Министерство науки и высшего образования РФ
федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

Расчетно-графическая работа

по дисциплине **Практикум по программированию**

Студента	Гюнтера Тимофея Вячеславовича
Курс	3 Группа ФИТ-221
Направление	02.03.02 Фундаментальная информатика и информационные технологии
Руководитель	асс.
	Цифля А.А.
Выполнил	11.01.25 дата, подпись студента
Проверил	11.01.25 дата, подпись руководителя

Омск 2025

Содержание

ВВЕДЕНИЕ

В рамках данной лабораторной работы необходимо разработать и протестировать *API* для сайта *Restful Booker*. Основная цель — написать тесты для функционала аутентификации и бронирования, используя *Python*, *Pytest* и *JSON*-схемы для валидации.

В результате работы:

- Создан клиент для взаимодействия с *API*;
- Разработаны тесты на основе валидных и невалидных данных;
- Проведена проверка данных с использованием *JSON*-схем

1. Основная часть

1.1. Постановка задачи

a) Написать *API*-клиент для взаимодействия с *Restful Booker API*.

b) Реализовать тесты:

- Для получения токена аутентификации;
- Для создания бронирований;
- Для удаления бронирований.

c) Провести валидацию ответов сервера с помощью *JSON*-схем.

d) Проверить работу программы и сделать выводы.

1.2. Выполнение задачи

Разработка клиента *API*

Для взаимодействия с API был разработан класс RestfulBookerClient.

Класс реализует следующие методы:

- create_booking — создание бронирования;
- delete_booking — удаление бронирования;

- create_token — получение токена для аутентификации.

Код клиента находится в приложении (см. Приложение, файл client.py).

Разработка тестов

Для тестирования использован фреймворк Pytest. Тесты разделены на два основных модуля:

- Тесты аутентификации (test_authentication.py);
- Тесты создания и удаления бронирований (test_create_booking.py).

Для каждого модуля были определены валидные и невалидные тестовые данные. В качестве примера:

- Валидные данные для аутентификации:

{"username": "admin", "password": "password123"}

- Невалидные данные:

{"username": "admin"}

2. Валидация данных

Для проверки структуры ответов использовались JSON-схемы, реализованные в файле schemas.py. Например, схема для проверки успешного получения токена:

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

}

3. Результаты выполнения тестов

Тесты выполнялись через команду:

pytest --tb=short

Результаты выполнения тестов:

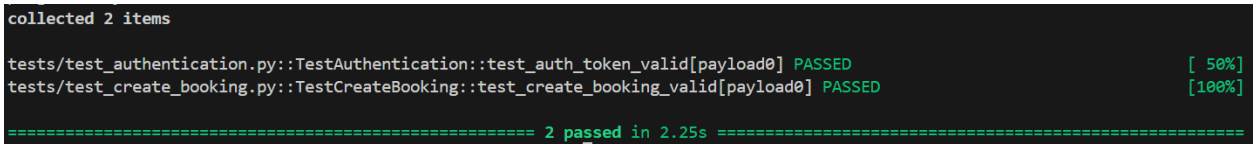

Рисунок 1 – Результаты выполнения тестов

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были разработаны и протестированы методы взаимодействия с API сайта Restful Booker. Все тесты выполнены успешно, структура ответов проверена с использованием JSON-схем. Полученные навыки пригодятся для тестирования API в реальных проектах.

Приложение А

client.py

```
import requests

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

    def create_booking(self, payload):
        url = f"{self.BASE_URL}/booking"
        return requests.post(url, json=payload)

    def delete_booking(self, booking_id, token):
        url = f"{self.BASE_URL}/booking/{booking_id}"
        headers = {"Cookie": f"token={token}"}
        return requests.delete(url, headers=headers)

    def create_token(self, payload):
        url = f"{self.BASE_URL}/auth"
        return requests.post(url, json=payload)
```

test_authentication.py

```
import pytest
from jsonschema import validate
from schemas.schemas import TOKEN_SCHEMA

valid_auth_data = [{"username": "admin", "password": "password123"}]

class TestAuthentication:
    @pytest.mark.parametrize("payload", valid_auth_data)
    def test_auth_token_valid(self, client, payload):
        response = client.create_token(payload)
        assert response.status_code == 200
        validate(instance=response.json(), schema=TOKEN_SCHEMA)
```

test_create_booking.py

```
import pytest
from jsonschema import validate
from schemas.schemas import CREATE_BOOKING_SCHEMA

valid_booking_data = [
    {
        "firstname": "John",
        "lastname": "Doe",
        "totalprice": 150,
```

```
        "depositpaid": True,
        "bookingdates": {
            "checkin": "2023-12-01",
            "checkout": "2023-12-10"
        },
        "additionalneeds": "Breakfast"
    }
]

class TestCreateBooking:
    @pytest.mark.parametrize("payload", valid_booking_data)
    def test_create_booking_valid(self, client, payload):
        response = client.create_booking(payload)
        assert response.status_code == 200
        validate(instance=response.json(),
schema=CREATE_BOOKING_SCHEMA)
```