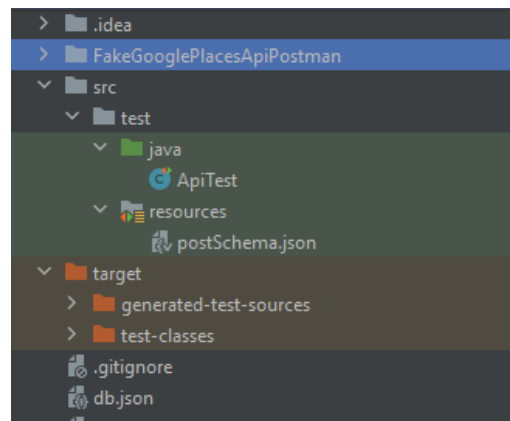


## 15. Java\_QA (REST&SOAP)

- 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>
</dependencies>
```

```

<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-0V3QR0S MINGW64 ~/Desкто
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:

✓ Test Results	6 s 661 ms
✓ ApiTest	6 s 661 ms
✓ postNewPost()	4 s 771 ms
✓ getSecondPosts()	172 ms
✓ changeThePost()	203 ms
✓ deleteThePost()	156 ms
✓ checkJSONSchemaValidation()	1 s 266 ms
✓ checkJSONSchemaValidationWithDraftVersion()	93 ms

## 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:

```
1 {
2   "location": {
3     "lat": -38.383494,
4     "lng": 33.427362
5   },
6   "accuracy": 50,
7   "name": "Frontline house",
8   "phone_number": "(+91) 983 893 3937",
9   "address": "29, side layout, cohen 09",
10  "types": [
11    "shoe park",
12    "shop"
13  ],
14  "website": "http://google.com",
```

## Параметры и адрес:

POST ▼ <https://rahulshettyacademy.com/maps/api/place/add/json?key=qaclick123>

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

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	key	qaclick123

Сценарии тестирования: проверка статус-кода, и добавление переменной place\_ID для дальнейшей работы

POST ▼ <https://rahulshettyacademy.com/maps/api/place/add/json?key=qaclick123>

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

```
1
2 pm.test("Status code is 200", function () {
3   pm.response.to.have.status(200);
4 });
5
6 pm.test("Set variable", function () {
7   var jsonData = pm.response.json();
8   pm.environment.set("placeID", jsonData.place_id);
9 });
```

Get-запрос: (получение информации о локации, используя переменную place\_ID)

GET ▼ [https://rahulshettyacademy.com/maps/api/place/get/json?place\\_id={{placeID}}&key=qaclick123](https://rahulshettyacademy.com/maps/api/place/get/json?place_id={{placeID}}&key=qaclick123)

Params ● Authorization Headers (6) Body Pre-request Script Tests ● Settings

Query Params

	KEY	VALUE	
<input checked="" type="checkbox"/>	place_id	{{placeID}}	
<input checked="" type="checkbox"/>	key	qaclick123	

Сценарии тестирования: проверка статус-кода и соответствия адреса

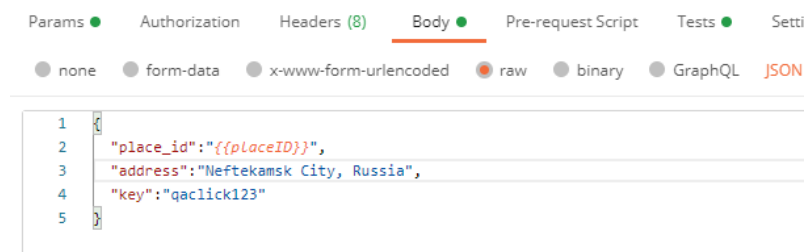
```

1  pm.test("Verification of status code", function () {
2      if (pm.environment.get("placeID")) {
3          pm.response.to.have.status(200);
4          pm.test("Address is correct", function () {
5              var jsonData = pm.response.json();
6              pm.expect(jsonData.address).to.eql("29, side layout, cohen 09");
7          });
8      } else
9      {
10         pm.response.to.have.status(404);
11         postman.setNextRequest(null);
12     }

```

Put-запрос (изменение адреса):

Тело и параметры запроса:



Сценарии тестирования: проверка статус-кода и соответствия адреса

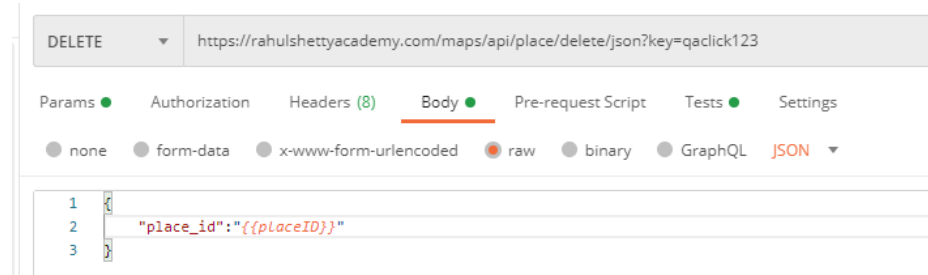
```

1  pm.test("Status code is 200", function () {
2      pm.response.to.have.status(200);
3  });
4
5  pm.test("Update address", function () {
6      var jsonData = pm.response.json();
7      pm.expect(jsonData.msg).to.eql("Address successfully updated");
8  });

```

Delete-запрос (удаление локации):

Тело запроса:



Сценарии тестирования: проверка статус-кода и удаление переменной place\_ID

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

```
1 pm.test("Status code is 200", function () {
2     pm.response.to.have.status(200);
3     pm.environment.set("placeID", null);
4 });
5
6 postman.setNextRequest("Get Place");
```

Автоматизированный запуск:

The screenshot shows the Postman Collection Runner interface. At the top, it displays 'Collection Runner' and 'Run Results' tabs. The collection is named 'GoogleMapsApi' and has 8 passed tests and 0 failed tests. Below this, the test results for 'Iteration 1' are shown. The tests are as follows:

Test Name	Method	URL	Path	Status	Time	Size
POST Add Place	POST	https://rahulshettyacademy...	/ Add Place	200 OK	735 ms	610 B
Status code is 200				200 OK		
Set variable						
GET Get Place	GET	https://rahulshettyacademy...	/ Get Place	200 OK	1641 ms	672 B
Address is correct						
Verification of status code						
PUT Update Place	PUT	https://rahulshettyacademy...	/ Update Place	200 OK	1961 ms	452 B
Status code is 200				200 OK		
Update address						
DELETE Remove Place	DELETE	https://rahulshettyacademy...	/ Remove Place	200 OK	1961 ms	429 B
Status code is 200				200 OK		
GET Get Place	GET	https://rahulshettyacademy...	/ Get Place	404 Not Found	1610 ms	488 B

At the bottom, there is a 'Console' tab.

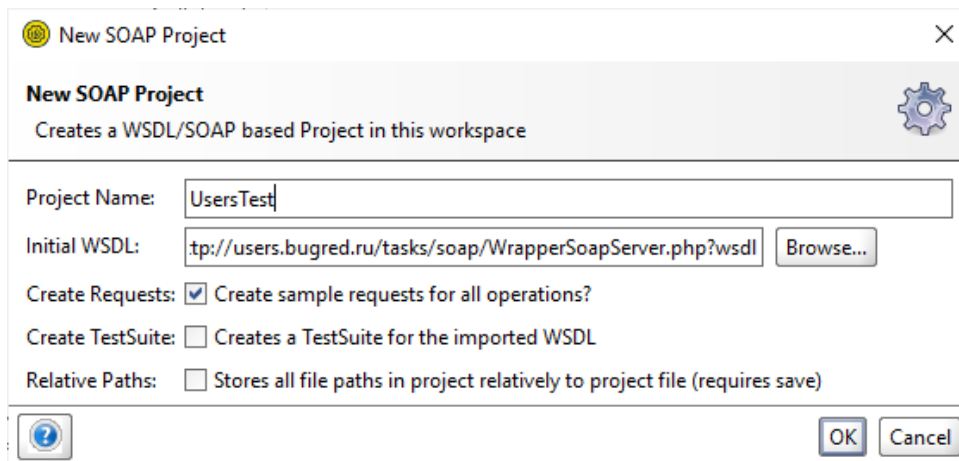
Коллекция лежит в корне проекта, папка: Fake Google Places Api Postman.

SOAP тестирование

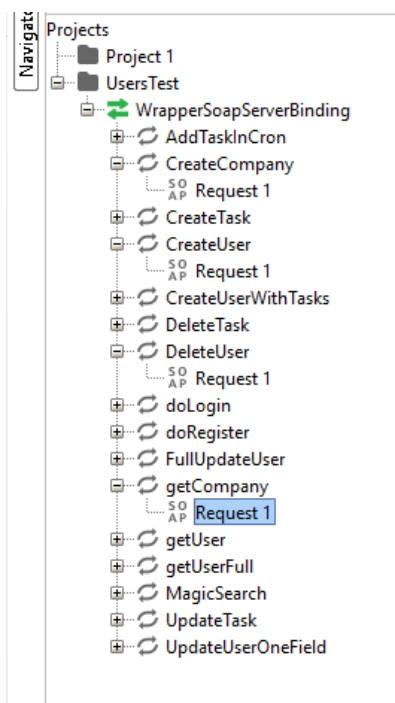
Сценарий: демонстрация работы с системой “users.bugred.ru” в программной среде SoapUI.

Создание проекта и добавление WSDL:

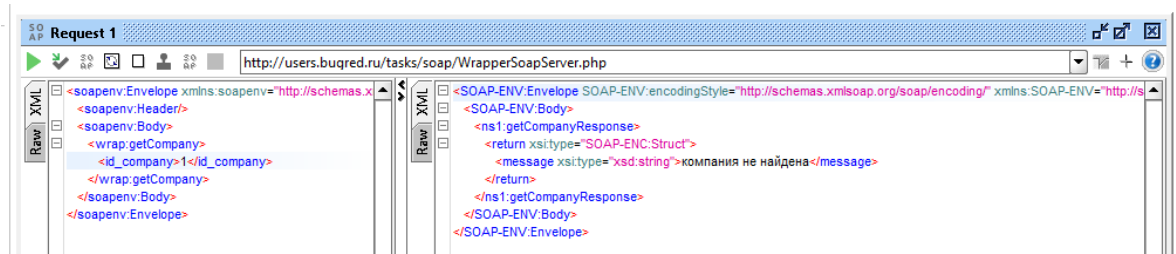




Структура проекта:



Пример отправки запроса по шаблону:



Файл проекта лежит в папке: SoapTest.