Задание

Интернет-магазин подарков хочет запустить акцию в разных регионах. Чтобы стратегия продаж была эффективной, необходимо произвести анализ рынка.

У магазина есть поставщик, регулярно присылающий выгрузки данных с информацией о жителях. Проанализировав их, можно выявить спрос на подарки в разных городах у жителей разных возрастных групп по месяцам.

Ваша задача - разработать на python REST API сервис, который сохраняет переданные ему наборы данных (выгрузки от поставщика) с жителями, позволяет их просматривать, редактировать информацию об отдельных жителях, а также производить анализ возрастов жителей по городам и анализировать спрос на подарки в разных месяцах для указанного набора данных.

Должна быть реализована возможность параллельно загрузить несколько наборов данных с разными идентификаторами. А также независимо друг от друга изменять и анализировать их.

Сервис необходимо развернуть на предоставленной виртуальной машине на 0.0.0.0:8080.

Описание обработчиков REST API

1: POST /imports

Принимает на вход набор с данными о жителях в формате `json` (поля не могут быть `null`) и сохраняет его с уникальным идентификатором.

Поставщик предупредил, что в разных выгрузках `citizen_id` не уникален и может повторяться у разных жителей, не закладывайтесь на то, что `citizen_id` будут уникальны между выгрузками от поставщика.

Родственные связи `relatives` актуальны только в рамках одной выгрузки.

```
POST /imports
{
  "citizens": [{
    "citizen_id": 1,
    "town": "Москва",
```

```
    "street": "Льва Толстого",
    "building": "16к7стр5",
    "appartement": 7,
    "name": "Иванов Иван Иванович",
    "birth_date": "01.02.2000",
    "gender": "male",
    "relatives": [2, 28]  # id родственников
  }, ...]
}
```

В случае успеха возвращается ответ с HTTP статусом `201 Created` и идентификатором импорта:

```
HTTP 201
{
  "data": {
    "import_id": 1
  }
}
```

2: PATCH /imports/$import_id/citizens/$citizen_id

Изменяет информацию о жителе в указанном наборе данных.

На вход подается JSON в котором можно указать любые данные о жителе (`name`, `gender`, `birth_date`, `relatives`, `town`, `street`, `building`, `appartement`), кроме `citizen_id`.

В запросе должно быть указано хотя бы одно поле, значения не могут быть `null`.

```
PATCH /imports/6/citizens/1
{
    "town": "Керчь",
    "street": "Иосифа Бродского"
}
```

Возвращается актуальная информация о жителе.

```
HTTP 200
{
  "data": {
    "citizen_id": 1,
    "town": "Керчь",
```

```
    "street": "Иосифа Бродского",
    "building": "16к7стр5",
    "appartement": 7,
    "name": "Иванов Иван Иванович",
    "birth_date": "01.02.2000",
    "gender": "male",
    "relatives": [2, 28]
  }
}
```

3: GET /imports/$import_id/citizens

Возвращает список всех жителей для указанного набора данных.

```
HTTP 200
{
  "data": [{
    "citizen_id": 1,
    "town": "Керчь",
    "street": "Иосифа Бродского",
    "building": "16к7стр5",
    "appartement": 7,
    "name": "Иванов Иван Иванович",
    "birth_date": "01.02.2000",
    "gender": "male",
    "relatives": [2, 28]
  }, ...]
}
```

4: GET /imports/$import_id/citizens/birthdays

Возвращает жителей и количество подарков, которые они будут покупать своим ближайшим родственникам (1-го порядка), сгруппированных по месяцам из указанного набора данных.

Ключом должен быть месяц (нумерация должна начинаться с единицы, `"1"` - январь, `"2"` - февраль и т.п.).

Если в импорте в каком-либо месяце нет ни одного жителя с днями рождения ближайших родственников, значением такого ключа должен быть пустой список (см. пример ниже для марта).

```
HTTP 200
{
```

```
  "data": {
    "1": [{
      "citizen_id": 1,
      "presents": 20
    }],
    "2": [{
      "citizen_id": 2,
      "presents": 7
    }],
    "3": [],
    ...
    "12": [{
      "citizen_id": 3,
      "presents": 4
    }, {
      "citizen_id": 8,
      "presents": 2
    }]
  }
}
```

5: GET /imports/$import_id/towns/stat/percentile/age

Возвращает статистику по городам для указанного набора данных в разрезе возраста жителей: p50, p75, p99, где число - это значение перцентиля.

```
HTTP 200
{
  "data": [
    {
      "town": "Москва",
      "p50": 20,
      "p75": 45,
      "p99": 100
    },
    {
      "town": "Санкт-Петербург",
      "p50": 17,
      "p75": 35,
      "p99": 80
    }
  ]
}
```

Что означает:

- `"p50": 20,` - 50% жителей меньше 20 лет
- `"p75": 45,` - 75% жителей меньше 45 лет

На что обратить внимание

Для прохождения проверки обратите внимание на следующее:

- Статусы HTTP ответов
- Структура json на входе и выходе
- Типы данных (строки, числа)
- Формат даты
- Таймаут на вызов каждого обработчика - 10 секунд при количестве жителей 10,000
- URL без trailing slash
- Реализация перцентилей должна соответствовать numpy.percentile с `interpolation='linear'`

Как производится оценка задания

Задание считается выполненным, если в REST API реализованы и проходят валидацию три первых обработчика. Также учитывается:

- Наличие реализованных обработчиков:
  - 4: `GET /imports/$import_id/citizens/birthdays`
  - 5: `GET /imports/$import_id/towns/stat/percentile/age`
- Наличие валидации входных данных (на некорректные входные данные сервис отвечает HTTP статусом `400 Bad Request`);
- Наличие файла `README` в корне репозитория с инструкциями по установке, развертыванию и запуску тестов;
- Явно описанные внешние python-библиотеки (зависимости);
- Наличие тестов;
- Автоматическое возобновление работы REST API после перезагрузки виртуальной машины.