Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления» Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования» Отчёт по лабораторной работе №6

«Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Выполнил:
студент группы ИУ5-35Б
Александров Георгий

Проверил: преподаватель каф. ИУ5
Нардид Анатолий Николаевич

Москва, 2025 г.

Описание задания

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Текст программы

bot2.py

```
import logging
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filters, CallbackContext, ConversationHandler
import requests
import os

logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)

# ⚠️ ТОКЕНЫ ВСТАВЛЕНЫ НАПРЯМУЮ (УДАЛИТЕ ПОСЛЕ ТЕСТА!)
TELEGRAM_TOKEN = "8588242112:AAGFNW1A7u9Ghj0FA7P7jgMq_uDYEhH5ar4"
WEATHER_API_KEY = "44fa02cd58e6789423b0a4ad4589acfe"

# Определяем состояния конечного автомата
START, ENTER_CITY, SHOW_WEATHER = range(3)

async def start(update: Update, context: CallbackContext) -> int:
    """Начало диалога, состояние START."""
    await update.message.reply_text(
        "Привет! Я погодный бот.\n"
        "Введите название города:"
    )
    return ENTER_CITY   # Переход в состояние ожидания города

async def get_weather(update: Update, context: CallbackContext) -> int:
    """Получение погоды, состояние ENTER_CITY -> SHOW_WEATHER."""
    city = update.message.text
    url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER_API_KEY}&units=metric&lang=ru"

    try:
        response = requests.get(url)

        if response.status_code == 200:
            data = response.json()
            temp = data['main']['temp']
            feels_like = data['main']['feels_like']
            humidity = data['main']['humidity']
            description = data['weather'][0]['description']
            city_name = data['name']
```

```
            weather_text = (
                f"Погода в {city_name}:\n"
                f"Температура: {temp}°C\n"
                f"Ощущается как: {feels_like}°C\n"
                f"Влажность: {humidity}%\n"
                f"Описание: {description}"
            )

            await update.message.reply_text(weather_text)
            await update.message.reply_text(
                "Введите новый город для погоды "
                "или /start для перезапуска."
            )
            return SHOW_WEATHER   # Переход в состояние после показа погоды
        else:
            await update.message.reply_text(
                "Город не найден. Попробуйте ещё раз:"
            )
            return ENTER_CITY   # Остаемся в состоянии ожидания города

    except Exception as e:
        logging.error(f"Ошибка: {e}")
        await update.message.reply_text(
            "Произошла ошибка. Попробуйте позже.\n"
            "Введите город:"
        )
        return ENTER_CITY

async def handle_weather_shown(update: Update, context: CallbackContext) -> int:
    """Обработка ввода после показа погоды, состояние SHOW_WEATHER."""
    # Любое сообщение в этом состоянии считаем новым городом
    return await get_weather(update, context)

async def cancel(update: Update, context: CallbackContext) -> int:
    """Завершение диалога."""
    await update.message.reply_text("До свидания!")
    return ConversationHandler.END

def main() -> None:
    """Запуск бота с явным конечным автоматом."""
    application = Application.builder().token(TELEGRAM_TOKEN).build()

    # Создаем ConversationHandler для управления состояниями
    conv_handler = ConversationHandler(
        entry_points=[CommandHandler('start', start)],
        states={
            START: [
```

```
            MessageHandler(filters.TEXT & ~filters.COMMAND, start)
        ],
        ENTER_CITY: [
            MessageHandler(filters.TEXT & ~filters.COMMAND, get_weather)
        ],
        SHOW_WEATHER: [
            MessageHandler(filters.TEXT & ~filters.COMMAND, handle_weather_shown),
            CommandHandler('start', start)
        ]
    },
    fallbacks=[CommandHandler('cancel', cancel)],
    # Разрешаем повторный вход в состояния
    allow_reentry=True
)

application.add_handler(conv_handler)
application.run_polling()

if __name__ == '__main__':
    main()
```

Листинг кода