[Муниципальное образовательное учреждение средняя общеобразовательная школа «Образовательный комплекс № 3 им. Н.П. Гусева»](https://school42.edu.yar.ru/)

**Практико-ориентированный проект**

**Разработка погодного бота для мессенджера “Telegram”**

Заводчиков Георгий,

ученик 9 «Б» класса

руководитель

Заводчиков М.А., к.ф.-м.н., доцент кафедры алгебры и математической логики ЯрГУ им. П.Г. Демидова

Ярославль

2025

## 

**Содержание**

[Введение 3](#_Toc12633)

[1. Схема работы Telegram-бота 4](#_Toc6584)

[2. Организация работы с пользователем. Интерфейс 4](#_Toc29254)

[3. Обработка запросов с помощью погодного API 6](#_Toc14094)

[Заключение 8](#_Toc13072)

[Список использованной литературы 9](#_Toc32490)

[Приложение 1 Листинг программы «Run.py» 10](#_Toc16109)

[Приложение 2 Листинг программы «Keyboard.py» 11](#_Toc10908)

[Приложение 3 Листинг программы «Weather\_handlers.py» 12](#_Toc24857)

[Приложение 4 Листинг программы «User\_processor.py» 15](#_Toc6901)

## Введение

Бот в Telegram ― это программа, которая автоматизирует определённые задачи и взаимодействие с пользователями в мессенджере Telegram. Это специальный аккаунт, который даёт пользователям возможность выполнять разные задачи посредством мессенджера. Информационные боты предоставляют пользователю информацию: погоду, новости, котировки, расписание, переводы. Медиа-боты предоставляют доступ к медиа-контенту, такому как фотографии, видео, аудиозаписи и другие мультимедийные материалы.

**В Telegram достаточно много ботов предоставляющих информацию о погоде в различных регионах, однако нет бота с удобным интерфейсом не привязанного к конкретному региону и времени выдачи сообщений.**

***Объект исследования*** - Возможности языка Python

***Предмет исследования*** - Разработка **Telegram-бота на языке программирования** Python**.**

***Цель работы:*** Создать **Telegram-бота, который будет сообщать пользователю погоду в настоящий момент времени или прогноз.**

***Задачи проектной работы***:

1. Изучить библиотеку Python для написания **Telegram-бота.**
2. Изучить библиотеку Python для запросов к API.
3. Написать систему способную осуществить связь: пользователь - сервер - API.

## Схема работы Telegram-бота

Приведём основные понятия, используемые при разработке **Telegram-бота для для сообщения прогноза погоды.**

Telegram-бот - это специальная программа внутри мессенджера, которая функционирует по заданному сценарию. Пользователь управляет её работой с помощью кнопок или текстовых команд из меню.

API (от англ. application programming interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

Request - (от англ. запрос) запрос к чему либо, с какими либо командами, в конкретно нашем случае, я посылая запрос на получения файла с погодой.

Рассмотрим схему разработанного **Telegram-бота**

1. Пользователь запускает бота.
2. Бот запрашивает позицию.
3. Пользователь отправляет геопозицию.
4. Бот обрабатывает позицию, сохраняя её в базе данных.
5. Бот спрашивает у пользователя формат: прогноз или погода в настоящее время.
6. Если был выбран прогноз, бот запрашивает у пользователя день, на который должен быть выведен прогноз.
7. Программа запрашивает в API погоду в нужном формате в зависимости от выбора пользователя.
8. Бот присылает погоду пользователю в нужном формате.

## 2. Организация работы с пользователем. Интерфейс

Для работы с клиентом используется с**амописный** Telegram-бот. Когда пользователь отправляет сообщение (рис. 1-6), оно попадает в “поток” программы.

Над каждой функции, которая должна работать с сообщениями стоит специальный фильтр, который выбирает на какие сообщения реагировать. Так, например, когда пользователь посылает свою геопозицию (рис. 2), это сообщение отсеивается на каждой функции, кроме нужной, геопозиция записывается в файл с форматом json.

Далее после получения данных о желаемом ответе - времени прогноза, программа передает request, содержащий геопозицию пользователя, к OpenweatherApi, в ответ возвращается json-файл, содержащий информацию о погоде (температура, скорость ветра, облачность и. т. д.). Программа обрабатывает этот файл, выбирая в ответ пользователя, всё самое полезное, потом генерирует само сообщение и отправляет его клиенту. Листинги соответствующих программ приведены в приложениях.

|  |  |
| --- | --- |
| start_bot | bot_geometka_otpravlena |
| *Рис. 1 Запуск бота* | *Рис. 2 Отправка геометки и выбор режима* |

## 3. Обработка запросов с помощью погодного API

Рассмотрим, как программа отправляет запрос. В данном случае запрос содержит немного параметров: широта, долгота, api-key. Широта и долгота извлекается из json-файла с пользовательскими геопозициями.

Аpi-key - это ключ для запроса к API без него возникнет ошибка 401 - «некорректный api-key». Ключ можно получить в личном кабинете сайта API.

Есть два типа запросов, один - погода сейчас (рис. 3), другой - прогноз (рис. 4 - 6). В обоих случаях при получении ответа, программа извлекает из него нужные данные - это температура в кельвинах, скорость ветра в м/с и влажность в %, переводит величины и составляет красивое сообщение. Листинги соответствующих программ приведены в приложениях.

|  |  |
| --- | --- |
| botnow | botprognose |
| *Рис. 3 Режим «Погода сейчас»* | *Рис. 4 Режим «Прогноз на завтра»* |
| bot_poslezavtra | bot_chetverg |
| *Рис. 5 Режим «Прогноз на послезавтра»* | *Рис. 6 Режим «Прогноз на третий день»* |

## Заключение

В ходе работы над проектом я проанализировал существующие подходы к программированию Telegram-ботов на языке Python, а также к получению данных из api-запросов. Улучшил свои навыки программирования на языке Python. В итоге у меня получилась полностью готовая система для сообщения погоды пользователю по средствам Telegram.

Исходный код проекта расположен по адресу: https://github.com/Oedada/BotTelegram

## Список использованной литературы

1. Youtube: Курс по разработке Telegram-ботов. Web: <https://www.youtube.com/watch?v=qRyshRUA0xM&list=PLhxfpj6Vm4rpxdSpvKI00rso8Trhq-_Hg> (Дата обращения: 12.11.2024)
2. Docs Aiohttp: Официальная документация библиотеки для асинхронных запросов. Web: <https://docs.aiohttp.org/en/stable/> (Дата обращения: 24.11.2024)
3. Примеры работы модуля json для python. Web: <https://python-scripts.com/json> (Дата обращения: 29.12.2024)
4. Docs Python: Официальная документация библиотеки для работы с json-файлами. Web: <https://docs.python.org/3/library/json.html>
5. Docs: Официальная документация для написания Telegram-бота. Web: <https://docs.aiogram.dev/en/stable/>

## Приложение 1 Листинг программы «Run.py»

import asyncio

import logging

from aiogram import Bot, Dispatcher

from app.weather\_handlers import router as weather\_router

from config import TOKEN

bot = Bot(token=TOKEN)

dp = Dispatcher()

async def main():

    dp.include\_router(weather\_router)

    await dp.start\_polling(bot)

if \_\_name\_\_ == '\_\_main\_\_':

    logging.basicConfig(level=logging.INFO)

    try:

        asyncio.run(main())

    except KeyboardInterrupt:

        print("exit")

## Приложение 2 Листинг программы «Keyboard.py»

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton, InlineKeyboardButton, InlineKeyboardMarkup

from aiogram.utils.keyboard import InlineKeyboardBuilder

from datetime import datetime, timedelta

twoday = (datetime.now() + timedelta(2)).weekday()

threeday = (datetime.now() + timedelta(3)).weekday()

weekdays = {0: "Понедельник", 1:"Вторник", 2: "Среда", 3: "Четверг", 4: "Пятница", 5: "Суббота", 6: "Воскресение"}

change\_pos = InlineKeyboardMarkup(inline\_keyboard=[

    [InlineKeyboardButton(text='Погода сейчас',  callback\_data='weather\_now')],

    [InlineKeyboardButton(text='Прогноз',  callback\_data='wheather\_forecast')]

])

select\_day = InlineKeyboardMarkup(inline\_keyboard=[

    [InlineKeyboardButton(text='Завтра',  callback\_data='tomorrow')],

    [InlineKeyboardButton(text='Послезавтра',  callback\_data='after\_tomorrow')],

    [InlineKeyboardButton(text=weekdays[twoday],  callback\_data='aa\_tomorrow')],

    [InlineKeyboardButton(text=weekdays[threeday],  callback\_data='aaa\_tomorrow')]

])

## Приложение 3 Листинг программы «Weather\_handlers.py»

from aiogram import F, Router

from aiogram.types import Message, CallbackQuery, Location

from aiogram.filters import CommandStart

import aiohttp

from config import API\_KEY

from app.user\_processor import users\_db

from datetime import datetime, timedelta

twoday = (datetime.now() + timedelta(2)).weekday()

threeday = (datetime.now() + timedelta(3)).weekday()

weekdays = {0: "Понедельник", 1:"Вторник", 2: "Среда", 3: "Четверг", 4: "Пятница", 5: "Суббота", 6: "Воскресение"}

import app.keyboard as kb

db = users\_db()

router = Router()

@router.message(CommandStart())

async def cmd\_start(message: Message):

    await message.answer("Здраствуйте отправте вашу геопозицию, для определения погоды.")

@router.message(F.location)

async def change\_city(message: Message):

    '''Once again sends user a message with list of cities'''

    db.set\_user\_param(message.from\_user.id, 'latitude', "%.20f" % message.location.latitude)

    db.set\_user\_param(message.from\_user.id, 'longitude', "%.20f" % message.location.longitude)

    db.save()

    await message.answer('Ваша позиция обработана, выберите, какую погоду вы хотите получить.', reply\_markup=kb.change\_pos)

@router.callback\_query(F.data == "weather\_now")

async def weather\_now(callback: CallbackQuery):

    data = db.get\_user\_data(str(callback.from\_user.id))

    lon = data['longitude']

    lat = data['latitude']

    weather = await get\_weather(lon, lat, time=0)

    return callback.message.answer(weather)

@router.callback\_query(F.data == "wheather\_forecast")

async def wheather\_forecast(callback: CallbackQuery):

    return callback.message.answer("Выберите день", reply\_markup=kb.select\_day)

@router.callback\_query(F.data == "tomorrow")

async def wheather\_tommorow(callback: CallbackQuery):

    data = db.get\_user\_data(str(callback.from\_user.id))

    lon = data["longitude"]

    lat = data['latitude']

    return callback.message.answer("Завтра: \n" + (await get\_weather(lon, lat, 1)))

@router.callback\_query(F.data == "after\_tomorrow")

async def wheather\_tommorow(callback: CallbackQuery):

    data = db.get\_user\_data(str(callback.from\_user.id))

    lon = data["longitude"]

    lat = data['latitude']

    return callback.message.answer("Послезавтра: \n" + (await get\_weather(lon, lat, 2)))

@router.callback\_query(F.data == "aa\_tomorrow")

async def wheather\_tommorow(callback: CallbackQuery):

    data = db.get\_user\_data(str(callback.from\_user.id))

    lon = data["longitude"]

    lat = data['latitude']

    return callback.message.answer(weekdays[twoday] + ":\n" + (await get\_weather(lon, lat, 3)))

@router.callback\_query(F.data == "aaa\_tomorrow")

async def wheather\_tommorow(callback: CallbackQuery):

    data = db.get\_user\_data(str(callback.from\_user.id))

    lon = data["longitude"]

    lat = data['latitude']

    return callback.message.answer(weekdays[threeday] + ":\n" + (await get\_weather(lon, lat, 4)))

async def get\_weather(longitude, latitude, time) -> str:

    '''Return string contain information about weather'''

    if time == 0:

        params = {

            'lat': latitude,

            'lon': longitude,

            'appid': API\_KEY

        }

        weather = await get\_api\_response('https://api.openweathermap.org/data/2.5/weather/', params)

        temp = str(round(weather['main']['temp'] - 273.15, 1)) + ' °С'

        cloud = ('пасмурно' if weather['clouds']['all'] >= 80 else 'ясно')

        wind = str(round(weather['wind']['speed'], 1)) + ' м/с'

        answer = f'Сейчас {temp}, {cloud}, скорость ветра {wind}'

        return answer

    else:

        params = {

            'lat': latitude,

            'lon': longitude,

            'appid': API\_KEY

        }

        weathers = await get\_api\_response('https://pro.openweathermap.org/data/2.5/forecast', params)

        weathers = weathers["list"]

        true\_weather = []

        for wheather in weathers:

            weather\_time = datetime.utcfromtimestamp(wheather["dt"]).strftime('%Y-%m-%d %H:%M:%S')

            if weather\_time.split(" ")[0] != datetime.now().strftime('%Y-%m-%d'):

                true\_weather.append(wheather)

        weathers = true\_weather

        morning = true\_weather[3+(time-1)]

        noon = true\_weather[5+(time-1)  ]

        evening = true\_weather[7+(time-1)]

        night =  true\_weather[8+(time-1)]

        temp = str(round(morning['main']['temp'] - 273.15, 1)) + ' °С'

        cloud = ('пасмурно' if morning['clouds']['all'] >= 80 else 'ясно')

        wind = str(round(morning['wind']['speed'], 1)) + ' м/с'

        morning\_string = f'{temp}, {cloud}, скорость ветра {wind}'

        temp = str(round(noon['main']['temp'] - 273.15, 1)) + ' °С'

        cloud = ('пасмурно' if noon['clouds']['all'] >= 80 else 'ясно')

        wind = str(round(noon['wind']['speed'], 1)) + ' м/с'

        noon\_string = f'{temp}, {cloud}, скорость ветра {wind}'

        temp = str(round(evening['main']['temp'] - 273.15, 1)) + ' °С'

        cloud = ('пасмурно' if evening['clouds']['all'] >= 80 else 'ясно')

        wind = str(round(evening['wind']['speed'], 1)) + ' м/с'

        evening\_string = f'{temp}, {cloud}, скорость ветра {wind}'

        temp = str(round(night['main']['temp'] - 273.15, 1)) + ' °С'

        cloud = ('пасмурно' if night['clouds']['all'] >= 80 else 'ясно')

        wind = str(round(night['wind']['speed'], 1)) + ' м/с'

        night\_string = f'{temp}, {cloud}, скорость ветра {wind}'

        answer = "Утром: " + morning\_string + "\n" + "Днём: " + noon\_string + "\n" + "Вечером: " + evening\_string + "\n" + "Ночь: " + night\_string

        return answer

async def get\_api\_response(url, params):

    '''Get all weather information from <https://api.openweathermap.org/data/2.5/weather>.

    Return dictionairy if it is successful else error code.'''

    async with aiohttp.ClientSession() as session:

        async with session.get(url, params=params) as response:

            if response.status == 200:

                content = await response.json()

                return content

            else:

                print(f"failed to get data from {url} with error code: {response.status}")

                return None

## Приложение 4 Листинг программы «User\_processor.py»

import json

class users\_db:

    def \_\_init\_\_(self):

        with open('app/users.json', 'r') as file:

            self.data = json.load(file)

            file.close()

    def get\_data(self) -> dict:

        return self.data

    def set\_user\_param(self, id: str, param: str, value) -> None:

        if id in self.data.keys():

            self.data[id][param] = value

        else:

            self.data[id] = {param: value}

    def get\_user\_data(self, id) -> dict:

        return self.data[id]

    def save(self) -> None:

        with open('app/users.json', 'w') as file:

            json.dump(self.data, file)

            file.close()

**ПАСПОРТ ПРОЕКТА**

1. ФИ обучающегося Заводчиков Георгий
2. Класс 9 «Б»
3. ФИО руководителя проекта Заводчиков Михаил Александрович
4. Тема проекта, название проекта (предметная область): Создание Telegram-бота.
5. Цель проекта: Создать **Telegram-бота, который будет сообщать пользователю погоду в настоящий момент времени или прогноз.**
6. Задачи проекта
7. Изучить библиотеку Python для написания **Telegram-бота.**
8. Изучить библиотеку Python для запросов к API.
9. Написать систему способную осуществить связь: пользователь - сервер - API.
10. Актуальность выбранной темы: **В Telegram достаточно много ботов предоставляющих информацию о погоде в различных регионах, однако нет бота с удобным интерфейсом не привязанного к конкретному региону.**
11. Тип проекта: практико-ориентированный
12. Этапы работы над проектом:

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Что делал | Затрачено  времени | Вопрос / затруднение |
| Сентябрь 2024 | Выбор темы проекта | 2 недели | Разработка **Telegram-ботов в настоящее время очень востребована, поэтому изучение этой сферы программирования для меня интересно.** |
| Октябрь 2024 | Выбор необходимых составляющих и инструментов. | 1 неделя |  |
| Ноябрь-декабрь 2024 | Изучение библиотек | 1 месяц | В решении возникающих проблемм мне помог Stackoverflow: **сайт вопросов и ответов для профессиональных разработчиков программного обеспечения, энтузиастов программирования и системных администраторов**. Web: 1)https://ru.stackoverflow.com/questions/1514493/геолокация-пользователя-с-помощью-aiogram-python  2)https://ru.stackoverflow.com/questions/1584355/Не-работает-обработчик-событий-router-callback-query 3)https://ru.stackoverflow.com/questions/1557132/Ошибка-при-вызове-inline-кнопики-aiogram-3 |
| Январь-февраль 2025 | написание программы | 1 месяц |  |

1. Методы и средства реализации проекта: теоретическое (поиск информации в сети Internet), эмпирические (подбор парамеров программы для настройки режимов работы), практические (написание программы)
2. Список источников информации
3. Курс по разработке Telegram-ботов. URL: <https://www.youtube.com/watch?v=qRyshRUA0xM&list=PLhxfpj6Vm4rpxdSpvKI00rso8Trhq-_Hg> (Дата обращения: 12.01.2025)
4. Официальная документация библиотеки для асинхронных запросов. URL: <https://docs.aiohttp.org/en/stable/> (Дата обращения: 12.01.2025)
5. Примеры работы модуля json для python. URL: <https://python-scripts.com/json> (Дата обращения: 12.01.2025)
6. Поиск яндекс с нейро. URL: [https://ya.ru/neuro-search/](https://ya.ru/neuro-search/?c=&campaign_id=&adgroup=&referrer=appmetrica_tracking_id%3D461069252380657226%26ym_tracking_id%3D3380943345351442812&from=about) (Дата обращения: 12.01.2025)
7. Описание конечного продукта. Конечный продукт представляет собой **Telegram-бот, который будет сообщать пользователю погоду в настоящий момент времени или прогноз**
8. Самоанализ и самооценка

В ходе работы над проектом я познакомился с основами программирования **Telegram-ботов** на python. Мне понравилось изучать теоретические вещи и находить им практическое применение.