

ТЕСТОВОЕ ЗАДАНИЕ

BACKEND

Необходимо разработать REST API для ведения учета питомцев (собак и кошек) и сделать возможность выгрузить список питомцев в json через командную строку.

Требуется использовать:

Python, Django, PostgreSQL и Docker (Compose).

или

Go, Gorm, PostgreSQL и Docker (Compose).

Результат задания нужно залить на **публичный git**, написать к нему **README** (инструкция, как разворачивать), где-нибудь (Heroku, DigitalOcean, AWS, etc) **развернуть и наполнить демо-данными**.

API

A. Необходимо реализовать API Key Authentication:

- в конфиге проекта нужно добавить параметр **API_KEY**
- все обработчики запросов должны принимать header **X-API-KEY**
- переданный **X-API-KEY** сравнивается с эталонным **API_KEY**
если совпадает - обрабатываем запрос
не совпадает - **401 Unauthorized**

B. Необходимо реализовать следующие endpoint'ы:

1. POST /pets

Создать питомца

request body

```
{  
    // нужно спроектировать  
}
```

response body

```
{
  "id": "433a203f-5480-442b-b599-01060d988d87",
  "name": "boy",
  "age": 7,
  "type": "dog",
  "photos": [],
  "created_at": "2021-05-18T19:10:17"
}
```

2. POST /pets/{id}/photo

Загрузить фотографию питомца

form data

file: `binary`

response body

```
{
  "id": "4cb15cbd-243e-427d-bb60-76f591b13cf8",
  "url": "https://address/filename.extension"
}
```

3. GET /pets

Получить список питомцев

query parameters

limit: integer (optional, default=20)

offset: integer (optional, default=0)

has_photos: boolean (optional)

*has_photos: **true** - вернуть записи с фотографиями*

*has_photos: **false** - вернуть записи без фотографий*

has_photos was not provided - вернуть все записи

response body

```
{
  "count": 341,
  "items": [
    {
```



```

        "id": "433a203f-5480-442b-b599-01060d988d87",
        "name": "boy",
        "age": 7,
        "type": "dog",
        "photos": [{
            "id": "4cb15cbd-243e-427d-bb60-76f591b13cf8",
            "url": "https://address/filename.extension"
        }],
        "created_at": "2021-05-18T19:10:17"
    },
    {
        "id": "6796fa0d-f405-4363-881d-6d3694a9655c",
        "name": "girl",
        "age": 3,
        "type": "cat"
        "photos": [
            {
                "id": "4cb15cbd-243e-427d-bb60-76f591b13cf8",
                "url": "https://address/filename.extension"
            },
            {
                "id": "ce8c4656-0860-48fd-9d96-093270f423a1",
                "url": "https://address/filename.extension"
            }
        ],
        "created_at": "2021-04-11T22:39:58"
    }
]
}

```

4. DELETE /pets Удалить питомцев

response body

```

{
  "ids": [
    "433a203f-5480-442b-b599-01060d988d87",
    "587e5358-6407-4fff-9f86-853ce1849ac7",
    "6796fa0d-f405-4363-881d-6d3694a9655c"
  ]
}

```

response body

```
{
  "deleted": 2,
  "errors": [
    {
      "id": "587e5358-6407-4fff-9f86-853ce1849ac7",
      "error": "Pet with the matching ID was not found."
    }
  ]
}
```

NOTE: при удалении питомца нужно удалять его фотографии (записи в базе и файлы)

CLI

Необходимо реализовать возможность выгрузки питомцев из командной строки в **stdout** в JSON формате.

Команда должна принимать на вход НЕобязательный параметр

has_photos: boolean

Формат выгрузки:

```
{
  "pets": [
    {
      "id": "433a203f-5480-442b-b599-01060d988d87",
      "name": "boy",
      "age": 7,
      "type": "dog",
      "photos": ["https://address/filename.extension"],
      "created_at": "2021-05-18T19:10:17"
    },
    {
      "id": "6796fa0d-f405-4363-881d-6d3694a9655c",
      "name": "girl",
      "age": 3,
      "type": "cat",
      "photos": [
        "https://address/filename.extension",
        "https://address/filename.extension"
      ],
      "created_at": "2021-04-11T22:39:58"
    }
  ]
}
```