## **Державний університет інформаційно-комунікаційних технологій**

**Практична робота №2**

**з дисципліни: Технології штучного інтелекту**

**Тема: Проектування чат-ботів в Python з базами даних**

Виконав:  
студент ДУІКТ  
Тертишний В.Ю.

група: ШІДМ-51

м.Київ

**Мета:** Ознайомитися з принципами проектування чат-ботів у Руthon із використанням баз даних. Навчитися створювати чат-боти, які взаємодіють із базами даних для зберігання та обробки інформації, а також забезпечити збереження та доступ до даних користувачів. Опанувати навички роботи з бібліотеками для взаємодії з базами даних (501, і т.д) та реалізувати функціонал ботів, що використовує дані для персоналізації відповідей і надання динамічного контенту.

**Код:**

import os

import telebot

from telebot import types

import sqlite3

import random

import requests

from dotenv import load\_dotenv

load\_dotenv()

TOKEN = os.getenv('TOKEN')

bot = telebot.TeleBot(TOKEN)

# Підключення до бази даних

conn = sqlite3.connect('users.db', check\_same\_thread=False)

cursor = conn.cursor()

# Створення таблиць

cursor.execute('''CREATE TABLE IF NOT EXISTS users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT,

phone TEXT,

unique\_id INTEGER,

chat\_id INTEGER

)''')

# Створення таблиці з новою структурою

cursor.execute('''CREATE TABLE IF NOT EXISTS food\_orders (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

food\_name TEXT,

quantity INTEGER,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY(user\_id) REFERENCES users(id)

)''')

conn.commit()

# Словник для зберігання станів користувачів

user\_states = {}

# Головне меню

def main\_menu():

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

markup.add('Замовити корм','Мої замовлення')

markup.add('Погода', 'Конвертер валют')

markup.add('Зареєструватись', 'Мій профіль')

return markup

# Конфігурація для API

WEATHER\_API\_KEY = os.getenv('WEATHER\_API\_KEY')

CURRENCY\_API\_URL = 'https://api.exchangerate-api.com/v4/latest/USD'

# Обробка старту та основного меню

@bot.message\_handler(commands=['start'])

def start(message):

bot.send\_message(message.chat.id, "Вітаємо у боті для замовлення корму для декоративних тварин! Виберіть дію:", reply\_markup=main\_menu())

# Реєстрація користувача

@bot.message\_handler(func=lambda message: message.text == 'Зареєструватись')

def register\_start(message):

# Check if user is already registered

cursor.execute('SELECT \* FROM users WHERE chat\_id = ?', (message.chat.id,))

existing\_user = cursor.fetchone()

if existing\_user:

bot.send\_message(message.chat.id, "Ви вже зареєстровані в системі. Використайте 'Мій профіль' для перегляду даних.")

return

user\_states[message.chat.id] = {'step': 'waiting\_name'}

bot.send\_message(message.chat.id, "Введіть ваше ім'я:")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_name')

def process\_name(message):

user\_states[message.chat.id]['name'] = message.text

user\_states[message.chat.id]['step'] = 'waiting\_phone'

bot.send\_message(message.chat.id, "Введіть ваш номер телефону:")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_phone')

def process\_phone(message):

phone = message.text

name = user\_states[message.chat.id]['name']

unique\_id = random.randint(1000, 9999)

cursor.execute('INSERT INTO users (name, phone, unique\_id, chat\_id) VALUES (?, ?, ?, ?)',

(name, phone, unique\_id, message.chat.id))

conn.commit()

bot.send\_message(message.chat.id,

f"Реєстрація успішна!\nВаш ID: {unique\_id}\nІм'я: {name}\nТелефон: {phone}",

reply\_markup=main\_menu())

del user\_states[message.chat.id]

# Перегляд профілю

@bot.message\_handler(func=lambda message: message.text == 'Мій профіль')

def show\_profile(message):

cursor.execute('SELECT \* FROM users WHERE chat\_id = ?', (message.chat.id,))

user = cursor.fetchone()

if user:

bot.send\_message(message.chat.id,

f"Ваш профіль:\nID: {user[3]}\nІм'я: {user[1]}\nТелефон: {user[2]}")

else:

bot.send\_message(message.chat.id, "Ви не зареєстровані. Використайте команду 'Зареєструватись'")

# Погода

@bot.message\_handler(func=lambda message: message.text == 'Погода')

def weather\_start(message):

user\_states[message.chat.id] = {'step': 'waiting\_city'}

bot.send\_message(message.chat.id, "Введіть назву міста:")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_city')

def process\_weather(message):

weather\_info = get\_weather(message.text)

bot.send\_message(message.chat.id, weather\_info, reply\_markup=main\_menu())

del user\_states[message.chat.id]

# Конвертер валют

@bot.message\_handler(func=lambda message: message.text == 'Конвертер валют')

def currency\_start(message):

user\_states[message.chat.id] = {'step': 'waiting\_amount'}

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

markup.add('Назад')

bot.send\_message(message.chat.id, "Введіть суму для конвертації:", reply\_markup=markup)

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_amount')

def process\_amount(message):

if message.text == 'Назад':

bot.send\_message(message.chat.id, "Операцію скасовано", reply\_markup=main\_menu())

del user\_states[message.chat.id]

return

try:

amount = float(message.text)

user\_states[message.chat.id]['amount'] = amount

user\_states[message.chat.id]['step'] = 'waiting\_from\_currency'

bot.send\_message(message.chat.id, "Введіть вихідну валюту (наприклад, USD):")

except ValueError:

bot.send\_message(message.chat.id, "Будь ласка, введіть коректне число")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_from\_currency')

def process\_from\_currency(message):

if message.text == 'Назад':

bot.send\_message(message.chat.id, "Операцію скасовано", reply\_markup=main\_menu())

del user\_states[message.chat.id]

return

user\_states[message.chat.id]['from\_currency'] = message.text.upper()

user\_states[message.chat.id]['step'] = 'waiting\_to\_currency'

bot.send\_message(message.chat.id, "Введіть цільову валюту (наприклад, EUR):")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_to\_currency')

def process\_to\_currency(message):

if message.text == 'Назад':

bot.send\_message(message.chat.id, "Операцію скасовано", reply\_markup=main\_menu())

del user\_states[message.chat.id]

return

state = user\_states[message.chat.id]

result = convert\_currency(state['amount'], state['from\_currency'], message.text.upper())

bot.send\_message(message.chat.id, result, reply\_markup=main\_menu())

del user\_states[message.chat.id]

# Функція для запиту погоди

def get\_weather(city):

url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER\_API\_KEY}&units=metric"

response = requests.get(url)

print(response)

if response.status\_code == 200:

data = response.json()

return f"Погода в {city}:\nТемпература: {data['main']['temp']}°C\nОпис: {data['weather'][0]['description']}"

elif response.status\_code == 429:

return "Перевищено ліміт запитів. Будь ласка, спробуйте пізніше."

else:

return "Місто не знайдено. Спробуйте ще раз."

# Функція для конвертації валют

def convert\_currency(amount, from\_currency, to\_currency):

response = requests.get(CURRENCY\_API\_URL)

if response.status\_code == 200:

rates = response.json()['rates']

if from\_currency in rates and to\_currency in rates:

converted = amount / rates[from\_currency] \* rates[to\_currency]

return f"{amount} {from\_currency} = {converted:.2f} {to\_currency}"

else:

return "Одна з валют не підтримується."

else:

return "Помилка отримання даних. Спробуйте пізніше."

@bot.message\_handler(func=lambda message: message.text == 'Переглянути корм')

def view\_food(message):

bot.send\_message(message.chat.id, "Ось список доступного корму:\n1. Корм для гризунів\n2. Корм для рибок\n3. Корм для птахів", reply\_markup=main\_menu())

# Замовлення корму

@bot.message\_handler(func=lambda message: message.text == 'Замовити корм')

def order\_food(message):

# Перевірка чи користувач зареєстрований

cursor.execute('SELECT id FROM users WHERE chat\_id = ?', (message.chat.id,))

user = cursor.fetchone()

if not user:

bot.send\_message(message.chat.id, "Спочатку потрібно зареєструватися!", reply\_markup=main\_menu())

return

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

markup.add('Корм для гризунів', 'Корм для рибок', 'Корм для птахів')

markup.add('Назад')

bot.send\_message(message.chat.id, "Виберіть тип корму:", reply\_markup=markup)

user\_states[message.chat.id] = {'step': 'waiting\_food\_type'}

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_food\_type')

def handle\_food\_type(message):

if message.text == 'Назад':

bot.send\_message(message.chat.id, "Повернення до головного меню", reply\_markup=main\_menu())

del user\_states[message.chat.id]

return

if message.text in ['Корм для гризунів', 'Корм для рибок', 'Корм для птахів']:

user\_states[message.chat.id] = {'step': 'waiting\_quantity', 'food\_name': message.text}

bot.send\_message(message.chat.id, "Введіть кількість упаковок (від 1 до 10):")

else:

bot.send\_message(message.chat.id, "Будь ласка, виберіть корм із запропонованих варіантів")

@bot.message\_handler(func=lambda message: message.chat.id in user\_states and user\_states[message.chat.id]['step'] == 'waiting\_quantity')

def handle\_food\_order(message):

try:

quantity = int(message.text)

if not 1 <= quantity <= 10:

raise ValueError("Кількість повинна бути від 1 до 10")

# Отримуємо user\_id

cursor.execute('SELECT id FROM users WHERE chat\_id = ?', (message.chat.id,))

user = cursor.fetchone()

# Зберігаємо замовлення в базі даних

cursor.execute('INSERT INTO food\_orders (user\_id, food\_name, quantity) VALUES (?, ?, ?)',

(user[0], user\_states[message.chat.id]['food\_name'], quantity))

conn.commit()

bot.send\_message(

message.chat.id,

f"Ваше замовлення на {quantity} упаковок '{user\_states[message.chat.id]['food\_name']}' успішно збережено!",

reply\_markup=main\_menu()

)

del user\_states[message.chat.id]

except ValueError as e:

if str(e) == "Кількість повинна бути від 1 до 10":

bot.send\_message(message.chat.id, str(e))

else:

bot.send\_message(message.chat.id, "Будь ласка, введіть коректне число")

except Exception as e:

bot.send\_message(message.chat.id, "Сталася помилка при оформленні замовлення. Спробуйте ще раз.", reply\_markup=main\_menu())

del user\_states[message.chat.id]

# Консультація

@bot.message\_handler(func=lambda message: message.text == 'Консультація щодо корму')

def consultation\_handler(message):

bot.send\_message(message.chat.id, "Напишіть ваше питання, і ми зв’яжемося з вами якнайшвидше.", reply\_markup=main\_menu())

# Додаємо новий обробник для перегляду замовлень

@bot.message\_handler(func=lambda message: message.text == 'Мої замовлення')

def show\_orders(message):

# Перевіряємо чи користувач зареєстрований

cursor.execute('SELECT id FROM users WHERE chat\_id = ?', (message.chat.id,))

user = cursor.fetchone()

if not user:

bot.send\_message(message.chat.id, "Спочатку потрібно зареєструватися!", reply\_markup=main\_menu())

return

# Отримуємо всі замовлення користувача

cursor.execute('''

SELECT food\_name, quantity, datetime(created\_at, 'localtime')

FROM food\_orders

WHERE user\_id = ?

ORDER BY created\_at DESC

LIMIT 10

''', (user[0],))

orders = cursor.fetchall()

if not orders:

bot.send\_message(message.chat.id, "У вас поки немає замовлень.", reply\_markup=main\_menu())

return

# Формуємо повідомлення зі списком замовлень

message\_text = "Ваші останні замовлення:\n\n"

for i, (food\_name, quantity, order\_date) in enumerate(orders, 1):

message\_text += f"{i}. {food\_name} - {quantity} шт.\n Дата: {order\_date}\n\n"

bot.send\_message(message.chat.id, message\_text, reply\_markup=main\_menu())

# Обробка невідомих повідомлень

@bot.message\_handler(func=lambda message: True)

def echo\_all(message):

if message.text == 'Назад':

bot.send\_message(message.chat.id, "Повернення до головного меню", reply\_markup=main\_menu())

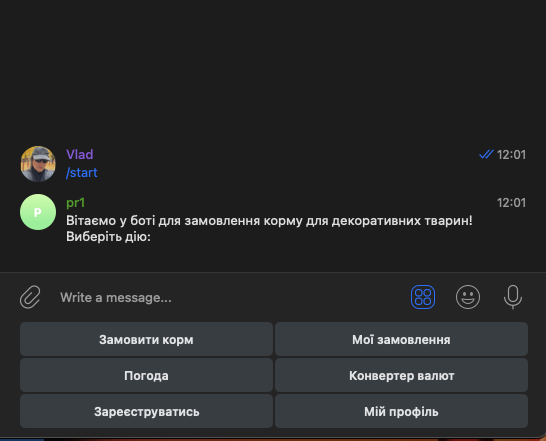
if message.chat.id in user\_states:

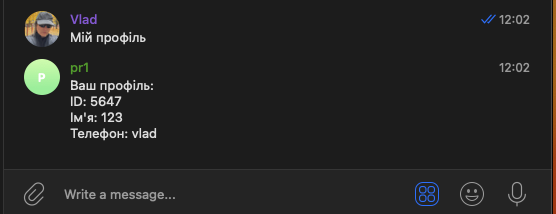
del user\_states[message.chat.id]

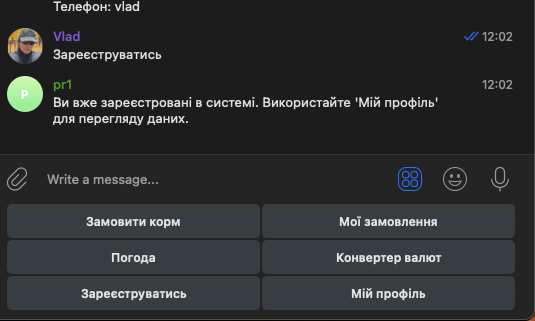
else:

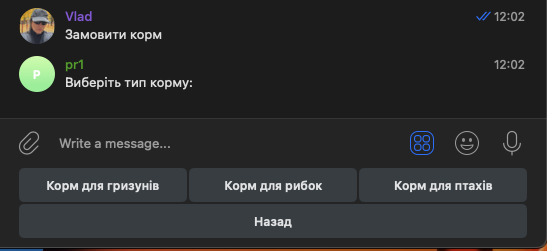
bot.send\_message(message.chat.id, "Не розумію команду. Використовуйте меню для навігації.", reply\_markup=main\_menu())

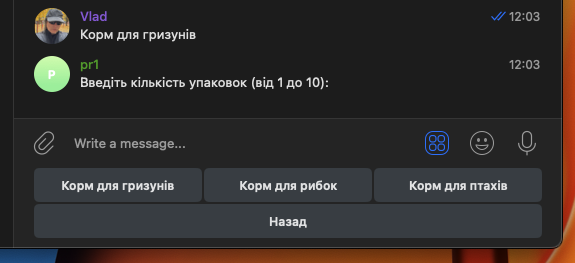
bot.polling(none\_stop=True)

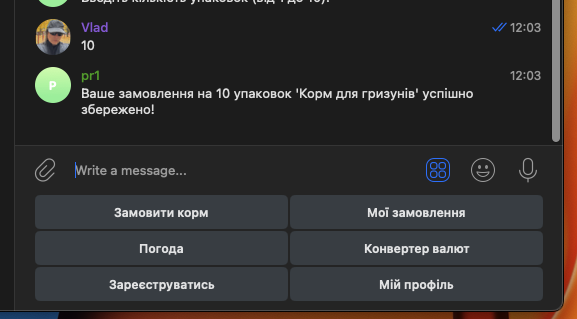
**Робота коду:**  


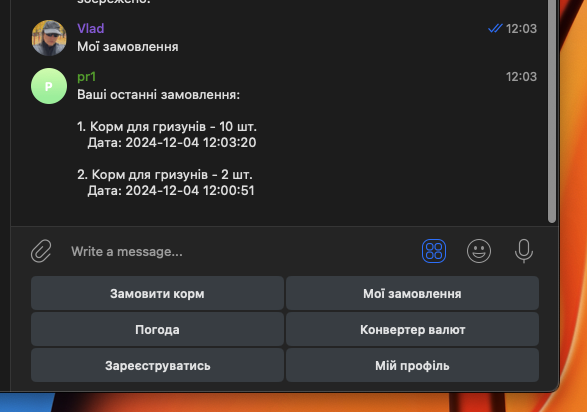












**Висновки**

У ході виконання поставленої мети було здійснено ознайомлення з принципами проектування чат-ботів у Python із використанням баз даних. Було розглянуто основні етапи створення чат-ботів, які взаємодіють із базами даних, що забезпечує можливість збереження та обробки інформації про користувачів. Зокрема, розроблено функціонал для реєстрації користувачів, збереження їхніх даних у базі, а також динамічної персоналізації відповідей на основі цих даних. У процесі роботи було опановано використання бібліотек для роботи з базами даних, таких як SQLite, а також інтеграцію API для реалізації динамічного контенту, включаючи прогнози погоди та конвертацію валют. Реалізація цих функцій дозволила створити багатофункціональний чат-бот із широкими можливостями взаємодії, що відповідає сучасним вимогам автоматизації та персоналізації сервісів.