

ПРОГРАММА

ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

**“Python-разработка для БПЛА: паттерны проектирования, API-интерфейсы и библиотеки для оптимизации решений”**

**Задание 9.**

**Проектирование RESTful интерфейсов**

Преподаватель курса

**Максимов Егор Васильевич**

Выполнил

**Зубков Евгений Юрьевич**

**№ потока БПЛА\_256-1, группа 1**

**Москва 2024 г.**

**Задание 9. Проектирование RESTful интерфейсов**

**Инструкция:**

Шаг 1: Определение Ресурсов

1. Определите ресурсы для управления БПЛА, например:

- "БПЛА" с атрибутами `id`, `name`, `status`, `location`.

- "Миссия" с атрибутами `id`, `name`, `start\_time`, `end\_time`, `route`.

Шаг 2: Определение CRUD Операций

2. Реализуйте следующие CRUD операции для ресурсов "БПЛА" и "Миссия":

- Создание нового БПЛА (HTTP метод: POST, Эндпоинт: `/drones`).

- Получение списка всех БПЛА (HTTP метод: GET, Эндпоинт: `/drones`).

- Получение информации о конкретном БПЛА (HTTP метод: GET, Эндпоинт: `/drones/{id}`).

- Обновление информации о БПЛА (HTTP метод: PUT, Эндпоинт: `/drones/{id}`).

- Удаление БПЛА (HTTP метод: DELETE, Эндпоинт: `/drones/{id}`).

- Создание новой миссии (HTTP метод: POST, Эндпоинт: `/missions`)

- Получение списка всех миссий (HTTP метод: GET, Эндпоинт: `/missions`).

- Получение информации о конкретной миссии (HTTP метод: GET, Эндпоинт: `/missions/{id}`).

- Обновление информации о миссии (HTTP метод: PUT, Эндпоинт: `/missions/{id}`).

- Удаление миссии (HTTP метод: DELETE, Эндпоинт: `/missions/{id}`).

Шаг 3: Определение Формата Данных

3. Определите формат представления данных, например, в формате JSON.

4. Структура данных:

- Запрос на создание/обновление БПЛА: `{"name": "Название", "status": "Статус", "location": "Местоположение"}`

- Ответ на запрос получения списка БПЛА: `[{"id": 1, "name": "БПЛА 1", "status": "В полете", "location": "Координаты 1"}, {"id": 2, "name": "БПЛА 2", "status": "На земле", "location": "Координаты 2"}]`

- Запрос на создание/обновление миссии: `{"name": "Название", "start\_time": "Время начала", "end\_time": "Время окончания", "route": "Маршрут"}`

- Ответ на запрос получения списка миссий: `[{"id": 1, "name": "Миссия 1", "start\_time": "Время начала 1", "end\_time": "Время окончания 1", "route": "Маршрут 1"}, {"id": 2, "name": "Миссия 2", "start\_time": "Время начала 2", "end\_time": "Время окончания 2", "route": "Маршрут 2"}]`

5. Коды состояния HTTP:

- 200 OK: Успешный запрос.

- 201 Created: Ресурс успешно создан.

- 404 Not Found: Ресурс не найден.

- 400 Bad Request: Некорректный запрос.

- 500 Internal Server Error: Внутренняя ошибка сервера.

**Решение:**

**Сервер:**

# Задание - проектирование RESTful интерфейсов  
from flask import Flask, Response, request, jsonify  
import requests  
import logging  
  
  
# Настройка логирования  
logging.basicConfig(level=logging.INFO, filemode="w")  
  
# Приложение  
app = Flask(\_\_name\_\_)  
# "БПЛА" с атрибутами `id`, `name`, `status`, `location`  
drones = {}  
# "Миссия" с атрибутами `id`, `name`, `start\_time`, `end\_time`, `route`  
missions = {}  
  
  
# Получение списка всех БПЛА (HTTP метод: GET, Эндпоинт: `/drones`)  
@app.route('/drones', methods=['GET'])  
def get\_drones():  
 return jsonify(drones), 200  
  
  
# Получение информации о конкретном БПЛА (HTTP метод: GET, Эндпоинт: `/drones/{id}`)  
@app.route('/drones/<id>', methods=['GET'])  
def get\_drone(id):  
 drone = drones.get(id)  
 if drone:  
 return jsonify(drone), 200  
 return jsonify({'error': f'Drone ID:{id} not found'}), 404  
  
  
# Создание нового БПЛА (HTTP метод: POST, Эндпоинт: `/drones`)  
@app.route('/drones', methods=['POST'])  
def create\_drone():  
 id = request.json.get('id')  
 if id:  
 drones[id] = request.json  
 drones[id]['status'] = 'active'  
 logging.info(f"Drone ID:{id} created")  
 return jsonify({'message': f'Drone ID:{id} created'}), 201  
 return jsonify({'error': 'Drone ID required'}), 400  
  
  
# Обновление информации о БПЛА (HTTP метод: PUT, Эндпоинт: `/drones/{id}  
@app.route('/drones/<id>', methods=['PUT'])  
def update\_drone(id):  
 drone = drones.get(id)  
 if drone:  
 drone.update(request.json)  
 logging.info(f"Drone ID:{id} updated")  
 return jsonify({'message': f'Drone ID:{id} updated'}), 200  
 return jsonify({'error': f'Drone ID:{id} not found'}), 404  
  
  
# Удаление БПЛА (HTTP метод: DELETE, Эндпоинт: `/drones/{id}`)  
@app.route('/drones/<id>', methods=['DELETE'])  
def delete\_drone(id):  
 if id in drones:  
 del drones[id]  
 logging.info(f"Drone ID:{id} deleted")  
 return jsonify({'message': f'Drone ID:{id} deleted'}), 200  
 return jsonify({'error': f'Drone ID:{id} not found'}), 404  
  
  
# Создание новой миссии (HTTP метод: POST, Эндпоинт: `/missions`)  
@app.route('/missions', methods=['POST'])  
def create\_mission():  
 mission\_id = request.json.get('id')  
 if mission\_id:  
 missions[mission\_id] = request.json  
 logging.info(f"Missions ID:{mission\_id} created")  
 return jsonify({'message': 'Mission created'}), 201  
 return jsonify({'error': 'Mission ID required'}), 400  
  
  
# Получение списка всех миссий (HTTP метод: GET, Эндпоинт: `/missions`)  
@app.route('/missions', methods=['GET'])  
def get\_missions():  
 return jsonify(missions), 200  
  
  
# Получение информации о конкретной миссии (HTTP метод: GET, Эндпоинт: `/missions/{id}`)  
@app.route('/missions/<id>', methods=['GET'])  
def get\_mission(id):  
 mission = missions.get(id)  
 if mission:  
 return jsonify(mission), 200  
 return jsonify({'error': f'Mission ID:{id} not found'}), 404  
  
  
# Обновление информации о миссии (HTTP метод: PUT, Эндпоинт: `/missions/{id}`)  
@app.route('/missions/<id>', methods=['PUT'])  
def update\_mission(id):  
 mission = missions.get(id)  
 if mission:  
 mission.update(request.json)  
 logging.info(f"Mission ID:{id} updated")  
 return jsonify({'message': f'Mission ID:{id} updated'}), 200  
 return jsonify({'error': f'Mission ID:{id} not found'}), 404  
  
  
# Удаление миссии (HTTP метод: DELETE, Эндпоинт: `/missions/{id}`)  
@app.route('/missions/<id>', methods=['DELETE'])  
def delete\_mission(id):  
 if id in missions:  
 del missions[id]  
 logging.info(f"Mission ID:{id} deleted")  
 return jsonify({'message': f'Mission ID:{id} deleted'}), 200  
 return jsonify({'error': f'Mission ID:{id} not found'}), 404  
  
  
# Отправка миссии <ID> на БПЛА <ID>  
@app.route('/missions/<mission\_id>/<drone\_id>/send', methods=['POST'])  
def send\_mission\_to\_drone(mission\_id, drone\_id):  
 mission = missions.get(mission\_id)  
 if mission:  
 drone = drones.get(drone\_id)  
 if drone:  
 if drone['status'] == 'active':  
 drone\_url = drone['control\_url'] + '/missions'  
 response = requests.post(drone\_url, json=mission)  
 if response.status\_code == 201:  
 drones['location'] = mission['route'][0]  
 drones['status'] = 'mission'  
 logging.info(f"Mission ID:{mission\_id} sent to Drone ID:{drone\_id}")  
 else:  
 return jsonify({'error': 'Drone is not active'}), 400  
 return jsonify({'message': f'Mission ID:{mission\_id} sent to Drone ID:{drone\_id}'}), 200  
 return jsonify({'error': f'Drone ID:{drone\_id} not found'}), 404  
 return jsonify({'error': f'Mission ID:{mission\_id} not found'}), 404  
  
  
def main():  
 app.run(host='0.0.0.0', port=5000)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Дрон 1:**

from flask import Flask, Response, request, jsonify  
import requests  
  
# Конфигурация БПЛА  
app = Flask(\_\_name\_\_)  
# "БПЛА" с атрибутами `id`, `name`, `status`, `location`.  
drone = {  
 "id": "B001",  
 "name": "Autel Robotics s.v.Defender",  
 "status": "landed",  
 "location": {  
 "latitude": 50.45,  
 "longitude": 30.52,  
 "altitude": 0  
 },  
 "control\_url": "http://localhost:8080"  
}  
  
# "Миссия" с атрибутами `id`, `name`, `start\_time`, `end\_time`, `route`  
mission = {  
 "id": "M001",  
 "name": "Перевозка груза",  
 "start\_time": "2024-08-08 09:00",  
 "end\_time": "2024-08-08 12:00",  
 "route": [  
 {"latitude": 50.45, "longitude": 30.52, "altitude": 100},  
 {"latitude": 50.47, "longitude": 30.55, "altitude": 200},  
 {"latitude": 50.49, "longitude": 30.58, "altitude": 300}  
 ]  
}  
  
BASE\_URL = "http://localhost:5000/drones"  
  
  
# Создание нового БПЛА (HTTP метод: POST, Эндпоинт: `/drones`)  
def register\_drone():  
 payload = {  
 \*\*drone  
 }  
 response = requests.post(BASE\_URL, json=payload)  
 if response.status\_code == 201:  
 drone['status'] = 'active'  
 return response.json()  
  
  
# Удаление БПЛА (HTTP метод: DELETE, Эндпоинт: `/drones/{id}`)  
def delete\_drone():  
 url = BASE\_URL + '/' + drone['id']  
 response = requests.delete(url)  
 if response.status\_code == 200:  
 drone['status'] = 'landed'  
 print(f"Drone ID: {drone['id']} deleted")  
 #return response.json()  
  
  
# Получение миссии с сервера  
@app.route('/missions', methods=['POST'])  
def create\_mission():  
 #print(request.json)  
 mission\_id = request.json.get('id')  
 if mission\_id:  
 mission[mission\_id] = request.json  
 return jsonify({'message': 'Mission created on Drone'}), 201  
 return jsonify({'error': 'Mission ID required'}), 400  
  
  
# Взлет  
@app.route("/takeoff", methods=["POST"])  
def takeoff():  
 altitude = request.json.get("altitude")  
 # Логика взлета  
 print(f"Взлетаем на высоту: {altitude} метров")  
 return jsonify({"message": f"Взлет на высоту {altitude} выполнен"}), 200  
  
  
# Посадка  
@app.route("/land", methods=["POST"])  
def land():  
 altitude = request.json.get("altitude")  
 # Логика посадки  
 print(f"Посадка с высоты: {altitude} метров")  
 return jsonify({"message": f"Посадка с высоты {altitude} выполнена"}), 200  
  
  
def main():  
 response = register\_drone()  
 print(f"БПЛА зарегистрирован для выполнения задач: {response['message']}")  
 app.run(port=8080)  
 delete\_drone()  
 print("БПЛА выключен")  
  
  
# Запуск приложения при запуске скрипта  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Дрон 2:**

from flask import Flask, Response, request, jsonify  
import requests  
  
# Конфигурация БПЛА  
app = Flask(\_\_name\_\_)  
# "БПЛА" с атрибутами `id`, `name`, `status`, `location`.  
drone = {  
 "id": "B002",  
 "name": "DJI mavic 3 Pro",  
 "status": "landed",  
 "location": {  
 "latitude": 51.45,  
 "longitude": 30.52,  
 "altitude": 0  
 },  
 "control\_url": "http://localhost:8081"  
}  
  
# "Миссия" с атрибутами `id`, `name`, `start\_time`, `end\_time`, `route`  
mission = {  
 "id": "M002",  
 "name": "Патрулирование",  
 "start\_time": "2024-08-08 09:00",  
 "end\_time": "2024-08-08 12:00",  
 "route": [  
 {"latitude": 50.45, "longitude": 30.52, "altitude": 100},  
 {"latitude": 50.47, "longitude": 30.55, "altitude": 200},  
 {"latitude": 50.49, "longitude": 30.58, "altitude": 300}  
 ]  
}  
  
BASE\_URL = "http://localhost:5000/drones"  
  
  
# Создание нового БПЛА (HTTP метод: POST, Эндпоинт: `/drones`)  
def register\_drone():  
 payload = {  
 \*\*drone  
 }  
 response = requests.post(BASE\_URL, json=payload)  
 if response.status\_code == 201:  
 drone['status'] = 'active'  
 return response.json()  
  
  
# Удаление БПЛА (HTTP метод: DELETE, Эндпоинт: `/drones/{id}`)  
def delete\_drone():  
 url = BASE\_URL + '/' + drone['id']  
 response = requests.delete(url)  
 if response.status\_code == 200:  
 drone['status'] = 'landed'  
 print(f"Drone ID: {drone['id']} deleted")  
 #return response.json()  
  
  
# Получение миссии с сервера  
@app.route('/missions', methods=['POST'])  
def create\_mission():  
 #print(request.json)  
 mission\_id = request.json.get('id')  
 if mission\_id:  
 mission[mission\_id] = request.json  
 return jsonify({'message': 'Mission created on Drone'}), 201  
 return jsonify({'error': 'Mission ID required'}), 400  
  
  
# Взлет  
@app.route("/takeoff", methods=["POST"])  
def takeoff():  
 altitude = request.json.get("altitude")  
 # Логика взлета  
 print(f"Взлетаем на высоту: {altitude} метров")  
 return jsonify({"message": f"Взлет на высоту {altitude} выполнен"}), 200  
  
  
# Посадка  
@app.route("/land", methods=["POST"])  
def land():  
 altitude = request.json.get("altitude")  
 # Логика посадки  
 print(f"Посадка с высоты: {altitude} метров")  
 return jsonify({"message": f"Посадка с высоты {altitude} выполнена"}), 200  
  
  
def main():  
 response = register\_drone()  
 print(f"БПЛА зарегистрирован для выполнения задач: {response['message']}")  
 app.run(port=8081)  
 delete\_drone()  
 print("БПЛА выключен")  
  
  
# Запуск приложения при запуске скрипта  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Клиент:**

import requests  
from pprint import pprint  
  
  
BASE\_URL = 'http://localhost:5000'  
  
  
# Постановка задачи БПЛА  
def send\_mission\_to\_drone(mission\_id, drone\_id):  
 url = f"{BASE\_URL}/missions/{mission\_id}/{drone\_id}/send"  
 response = requests.post(url)  
 return response.json()  
  
  
# Получение списка всех БПЛА (HTTP метод: GET, Эндпоинт: `/drones`).  
def get\_drones():  
 url = f"{BASE\_URL}/drones"  
 response = requests.get(url)  
 return response.json()  
  
  
# Получение списка всех миссий (HTTP метод: GET, Эндпоинт: `/missions`)  
def get\_missions():  
 url = f"{BASE\_URL}/missions"  
 response = requests.get(url)  
 return response.json()  
  
  
# Получение информации о конкретной миссии (HTTP метод: GET, Эндпоинт: `/missions/{id}`)  
def get\_mission(id):  
 url = f"{BASE\_URL}/missions/{id}"  
 response = requests.get(url)  
 return response.json()  
  
# Получение информации о дроне (HTTP метод: GET, Эндпоинт: `/drones/{drone\_id}`)  
def get\_drone(drone\_id):  
 url = f"{BASE\_URL}/drones/{drone\_id}"  
 response = requests.get(url)  
 return response.json()  
  
  
# Обновление информации о БПЛА (HTTP метод: PUT, Эндпоинт: `/drones/{id}  
def update\_drone(drone\_id, data):  
 url = f"{BASE\_URL}/drones/{drone\_id}"  
 payload = {  
 \*\*data  
 }  
 response = requests.put(url, json=payload)  
 return response.json()  
  
  
# Обновление информации о миссии (HTTP метод: PUT, Эндпоинт: `/missions/{id}`)  
def update\_mission(id, data):  
 url = f"{BASE\_URL}/missions/{id}"  
 payload = {  
 \*\*data  
 }  
 response = requests.put(url, json=payload)  
 return response.json()  
  
  
# Создание новой миссии  
def create\_mission(data):  
 url = f"{BASE\_URL}/missions"  
 payload = {  
 \*\*data  
 }  
 response = requests.post(url, json=payload)  
 return response.json()  
  
  
def main():  
 # Создание новой миссии  
 data = {  
 "id": "M001",  
 "name": "Перевозка груза",  
 "start\_time": "2024-08-08 09:00",  
 "end\_time": "2024-08-08 12:00",  
 "route": [  
 {"latitude": 50.45, "longitude": 30.52, "altitude": 100},  
 {"latitude": 50.47, "longitude": 30.55, "altitude": 200},  
 {"latitude": 50.49, "longitude": 30.58, "altitude": 300}  
 ]  
 }  
 create\_mission(data)  
 print(f"Миссия: {data['id']} {data['name']} создана")  
 print("")  
 data = {  
 "id": "M002",  
 "name": "Патрулирование",  
 "start\_time": "2024-08-08 02:00",  
 "end\_time": "2024-08-08 10:00",  
 "route": [  
 {"latitude": 51.45, "longitude": 30.52, "altitude": 100},  
 {"latitude": 51.47, "longitude": 30.55, "altitude": 200},  
 {"latitude": 51.49, "longitude": 30.58, "altitude": 300}  
 ]  
 }  
 create\_mission(data)  
 print(f"Миссия: {data['id']} {data['name']} создана")  
 print("")  
  
 # Получение списка всех БПЛА  
 print("Request info on all active UAVs")  
 drones = get\_drones()  
 pprint(drones)  
 print("")  
  
 # Получение информации о конкретном БПЛА  
 drone\_id = "B001"  
 print(f"Request info on Drone ID: {drone\_id}...")  
 drones = get\_drone(drone\_id)  
 pprint(drones)  
 print("")  
  
 # Получение списка всех миссий  
 print("Request info on all missions")  
 missions = get\_missions()  
 pprint(missions)  
 print("")  
  
 # Получение информации о конкретной миссии  
 mission\_id = "M001"  
 print(f"Request info on Mission ID: {mission\_id}...")  
 mission = get\_mission(mission\_id)  
 pprint(mission)  
 print("")  
  
 # Передача миссии БПЛА дрону  
 # mission\_id - идентификатор миссии  
 # drone\_id - идентификатор дрона  
 # send\_mission\_to\_drone(mission\_id, drone\_id)  
 print("Send mission to Drone ...")  
 drone\_id = "B001"  
 mission\_id = "M001"  
 response = send\_mission\_to\_drone(mission\_id, drone\_id)  
 print(f"success: {response.get('message')}")  
 print("")  
  
 # Обновление информации о БПЛА (HTTP метод: PUT, Эндпоинт: `/drones/{id}  
 drone\_id = "B001"  
 payload = {  
 "name": "Autel Robotics EVO MAX4N s.v.Defender",  
 "status": "landed"  
 }  
 print(f"Update info on Drone ID: {drone\_id}...")  
 drones = update\_drone(drone\_id, payload)  
 print(f"success: {response.get('message')}")  
 print("")  
  
 # Обновление информации о миссии  
 mission\_id = "M001"  
 payload = {  
 "name": "Перевозка груза (новое название)",  
 "start\_time": "2024-08-08 10:00",  
 "end\_time": "2024-08-08 11:00"  
 }  
 print(f"Update info on Mission ID: {mission\_id}...")  
 missions = update\_mission(mission\_id, payload)  
 print(f"success: {response.get('message')}")  
 print("")  
  
 # Удаление миссии  
 mission\_id = "M001"  
 print(f"Delete Mission ID: {mission\_id}...")  
 response = requests.delete(f"{BASE\_URL}/missions/{mission\_id}")  
 print(f"success: {response.status\_code == 200}")  
 print("")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Вывод сервера:**

**C:\DATA\BPLAPractise\pythonProject\.venv\Scripts\python.exe C:\DATA\BPLAPractise\pythonProject\homework\_3\_9\_server.py**

**\* Serving Flask app 'homework\_3\_9\_server'**

**\* Debug mode: off**

**INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

**\* Running on all addresses (0.0.0.0)**

**\* Running on http://127.0.0.1:5000**

**\* Running on http://192.168.10.7:5000**

**INFO:werkzeug:Press CTRL+C to quit**

**INFO:root:Drone ID:B002 created**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:04] "POST /drones HTTP/1.1" 201 -**

**INFO:root:Drone ID:B001 created**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:13] "POST /drones HTTP/1.1" 201 -**

**INFO:root:Missions ID:M001 created**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:22] "POST /missions HTTP/1.1" 201 -**

**INFO:root:Missions ID:M002 created**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:24] "POST /missions HTTP/1.1" 201 -**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:26] "GET /drones HTTP/1.1" 200 -**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:28] "GET /drones/B001 HTTP/1.1" 200 -**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:30] "GET /missions HTTP/1.1" 200 -**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:32] "GET /missions/M001 HTTP/1.1" 200 -**

**INFO:root:Mission ID:M001 sent to Drone ID:B001**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:36] "POST /missions/M001/B001/send HTTP/1.1" 200 -**

**INFO:root:Drone ID:B001 updated**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:38] "PUT /drones/B001 HTTP/1.1" 200 -**

**INFO:root:Mission ID:M001 updated**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:41] "PUT /missions/M001 HTTP/1.1" 200 -**

**INFO:root:Mission ID:M001 deleted**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:43] "DELETE /missions/M001 HTTP/1.1" 200 -**

**INFO:root:Drone ID:B002 deleted**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:30:57] "DELETE /drones/B002 HTTP/1.1" 200 -**

**INFO:root:Drone ID:B001 deleted**

**INFO:werkzeug:127.0.0.1 - - [07/Aug/2024 16:31:05] "DELETE /drones/B001 HTTP/1.1" 200 -**

**Process finished with exit code 0**

**Вывод Дрона 1:**

**C:\DATA\BPLAPractise\pythonProject\.venv\Scripts\python.exe C:\DATA\BPLAPractise\pythonProject\homework\_3\_9\_drone\_1.py**

**БПЛА зарегистрирован для выполнения задач: Drone ID:B001 created**

**\* Serving Flask app 'homework\_3\_9\_drone\_1'**

**\* Debug mode: off**

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

**\* Running on http://127.0.0.1:8080**

**Press CTRL+C to quit**

**127.0.0.1 - - [07/Aug/2024 16:30:36] "POST /missions HTTP/1.1" 201 -**

**Drone ID: B001 deleted**

**БПЛА выключен**

**Process finished with exit code 0**

**Вывод Дрона 2:**

**C:\DATA\BPLAPractise\pythonProject\.venv\Scripts\python.exe C:\DATA\BPLAPractise\pythonProject\homework\_3\_9\_drone\_2.py**

**БПЛА зарегистрирован для выполнения задач: Drone ID:B002 created**

**\* Serving Flask app 'homework\_3\_9\_drone\_2'**

**\* Debug mode: off**

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

**\* Running on http://127.0.0.1:8081**

**Press CTRL+C to quit**

**Drone ID: B002 deleted**

**БПЛА выключен**

**Process finished with exit code 0**

**Вывод Клиента:**

**C:\DATA\BPLAPractise\pythonProject\.venv\Scripts\python.exe C:\DATA\BPLAPractise\pythonProject\homework\_3\_9\_client.py**

**Миссия: M001 Перевозка груза создана**

**Миссия: M002 Патрулирование создана**

**Request info on all active UAVs**

**{'B001': {'control\_url': 'http://localhost:8080',**

**'id': 'B001',**

**'location': {'altitude': 0, 'latitude': 50.45, 'longitude': 30.52},**

**'name': 'Autel Robotics s.v.Defender',**

**'status': 'active'},**

**'B002': {'control\_url': 'http://localhost:8081',**

**'id': 'B002',**

**'location': {'altitude': 0, 'latitude': 51.45, 'longitude': 30.52},**

**'name': 'DJI mavic 3 Pro',**

**'status': 'active'}}**

**Request info on Drone ID: B001...**

**{'control\_url': 'http://localhost:8080',**

**'id': 'B001',**

**'location': {'altitude': 0, 'latitude': 50.45, 'longitude': 30.52},**

**'name': 'Autel Robotics s.v.Defender',**

**'status': 'active'}**

**Request info on all missions**

**{'M001': {'end\_time': '2024-08-08 12:00',**

**'id': 'M001',**

**'name': 'Перевозка груза',**

**'route': [{'altitude': 100, 'latitude': 50.45, 'longitude': 30.52},**

**{'altitude': 200, 'latitude': 50.47, 'longitude': 30.55},**

**{'altitude': 300, 'latitude': 50.49, 'longitude': 30.58}],**

**'start\_time': '2024-08-08 09:00'},**

**'M002': {'end\_time': '2024-08-08 10:00',**

**'id': 'M002',**

**'name': 'Патрулирование',**

**'route': [{'altitude': 100, 'latitude': 51.45, 'longitude': 30.52},**

**{'altitude': 200, 'latitude': 51.47, 'longitude': 30.55},**

**{'altitude': 300, 'latitude': 51.49, 'longitude': 30.58}],**

**'start\_time': '2024-08-08 02:00'}}**

**Request info on Mission ID: M001...**

**{'end\_time': '2024-08-08 12:00',**

**'id': 'M001',**

**'name': 'Перевозка груза',**

**'route': [{'altitude': 100, 'latitude': 50.45, 'longitude': 30.52},**

**{'altitude': 200, 'latitude': 50.47, 'longitude': 30.55},**

**{'altitude': 300, 'latitude': 50.49, 'longitude': 30.58}],**

**'start\_time': '2024-08-08 09:00'}**

**Send mission to Drone ...**

**success: Mission ID:M001 sent to Drone ID:B001**

**Update info on Drone ID: B001...**

**success: Mission ID:M001 sent to Drone ID:B001**

**Update info on Mission ID: M001...**

**success: Mission ID:M001 sent to Drone ID:B001**

**Delete Mission ID: M001...**

**success: True**

**Process finished with exit code 0**