**Тестовое задание**

# Время выполнения (ориентир)

подготовка окружения - не более 10 минут (настройка выполняется автоматически через подготовленный файл **docker-compose**, см. инструкцию далее) выполнение задания - не более часа (время выполнения зависит от уровня кандидата); валидация данных не требуется

использование ORM для выполнения запросов к БД не обязательно, можно использовать DB::select

# Подготовка к выполнению задания (настройка окружения)

Для вашего удобства подготовлен файл **docker-compose.yml** для быстрого развертывания окружения с уже подготовленной базой данных [PostgreSQL](https://www.postgresql.org/) и пустым проектом на [Laravel.](https://laravel.com/)

Для запуска необходимо наличие установленного ПО [docker.](https://docs.docker.com/engine/install/)

В файле **.env** можете при необходимости (например, если порты 7777 и/или 5555 уже используются) настроить следующие параметры:

**PORT** - номер порта, по которому будет доступен сайт (по умолчанию 7777, т.е. после запуска сервера в браузере сайт можно будет открыть по адресу <http://localhost:7777/>)

**DB\_PORT\_EXTERNAL** - порт СУБД PostgreSQL, через который можно будет подключиться, если захотите поработать с базой данных напрямую, например через [DBeaver](https://dbeaver.io/download/) или иное ПО (по умолчанию порт 5555)

В командной строке, находясь в директории с проектом, запустите

Предупреждение: *при первом запуске будут скачиваться необходимые docker-образы, а также зависимости через Composer; имейте это в виду, если используете лимитное подключение к internet с ограничением трафика*.

# Исходные данные

В базе данных содержатся три таблицы:

aircrafts - справочник бортовых номеров airports - справочник аэропортов

flights - основная таблица в данном задании - таблица перелетов aircraft\_id - id бортового номера airport\_id1 - id аэропорта вылета airport\_id2 - id аэропорта посадки takeoff - время вылета landing - время посадки

cargo\_load - объём загрузки в аэропорту **вылета** cargo\_offload - объём выгрузки в аэропорту **посадки**

Со структурой таблиц можно также ознакомиться непосредственно в БД, либо в скрипте, из которого она автоматически инициализируется при развертывании через **dockercompose**:

**entrypoint/database**:

**010-db\_init.sql** - создание таблиц

**020-aircrafts.sql** - заполнение тестовыми данными таблицы бортовых номеров

**030-airports.sql** - заполнение тестовыми данными таблицы аэропортов

**040-flights.sql** - заполнение тестовыми данными таблицы перелетов

# Постановка задачи

Реализуйте API endpoint:

HTTP метод: GET URL: /api/aircraft\_airports

Параметры:

tail - бортовой номер воздушного судна

date\_from - начало периода (формат: yyyy-mm-dd hh:mm ) date\_to - конец периода (формат: yyyy-mm-dd hh:mm )

Endpoint возвращает в формате JSON массив аэропортов, в которых находилось выбранное воздушное судно за заданный период с указанием:

airport\_id - id аэропорта code\_iata - код IATA аэропорта code\_icao - код ICAO аэропорта

cargo\_offload - объёма разгрузки **в этом аэропорту** cargo\_load - объёма загрузки **в этом аэропорту** landing - времени посадки **в этот аэропорт**

takeoff - времени вылета **из этого аэропорта**

# Пример

Результат вызова:

Для понимания, этот ответ можно трактовать так:

Борт TEST-001 за период с 01.01.2023 22:00 по 02.01.2023 15:00 пребывал в трёх аэропортах:

1. BHA/SESV - сюда сел в 01.01.23 20:54 , по прилёту не разгружался, взял на борт 10 тонн груза, в следующий аэропорт вылетел в 01.01.23 23:36
2. COA/KO22 - сюда сел в 02.01.23 04:00 , по прилету не разгружался, взял на борт 90 тонн груза, вылетел в 02.01.23 09:06
3. DKR/GOOY - сюда сел в 02.01.23 14:06:49 , разгрузил 80 тонн груза, погрузок не было, вылетел отсюда в 02.01.23 15:43

2003 © Volga-Dnepr Airlines