

EXPERIMENT 5

Aim:

Testing of restful web service using Postman

Requirements:

Django, Web Browser(Chrome), POSTMAN App

Problem Statement:

Olympic management system is an app which targets to provide various sports event organizers a platform to promote their events and also to give the general users/participants information about the sports events in which they're interested

Theory:

1) Postman:

Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated.

Postman is an application that allows us to test APIs utilizing a graphical user interface.

Some of Postman's advantages include the collection feature and the possibility to create different testing environments.

Postman is a user-friendly tool that helps us optimize our time when executing tests.

2) Methods

Postman offers many endpoint interaction methods. The following are some of the most used, including their functions:

GET: Obtain information

POST: Add information

PUT: Replace information

PATCH: Update certain information

DELETE: Delete information

3) Response Codes

When testing APIs with Postman, we usually obtain different response codes. Some of the most common include:

- **100 Series** - Temporal responses, for example, '102 Processing'.

- **200 Series** - Responses where the client accepts the request and the server processes it successfully, for instance, '200 Ok'.
- **300 Series** - Responses related to URL redirection, for example, '301 Moved Permanently.'
- **400 Series** - Client error responses, for instance, '400 Bad Request'.
- **500 Series** - Server error responses, for example, '500 Internal Server Error.'

4) Collections

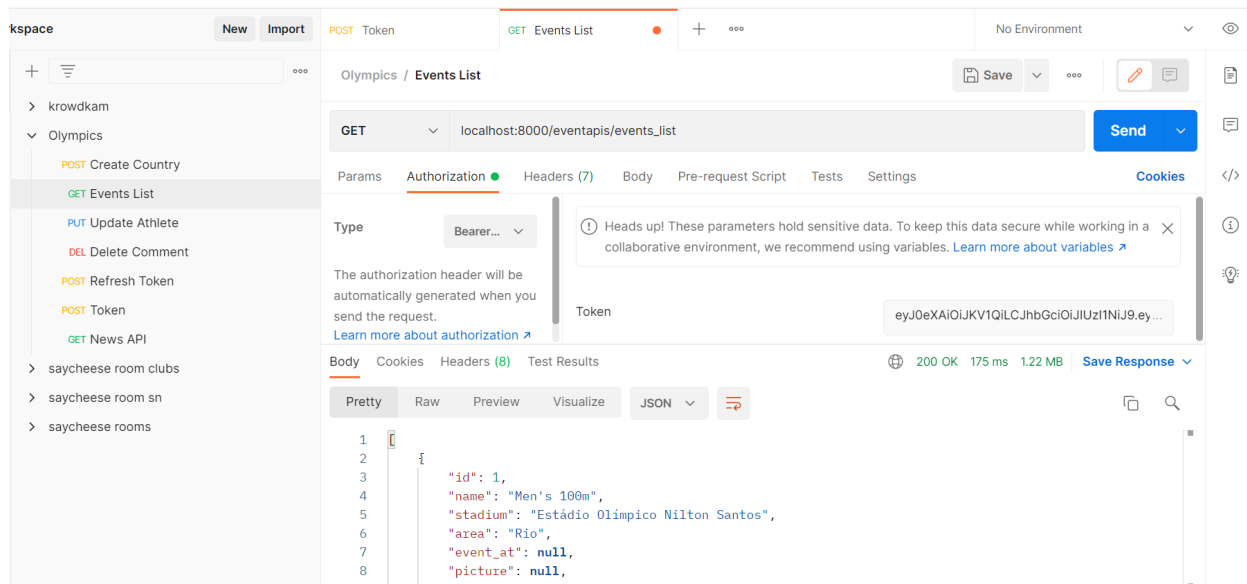
Postman gives the possibility to group different requests. This feature is known as 'collections' and helps organize tests. These collections are folders where requests are stored and can be structured in whichever way the team prefers. It is also possible to export-import them.

5) Environments

Postman also allows us to create different environments through the generation/use of variables; for example, a URL variable that is aimed towards different test environments (dev-QA), enabling us to execute tests in different environments using existing requests.

Implementation:

1) GET Request:

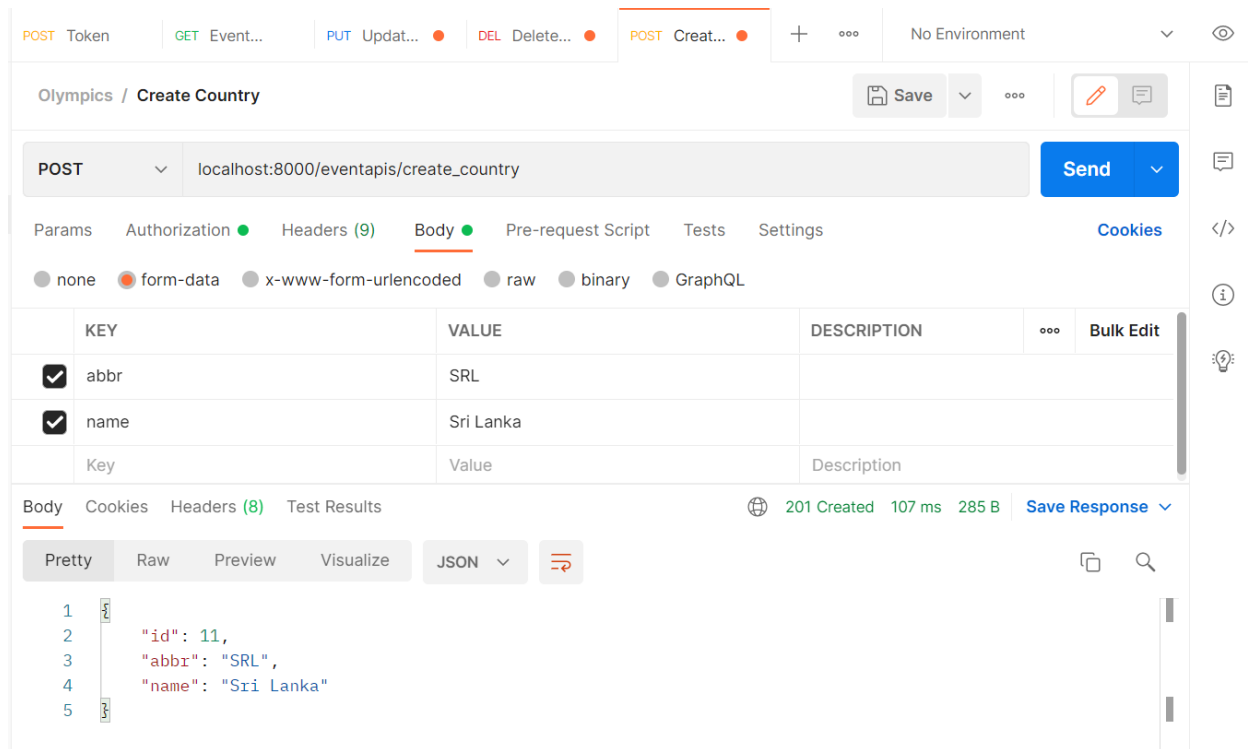


The screenshot displays the Postman interface for a GET request. The request is named 'Events List' and is located within a collection named 'Olympics'. The URL is 'localhost:8000/eventapis/events_list'. The request is configured with a Bearer token authorization. The response is a 200 OK status with a 175 ms response time and a 1.22 MB body size. The response body is a JSON object containing event details.

```

{
  "id": 1,
  "name": "Men's 100m",
  "stadium": "Estádio Olímpico Nilton Santos",
  "area": "Rio",
  "event_at": null,
  "picture": null,
  "event_time": null
}

```



2) POST Request:

3) PUT Request:

4) DELETE Request:

Conclusion:

From the above experiment, I have learnt the following:

- Knowledge about Postman app.
- Testing of APIs.
- Hands-on experience of using Postman to test APIs from the given problem statement.

References:

1. <https://www.guru99.com/postman-tutorial.html>
2. <https://www.encora.com/insights/what-is-postman-api-test>
3. <https://medium.com/aubergine-solutions/api-testing-using-postman-323670c89f6d>.