15. Java_QA (REST&SOAP)

- Методика тестирования API на примере библиотеки RestAssured

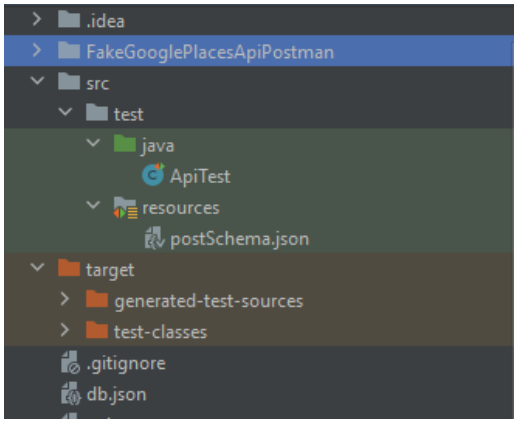
- GET, POST, PUT, DELETE запросы

- Представление и валидация данных в формате JSON

- Автоматизация тестирования в программной среде Postman

- Примеры SOAP тестирования в среде SoapUI

Структура проекта:



Pom.xml: (добавление библиотек)

```
<properties>
   <restassured.version>2.9.0</restassured.version>
   <junit.version>5.5.2</junit.version>
</properties>
<dependencies>
   <dependency>
      <groupId>com.jayway.restassured</groupId>
      <artifactId>rest-assured</artifactId>
      <version>${restassured.version}</version>
      <scope>test</scope>
   </dependency>
   <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>${junit.version}</version>
   </dependency>

   <dependency>
      <groupId>io.rest-assured</groupId>
      <artifactId>json-schema-validator</artifactId>
      <version>4.1.2</version>
   </dependency>
```

```
   <dependency>
      <groupId>javax.xml.bind</groupId>
      <artifactId>jaxb-api</artifactId>
      <version>2.3.1</version>
   </dependency>

   <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-simple</artifactId>
      <version>1.7.30</version>
      <scope>test</scope>
   </dependency>

   <dependency>
      <groupId>org.json</groupId>
      <artifactId>json</artifactId>
      <version>20180813</version>
      <scope>test</scope>
   </dependency>
</dependencies>
```

Сценарий: попытка отправки REST – запросов, с последующей проверкой данных, на примере json-сервера.

Запуск Json-сервера:

```
nar@DESKTOP-OV3QROS MINGW64 ~/Deskto
npm -version \
npm install -g json-server \
json-server --watch db.json
```

Структура БД db.json:

```
{
  "posts": [
    {
      "id": 1,
      "title": "json-server",
      "author": "typicode"
    }
  ],
  "comments": [
    {
      "id": 1,
      "body": "some comment",
      "postId": 1
    }
  ],
  "profile": {
    "name": "typicode"
  }
}
```

Исходный код:

Создание объекта для логов, и хранения параметров запроса:

```
private static final Logger LOGGER = LoggerFactory.getLogger(ApiTest.class);
private static JSONObject requestParams = new JSONObject();
```

Добавление новой записи, post-метод:

```
@Test
@Order(1)
public void postNewPost() {
    requestParams.put("id", "2");
    requestParams.put("title", "Second");
    requestParams.put("author", "Linar");
    given().
            contentType("application/json").body(requestParams.toString()).
            when().
            post("http://localhost:3000/posts").
            then().
            statusCode(201);
    LOGGER.info("New post has been added." + "\n");

    when().
            get("http://localhost:3000/posts/2").
            then().
            statusCode(200).
            assertThat().body("author", equalTo("Linar"));
    Response response = get("http://localhost:3000/posts/2");
    LOGGER.info("New post has data: " + response.getBody().asString());
}
```

Получение добавленной записи get-запросом:

```
@Test
@Order(2)
public void getSecondPosts() {
    when().
            get("http://localhost:3000/posts/2").
            then().
            statusCode(200).
            assertThat().body("author", equalTo("Linar"), "id", equalTo(2));
    Response response = get("http://localhost:3000/posts/2");
    LOGGER.info("Second post has data: " + response.getBody().asString());
}
```

Put-запрос, изменение параметра “имя”:

```
@Test
@Order(3)
public void changeThePost() {
    given().
            contentType("application/json").body("{\n" +
            "       \"id\": 2,\n" +
            "       \"title\": \"second\",\n" +
            "       \"author\": \"noname\"\n" +
            "    }\n").
            when().
            put("http://localhost:3000/posts/2").
            then().
```

```
            statusCode(200);
    LOGGER.info("Post has been edited." + "\n");

    when().
            get("http://localhost:3000/posts/2").
            then().
            statusCode(200).
            assertThat().body("title", equalTo("second"))
            .body("author", equalTo("noname"));
    Response response = get("http://localhost:3000/posts/2");
    LOGGER.info("New data is : " + response.getBody().asString());
}
```

Удаление, delete-запрос:

```
@Test
@Order(4)
public void deleteThePost() {
    when().
            delete("http://localhost:3000/posts/2").
            then().
            statusCode(200).
            assertThat().body("{}", Matchers.nullValue());
    LOGGER.info("Post has been deleted .");
}
```

Генерация json-схемы:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "id": {
      "type": "integer"
    },
    "title": {
      "type": "string"
    },
    "author": {
      "type": "string"
    }
  },
  "required": [
    "id",
    "title",
    "author"
  ]
}
```

Сценарий валидации:

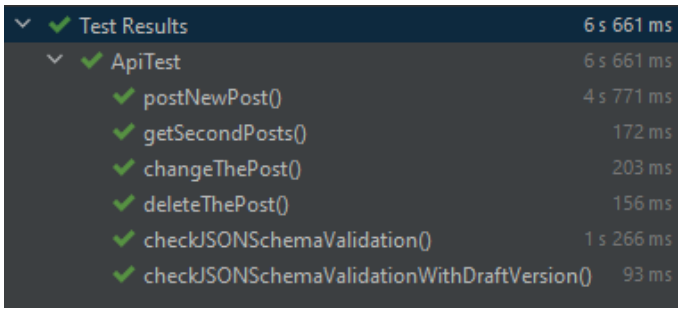
```
@Test
public void checkJSONSchemaValidation() {
    get("http://localhost:3000/posts/1").then().assertThat()
            .body(matchesJsonSchemaInClasspath("postSchema.json"));
}
```

Валидация с учетом драфт-версии:

```
@Test
public void checkJSONSchemaValidationWithDraftVersion() {
    JsonSchemaFactory jsonSchemaFactory = JsonSchemaFactory
            .newBuilder()
            .setValidationConfiguration(
                    ValidationConfiguration.newBuilder()
                            .setDefaultVersion(SchemaVersion.DRAFTV4)
                            .freeze()).freeze();

    get("http://localhost:3000/posts/1")
            .then()
            .assertThat()
            .body(matchesJsonSchemaInClasspath("postSchema.json").using(
                    jsonSchemaFactory));
}
```

Результат запуска:

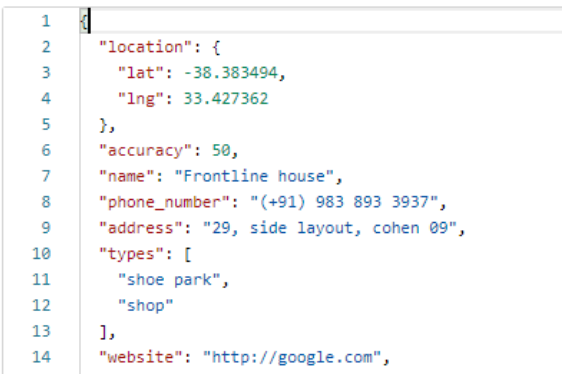


Пример автоматизации Postman:

Сценарий: Имитация тестирования GoogleMaps сервисов, на стороннем ресурсе, отправка REST запросов и создание тестов встроенными средствами javaScript, использование environment variables. Ключ для отправки запросов - qaclick123.

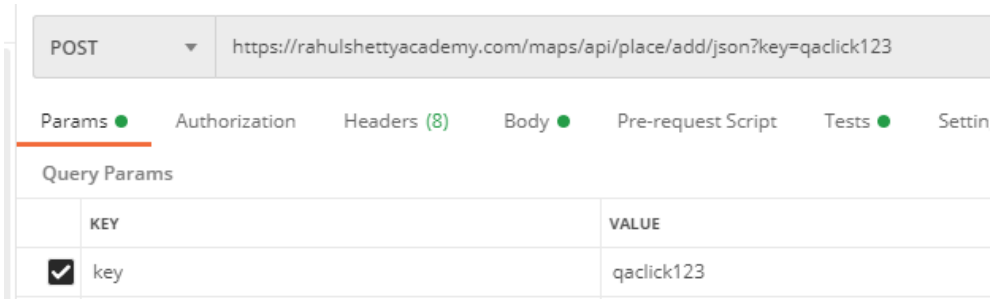
Post-запрос: (добавление локации)

Тело запроса в формате Json:

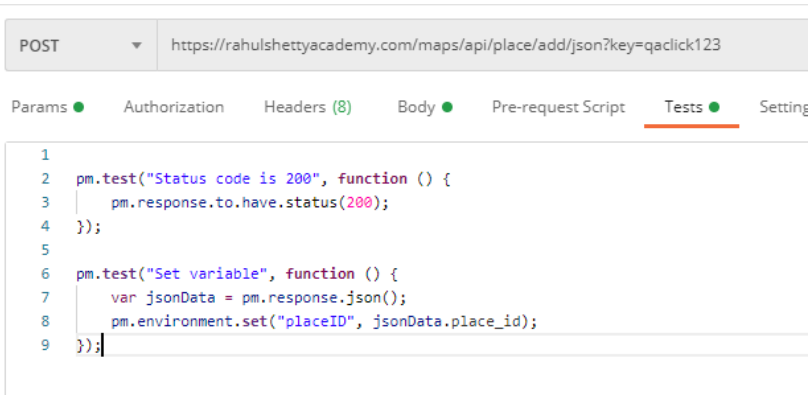


```
{
  "location": {
    "lat": -38.383494,
    "lng": 33.427362
  },
  "accuracy": 50,
  "name": "Frontline house",
  "phone_number": "(+91) 983 893 3937",
  "address": "29, side layout, cohen 09",
  "types": [
    "shoe park",
    "shop"
  ],
  "website": "http://google.com",
```

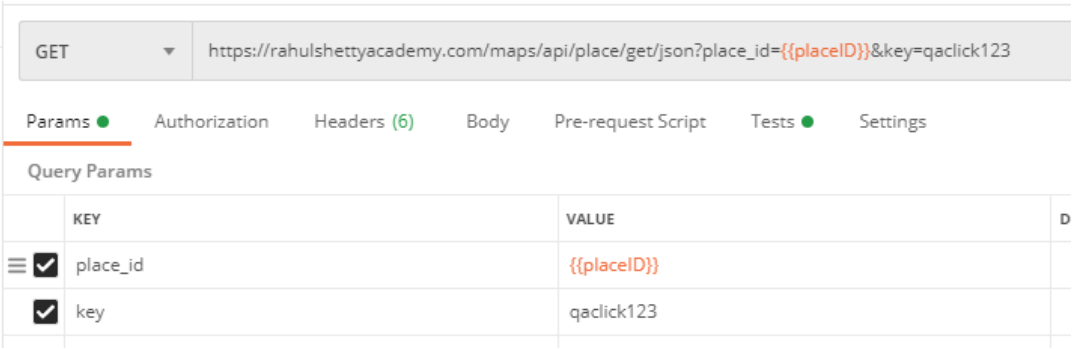
Параметры и адрес:



Сценарии тестирования: проверка статус-кода, и добавление переменной place_ID для дальнейшей работы



Get-запрос: (получение информации о локации, используя переменную place_ID)

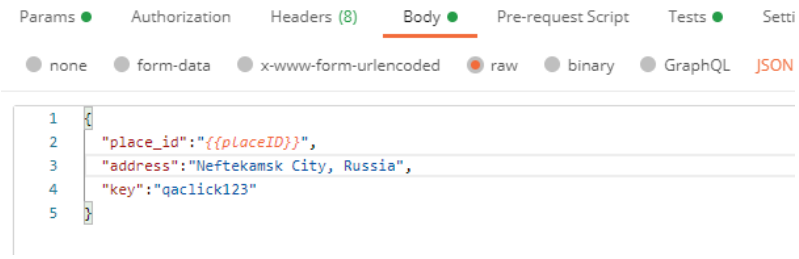


Сценарии тестирования: проверка статус-кода и соответствия адреса

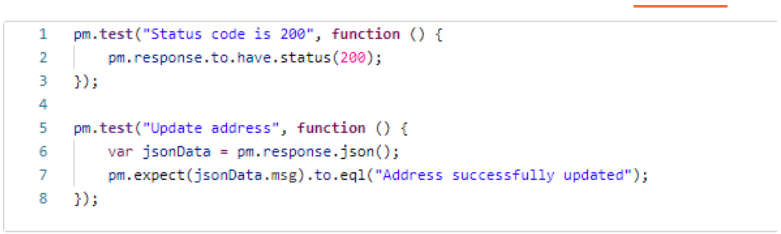
```
pm.test("Verification of status code", function () {
    if (pm.environment.get("placeID")) {
        pm.response.to.have.status(200);
        pm.test("Address is correct", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.address).to.eql("29, side layout, cohen 09");
    });
    } else
    {
        pm.response.to.have.status(404);
        postman.setNextRequest(null);
    }
```

Put-запрос (изменение адреса):

Тело и параметры запроса:

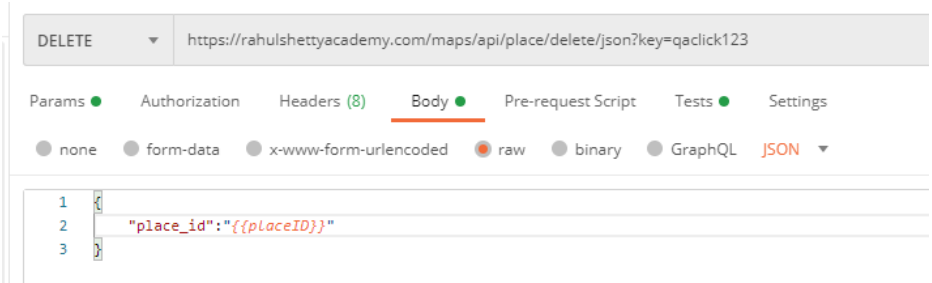


Сценарии тестирования: проверка статус-кода и соответствия адреса



Delete-запрос (удаление локации):

Тело запроса:



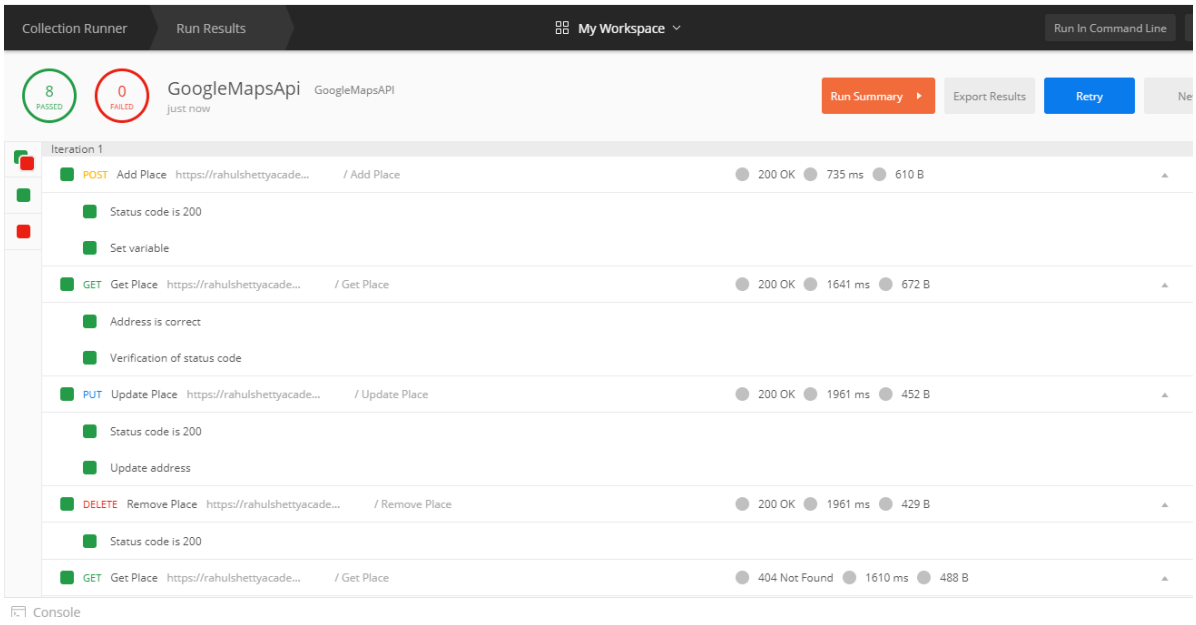
Сценарии тестирования: проверка статус-кода и удаление переменной place_ID

Params ● Authorization Headers (8) Body ● Pre-request Script

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
    pm.environment.set("placeID", null);
});

postman.setNextRequest("Get Place");
```

Автоматизированный запуск:

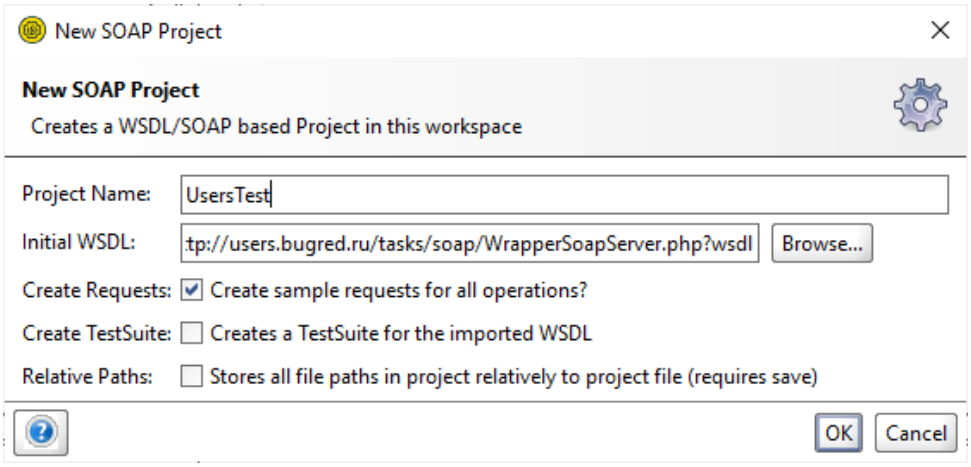


Коллекция лежит в корне проекта, папка: Fake_Google_Places Api_Postman.

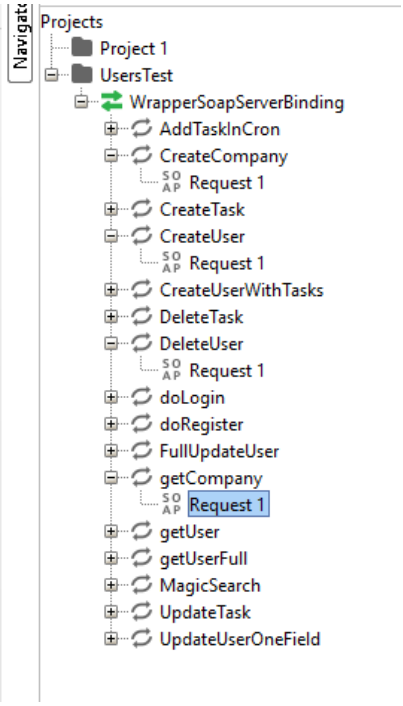
SOAP тестирование

Сценарий: демонстрация работы с системой “users.bugred.ru” в программной среде SoapUI.

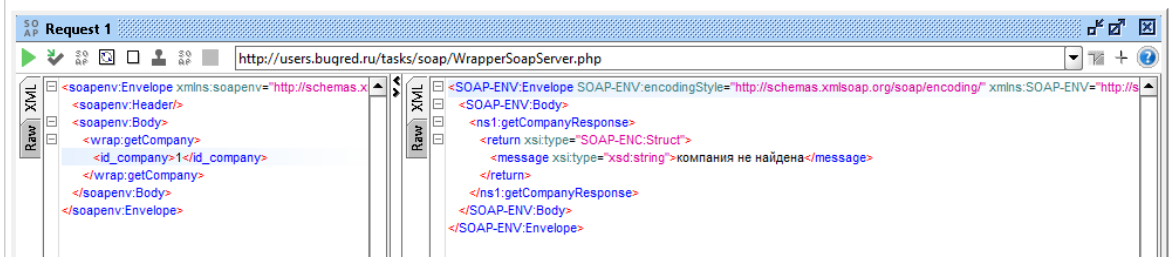
Создание проекта и добавление WSDL:



Структура проекта:



Пример отправки запроса по шаблону:



Файл проекта лежит в папке: SoapTest.