

Методические указания

Урок 32.1 Введение в Telegram API, Настройка Webhook для чат-бота

Задачи урока:

- Настройка Webhook для чат-бота

0. Подготовка к уроку

До начала урока преподавателю необходимо:

- 1) Просмотреть, как ученики справились с домашним заданием
- 2) Прочитать методичку

1. Введение в Telegram API

Учитель: На прошлых занятиях мы с вами поработали с API телеграм, реализовали простую клавиатуру, но не использовали инлайн клавиатуру. Для начала разберемся, чем отличается инлайн клавиатура от обычной.

Как мы видели при нажатии на кнопку обычной клавиатуры у нас происходит отправка текста боту, который написан на кнопке. В инлайн же клавиатуре у нас есть возможность переслать какие либо данные при нажатии на кнопку, либо же перейти на необходимый url.

Для реализации встроенной(инлайн) клавиатуры, нам необходимо ознакомиться с двумя объектами, предоставляемыми Telegram API

InlineKeyboardMarkup

Этот объект представляет встроенную клавиатуру, которая появляется под соответствующим сообщением.

Поле	Тип	Описание
inline_keyboard	Массив массивов с InlineKeyboardButton	Массив строк, каждая из которых является массивом объектов InlineKeyboardButton.

InlineKeyboardButton

Этот объект представляет одну кнопку встроенной клавиатуры. Вы обязательно должны задействовать ровно одно опциональное поле.

Поле	Тип	Описание
text	String	Текст на кнопке
url	String	Опционально. URL, который откроется при нажатии на кнопку
callback_data	String	Опционально. Данные, которые будут отправлены в callback_query при нажатии на кнопку
switch_inline_query	String	<p>Опционально. Если этот параметр задан, то при нажатии на кнопку приложение предложит пользователю выбрать любой чат, откроет его и вставит в поле ввода сообщения юзернейм бота и определённый запрос для встроенного режима. Если отправлять пустое поле, то будет вставлен только юзернейм бота.</p> <p>Примечание: это нужно для того, чтобы быстро переключаться между диалогом с ботом и встроенным режимом с этим же ботом. Особенно полезно в сочетаниях с действиями switch_pm... – в этом случае пользователь вернётся в исходный чат автоматически, без ручного выбора из списка.</p>

Для начала давайте немного реализуем еще одну версию пулинга. Создадим переменную offset для указания в дальнейшем сдвига по нашим сообщениям

```
def main():  
    offset = 0
```

Далее реализуем основной бесконечный цикл, в котором будем проверять нет ли ошибки при запросе к телеграм API. В случае ошибки используем continue

```
while True:  
    dt = dict(offset=offset, timeout=timeout)  
    try:  
        req = requests.post(geturl, data=dt, timeout=None).json()  
    except ValueError:  
        continue
```

В цикле также обязательно проверим есть ли в ответе, который пришел ключи 'ok' и 'result'. В случае отсутствия ключей, точно также используем continue

```
if not req['ok'] or not req['result']:
    continue
```

Если все в порядке, то мы начинаем перебирать значения по ключу 'result' и получим значения ключей 'message' и id пользователя

```
for r in req['result']:
    message = r['message']
    id = message['chat']['id']
```

Теперь проверим если есть 'text' в 'message', то мы словарь с необходимыми данными и отправляем эти данные на сервер, предварительно преобразовав в json. Также изменим значение сдвига на 1 и отправим пользователю сообщение.

```
if 'text' in message:
    dt = dict(chat_id=id, text='reply')
    requests.post(sendurl, data=dt).json()
    offset = r['update_id'] + 1
```

Теперь пришло время передать в нашем запросе инлайн клавиатуру. Как мы помним клавиатура состоит из массива массивов инлайн кнопок. Очень важно помнить, что при создании инлайн кнопки, мы обязательно должны указать хотя бы одно необязательное поле. Для начала создадим клавиатуру и сохраним в нее нашу клавиатуру в виде json

```
if 'text' in message:
    keyboard = json.dumps({'inline_keyboard': [
        [{'text': 'Да', 'callback_data': '1'}]]})
```

и передадим в запрос

```
dt = dict(chat_id=id, text='reply', reply_markup=keyboard)
requests.post(sendurl, data=dt).json()
```

Проверяем. Отлично. При отправке какого то текста нашему боту, в ответ он отправляет нам текст и выводит кнопку. Давайте добавим еще одну кнопку рядом с созданной

```
keyboard = json.dumps({'inline_keyboard': [
    [{'text': 'Да', 'callback_data': '1'}, {'text': 'Нет', 'callback_data': '2'}]})
```

Готово. Давайте добавим еще одну кнопку на ряд ниже

```
keyboard = json.dumps({'inline_keyboard': [
    [{'text': 'Да', 'callback_data': '1'}, {'text': 'Нет', 'callback_data': '2'}],
    [{'text': 'Не знаю', 'callback_data': '0'}]}])
```

Как мы видим добавлять инлайн клавиатуру не так сложно. А давайте изменим нижнюю кнопку так, чтобы она вела нас на какой то адрес в сети

```
keyboard = json.dumps({'inline_keyboard': [
    [{'text': 'Да', 'callback_data': '1'}, {'text': 'Нет', 'callback_data': '2'}],
    [{'text': 'Google', 'url': 'www.google.ru'}]}])
```

Готово!

Код

```
import json

import requests

TOKEN = 'Ваш токен'

prefix = 'https://api.telegram.org/bot'
key = TOKEN
geturl = prefix + key + '/getUpdates'
sendurl = prefix + key + '/sendMessage'
timeout = 60

class InlineKeyboardButton:
    def __init__(self):
        self.text = 'Button'

class InlineKeyboardMarkup:
    def __init__(self):
        self.keyboard = [InlineKeyboardButton().__getattr__('text')]

def main():
    offset = 0
    while True:
        dt = dict(offset=offset, timeout=timeout)
        try:
            req = requests.post(geturl, data=dt, timeout=None).json()
        except ValueError:
            continue
        if not req['ok'] or not req['result']:
            continue
        for r in req['result']:
            message = r['message']
```

```
id = message['chat']['id']
if 'text' in message:
    keyboard = json.dumps({'inline_keyboard': [
        [{'text': 'Да', 'callback_data': '1'}, {'text': 'Нет', 'callback_data': '2'}],
        [{'text': 'Google', 'url': 'www.google.ru'}]})

    dt = dict(chat_id=id, text='reply', reply_markup=keyboard)
    print(requests.post(sendurl, data=dt).json())
    offset = r['update_id'] + 1

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

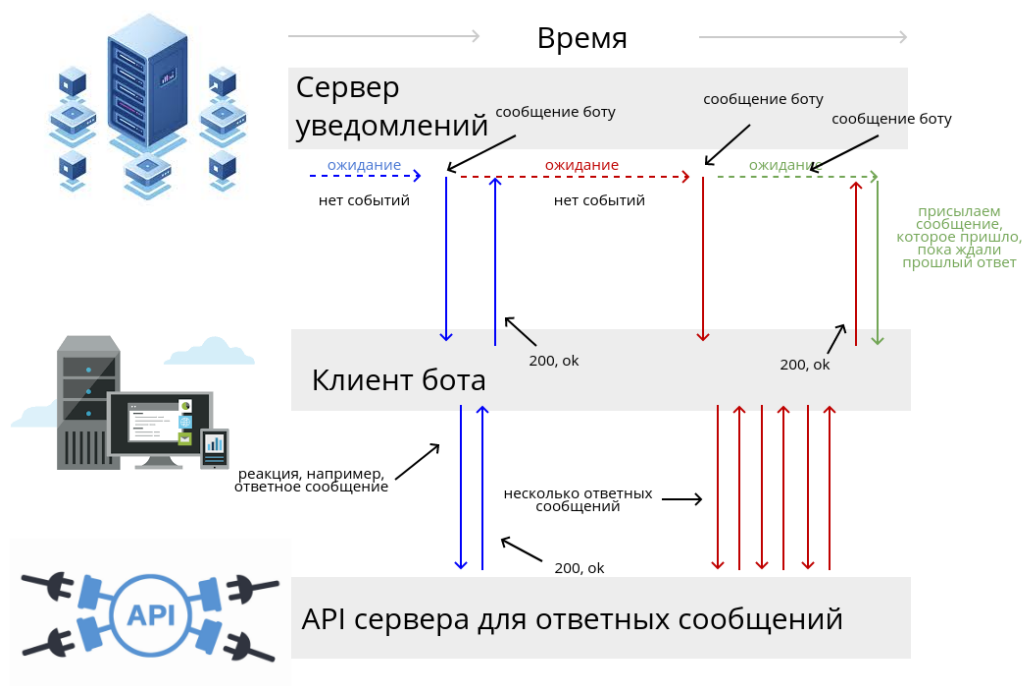
2. Настройка Webhook для чат бота

С Telegram API мы немного поработали. Конечно же ботов в данном курсе мы будем писать не на чистом API, а используя библиотеку Aiogram.

Разрабатывая телеграмм ботов мы часто не задумываемся какое количество человек ими будет пользоваться. До этого мы писали примеры с использованием технологии pooling.

Polling — это когда клиент всё время спрашивает у сервера “есть что-нибудь новенькое?”, а сервер присылает события (например, новые сообщения) или пишет, что ничего не происходило.

Webhook — это то же самое, только наоборот. Теперь если что-то случается (например, новое сообщение) — сервер сам пишет клиенту



Давайте рассмотрим схему подробнее. Есть 2 основных действующих лица: клиент и сервер уведомлений. Как только вашему боту написал пользователь — сервер уведомлений шлёт вам JSON-объект, где указывает кто писал, что писал и так далее. Дальше клиент сам решает, как на это сообщение отреагировать. Он может прислать одно сообщение, может несколько, а может вообще промолчать. В конце он отвечает серверу “Ок, 200”, обозначая этим, что запрос обработан.

Чтобы клиент мог писать в ответ, обычно создаётся отдельное API, где реализованы запросы на отправку сообщений и так далее. API и сервер уведомлений — это обычно одна та же соцсеть, просто разные микросервисы. Один занимается только рассылкой уведомлений, другой — полноценное API для разработчиков.

Обе технологии занимаются одним и тем же: приносят вам сообщения о событиях. Тем не менее, между ними огромная разница: polling легко реализуется обычной библиотекой requests, в то время как для webhook обязательно нужен сайт, который будет принимать и обрабатывать запросы от сервера уведомлений.

Взамен webhook снижает нагрузку на сеть: вы обмениваетесь сообщениями с сервером уведомлений только “по делу”, в то время как polling постоянно меняется пустыми сообщениями:

Клиент: Обновления есть?

Сервер: Обновлений нет.

Клиент: А теперь?

Сервер: Всё ещё нет

...

При этом вы не теряете мгновенность ответа, как при long polling, потому что сервер по-прежнему пишет вам сразу, как только ему написал пользователь.

Для простоты мы с вами будем использовать не Telegram API, а библиотеку aiogram. Начнем мы с реализации пуллинга.

Создадим простой шаблон бота на aiogram

```
import logging

from aiogram import Bot, Dispatcher, executor, types

API_TOKEN = 'Ваш токен'

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)

if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)
```

Обработка ввода команды /start пользователем

```
@dp.message_handler(commands=['start'])
async def start_func(message: types.Message):
    await message.answer('Вы ввели команду /start')
```

Протестировали. Отлично. Выгрузим на сервис <https://www.pythonanywhere.com/>. Пока что реализуем в данном примере пуллинг. Реализацию вебхуков мы рассмотрим в следующих занятиях. Переходим на главную

Host, run, and code Python in the cloud!

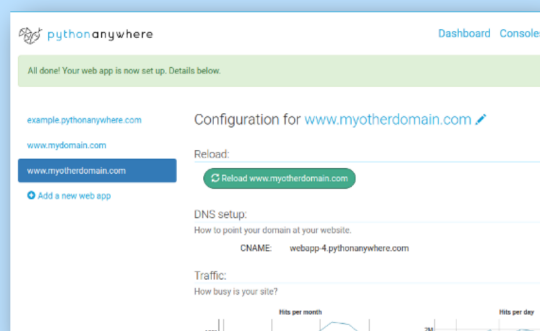
Get started for free. Our basic plan gives you access to machines with a full Python environment already installed. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at \$5/month.

Start running Python online in less than a minute! »

Watch our one-minute video »

Not convinced? Read what our users are saying!



Далее выбираем тестовый бесплатный аккаунт



Plans and pricing

Beginner: Free!

A **limited account** with one web app at `your-username.pythonanywhere.com`, restricted outbound Internet access from your apps, low CPU/bandwidth, no IPython/Jupyter notebook support.
It works and it's a great way to get started!

[Create a Beginner account](#)

Education accounts

Are you a teacher looking for a place your students can code Python? You're not alone. Click through to find out more about our [Education beta](#).

All of our paid plans come with a **no-quibble 30-day money-back guarantee** – you're billed monthly and you can cancel at any time. The minimum contract length is just one month. You get unrestricted Internet access from your applications, unlimited in-browser Python, Bash and database consoles, and full SSH access to your account. All accounts (including free ones) have screen-sharing with other PythonAnywhere accounts, and free SSL support (though you'll need to get a certificate for your own domains).

Hacker	\$5/month	Web dev	\$12/month	Startup	\$99/month	Custom	\$5 to \$500/month
Run your Python code in the cloud from one web app and the console		If you want to host small Python-based websites for you or for your clients		Start a business and don't worry about having to scale to handle traffic spikes		Want a combination that's not on the list? Create your own! All custom plans have:	

Регистрируемся



Create your account

Username:

Email:

Password:


Password (again):

☐ I agree to the [Terms and Conditions](#) and the [Privacy and Cookies Policy](#), and confirm that I am at least 13 years old.

[Register](#)

We promise not to spam or pass your details on to anyone else.

Готово

pythonanywhere
by ANACONDA

DashboardConsolesFilesWebTasksDatabases

Warning

You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).

Dashboard

Welcome, [bakeevrus](#)

CPU Usage: 0% used – 0.00s of 100s. Resets in 23 hours, 59 minutes [More Info](#)

File storage: 0% full – 48.0 KB of your 512.0 MB quota [More Info](#)

Upgrade Account

Recent Consoles

+ 5 -

You have no recent consoles.

New console:

\$ Bash

>>> Python

More...

Recent Files

+ 5 -

You have no recently edited files.

+ Open another file

Browse files

Recent Notebooks

+ 5 -

Your account does not support Jupyter Notebooks. Upgrade your account to get access!


All Web apps

You don't have any web apps.

Open Web tab

Copyright © 2011-2023 PythonAnywhere LLP — [Terms](#) — [Privacy & Cookies](#)


Переходим во вкладку Files и создаем новую папку под бота

pythonanywhere
by ANACONDA

DashboardConsoles**Files**WebTasksDatabases

Warning

You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).

/home/  bakeevrus

[Open Bash console here](#)0% full – 168.0 KB of your 512.0 MB quota [More Info](#)

Directories

Enter new directory name

New directory

.cache/

.local/


.virtualenvs/



bot/


Files



Enter new file name, eg hello.py


New file



 .bashrc


  2023-03-09 20:01560 bytes



 .gitconfig


  2023-03-09 20:01266 bytes



 .profile


  2023-03-09 20:0179 bytes



 .pythonstartup.py

  2023-03-09 20:0177 bytes

 .vimrc

  2023-03-09 20:014.6 KB

 README.txt

  2023-03-09 20:01232 bytes


Upload a file

100MiB maximum size

Copyright © 2011-2023 PythonAnywhere LLP — [Terms](#) — [Privacy & Cookies](#)


Загружаем файл с ботом в папку

Send feedbackForumsHelpBlogAccountLog out

pythonanywhere
by ANACONDA

DashboardConsoles**Files**WebTasksDatabases

Warning You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).

/home/bakeevrus/  bot

Open Bash console here0% full – 172.0 KB of your 512.0 MB quota [More Info](#)

Directories


Enter new directory name



New directory

Files

Enter new file name, eg hello.py

New file

 bot.py

  2023-03-09 20:08 473 bytes


Upload a file

100MiB maximum size

Copyright © 2011-2023 PythonAnywhere LLP – [Terms](#) – [Privacy & Cookies](#)

Далее переходим в консоль

Send feedbackForumsHelpBlogAccountLog out

pythonanywhere
by ANACONDA

Dashboard**Consoles**FilesWebTasksDatabases

Warning You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).

CPU Usage: 0% used – 0.00s of 100s. Resets in 23 hours, 52 minutes [More Info](#)

Start a new console:

Python: [3.10](#) / [3.9](#) / [3.8](#) / [3.7](#) / [3.6](#) IPython: [3.10](#) / [3.9](#) / [3.8](#) / [3.7](#) / [3.6](#) PyPy: [2](#) / [3](#)
Other: [Bash](#) | [MySQL](#)
Custom: [+](#)

Your consoles:

You have no consoles. Click a link above to start one.

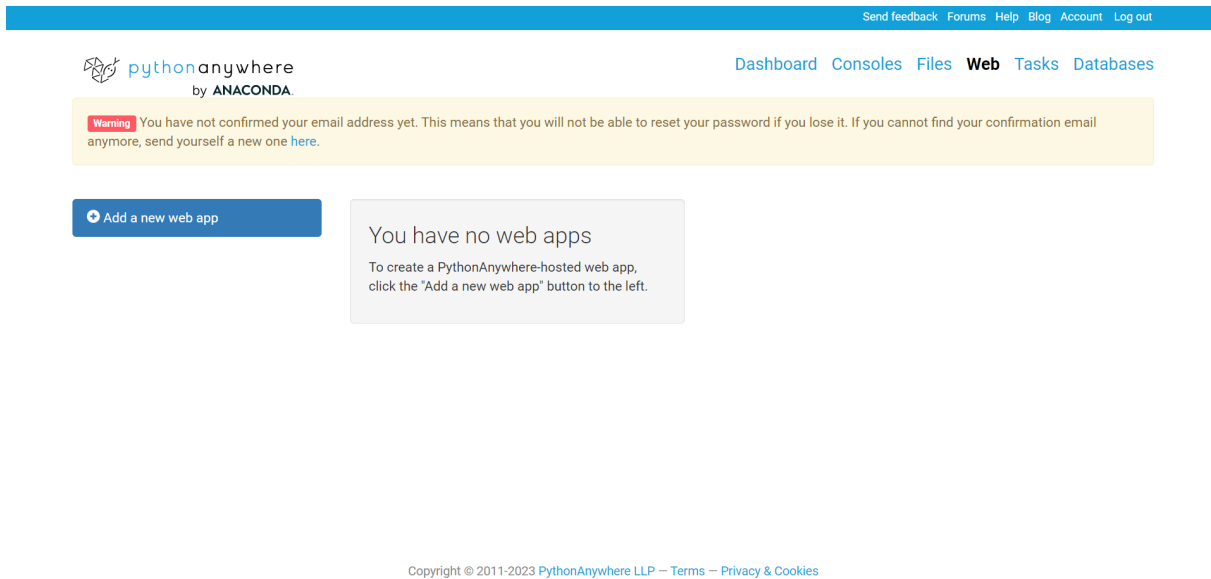
Consoles shared with you

No-one has shared any consoles with you :-)

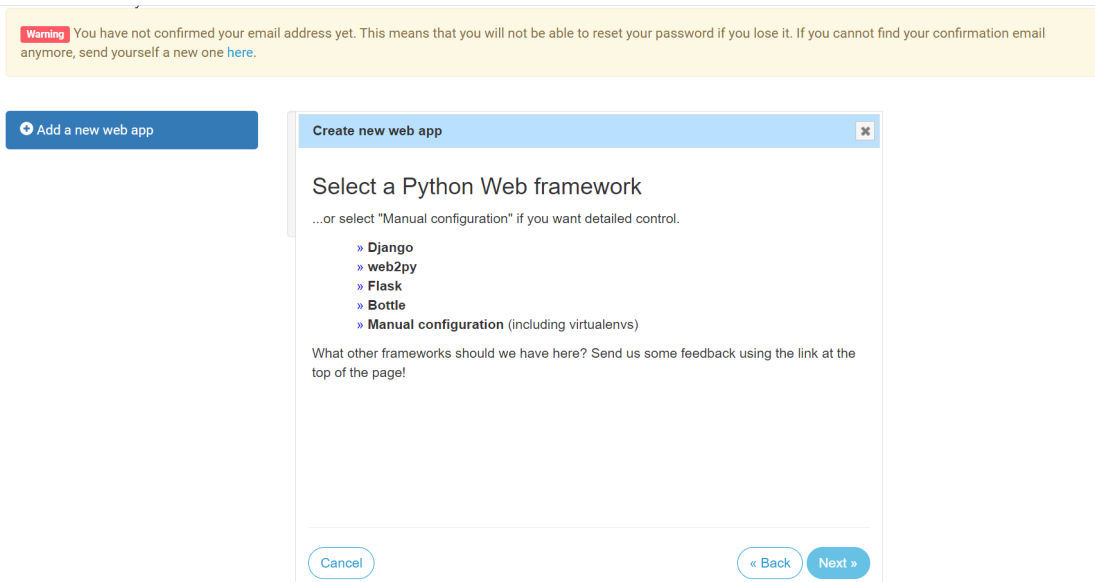
Running processes

Заходим в bash консоль и устанавливаем aiogram
pip install aiogram

После установки необходимых библиотек переходим во вкладку Web



И создаем новое веб приложение



выбираем последний пункт и далее версию python

Warning You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email anymore, send yourself a new one [here](#).

Add a new web app

Create new web app

Select a Python version

- » Python 2.7
- » Python 3.6
- » Python 3.7
- » Python 3.8
- » Python 3.9
- » Python 3.10

Cancel

« Back

Next »

Далее нажимаем next и ждем когда настроится наше приложение

Указываем путь до нашего файла

[Paying users](#)' sites stay up forever without any need to log in to keep them running.

Traffic:

How busy is your site?

This month (previous month)	0	(0)
Today (yesterday)	0	(0)
Hour (previous hour)	0	(0)

Want some more data? [Paying accounts](#) get pretty charts ;-)

Code:

What your site is running.

Source code:	/home/bakeevrus/bot	Go to directory
Working directory:	/home/bakeevrus/	Go to directory
WSGI configuration file:	/var/www/bakeevrus_pythonanywhere_com_wsgi.py	
Python version:	3.10 ✎	

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired

Далее переходим в консоль и запускаем бота

```
11:14 ~/bot python bot.py
```

Готово.

Внимание для использования данного сервиса, вам возможно потребуется использовать прокси данного сервиса.

```
proxy_url = 'http://proxy.server:3128'  
bot = Bot(token=API_TOKEN, proxy=proxy_url)
```

Полный код

```
import logging  
  
from aiogram import Bot, Dispatcher, executor, types  
  
API_TOKEN = 'Ваш токен'  
  
logging.basicConfig(level=logging.INFO)  
proxy_url = 'http://proxy.server:3128'  
bot = Bot(token=API_TOKEN, proxy=proxy_url)  
dp = Dispatcher(bot)  
  
@dp.message_handler(commands=['start'])  
async def start_func(message: types.Message):  
    await message.answer('Вы ввели команду /start')  
    print('hello')  
  
if __name__ == '__main__':  
    executor.start_polling(dp, skip_updates=True)
```

Для работы с вебхуками мы можем использовать стандартный шаблон из официальной документации. Как запустить бота с вебхуками мы рассмотрим в следующем занятии

шаблон

```
import logging

from aiogram import Bot, types
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from aiogram.dispatcher import Dispatcher
from aiogram.dispatcher.webhook import SendMessage
from aiogram.utils.executor import start_webhook

API_TOKEN = 'Ваш  токен'

# webhook settings
WEBHOOK_HOST = 'адрес сайта'
WEBHOOK_PATH = '/path/to/api'
WEBHOOK_URL = f"{WEBHOOK_HOST}{WEBHOOK_PATH}"

# webserver settings
WEBAPP_HOST = 'localhost' # or ip
WEBAPP_PORT = 3001

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
dp.middleware.setup(LoggingMiddleware())

async def on_startup(dp):
    await bot.set_webhook(WEBHOOK_URL)
    # insert code here to run it after start

async def on_shutdown(dp):
    logging.warning('Shutting down..')

    # insert code here to run it before shutdown

    # Remove webhook (not acceptable in some cases)
    await bot.delete_webhook()

    # Close DB connection (if used)
    await dp.storage.close()
    await dp.storage.wait_closed()

    logging.warning('Bye!')

if __name__ == '__main__':
    start_webhook(
        dispatcher=dp,
        webhook_path=WEBHOOK_PATH,
        on_startup=on_startup,
        on_shutdown=on_shutdown,
        skip_updates=True,
        host=WEBAPP_HOST,
```

```
)  
    port=WEBAPP_PORT,
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1

Реализовать эхо бота использующего webhook по шаблону(на сервис не выгружать). Нужно прописать только обработчик сообщений для эхо бота по шаблону