Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем
Кафедра «Прикладная математика и фундаментальная информатика»

Расчётно-графическая работа
по дисциплине «Практикум по программированию»

Студента Курпенова Куата Ибраимовича
фамилия, имя, отчество полностью

Курс 3, группа ФИТ-222

Направление 02.03.02 Фундаментальная информатика
и информационные технологии
код, наименование

Руководитель ассистент
должность, ученая степень, звание
Цифля А. А.
фамилия, инициалы

Выполнил 23.11.2024 г.
дата, подпись студента

Проверил 23.11.2024 г.
дата, подпись руководителя

Омск 2024

СОДЕРЖАНИЕ

1 Подготовка окружения

Для создания виртуального окружения используем модуль `virtualenv`. Создание и активация окружения выполняется следующими командами:

```
virtualenv3 venv && source ./venv/bin/activate
```

После активации виртуального окружения требуется установить модули для тестирования и для работы с `JSON`-схемами:

```
pip3 install pytest-playwright jsonschema
```

Чтобы была возможность повторно развернуть идентичное виртуальное окружение, зафиксируем требуемые зависимости в файле с зависимостями - `requirements.txt`:

```
pip3 freeze > requirements.txt
```

Для запуска тестов командой `pytest` создадим пустой файл `conftest.py`, чтобы определить корневую директорию проекта:

```
touch conftest.py
```

Виртуальное окружение готово к использованию!

2 Создание `API`-клиента

Так как одним из условий задания было конструирование запросов вне тест-кейсов, создадим файл с классом, содержащим функции для обращения к `API` сервиса:

```
# Библиотека для работы с запросами
import requests


class RestfulBookerAPI:
BASE_URL = "https://restful-booker.herokuapp.com"

# Функция для получения токена через API
def get_token(self, payload):
    url = f"{self.BASE_URL}/auth"
```

```
    response = requests.post(url, json=payload)
    return response

# Функция для создания бронирования через API
def create_booking(self, payload):
    url = f"{self.BASE_URL}/booking"
    response = requests.post(url, json=payload)
    return response

# Функция для удаления бронирования
def delete_booking(self, booking_number):
    url = f"{self.BASE_URL}/booking/{booking_number}"
    response = requests.delete(url)
    return response
```

3 Тестирование API-хэндлера получения токена

Для тестирования процесса получения токена нам нужно создать объект класса `RestfulBookingAPI`, вызвать метод для создания токена и сверить JSON-схему ответа сервиса с JSON-схемой ответа, указанной в документации:

```
from jsonschema import validate

from api_client import RestfulBookerAPI


class TestAuthToken:
def setup_class(self):
    self.api = RestfulBookerAPI()

def test_get_token_success(self):
    payload = {"username": "admin", "password": "password123"}

    schema = {
        "type": "object",
        "properties": {"token": {"type": "string"}},
        "required": ["token"],
    }

    response = self.api.get_token(payload)

    assert (
```

```
        response.status_code == 200
    ), f"Expected status code 200, got {response.status_code}"

    response_json = response.json()
    validate(instance=response_json, schema=schema)
```

4 Тестирование API-хэндлера бронирования

Тестирование создания бронирования несколько отличается от тестирования получения токена: у запроса другая схема ответа сервиса бронирования, а также требуется удаление бронирования после завершения тестирования, так как тест не должен оставлять после себя сущности на сервере:

```
from jsonschema import validate

from api_client import RestfulBookerAPI


class TestCreateBooking:
def setup_class(self):
    self.api = RestfulBookerAPI()

def test_create_booking_success(self):
    payload = {
        "firstname": "John",
        "lastname": "Doe",
        "totalprice": 123,
        "depositpaid": True,
        "bookingdates": {
            "checkin": "2024-01-01",
            "checkout": "2024-01-10",
        },
        "additionalneeds": "Breakfast",
    }

    schema = {
        "type": "object",
        "properties": {
            "bookingid": {"type": "integer"},
            "booking": {
                "type": "object",
                "properties": {
```

```
                    "firstname": {"type": "string"},
                    "lastname": {"type": "string"},
                    "totalprice": {"type": "integer"},
                    "depositpaid": {"type": "boolean"},
                    "bookingdates": {
                        "type": "object",
                        "properties": {
                            "checkin": {"type": "string"},
                            "checkout": {"type": "string"},
                        },
                        "required": ["checkin", "checkout"],
                    },
                    "additionalneeds": {"type": "string"},
                },
                "required": [
                    "firstname",
                    "lastname",
                    "totalprice",
                    "depositpaid",
                    "bookingdates",
                    "additionalneeds",
                ],
            },
        },
        "required": ["bookingid", "booking"],
    }

    response = self.api.create_booking(payload)

    assert (
        response.status_code == 200
    ), f"Expected status code 200, got {response.status_code}"

    response_json = response.json()
    validate(instance=response_json, schema=schema)

    self.api.delete_booking(response.json()["bookingid"])
```

5 Запуск тестирования и проверка результатов

После запуска тестов командой `pytest`, получаем следущий вывод:

```
========================== test session starts ==========================
platform linux -- Python 3.12.7, pytest-8.3.3, pluggy-1.5.0
rootdir: .../omstu/semester_5/testing/t4_playwright_restful_booker
plugins: base-url-2.1.0, playwright-0.5.2
collected 2 items

tests/test_auth.py .                                                  [ 50%]
tests/test_booking.py .                                               [100%]

=========================== 2 passed in 3.80s ===========================
```

Как видим из вывода `pytest`, тесты прошли успешно.