

Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет «Инфокоммуникационных Технологий»
Направление подготовки «Программирование в
инфокоммуникационных системах»

Тестирование программного обеспечения
Лабораторная работа №2

Выполнила:
Улитина Мария Сергеевна
Группа №3322
Проверил:
Кочубеев Николай Сергеевич

Санкт-Петербург
2024

Цель работы: научиться писать интеграционные тесты.

Задачи:

1. Выбор репозитория с GitHub

- Необходимо выбрать открытый проект на платформе GitHub. Подходят как публичные проекты, так и ваши собственные пет-проекты.

2. Анализ взаимодействий между модулями

- Изучите структуру выбранного проекта и определите, какие модули или компоненты взаимодействуют друг с другом.

3. Написание интеграционных тестов

- Создайте тесты, проверяющие корректность взаимодействия между ключевыми компонентами системы. Например, протестируйте взаимодействие между слоем бизнес-логики и базой данных, обработку данных между API и фронтендом, или другие важные интеграции.

- Напишите минимум 5 интеграционных тестов, проверяющих основные сценарии взаимодействий, включая граничные случаи и потенциальные ошибки.

Ход работы:

1-2. Выбор репозитория и анализ тестируемых функций.

Принято решение реализовывать задачи на языке python. Был выбран репозиторий по ссылке: <https://github.com/turkupy/beginners-python-weather>
Этот репозиторий представляет из себя сайт для изучения погоды в разных городах и странах. Воспользоваться функционалом можно по ссылке: <https://openweathermap.org/>

Проанализируем функционал:

Функциональные элементы

1. Frontend:

- интерфейсы для отображения данных о погоде на стороне клиента.

2. Backend:

- маршруты для обработки запросов к API OpenWeatherMap;
- форматирование и обработка ответов от API для отправки клиенту.

3. API:

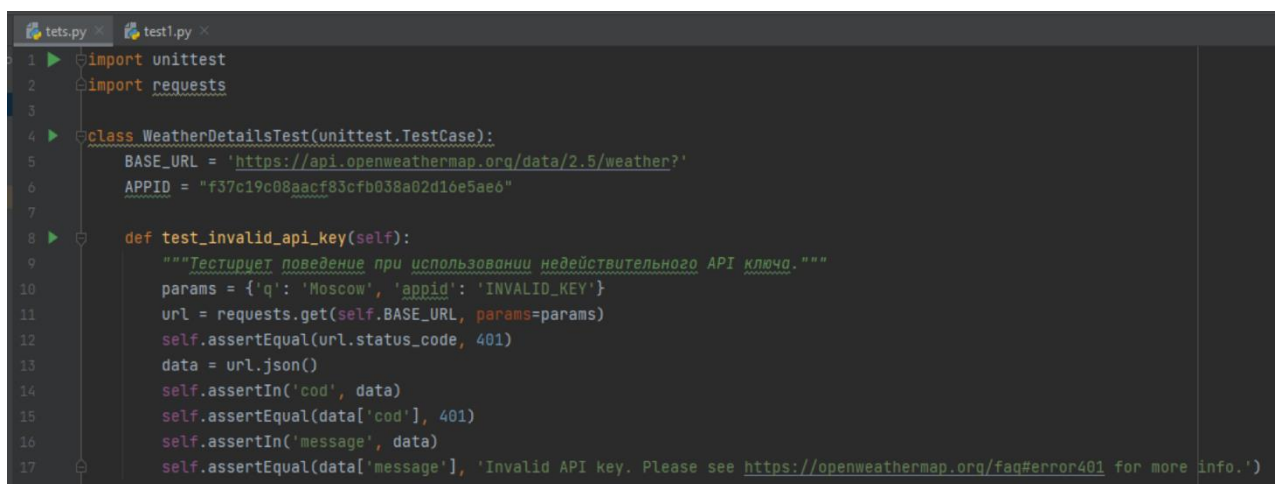
- сервер, который отвечает за обработку запросов к внешнему API;
- преобразование и передача данных между клиентской частью и внешним API.

Критические части системы, которые должны быть протестированы

- использование некорректного API;
- отсутствие города в запросе;
- передача некорректного значения температуры;
- запрос погоды для города;
- запрос погоды по географическим координатам.

3. Написание тестов

Для каждого пункта, описанного в критических частях системы, был создан интеграционный тест. Первые три теста приведены на рисунках 1 – 2.

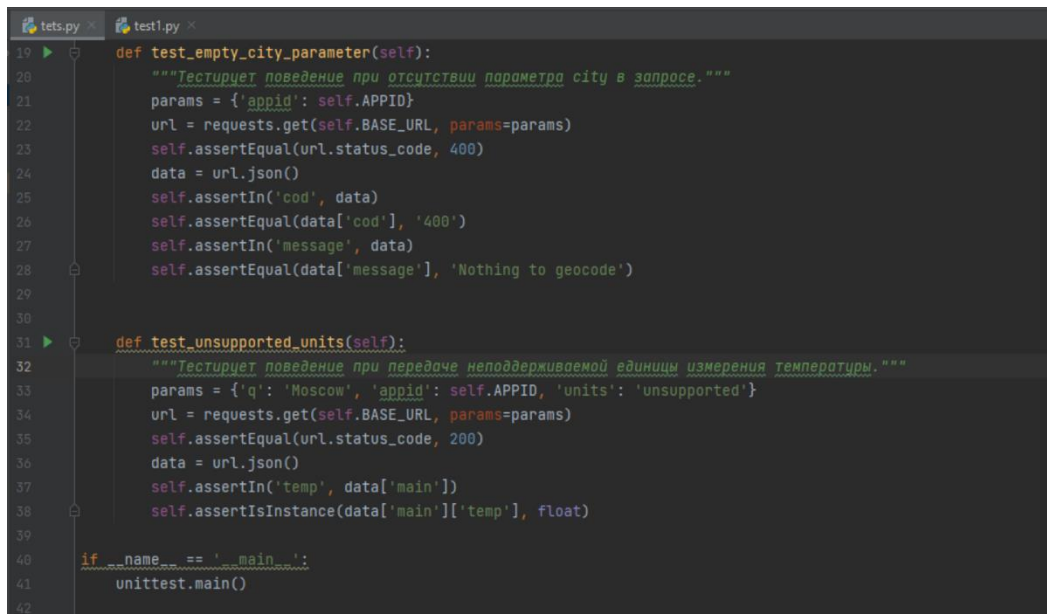


```

1  import unittest
2  import requests
3
4  class WeatherDetailsTest(unittest.TestCase):
5      BASE_URL = 'https://api.openweathermap.org/data/2.5/weather?'
6      APPID = "f37c19c08aacf83cfb038a02d16e5ae6"
7
8  def test_invalid_api_key(self):
9      """Тестирует поведение при использовании недействительного API ключа."""
10     params = {'q': 'Moscow', 'appid': 'INVALID_KEY'}
11     url = requests.get(self.BASE_URL, params=params)
12     self.assertEqual(url.status_code, 401)
13     data = url.json()
14     self.assertIn('cod', data)
15     self.assertEqual(data['cod'], 401)
16     self.assertIn('message', data)
17     self.assertEqual(data['message'], 'Invalid API key. Please see https://openweathermap.org/faq#error401 for more info.')

```

Рисунок 1 – Интеграционный тест 1.



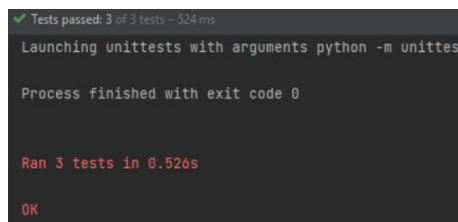
```

19 def test_empty_city_parameter(self):
20     """Тестирует поведение при отсутствии параметра city в запросе."""
21     params = {'appid': self.APPID}
22     url = requests.get(self.BASE_URL, params=params)
23     self.assertEqual(url.status_code, 400)
24     data = url.json()
25     self.assertIn('cod', data)
26     self.assertEqual(data['cod'], '400')
27     self.assertIn('message', data)
28     self.assertEqual(data['message'], 'Nothing to geocode')
29
30
31 def test_unsupported_units(self):
32     """Тестирует поведение при передаче неподдерживаемой единицы измерения температуры."""
33     params = {'q': 'Moscow', 'appid': self.APPID, 'units': 'unsupported'}
34     url = requests.get(self.BASE_URL, params=params)
35     self.assertEqual(url.status_code, 200)
36     data = url.json()
37     self.assertIn('temp', data['main'])
38     self.assertIsInstance(data['main']['temp'], float)
39
40 if __name__ == '__main__':
41     unittest.main()
42

```

Рисунок 2 – Интеграционные тесты 2-3.

На рисунке 3 результат запуска первых трех тестов. Все тесты пройдены.



```

✓ Tests passed: 3 of 3 tests - 524 ms
Launching unittests with arguments python -m unittests
Process finished with exit code 0

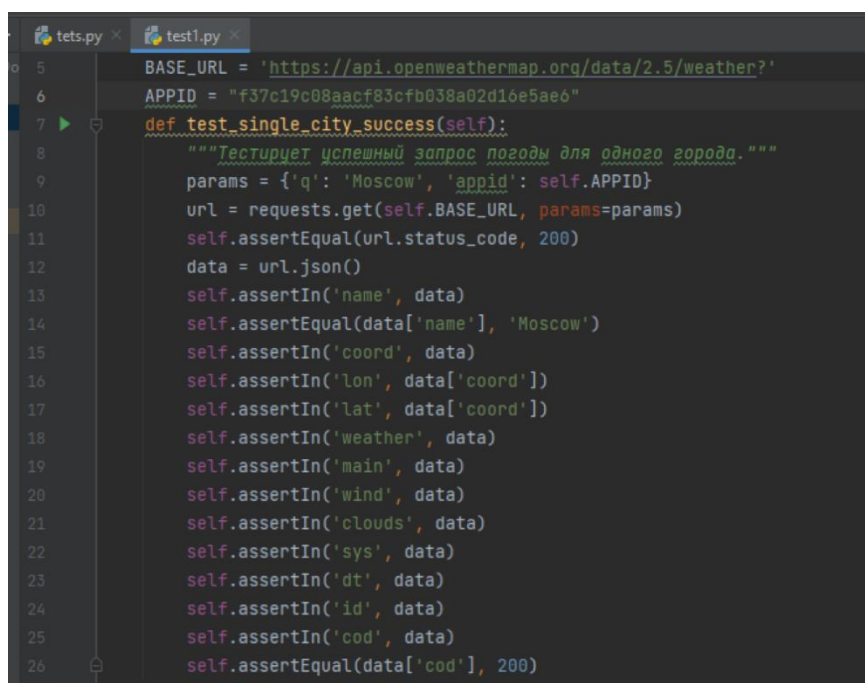
Ran 3 tests in 0.520s

OK

```

Рисунок 3 – Запуск тестов.

На рисунках 4-5 код двух других тестов.



```

5 BASE_URL = 'https://api.openweathermap.org/data/2.5/weather?'
6 APPID = "f37c19c08aacf83cfb038a02d16e5ae6"
7 def test_single_city_success(self):
8     """Тестирует успешный запрос погоды для одного города."""
9     params = {'q': 'Moscow', 'appid': self.APPID}
10    url = requests.get(self.BASE_URL, params=params)
11    self.assertEqual(url.status_code, 200)
12    data = url.json()
13    self.assertIn('name', data)
14    self.assertEqual(data['name'], 'Moscow')
15    self.assertIn('coord', data)
16    self.assertIn('lon', data['coord'])
17    self.assertIn('lat', data['coord'])
18    self.assertIn('weather', data)
19    self.assertIn('main', data)
20    self.assertIn('wind', data)
21    self.assertIn('clouds', data)
22    self.assertIn('sys', data)
23    self.assertIn('dt', data)
24    self.assertIn('id', data)
25    self.assertIn('cod', data)
26    self.assertEqual(data['cod'], 200)

```

Рисунок 4 – Интеграционный тест 4.

```

39 def test_by_coordinates(self):
40     """Тестирует успешный запрос погоды по географическим координатам."""
41     params = {'lat': 55.751244, 'lon': 37.618423, 'appid': self.APPID}
42     url = requests.get(self.BASE_URL, params=params)
43     self.assertEqual(url.status_code, 200)
44     data = url.json()
45     self.assertIn('name', data)
46     self.assertTrue('Moscow' in data['name']) # Москва должна быть в названии города
47     self.assertIn('coord', data)
48     self.assertIn('lon', data['coord'])
49     self.assertAlmostEqual(data['coord']['lon'], 37.62, delta=0.01)
50     self.assertIn('lat', data['coord'])
51     self.assertAlmostEqual(data['coord']['lat'], 55.75, delta=0.05)
52     self.assertIn('weather', data)
53     self.assertIn('main', data)
54     self.assertIn('wind', data)
55     self.assertIn('clouds', data)
56     self.assertIn('sys', data)
57     self.assertIn('dt', data)
58     self.assertIn('id', data)
59     self.assertIn('cod', data)
60     self.assertEqual(data['cod'], 200)
61

```

Рисунок 5 – Интеграционный тест 5.

Результат запуска на рисунке 6. Оба теста пройдены.

```

✓ Tests passed: 2 of 2 tests – 333 ms
Launching unittests with arguments python -

Ran 2 tests in 0.337s

OK
Process finished with exit code 0

```

Рисунок 6 – Результат запуска.

Вывод:

Выполнены все задачи лабораторной работы, а именно:

- Выбран репозиторий с GitHub
- Проанализированы тестируемые функциональности
- Написаны интеграционные тесты

Соответственно, достигнута цель работы – умение писать интеграционные тесты.