Техническая документация

Тестовое задание на позицию Python-разработчика.

Общее описание программы

Настоящий документ предназначен для ознакомления пользователя с функциональными возможностями написанной программы в рамках выполнения тестового задания для **стажировки** в компании **Infotecs** на позицию **Python-разработчика**.

Программа служит для реализации http-сервера для предоставления информации по географическим объектам с помощью REST API сервиса.

Были выполнены все задания, в том числе и **дополнительные**.

Выполнил задание: Цыденов Саян Баирович

Библиотеки

- Для реализации серверной составляющей был выбран фреймворк **Flask**
- В качестве обработчика данных используется **Pandas**

Реализованные методы программы

1. Поиск по идентификатору города
2. Поиск по страницам
3. Сравнивание двух городов
4. Поиск совпадения по части названия
5. Тестирование программы

Запуск программы

Зависимости расположены в файле requirements.txt.

Скрипт запускается следующим образом:

```
python3 script.py.
```

После запуска скрипта по адресу 127.0.0.1 и порту 8000 можно обращаться с указанными выше функциями.

Все методы возвращают данные в формате json. В полученном ответе содержится код ответа, описание и сами данные.

Описание методов программы

1. Поиск по идентификатору города

Метод получает на вход идентификатор города и возвращает информацию о нем.

Для обращению к серверу используется запрос:

> /geonameid/(Идентификатор города)

Например, запрос

> /geonameid/1489421

возвратит информацию о Томске, код ответа и описание ответа:


```
{
  "status": 200,
  "description": "OK",
  "info": [
    {
      "geonameid": 1489421,
      "name": "Tomsk Oblast",
      "asciiname": "Tomsk Oblast",
      "alternatenames": "Oblast de Tomsk,Tomsk,Tomsk Oblast,Tomsk Oblast',Tomsk
Oblast’,Tomskaja Oblast',Tomskaja oblast,Tomskaja oblast',Tomskaya
Oblast',Tomskaya Oblast’,Томская Область,Томская область",
      "latitude": 58.5,
      "longitude": 82.5,
      "feature class": "A",
      "feature code": "ADM1",
      "country code": "RU",
      "admin1 code": "75",
      "admin2 code": NaN,
      "admin3 code": NaN,
      "admin4 code": NaN,
      "population": 1049069,
      "elevation": NaN,
      "dem": 55,
      "timezone": "Asia/Krasnoyarsk",
      "modification date": "2020-07-08"
    }
  ]
}
```


В случае ненахождения города, будет возвращен статус 404, описание и пустой список городов:


```
{
  "status": 404,
  "description": "City is not found",
  "info": []
}
```


2. Поиск по страницам

Метод делит весь массив данных на равные части и возвращает информацию о городах из выбранной части.

На вход принимает нужную страницу и сколько данных должно находиться на каждой странице.

Для обращению к серверу используется запрос:

```
/page/(страница)&count=(делитель)
```

Например, запрос

```
/page/40&count=10
```

возвратит 40-ю страницу, на которой будет информация о 10 городах.

В случае **выхода за границы массива данных**, метод возвратит ошибку 404, описание "Out of range" и пустой массив городов.

3. Сравнивание двух городов

Метод получает на вход название двух городов и возвращает информацию о них, о том, какой город находится севернее и разницу в часовых поясах.

Если возникает конфликт одинаковых имен, то берется город с наибольшим населением.

Для обращению к серверу используется запрос:

```
/compare/(Город_1)&(Город_2)
```

Например, запрос

```
/compare/Москва&Томск
```

возвратит следующий ответ:


```
{
  "status": 200,
  "description": "OK",
  "north": "Томск",
  "text_description_north": "Город Томск находится севернее, чем город Москва.
<br>",
  "delta_time": "Разница в часовых поясах городов составляет 4 ч.",
  "info_about_cities": [
    {
      "geonameid": 524894,
      "name": "Moskva",
      "asciiname": "Moskva",
      "alternatenames":
"Maskva,Moscou,Moscow,Moscu,Moscú,Moskau,Moskou,Moskovu,Moskva,Məskeu,Москва,Məске
y",
      "latitude": 55.76167,
      "longitude": 37.60667,
```

```
      "feature class": "A",
      "feature code": "ADM1",
      "country code": "RU",
      "admin1 code": "48",
      "admin2 code": NaN,
      "admin3 code": NaN,
      "admin4 code": NaN,
      "population": 11503501,
      "elevation": NaN,
      "dem": 161,
      "timezone": "Europe/Moscow",
      "modification date": "2020-03-31"
    },
    {
      "geonameid": 1489421,
      "name": "Tomsk Oblast",
      "asciiname": "Tomsk Oblast",
      "alternatenames": "Oblast de Tomsk,Tomsk,Tomsk Oblast,Tomsk Oblast',Tomsk
Oblast’,Tomskaja Oblast',Tomskaja oblast,Tomskaja oblast',Tomskaya
Oblast',Tomskaya Oblast’,Томская Область,Томская область",
      "latitude": 58.5,
      "longitude": 82.5,
      "feature class": "A",
      "feature code": "ADM1",
      "country code": "RU",
      "admin1 code": "75",
      "admin2 code": NaN,
      "admin3 code": NaN,
      "admin4 code": NaN,
      "population": 1049069,
      "elevation": NaN,
      "dem": 55,
      "timezone": "Asia/Krasnoyarsk",
      "modification date": "2020-07-08"
    }
  ]
}
```


В случае неправильного написания названия:

> /compare/Неправильный-город&Томск

метод возвратит:


```
{
  "status": 404,
  "description": "Неправильный-город is not found"
}
```


4. Поиск совпадения по части названия

Метод получает на вход часть от названия населенного пункта и возвращает совпадающие имена из базы данных.

Количество соответствий ограничено подходящими альтернативными именами 10 пунктов с наибольшим населением.

Для обращению к серверу используется запрос:

> /guess_town_name/(Часть названия города)

Например, запрос

> /guess_town_name/Сан

возвратит:

```
{
  "status": 200,
  "guessed_names": [
    "Санкт-Петербург",
    "Санкт",
    "Санкт",
    "Санкт-Петербург",
    "Санчурск",
    "Сангаар",
    "Сангар",
    "Сандата",
    "Сандово",
    "Сангородок",
    "Санниково",
    "Санатория",
    "Санатория",
    "Санатория",
    "Санатория",
    "Санатория",
    "Санатория",
    "Санатория"
  ]
}
```

Тестирование кода

Для тестирования кода была написана функция test_with_random_results в файле data_handler.py. Она тестирует некоторые случаи, которые могут возникнуть при эксплуатации программы.

Далее будет показан пример вывода в консоль работы этой функции. Информация по городам будет намеренно сокращенна здесь для того, чтобы не загромождать документацию.

```
Проверка первого метода: OK {'status': 200, 'description': 'OK', 'info':
[{'geonameid': 2050401, 'name': 'Gora Kamni Shakhtanskiye', ... , 'modification
date': '2019-09-05'}]}

 Несуществующий geonameid {'status': 404, 'description': 'City is not found',
'info': []}
```

```
Проверка второго метода: OK {'status': 200, 'description': 'OK', 'pages':
[{'geonameid': 451747, 'name': 'Zyabrikovo', ... , 'modification date': '2011-07-
09'},

{'geonameid': 451748, 'name': 'Znamenka', ... , 'modification date': '2011-07-
09'},
... ,
{'geonameid': 451757, 'name': 'Zamush’ye', ... , 'modification date': '2011-07-
09'}]}

 Номер страницы задан больше, чем номер существующей {'status': 404,
'description': 'Out of range', 'pages': []}
```

```
Проверка третьего метода: OK {'status': 200, 'description': 'OK', 'north':
'Томск', 'text_description_north': 'Город Томск находится севернее, чем город
Москва. <br>', 'delta_time': 'Разница в часовых поясах городов составляет 4 ч.',
'info_about_cities': [{'geonameid': 1489421, 'name': 'Tomsk Oblast', 'asciiname':
'Tomsk Oblast',
...,
 'timezone': 'Asia/Krasnoyarsk', 'modification date': '2020-07-08'},

{'geonameid': 524894, 'name': 'Moskva', 'asciiname': 'Moskva', ... , 'timezone':
'Europe/Moscow', 'modification date': '2020-03-31'}]}

 Отсутствие города №1 {'status': 404, 'description': 'wrong_city1 is not found'}

 Отсутствие города №2 {'status': 404, 'description': 'wrong_city2 is not found'}
```

```
Проверка четвертого метода: OK {'status': 200, 'guessed_names': ['Санкт-
Петербург', 'Санкт', 'Санкт', 'Санкт-Петербург', 'Санкт-Петербургский', 'Санкт-
Петербургский']}
 Не найдено соответствие {'status': 404, 'guessed_names': []}
```