**Московский государственный технический**

**университет им. Н. Э. Баумана**

Факультет «Радиотехнический»

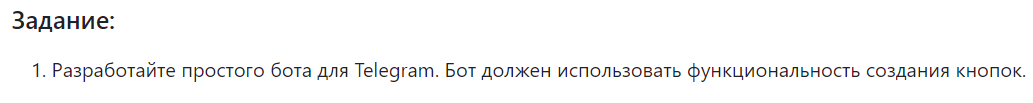
Кафедра РТ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б |  | преподаватель каф. ИУ5 |
| Суровец Е. А. |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2024 г.



**Текст программы**

import requests  
from aiogram import Bot, Dispatcher, types  
from aiogram.contrib.fsm\_storage.memory import MemoryStorage  
from aiogram.dispatcher import FSMContext  
from aiogram.dispatcher.filters.state import StatesGroup, State  
from aiogram.utils import executor  
from PIL import Image, ImageDraw, ImageFont  
import io  
  
TELEGRAM\_BOT\_TOKEN = '7417967380:AAEQgNS6qxK2zpQGTz5TwlYwT5txK-bZINk'  
UNSPLASH\_ACCESS\_KEY = '9c9odrBt8HLsPK5OQn3jKrRkTblBJu85d6vu2ZDNPS4'  
  
bot = Bot(token=TELEGRAM\_BOT\_TOKEN)  
storage = MemoryStorage()  
dp = Dispatcher(bot, storage=storage)  
  
  
def wrap\_text(text, font, max\_width):  
 lines = []  
 words = text.split(' ')  
 current\_line = ""  
  
 for word in words:  
 test\_line = f"{current\_line} {word}".strip()  
 if 44 \* (len(test\_line) // 2) <= max\_width:  
 current\_line = test\_line  
 else:  
 lines.append(current\_line)  
 current\_line = word  
  
 if current\_line:  
 lines.append(current\_line)  
 return lines  
  
  
class Form(StatesGroup):  
 waiting\_for\_word = State()  
 waiting\_for\_photo\_choice = State()  
 waiting\_for\_caption = State()  
  
  
@dp.message\_handler(commands=['start'])  
async def start\_command(message: types.Message):  
 await message.answer("Привет! Нажмите кнопку ниже, чтобы создать мем.", reply\_markup=main\_menu())  
  
  
def main\_menu():  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 button = types.KeyboardButton("Создать мем")  
 markup.add(button)  
 return markup  
  
  
@dp.message\_handler(lambda message: message.text == "Создать мем")  
async def create\_mem(message: types.Message):  
 await Form.waiting\_for\_word.set()  
 await message.answer("Пожалуйста, введите слово для поиска изображений.")  
  
  
@dp.message\_handler(state=Form.waiting\_for\_word)  
async def search\_photos(message: types.Message, state: FSMContext):  
 query = message.text  
 if not query:  
 await message.answer("Пожалуйста, введите слово для поиска.")  
 return  
  
 url = f'https://api.unsplash.com/search/photos?page=1&query={query}&client\_id={UNSPLASH\_ACCESS\_KEY}&per\_page=10'  
 response = requests.get(url)  
 data = response.json()  
  
 if data['total'] == 0:  
 await message.answer("Извините, ничего не найдено. Попробуйте другое слово.")  
 await state.finish()  
 await Form.waiting\_for\_word.set()  
 return  
  
 photos = data['results']  
 for photo in photos:  
 await bot.send\_photo(chat\_id=message.chat.id, photo=photo['urls']['small'])  
  
 await state.update\_data(photos=photos)  
  
 buttons = [types.InlineKeyboardButton(text=str(i + 1), callback\_data=str(i)) for i in range(len(photos))]  
 markup = types.InlineKeyboardMarkup(row\_width=2)  
 markup.add(\*buttons)  
  
 await Form.waiting\_for\_photo\_choice.set()  
 await message.answer("Выберите одно из фото, нажав на соответствующую цифру:", reply\_markup=markup)  
  
  
@dp.callback\_query\_handler(state=Form.waiting\_for\_photo\_choice)  
async def photo\_choice(callback\_query: types.CallbackQuery, state: FSMContext):  
 selected\_photo\_index = int(callback\_query.data)  
 user\_data = await state.get\_data()  
 photos = user\_data.get('photos')  
  
 if selected\_photo\_index < len(photos):  
 selected\_photo\_url = photos[selected\_photo\_index]['urls']['small']  
 await bot.send\_message(callback\_query.from\_user.id, "Вы выбрали фото. Теперь введите текст для надписи:")  
 await Form.waiting\_for\_caption.set()  
  
 *# Сохраняем выбранное фото в состоянии* await state.update\_data(selected\_photo\_url=selected\_photo\_url)  
 else:  
 await bot.send\_message(callback\_query.from\_user.id, "Неверный выбор. Попробуйте снова.")  
  
  
@dp.message\_handler(state=Form.waiting\_for\_caption)  
async def add\_caption(message: types.Message, state: FSMContext):  
 caption\_text = message.text  
 user\_data = await state.get\_data()  
 selected\_photo\_url = user\_data.get('selected\_photo\_url')  
  
 *# Скачиваем фото* response = requests.get(selected\_photo\_url)  
 image = Image.open(io.BytesIO(response.content))  
  
 *# Добавляем текст на изображение* draw = ImageDraw.Draw(image)  
 font\_size = 44  
 font\_path = "Impact.ttf"  
 font = ImageFont.truetype(font\_path, font\_size)  
  
 max\_width = image.width - 20  
 wrapped\_lines = wrap\_text(caption\_text, font, max\_width)  
 line\_height = font\_size + 8  
 y\_position = 12  
  
 for line in wrapped\_lines:  
 x\_position = 10  
 draw.text((x\_position, y\_position), line, fill="white", font=font, stroke\_width=3, stroke\_fill=(0, 0, 0))  
 y\_position += line\_height  
  
 *# Сохраняем изображение в байтовом потоке* byte\_io = io.BytesIO()  
 byte\_io.name = 'meme.png'  
 image.save(byte\_io, 'PNG')  
 byte\_io.seek(0)  
  
 *# Отправляем мем с текстом* await bot.send\_photo(chat\_id=message.chat.id, photo=byte\_io)  
  
 await message.answer("Ваш мем готов! Если хотите создать еще один, нажмите 'Создать мем'.")  
 await state.finish()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 executor.start\_polling(dp, skip\_updates=True)

**Результаты выполнения**

**Изображение выглядит как снимок экрана, текст, риф, Организм

Автоматически созданное описание**

**Изображение выглядит как текст, птица, снимок экрана, вода

Автоматически созданное описание**

**Изображение выглядит как текст, дерево, снимок экрана, растение

Автоматически созданное описание**