МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

Факультет “Информатика и системы управления”  
Кафедра “Системы обработки информации и управления”

A logo with a gold star and blue ribbon

Description automatically generated

Дисциплина «Парадигмы и конструкции языков программирования»

## Отчет по лабораторной работе №5\_6 «Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

**Выполнил:**Студент группы ИУ5-34Б  
Буш И. Е.  
**Преподаватель:**Нардид Анатолий Николаевич

Москва 2025

**Задание:**

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

Листинг кода

bot.py

import logging

import requests

import random

from datetime import datetime

import asyncio

from aiogram import Bot, Dispatcher, types

API\_TOKEN = "8581616381:AAHKTw4xbfbqwxh71Vtkw\_SSSJujdV5EuNE"

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API\_TOKEN)

dp = Dispatcher()

GAME\_STATE = {}

# Вспомогательные функции

def fetch\_today\_events():

    today = datetime.now()

    url = f"https://api.wikimedia.org/feed/v1/wikipedia/ru/onthisday/all/{today.month:02d}/{today.day:02d}"

    headers = {"User-Agent": "TelegramBot-Wiki/1.0"}

    try:

        r = requests.get(url, headers=headers, timeout=5)

        r.raise\_for\_status()

        data = r.json()

        events = data.get("events", [])

        holidays = data.get("holidays", [])

        return events[:10], holidays[:5]

    except:

        return [], []

def fetch\_random\_day\_events():

    import random

    month = random.randint(1, 12)

    if month == 2:

        day = random.randint(1, 28)

    elif month in [4, 6, 9, 11]:

        day = random.randint(1, 30)

    else:

        day = random.randint(1, 31)

    url = f"https://api.wikimedia.org/feed/v1/wikipedia/ru/onthisday/all/{month:02d}/{day:02d}"

    headers = {"User-Agent": "TelegramBot-Wiki/1.0"}

    try:

        r = requests.get(url, headers=headers, timeout=5)

        r.raise\_for\_status()

        data = r.json()

        events = data.get("events", [])

        holidays = data.get("holidays", [])

        return events, holidays

    except:

        return [], []

def format\_events(events, holidays):

    parts = []

    if holidays:

        parts.append("🎉 \*Праздники и памятные даты:\*")

        for h in holidays:

            parts.append(f"- {h.get('text')}")

    if events:

        parts.append("🕰 \*События в истории:\*")

        for e in events:

            parts.append(f"- {e.get('year')}: {e.get('text')}")

    return "\n".join(parts) if parts else "Информация недоступна."

# Хэндлеры

@dp.message()

async def handle\_messages(message: types.Message):

    text = message.text.lower()

    # /start

    if text == "/start":

        await message.reply(

            "Привет! Я бот Wikipedia.\n"

            "/today — события и праздники сегодня\n"

            "/guess — угадай год события\n"

            "/truefalse — игра «Правда или Ложь»\n"

            "/earlierlater — какая из двух произошла раньше?"

        )

        return

    # /today

    if text == "/today":

        events, holidays = fetch\_today\_events()

        await message.reply(format\_events(events, holidays), parse\_mode="Markdown")

        return

    # /guess

    if text == "/guess":

        events, \_ = fetch\_random\_day\_events()

        if not events:

            await message.reply("События недоступны.")

            return

        event = random.choice(events)

        GAME\_STATE[message.from\_user.id] = {"type": "guess", "year": event["year"]}

        await message.reply(f"📜 Событие:\n{event['text']}\nВ каком году это произошло?")

        return

    # проверка на угадай год

    state = GAME\_STATE.get(message.from\_user.id)

    if state and state["type"] == "guess":

        try:

            user\_year = int(message.text.strip())

            correct\_year = state["year"]

            if user\_year == correct\_year:

                await message.reply(f"✅ Верно! Это {correct\_year} год.")

            else:

                await message.reply(f"❌ Неверно. Правильный год: {correct\_year}.")

        except:

            await message.reply("Введите число — предполагаемый год события.")

        finally:

            del GAME\_STATE[message.from\_user.id]

        return

    # /truefalse

    if text == "/truefalse":

        events, \_ = fetch\_random\_day\_events()

        if not events:

            await message.reply("События недоступны.")

            return

        event = random.choice(events)

        year = event["year"]

        if random.choice([True, False]):

            fake\_year = year

            correct = True

        else:

            fake\_year = year + random.randint(-50, 50)

            correct = False

        GAME\_STATE[message.from\_user.id] = {"type": "truefalse", "correct": correct}

        await message.reply(f"❓ Событие:\n{event['text']}\nГод: {fake\_year}\nПравда или Ложь? (ответьте: Правда / Ложь)")

        return

    # роверка «Правда/Ложь»

    if state and state["type"] == "truefalse" and text in ["правда", "ложь"]:

        correct\_answer = state["correct"]

        if (text == "правда" and correct\_answer) or (text == "ложь" and not correct\_answer):

            await message.reply("✅ Верно!")

        else:

            await message.reply("❌ Неверно!")

        del GAME\_STATE[message.from\_user.id]

        return

        # /earlierlater

    if text == "/earlierlater":

        events, \_ = fetch\_random\_day\_events()

        if len(events) < 2:

            await message.reply("Недостаточно событий для игры.")

            return

        e1, e2 = random.sample(events, 2)

        GAME\_STATE[message.from\_user.id] = {

            "type": "earlierlater",

            "event1": e1,

            "event2": e2,

            "correct": "1" if e1["year"] < e2["year"] else "2"

        }

        await message.reply(

            f"⚔️ Какое событие произошло РАНЬШЕ?\n\n"

            f"1️⃣ {e1['text']}\n"

            f"2️⃣ {e2['text']}\n\n"

            "Ответьте: 1 или 2"

        )

        return

    # проверка /earlierlater

    if state and state["type"] == "earlierlater" and text in ["1", "2"]:

        correct = state["correct"]

        e1 = state["event1"]

        e2 = state["event2"]

        if text == correct:

            await message.reply(f"✅ Верно! Событие №{correct} произошло раньше.")

        else:

            earlier\_event = e1 if correct == "1" else e2

            await message.reply(

                f"❌ Неверно!\n"

                f"Раньше было:\n{earlier\_event['year']}: {earlier\_event['text']}"

            )

        del GAME\_STATE[message.from\_user.id]

        return

# Запуск

if \_\_name\_\_ == "\_\_main\_\_":

    asyncio.run(dp.start\_polling(bot))

Примеры работы программы

