hw\_num: 19

В этом задании вы создадите минимальное REST API на Flask для сайта фитнес клуба, используя уже знакомую модель данных и PeeWee ORM. Вы реализуете базовые маршруты для работы с тренерами и записями на тренировки: получение, создание, обновление и удаление.

**### Технологии: 🛠️**

- Python

- Flask

- PeeWee ORM

- JSON

- HTTP-методы (GET, POST, PUT, DELETE)

**## Задание 👷‍♂️**

**### 1. Настройка структуры проекта**

Создайте файл, который будет содержать:

- Инициализацию приложения Flask

- Модели данных на основе PeeWee (аналогично предыдущему заданию)

- Обработчики маршрутов

Модели должны соответствовать созданным в предыдущем ДЗ:

- `trainers` — тренера фитнес клуба

- `appointments` — запись на тренировку

**### 2. Маршруты API**

Реализуйте следующие маршруты в файле:

| Метод | Маршрут | Описание | Статус ответа |

|-------|---------|----------|--------------|

| GET | `/trainers` | Получить список всех тренеров | 200 |

| GET | `/trainers/<id>` | Получить информацию о тренере по ID | 200, 404 |

| POST | `/ trainers ` | Добавить нового тренера | 201 |

| PUT | `/ trainers /<id>` | Обновить информацию о тренере | 200, 404 |

| DELETE | `/ trainers /<id>` | Удалить тренера | 204, 404 |

| GET | `/appointments` | Получить все записи на тренировки с опциональной сортировкой | 200 |

| GET | `/appointments/<id>` | Получить запись по ID | 200, 404 |

| GET | `/appointments/trainers/<trainers\_id>` | Получить все записи для заданного тренера | 200 |

| POST | `/appointments` | Создать новую запись | 201 |

| PUT | `/appointments/<id>` | Обновить запись | 200, 404 |

| DELETE | `/appointments/<id>` | Удалить запись | 204, 404 |

**### 3. Параметры запросов**

Для маршрута `/appointments`:

- Добавьте поддержку параметров запроса:

  - `sort\_by` — поле, по которому производится сортировка

  - `direction` — направление сортировки (`asc` или `desc`)

Например: `/appointments?sort\_by=date&direction=desc`

**### 4. Формат JSON-ответов**

Все API-запросы должны возвращать данные в формате JSON с корректными HTTP-статусами.

Обязательно настройте корректное отображение кириллических символов.

Для корректного отображения кириллических символов в JSON используйте:

json.dumps(data, ensure\_ascii=False)

В контексте Flask:

>return json.dumps(data, ensure\_ascii=False), 200, {'Content-Type': 'application/json; charset=utf-8'}

**### 5. Обработка ошибок**

Реализуйте базовую обработку ошибок:

- При попытке получить несуществующий ресурс — возврат 404 с сообщением об ошибке

- При передаче некорректных данных — возврат 400 с информацией о проблеме

- При внутренних ошибках сервера — возврат 500 с описанием проблемы

**### 6. Конвертация данных**

Создайте функции-помощники для преобразования объектов PeeWee в словари Python:

| Функция | Описание |

|---------|----------|

| `trainers\_to\_dict(master)` | Преобразует объект Trainers в словарь |

| `appointments\_to\_dict(appointment)` | Преобразует объект Appointments в словарь с включением информации о тренере |

| `validate\_trainers\_data(data)` | Проверяет корректность данных для создания/обновления тренера |

| `validate\_appointments\_data(data)` | Проверяет корректность данных для создания/обновления записи |

**### 7. Запуск приложения**

В конце файла добавьте запуск приложения с включенным режимом отладки:

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

    # Создание таблиц, если они не существуют

    db.connect()

    db.create\_tables([Trainers, Appointments], safe=True)

    db.close()

    # Запуск веб-сервера

    app.run(debug=True)

**### 8. Пример структуры ответов**

**\*\*Получение списка тренеров:\*\***

```json

{

  "trainers": [

    {

      "id": 1,

      "first\_name": "Иван",

"middle\_name": "Иванович",

      "last\_name": "Иванов",

      "phone": "+7-900-123-45-67"

    },

    {

      "id": 2,

      "first\_name": "Пётр",

      "last\_name": "Петров",

      "middle\_name": null,

      "phone": "+7-900-765-43-21"

    }

  ]

}

```

**\*\*Получение записи на услугу по ID:\*\***

```json

{

  "appointments": {

    "id": 1,

    "client\_name": "Алексей Сидоров",

    "clients\_phone": "+7-999-888-77-66",

    "date": "2025-04-26 14:30:00",

    "trainers": {

      "id": 1,

      "first\_name": "Иван",

      "last\_name": "Иванов"

    },

  }

}

```

**## Таблица методов и функций**

| Тип | Название | Описание |

|-----|----------|----------|

| **\*\*PeeWee\*\*** | `Model.select()` | Выборка записей из таблицы |

| **\*\*PeeWee\*\*** | `Model.get(Model.id == id)` | Получение записи по ID |

| **\*\*PeeWee\*\*** | `Model.create(\*\*data)` | Создание новой записи |

| **\*\*PeeWee\*\*** | `Model.update(\*\*data).where(...)` | Обновление записи |

| **\*\*PeeWee\*\*** | `Model.delete().where(...)` | Удаление записи |

| **\*\*Flask\*\*** | `@app.route('/<path>', methods=['GET', 'POST', ...])` | Декоратор для определения маршрута |

| **\*\*Flask\*\*** | `request.json` | Получение данных JSON из запроса |

| **\*\*Flask\*\*** | `request.args.get('param')` | Получение параметра из URL |

| **\*\*Функция\*\*** | `master\_to\_dict(master)` | Преобразование объекта Master в словарь |

| **\*\*Функция\*\*** | `appointment\_to\_dict(appointment)` | Преобразование объекта Appointment в словарь |

| **\*\*Функция\*\*** | `validate\_master\_data(data)` | Валидация данных для мастера |

| **\*\*Функция\*\*** | `validate\_appointment\_data(data)` | Валидация данных для записи |

>**### Полезные методы PeeWee для сортировки**

>- `.order\_by(Model.field)` — сортировка по возрастанию

>- `.order\_by(Model.field.desc())` — сортировка по убыванию

>Например: `Appointments.select().order\_by(Appointments.date.desc())`

**## Примеры API-запросов**

**### Создание нового тренера (POST /trainers)**

Запрос:

```json

{

  "first\_name": "Сергей",

"middle\_name": "Сергеевич",

  "last\_name": "Сергеев",

  "phone": "+7-911-222-33-44"

}

```

**### Создание новой записи (POST /appointments)**

Запрос:

```json

{

  "client\_name": "Михаил Кузнецов",

  "client\_phone": "+7-922-333-55-66",

  "master\_id": 1,

}

```

>**### Критерии проверки 👌**

>

>1. **\*\*Структура и инициализация:\*\***

>

> - Корректная инициализация Flask-приложения и ORM-моделей

> - Создание таблиц при первом запуске, если они не существуют

>

>2. **\*\*API-маршруты:\*\***

>

> - Реализованы все указанные маршруты с правильными HTTP-методами

> - Корректные форматы ответов и коды статусов

> - Правильная обработка несуществующих ресурсов (404)

>

>3. **\*\*Работа с данными:\*\***

>

> - Корректное извлечение параметров из запроса

> - Работа с сортировкой для маршрута `/appointments`

> - Правильная валидация данных перед сохранением

>

>4. **\*\*Обработка кириллицы:\*\***

>

> - Корректная настройка JSON для работы с кириллицей

> - Содержимое всех ответов на русском языке отображается без экранирования

>

>5. **\*\*Конвертация и валидация:\*\***

>

> - Реализованы функции конвертации объектов PeeWee в словари

> - Проверка валидности входящих данных

>

>6. **\*\*Оформление кода:\*\***

>

> - Код соответствует PEP-8

> - Присутствуют аннотации типов

> - Присутствуют поясняющие комментарии на русском языке

---

При запуске должен запуститься Flask-сервер, к которому можно отправлять HTTP-запросы для работы с API фитнес клуба

.