

**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 №5  
**«Разработка простого бота 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 простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

## Текст программы

### bot1.py

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

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

TOKEN = os.getenv("TELEGRAM_BOT_TOKEN") or "8549908539:AAHxzTEb2EH-9WGstE556Lvf_kuzmTKo-9o"
WEATHER_API_KEY = os.getenv("WEATHER_API_KEY") or "44fa02cd58e6789423b0a4ad4589acfe"

MAIN_KEYBOARD = [
    ["🌤 Погода сейчас", "📅 Прогноз на завтра"],
    ["📍 Погода по местоположению", "⚙️ Помощь"]
]

async def start(update: Update, context: CallbackContext):
    user = update.effective_user
    reply_markup = ReplyKeyboardMarkup(MAIN_KEYBOARD, resize_keyboard=True)

    await update.message.reply_text(
        f"Привет, {user.first_name}! 🌞\n"
        f"Я бот для прогноза погоды.\n"
        f"Выберите действие на клавиатуре ниже:",
        reply_markup=reply_markup
    )

async def help_command(update: Update, context: CallbackContext):
    help_text = """
📌 *Доступные команды:*
/start - Запустить бота
/help - Получить справку
/pogoda [город] - Узнать погоду
    """

    # Используйте кнопки
    buttons = [
        {"text": "🌤 Погода сейчас - текущая погода", "callback_data": "/pogoda current"}, 
        {"text": "📅 Прогноз на завтра - прогноз", "callback_data": "/forecast tomorrow"}, 
        {"text": "📍 Погода по местоположению - отправьте геолокацию", "callback_data": "/location current"}, 
        {"text": "⚙️ Помощь - эта справка", "callback_data": "/help"}
    ]
    reply_markup = ReplyKeyboardMarkup(buttons, resize_keyboard=True)

    update.message.reply_text(help_text, reply_markup=reply_markup)
```

```

*Пример:* /pogoda Москва
"""

await update.message.reply_text(help_text, parse_mode='Markdown')

async def get_weather(city: str):
    try:
        url_current =
f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={WEATHER_API_KEY}&units=metric
&lang=ru"
        response = requests.get(url_current)
        data = response.json()

        if data["cod"] != 200:
            return None

        url_forecast =
f"http://api.openweathermap.org/data/2.5/forecast?q={city}&appid={WEATHER_API_KEY}&units=metric
&lang=ru&cnt=8"
        forecast_response = requests.get(url_forecast)
        forecast_data = forecast_response.json()

        return {
            'current': data,
            'forecast': forecast_data
        }
    except Exception as e:
        logger.error(f"Error getting weather: {e}")
        return None

async def weather_now(update: Update, context: CallbackContext):
    if context.args:
        city = ''.join(context.args)
    else:
        city = update.message.text.replace('☀️ Погода сейчас', "").strip()
        if not city:
            await update.message.reply_text("Пожалуйста, укажите город после команды или отправьте
его название.")
            return

    weather_data = await get_weather(city)

    if weather_data and weather_data['current']:
        data = weather_data['current']
        weather_info = (
            f"☀️ *Погода в {data['name']}*\n\n"
            f"🌡️ Температура: *{data['main']['temp']}°C*\n"
            f"💨 Ощущается как: *{data['main']['feels_like']}°C*\n"
        )

```

```

f" 🌬 Ветер: *{data['wind']['speed']} м/с*\n"
f" 💧 Влажность: *{data['main']['humidity']}%*\n"
f" ☁ Облачность: *{data['clouds']['all']}%*\n"
f" 📋 {data['weather'][0]['description'].capitalize()}""
)
await update.message.reply_text(weather_info, parse_mode='Markdown')
else:
    await update.message.reply_text("Не удалось получить данные о погоде. Проверьте название города.")

async def weather_tomorrow(update: Update, context: CallbackContext):
    if context.args:
        city = ''.join(context.args)
    else:
        city = update.message.text.replace('🗓 Прогноз на завтра', "").strip()
        if not city:
            await update.message.reply_text("Пожалуйста, укажите город после команды или отправьте его название.")
    return

    weather_data = await get_weather(city)

    if weather_data and weather_data['forecast']:
        tomorrow_data = weather_data['forecast']['list'][4]

        forecast_info = (
            f" 🗓 *Прогноз на завтра для {weather_data['current']['name']}*\n\n"
            f" 🌡 Температура: *{tomorrow_data['main']['temp']}°C*\n"
            f" 🌫 Ощущается как: *{tomorrow_data['main']['feels_like']}°C*\n"
            f" 🌬 Ветер: *{tomorrow_data['wind']['speed']} м/с*\n"
            f" 💧 Влажность: *{tomorrow_data['main']['humidity']}%*\n"
            f" 📋 {tomorrow_data['weather'][0]['description'].capitalize()}""
        )
        await update.message.reply_text(forecast_info, parse_mode='Markdown')
    else:
        await update.message.reply_text("Не удалось получить прогноз. Проверьте название города.")

async def handle_location(update: Update, context: CallbackContext):
    if update.message.location:
        lat = update.message.location.latitude
        lon = update.message.location.longitude

        try:
            url =
            f"http://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={WEATHER_API_KEY}&units=metric&lang=ru"
            response = requests.get(url)

```

```

data = response.json()

if data["cod"] == 200:
    weather_info = (
        f"📍 *Погода по вашему местоположению*\n\n"
        f"🏙️ Город: *{data['name']}*\n"
        f"🌡️ Температура: *{data['main']['temp']}°C*\n"
        f"💨 Ощущается как: *{data['main']['feels_like']}°C*\n"
        f"💨 Ветер: *{data['wind']['speed']} м/с*\n"
        f"📝 {data['weather'][0]['description'].capitalize()}"
    )
    await update.message.reply_text(weather_info, parse_mode='Markdown')
else:
    await update.message.reply_text("Не удалось определить погоду для этого
местоположения.")
except Exception as e:
    logger.error(f"Error with location: {e}")
    await update.message.reply_text("Произошла ошибка при обработке местоположения.")

async def handle_message(update: Update, context: CallbackContext):
    text = update.message.text

    if '🌤️ Погода сейчас' in text:
        await weather_now(update, context)
    elif '📅 Прогноз на завтра' in text:
        await weather_tomorrow(update, context)
    elif '📍 Погода по местоположению' in text:
        await update.message.reply_text(
            "Пожалуйста, отправьте ваше местоположение, нажав на скрепку 📌 и выбрав
'Mестоположение'"
        )
    elif '⚙️ Помощь' in text:
        await help_command(update, context)
    elif text.strip():
        await weather_now(update, context)

def main():
    application = Application.builder().token(TOKEN).build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("help", help_command))
    application.add_handler(CommandHandler("pogoda", weather_now))
    application.add_handler(CommandHandler("tomorrow", weather_tomorrow))

    application.add_handler(MessageHandler(filters.LOCATION, handle_location))

```

```

application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))

print("Бот запущен...")
application.run_polling(allowed_updates=Update.ALL_TYPES)

if __name__ == '__main__':
    main()

```

## Листинг кода

### Запуск

```

○ (venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\лабы\lab_python_fp> python bot.py
Бот запущен...
2025-12-25 03:01:13,490 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot8549908539:AAHxzTEb2EH-9WGstE55
6Lvf_kuzmTKo-9o/getMe "HTTP/1.1 200 OK"
2025-12-25 03:01:13,550 - httpx - INFO - HTTP Request: POST https://api.telegram.org/bot8549908539:AAHxzTEb2EH-9WGstE55
6Lvf_kuzmTKo-9o/deleteWebhook "HTTP/1.1 200 OK"

```

### Работа



