПрізвище: {last\_name}'

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': user\_info,

                'reply\_markup': {

                    'remove\_keyboard': True,

                }

            })

        else:

            send\_message("sendMessage", {

                'chat\_id': chat\_id,

                'text': f'Ти сказав: {text}'

            })

    except Exception as e:

        send\_message("sendMessage", {

            'chat\_id': chat\_id,

            'text': 'Щось пішло не так. Спробуйте ще раз.'

        })

def send\_message(method, data):

    return requests.post(TELEGRAM\_API\_URL + method, json=data)

def check\_user\_existence(phone\_number):

    try:

        with connection.cursor() as cursor:

            sql = "SELECT id FROM customers WHERE phone\_number = %s"

            cursor.execute(sql, (phone\_number,))

            result = cursor.fetchone()

            if result:

                return result['id']

            else:

                return None

    except Exception as e:

        print(f"Помилка при перевірці наявності користувача: {str(e)}")

        return None

def save\_user\_data(telegram\_id, phone\_number, name, last\_name):

    try:

        with connection.cursor() as cursor:

            sql = "INSERT INTO customers (telegram\_id, phone\_number, name, last\_name) VALUES (%s, %s, %s, %s)"

            cursor.execute(sql, (telegram\_id, phone\_number, name, last\_name))

            connection.commit()

            return cursor.lastrowid

    except Exception as e:

        print(f"Помилка при збереженні даних користувача: {str(e)}")

        connection.rollback()

        return None

def delete\_user\_data(telegram\_id):

    try:

        with connection.cursor() as cursor:

            sql = "DELETE FROM customers WHERE telegram\_id = %s"

            cursor.execute(sql, (telegram\_id,))

            connection.commit()

            return True

    except Exception as e:

        print(f"Помилка при видаленні даних користувача: {str(e)}")

        connection.rollback()

        return False