

Web API and Postman

Web APIs, HTTP Protocol, REST and Postman



SoftUni Team
Technical Trainers



SoftUni

Software University

<https://about.softuni.bg/>

1. Introduction to **Web APIs**
2. **HTTP** Fundamentals
3. Introduction to **REST**
4. **Web APIs** and **REST**
5. **Postman** Overview
6. Using the **Postman tool**

sli.do

#qa-fund



Web Services

Communication between
Systems and Components

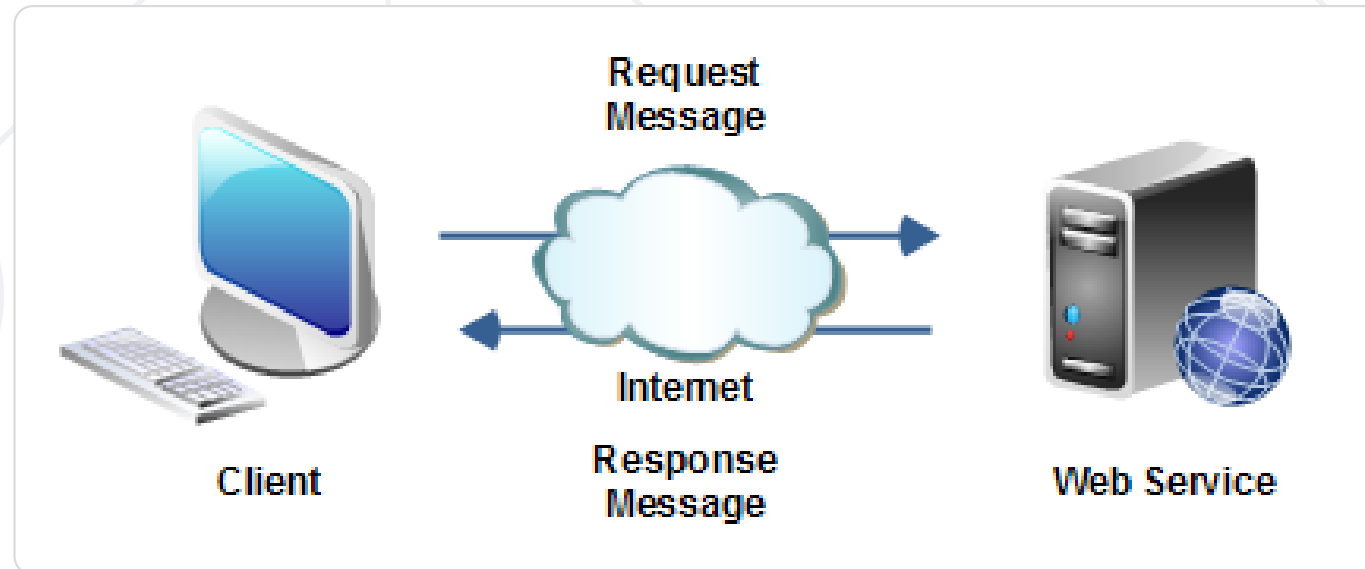
What is API?

- **API** == **A**pplication **P**rogramming **I**nterface
 - Programming interface, designed for communication between system components
 - Set of **functions** and **specifications** that software programs and components follow to talk to each other
- **API examples:**
 - **JDBC** – Java API for apps to talk with database servers
 - **Windows API** – Windows apps talk with Windows OS
 - **Web Audio API** – play audio in the Web browser with JS



What is Web Service?

- Web **services** implement **communication** between software **systems** or **components** of over the **network**
 - Using standard **protocols**, such as HTTP, JSON and XML
 - Exchanging **messages**, holding data and operations



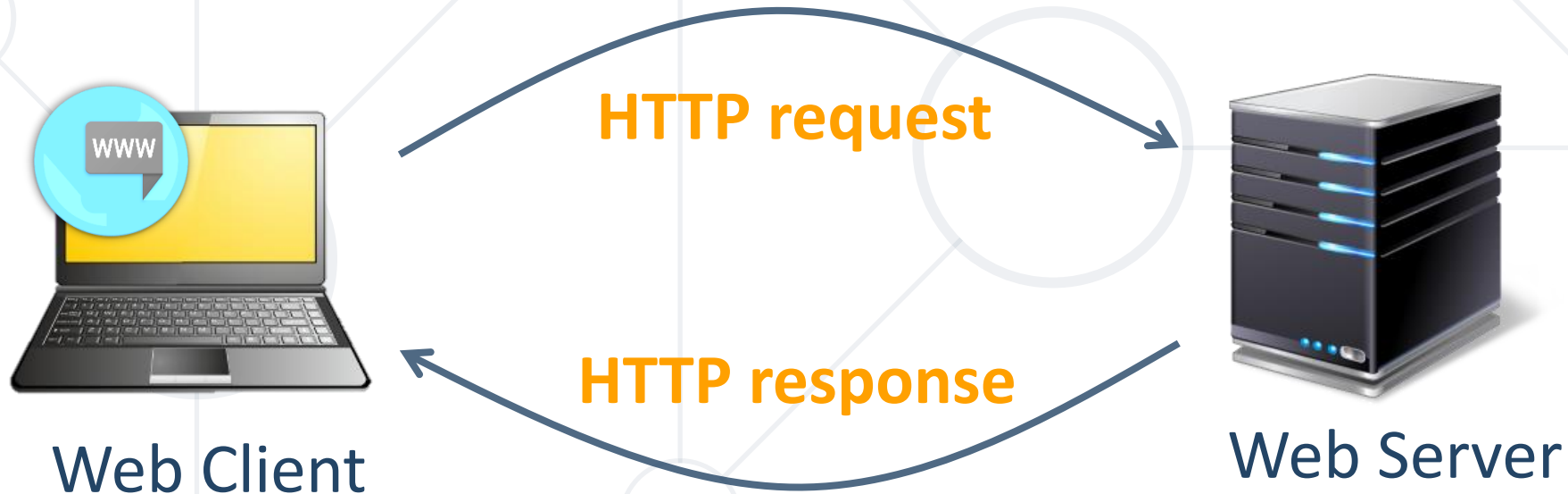
- **Web services** expose **back-end APIs** over the **network**
 - May use different **protocols** and **data formats**: **HTTP, REST, GraphQL, gRPC, JSON-RPC, JSON, BSON, XML, YML, SOAP...**
- **Web services** are hosted on a Web server (HTTP server)
 - Provide a set of functions, invokable from the Web (Web API)
- **RESTful APIs** is the most popular Web service standard
- **Example** of RESTful service (HTTP GET request, returns JSON):
 - GET <http://api.zippopotam.us/us/90210>

A background network diagram consisting of a grid of light gray lines intersecting at various points. At these intersections, there are several circles of different sizes, some solid light gray and some hollow, representing nodes in a network.

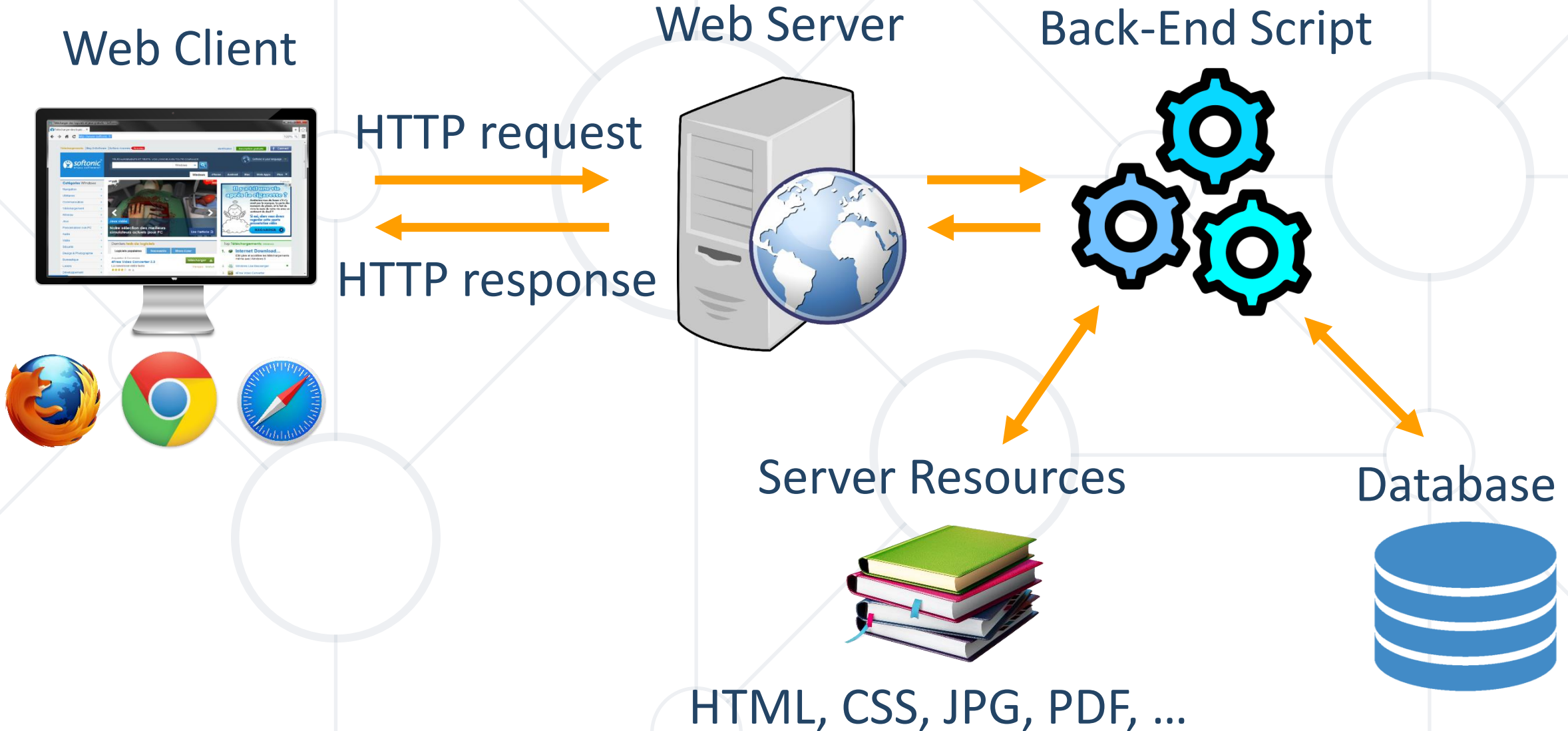
http://

HTTP Protocol – Basics

- **HTTP** (**H**yper**T**ext **T**ransfer **P**rotocol)
 - **Text-based** client-server protocol for the Internet
 - For transferring **Web resources** (HTML files, images, styles, etc.)
 - **Request-response** based









Web Server Work Model



HTTP Request Methods

- **HTTP request methods** specify the desired **action** to be performed on the requested resource (identified by URL)

Method		Description	Other Methods
GET		Retrieve a resource	
POST		Create / store a resource	
PUT		Update (replace) a resource	
DELETE		Delete (remove) a resource	
PATCH		Update resource partially (modify)	
HEAD		Retrieve the resource's headers	

CRUD == the four main functions of persistent storage

CONNECT
OPTIONS
TRACE

HTTP GET Request – Example

GET /users/SoftUni-Tech-Module/repos **HTTP/1.1**

Host: **api.github.com**

Relative URI, not full URL

HTTP request line

Accept: */*

The **host** is part of the URL

Accept-Language: en

HTTP headers

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)

AppleWebKit/537.36 (KHTML, like Gecko)

Chrome/54.0.2840.71 Safari/537.36

Connection: keep-alive

Cache-Control: no-cache

<CRLF>

The request body is empty

HTTP Response – Example

HTTP/1.1 200 OK

HTTP response status line

Date: Fri, 11 Feb 2021 16:09:18 GMT+2

Server: Apache/2.2.14 (Linux)

Accept-Ranges: bytes

Content-Length: 80

Content-Type: application/json

HTTP response headers

Describes the
returned content

<CRLF>

HTTP response body

```
{ "id": 1, "firstName": "Steve", "lastName": "Jobs",  
  "email": "steve@apple.com" }
```



XML

Specifics and Structure

- **XML** (e**X**tensible **M**arkup **L**anguage) is a **markup language** that is **used to store and transport data**.
- **Tree-like structure**
 - Each element has a start and an end tag, and
 - Can have attributes and child elements
 - Elements can also have text content
- An XML document has a **root element** that contains all other elements

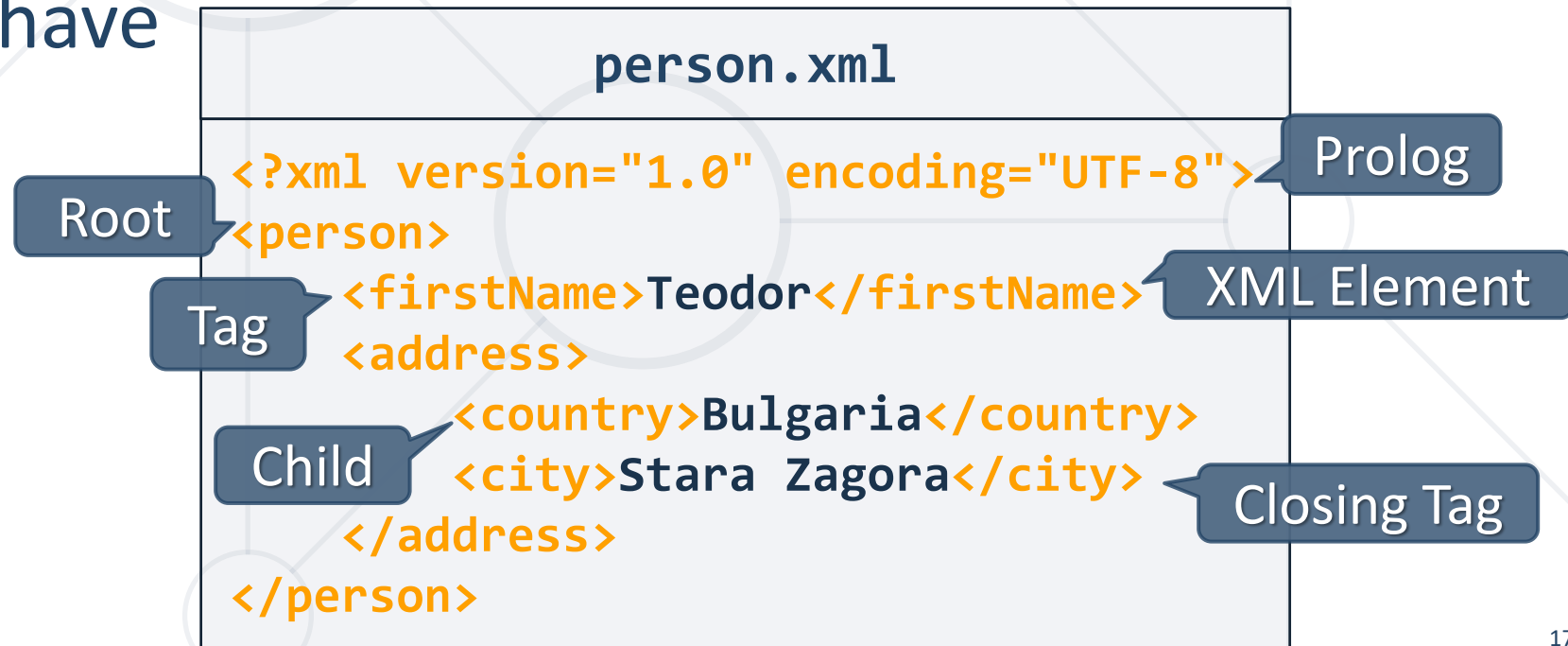
- An XML document consists of strings that:
 - Constitute **markup** – usually begin with **<** and end with **>**
 - Are **content** – placed between markup(**tags**)

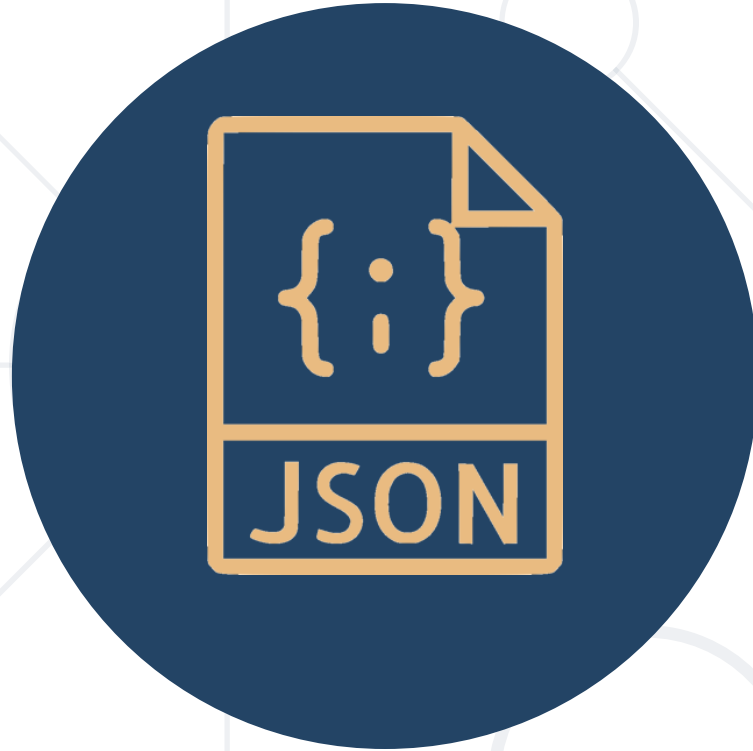
Markup tags
for Person
Object

```
person.xml
<?xml version="1.0" encoding="UTF-8">
<person>
  <firstName>Teodor</firstName>
</person>
```

Content
(Person Name)

- XML documents are formed as **element trees**
- An XML tree starts at a **root element** and branches from the root to **sub elements**
 - All elements can have child elements:





JSON

Specifics and Structure

- **JSON** (JavaScript Object Notation) is a **lightweight** data-interchange format
- Self-describing **data format**, based on JS object syntax
- **Simple, text-based, key-value** based
- **Easy** for people to **read and write** and **easy** for machines to **parse and generate**.
- Supports **several data types**:
 - Number, String, Boolean, Array, Object, null
- Used to **transmit data** between a server and a web application, as an **alternative to XML**

- Example of JSON string, holding an "issue":

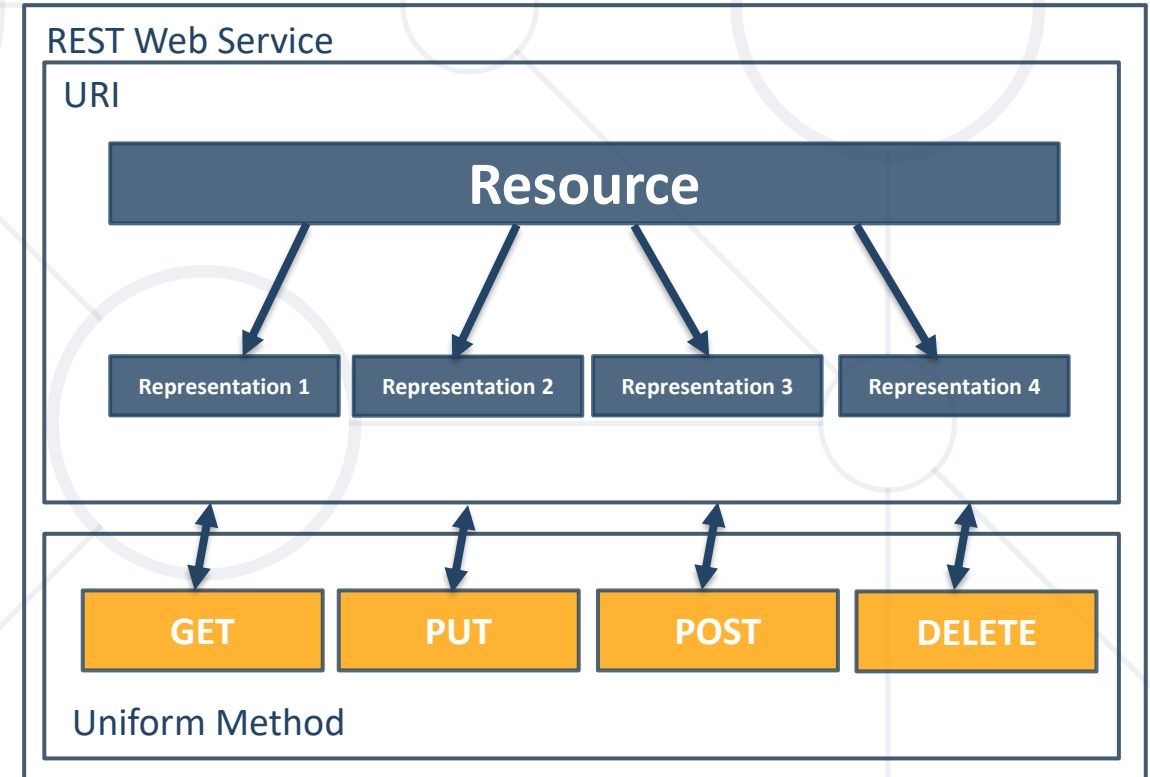
```
{  
  "issueId": "TEST-123",  
  "summary": "Unable to log in to the application",  
  "description": "When attempting to log in to the  
application, the login button does not respond and no  
error message is displayed.",  
  "severity": "Critical",  
  "status": "Open",  
  "priority": "High",  
  "createdAt": "2022-12-01T08:00:00Z",  
  "reporter": "jane.doe@example.com"  
}
```



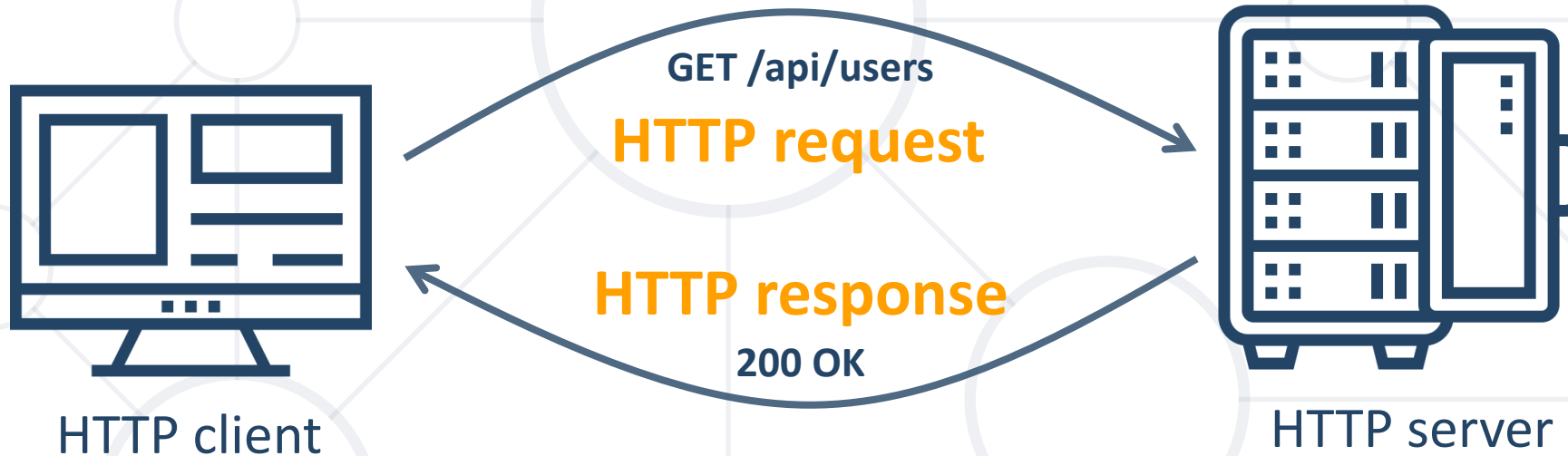
RESTful APIs

- **REST** (**R**epresentational **S**tate **T**ransfer) is an architectural style for building web services, that are lightweight, fast, and scalable
- RESTful web services **use** the **HTTP protocol** for communication
- Two key **principles**
 - **Stateless** - the server does not maintain any information about the client between requests
 - **Use of resource-based URLs** - each resource is identified by a unique URL, and the server responds to requests for that resource by returning the appropriate data.

- **Re**presentational **S**tate **T**ransfer (**REST**)
 - Architecture for **client-server communication** over HTTP
 - Resources have **URI** (address)
 - Can be **created** / **retrieved** / **modified** / **deleted** / etc.
- RESTful API / RESTful Service
 - Provides access to **server-side resources** via **HTTP** and **REST**



- **HTTP** is text-based client-server protocol for the Internet



- **RESTful APIs** are HTTP-based Web services (backend apps)

REST and RESTful Services – Example

- **Get all posts / specific post**

GET	http://some-service.org/api/posts
GET	http://some-service.org/api/posts/17

- **Create a new post**

POST	http://some-service.org/api/posts
------	---

- **Delete existing post**

DELETE	http://some-service.org/api/posts/17
--------	---

- **Replace / modify existing post**

PUT/PATCH	http://some-service.org/api/posts/17
-----------	---



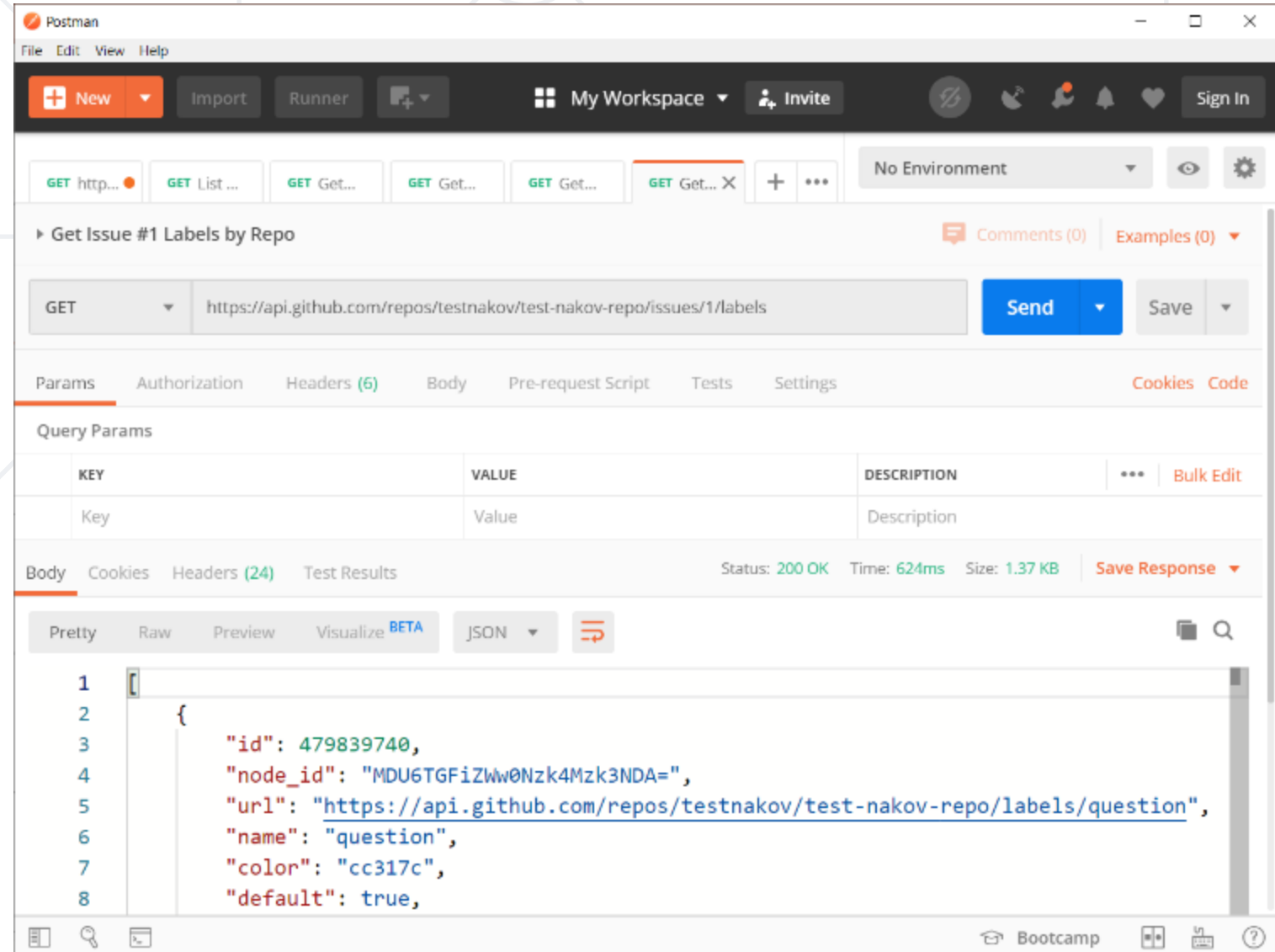
Postman

Testing Tool for RESTful APIs



Postman

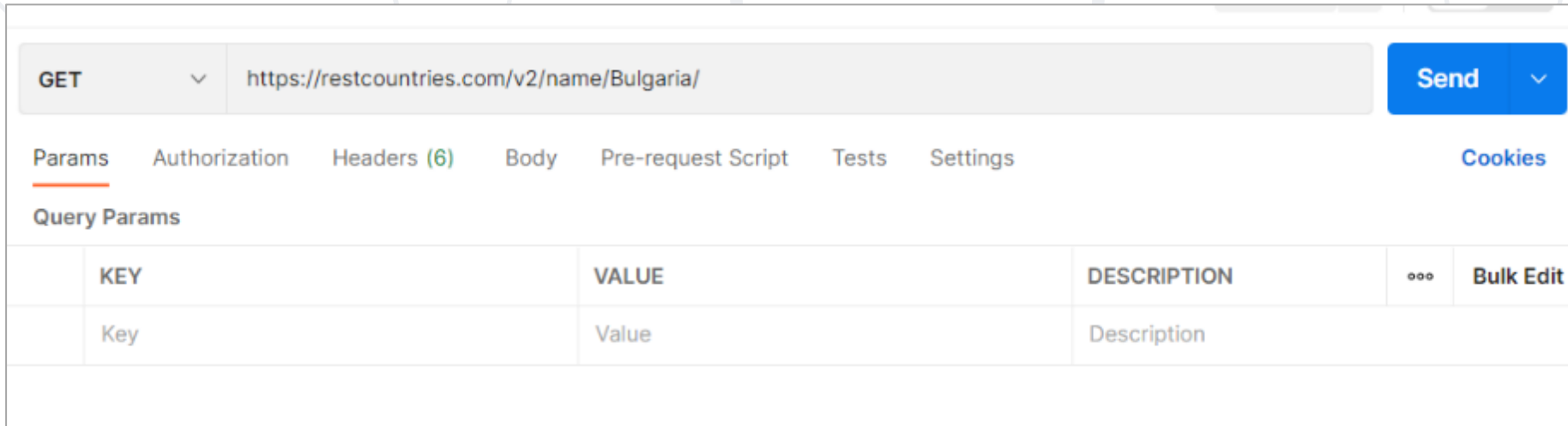
- HTTP client tool for developers and QAs
- Compose and send HTTP requests



- **Postman** is a **free**, and **easy-to-use**, **powerful** tool that can help streamline the process of API development and testing
- **Key features** include
 - **API requests** (GET / POST / PATCH / DELETE / ...)
 - **Collections** of requests
 - **Automation**
 - **Documenting**
 - **Integrations**
- Popular choice among developers and QAs, as it allows them to **quickly test** and **debug API requests** without writing a lot of code

Postman – Send Your First Request

- Create a new "GET" request to the following link
 - <https://restcountries.com/v2/name/Bulgaria/>



GET <https://restcountries.com/v2/name/Bulgaria/> Send

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

```
{
  "name": "Bulgaria",
  "topLevelDomain": [
    ".bg"
  ],
  "alpha2Code": "BG",
  "alpha3Code": "BGR",
  "callingCodes": [
    "359"
  ],
  "capital": "Sofia",
  "altSpellings": [
    "BG",
    "Republic of Bulgaria",
    "Република България"
  ],
  "subregion": "Eastern Europe",
  "region": "Europe",
  "population": 6927288,
  "latlng": [
    43.0,
    25.0
  ]
}
```

- You should receive detailed information about Bulgaria in JSON format

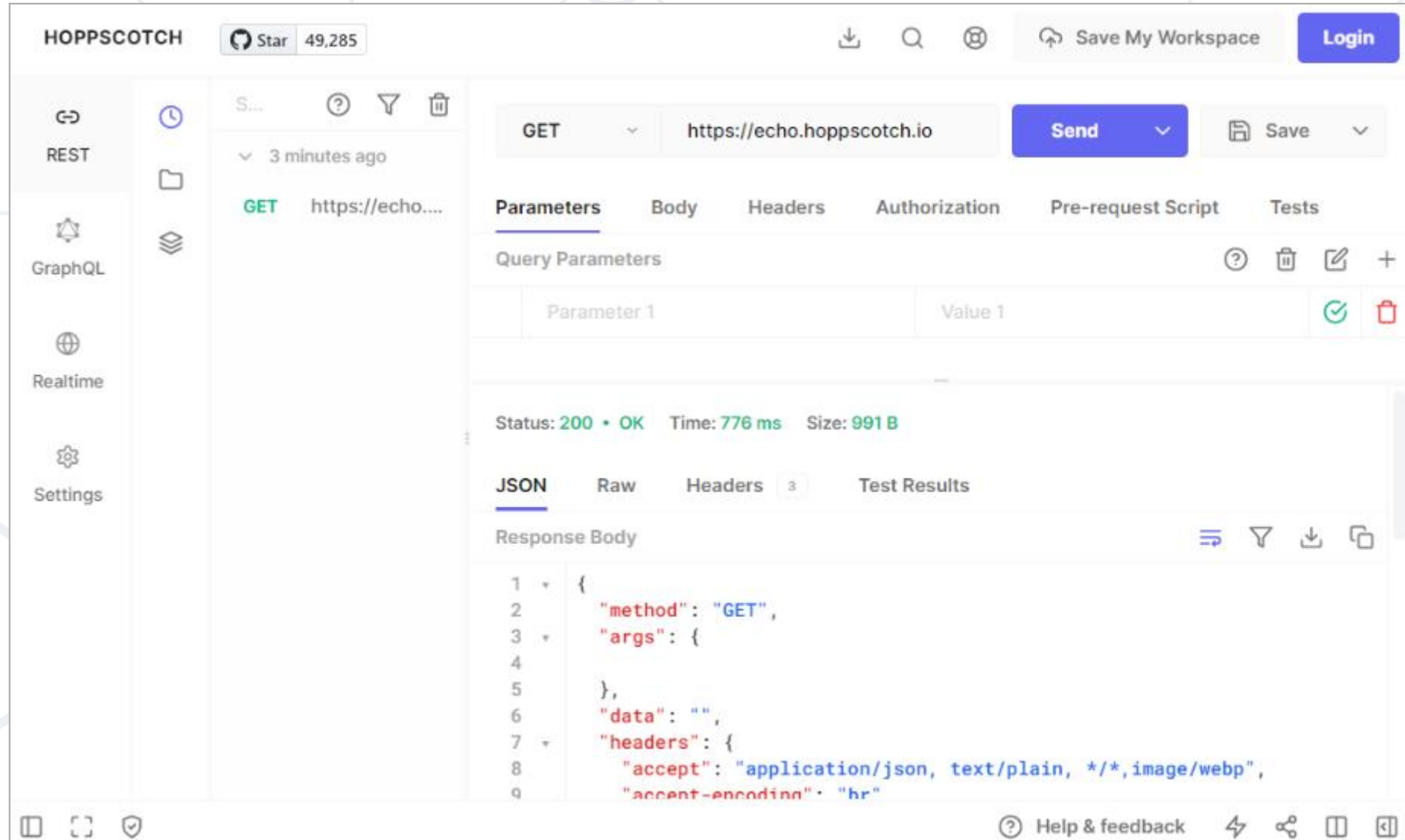
- Each API has **documentation**, where you can see how to use the API. You can find the documentation of this API here
 - <https://restcountries.com>
- Try a few more requests
 - GET only German speaking countries
 - GET only countries in Europe

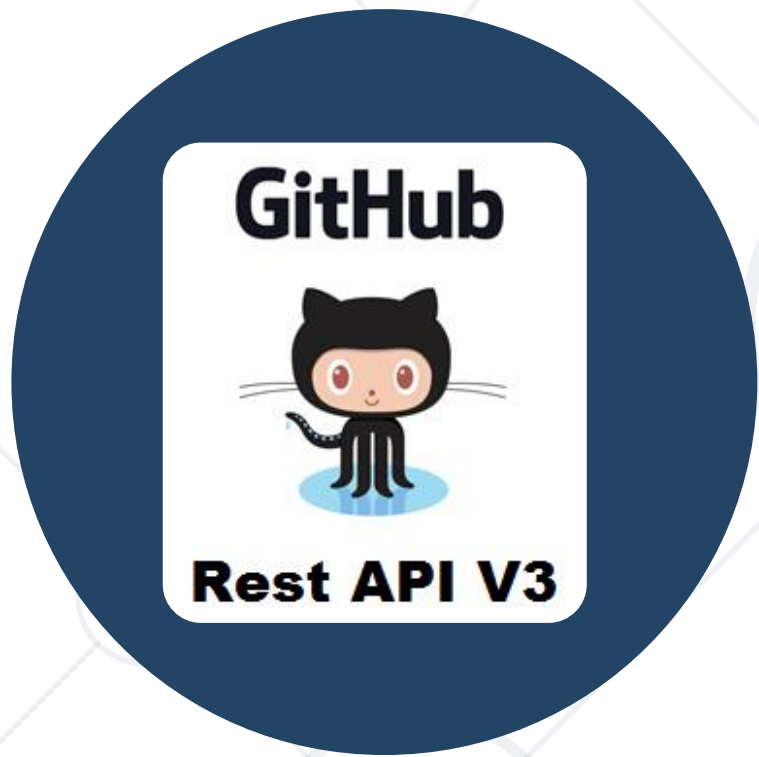
REST COUNTRIES PE

Get information about countries via a RESTful API

Current version: 3.1

- [Hoppscotch.io](https://hoppscotch.io)
- Postman alternative





The GitHub API

Accessing the RESTful API for GitHub Issues

- **GitHub** provides a **public API** for external apps
- It enables developers to **access** and **manipulate** the functionality of GitHub through a variety of methods
- Uses **RESTful principles**, which means that resources are **accessed via a URL**
- Operations on those resources are performed using **standard HTTP methods**, such as GET, POST, PUT, and DELETE
- Also supports **GraphQL** based API access

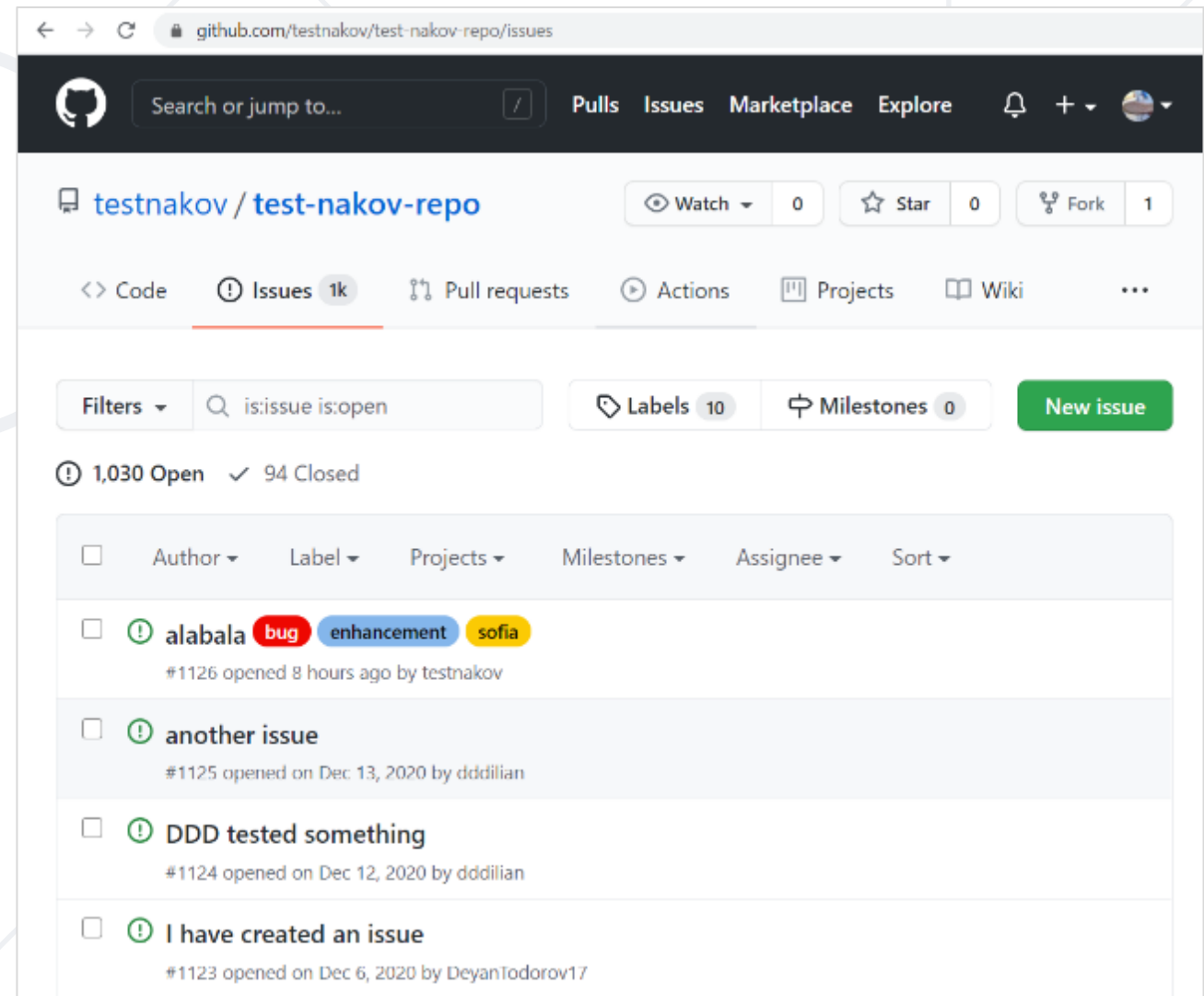
- **Reading** from a public GitHub project is open to everyone

GET	https://api.github.com/repos/testnakov/test-nakov-repo/issues/12
-----	---

- **Modifying** data in a GitHub project requires **authentication**
 - Get an **API access token** from your GitHub profile:
<https://github.com/settings/tokens/new>
 - Use **HTTP basic authentication**: username + token
- To **know more** about how to use the GitHub REST API, check the **documentation**
 - <https://docs.github.com/en/rest?apiVersion=2022-11-28>

Sample GitHub Project with Issues

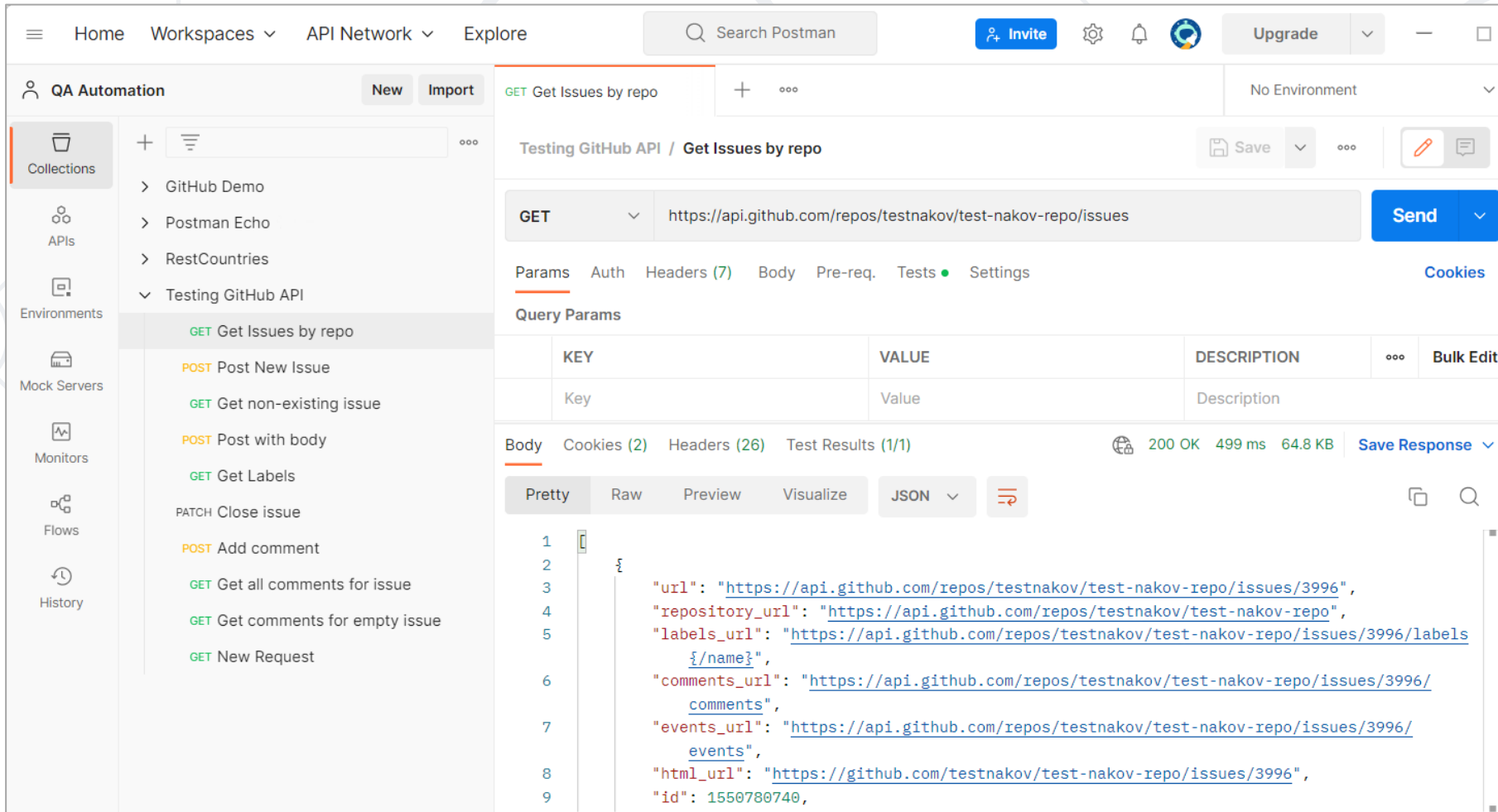
- We shall access the GitHub Issue tracker
 - Using its REST API
- Project URL:
 - <https://github.com/testnakov/test-nakov-repo/issues>
- API URL:
 - <https://api.github.com/repos/testnakov/test-nakov-repo/issues>



Postman Examples: Get Issues from GitHub

GET

<https://api.github.com/repos/testnakov/test-nakov-repo/issues>



The screenshot shows the Postman interface with a GET request configured. The request is named "Get Issues by repo" and is part of a collection called "Testing GitHub API". The URL is `https://api.github.com/repos/testnakov/test-nakov-repo/issues`. The response is displayed in the "Body" tab, showing a JSON object with various URLs and an ID.

Request Details:

- Method: GET
- URL: `https://api.github.com/repos/testnakov/test-nakov-repo/issues`

Response Details:

- Status: 200 OK
- Time: 499 ms
- Size: 64.8 KB

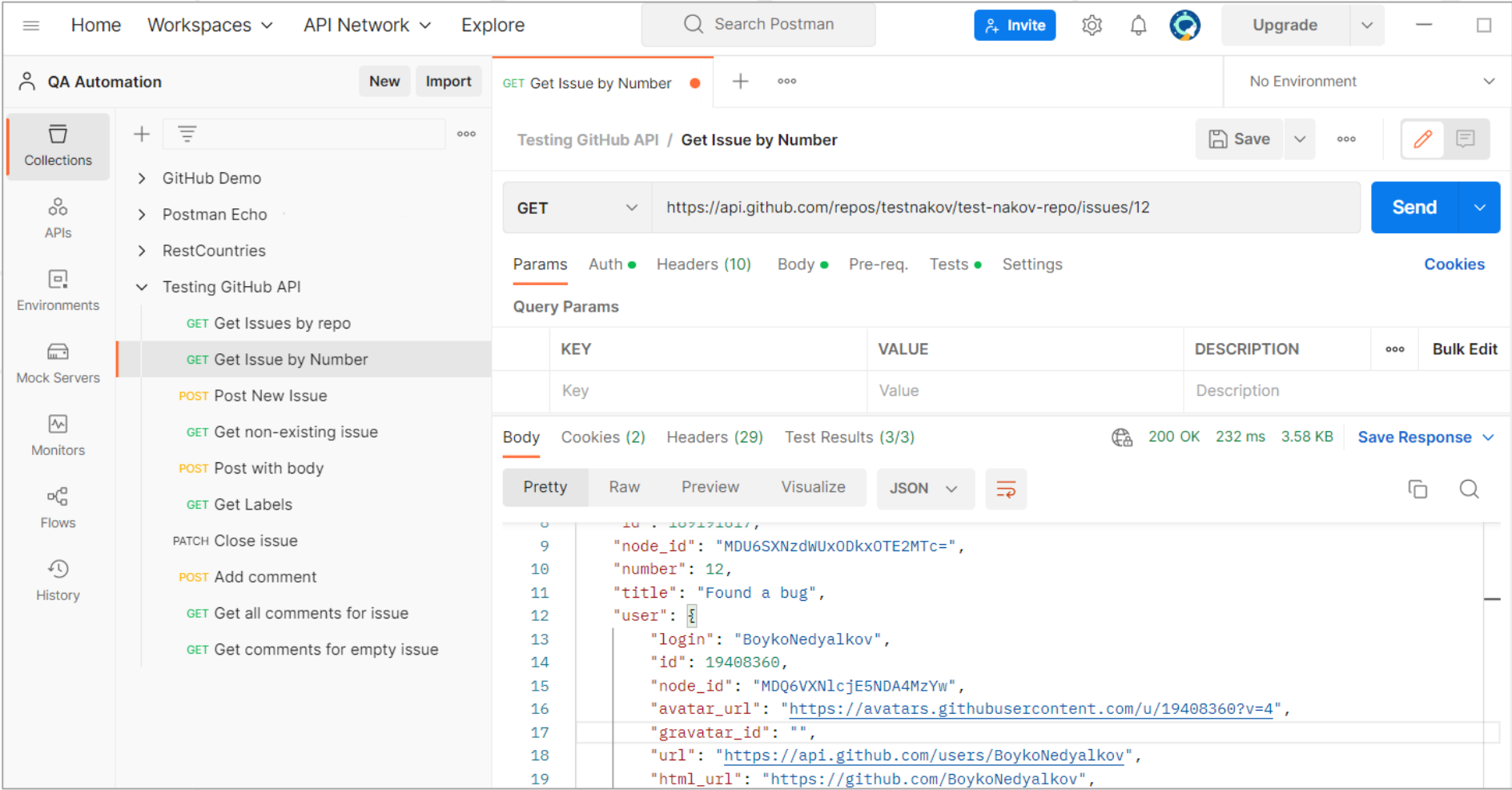
Response Body (JSON):

```
{
  "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3996",
  "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
  "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3996/labels/{name}",
  "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3996/comments",
  "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3996/events",
  "html_url": "https://github.com/testnakov/test-nakov-repo/issues/3996",
  "id": 1550780740,
```

Postman Examples: Get Issue by Number

GET

<https://api.github.com/repos/testnakov/test-nakov-repo/issues/12>



The screenshot shows the Postman interface with a GET request configured. The URL is `https://api.github.com/repos/testnakov/test-nakov-repo/issues/12`. The request is saved in a collection named "Testing GitHub API". The response is a JSON object representing a GitHub issue.

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

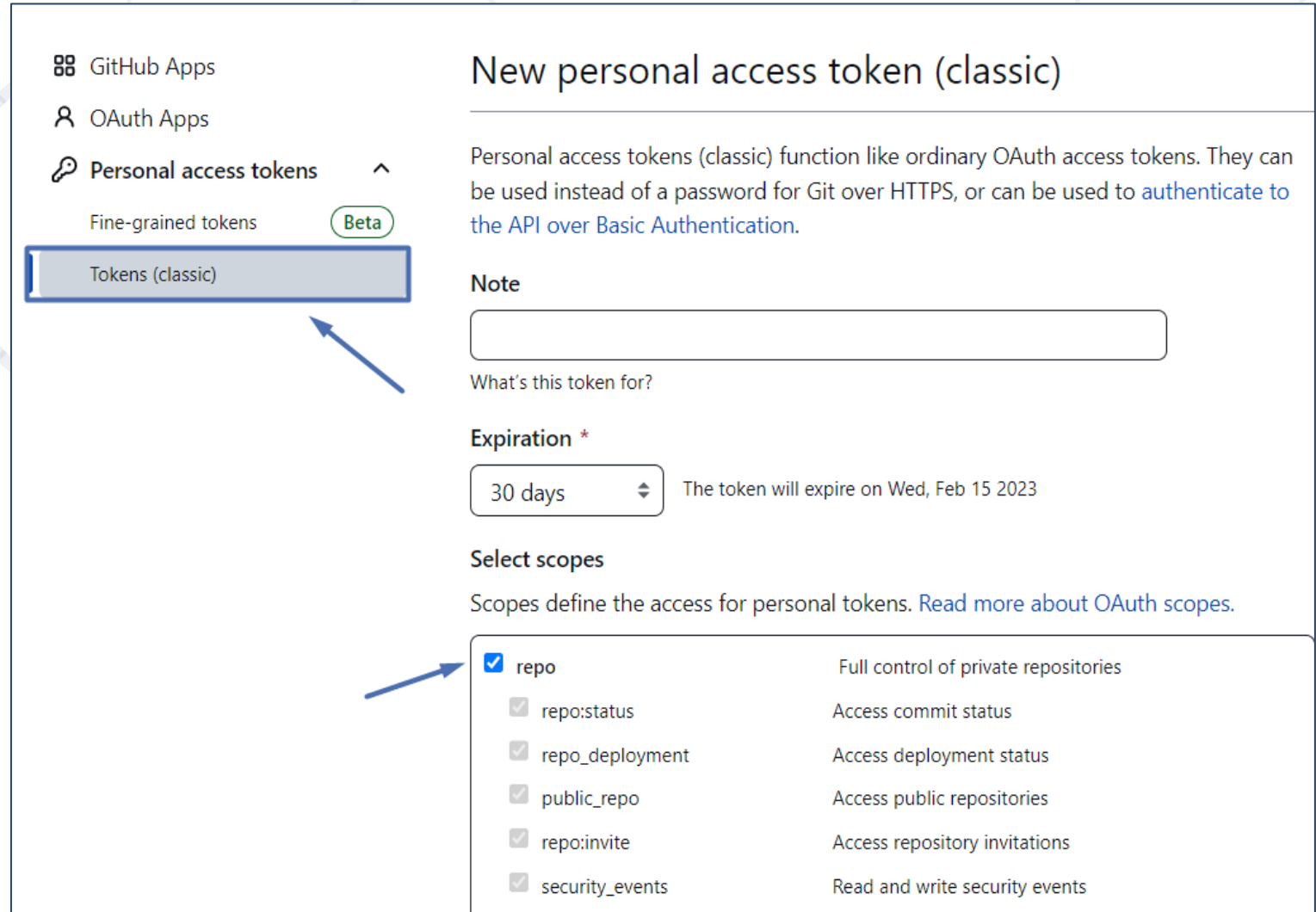
Body

```
8  {
9    "node_id": "MDU6SXNzdWUxODkxOTE2MTc=",
10   "number": 12,
11   "title": "Found a bug",
12   "user": {
13     "login": "BoykoNedyalkov",
14     "id": 19408360,
15     "node_id": "MDQ6VXNlcjE5NDA4MzYw",
16     "avatar_url": "https://avatars.githubusercontent.com/u/19408360?v=4",
17     "gravatar_id": "",
18     "url": "https://api.github.com/users/BoykoNedyalkov",
19     "html_url": "https://github.com/BoykoNedyalkov",
```

Postman Examples: Create New Issue

- Some GitHub API endpoints need **authentication**
- Create new **personal access token** for the GitHub API from your profile:

<https://github.com/settings/tokens/new>



GitHub Apps

OAuth Apps

Personal access tokens ^

Fine-grained tokens **Beta**

Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration *

30 days The token will expire on Wed, Feb 15 2023

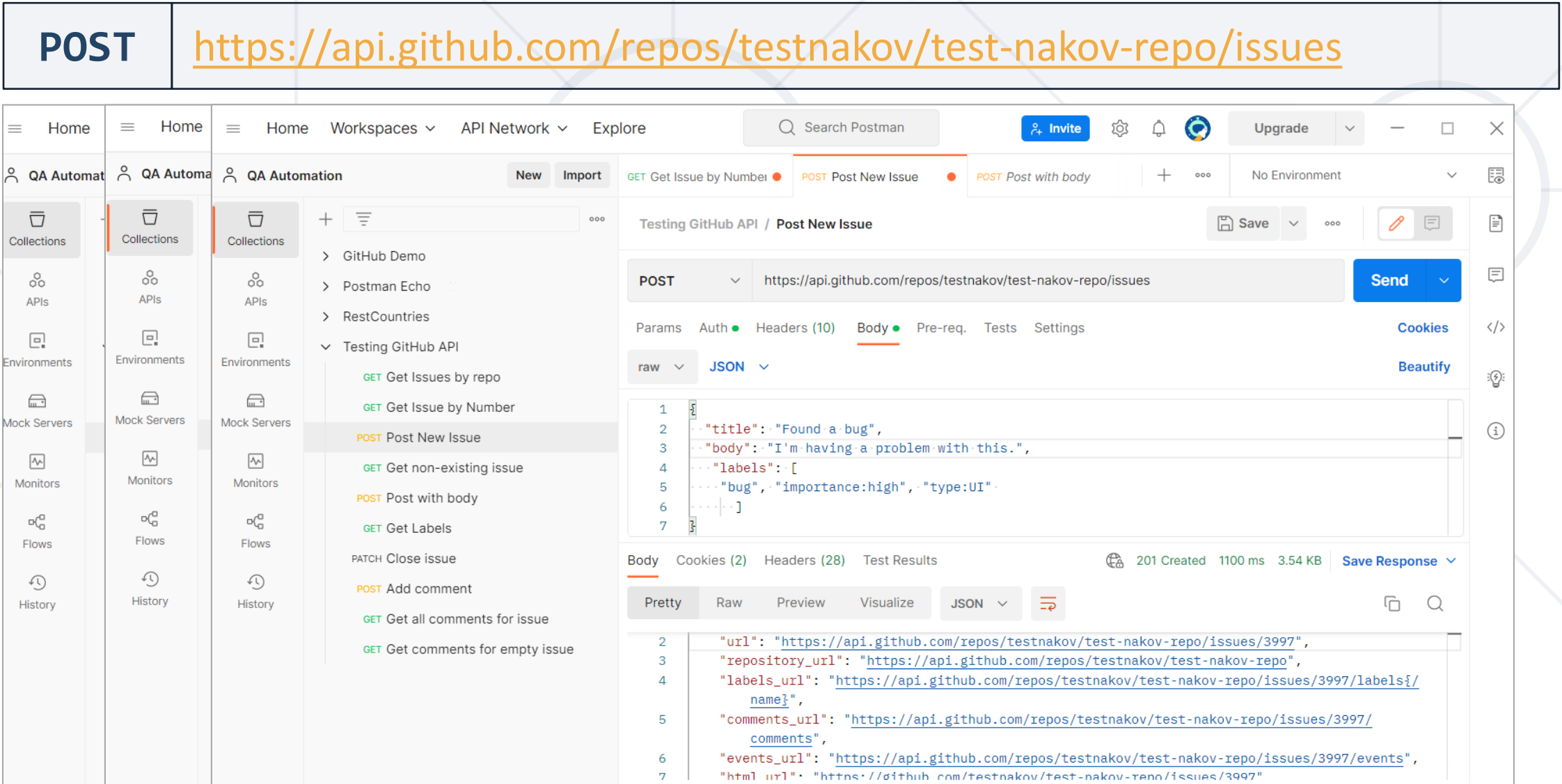
Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events

Postman Examples: Create New Issue

POST <https://api.github.com/repos/testnakov/test-nakov-repo/issues>



The screenshot displays the Postman application interface. On the left, a sidebar shows a collection named 'Testing GitHub API' with a sub-collection 'Post New Issue' selected. The main workspace shows the details of this POST request. The URL is `https://api.github.com/repos/testnakov/test-nakov-repo/issues`. The request body is in JSON format, containing fields for title, body, and labels. The response is also in JSON format, showing the created issue's details including its URL, repository URL, labels URL, comments URL, events URL, and HTML URL.

POST <https://api.github.com/repos/testnakov/test-nakov-repo/issues>

Testing GitHub API / Post New Issue

POST `https://api.github.com/repos/testnakov/test-nakov-repo/issues` **Send**

Params Auth Headers (10) **Body** Pre-req. Tests Settings Cookies Beautify

raw JSON

```
1 {
2   "title": "Found a bug",
3   "body": "I'm having a problem with this.",
4   "labels": [
5     "bug", "importance:high", "type:UI"
6   ]
7 }
```

Body Cookies (2) Headers (28) Test Results 201 Created 1100 ms 3.54 KB Save Response

Pretty Raw Preview Visualize JSON

```
2 "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3997",
3 "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
4 "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3997/labels/{name}",
5 "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3997/comments",
6 "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3997/events",
7 "html_url": "https://github.com/testnakov/test-nakov-repo/issues/3997"
```

- List user's all public repositories:

GET	https://api.github.com/users/testnakov/repos
-----	---

- Get all commits from a public repository:

GET	https://api.github.com/repos/testnakov/softuniada-2016/commits
-----	---

- Get all issues/issue #1 from a public repository

GET	/repos/testnakov/test-nakov-repo/issues
-----	---

GET	/repos/testnakov/test-nakov-repo/issues/1
-----	---

- Get the first issue from the "**test-nakov-repo**" repository
- Send a **GET** request to:
 - <https://api.github.com/repos/testnakov/test-nakov-repo/issues/:id>
 - Where **:id** is the current issue



- Get all labels for certain issue from a public repository:

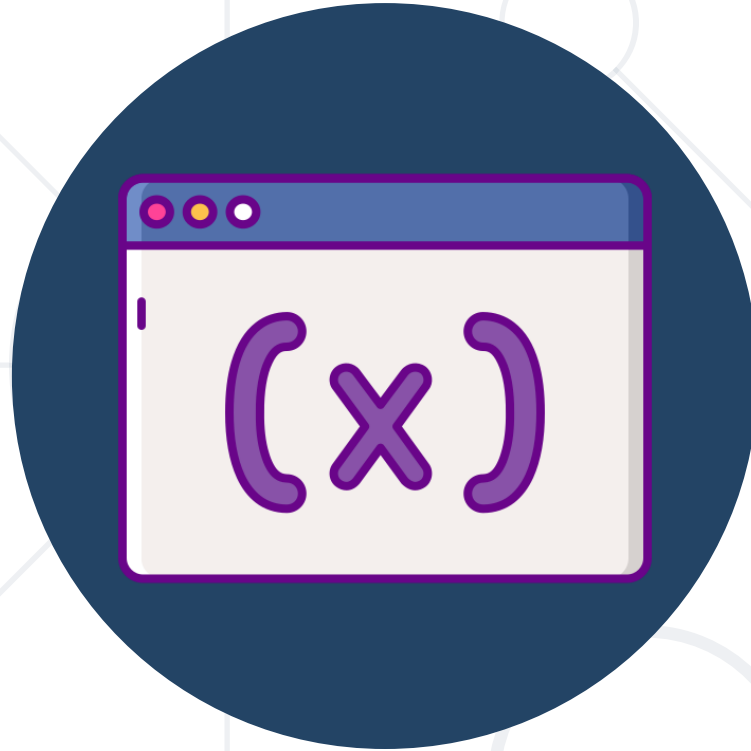
GET	https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/labels
------------	---

- Create a new issue to certain repository (with authentication)

POST	https://api.github.com/repos/testnakov/test-nakov-repo/issues
-------------	---

Headers	Authorization: Basic base64(user:pass)
----------------	--

Body	<pre>{"title": "Found a bug", "body": "I'm having a problem with this."}</pre>
-------------	--

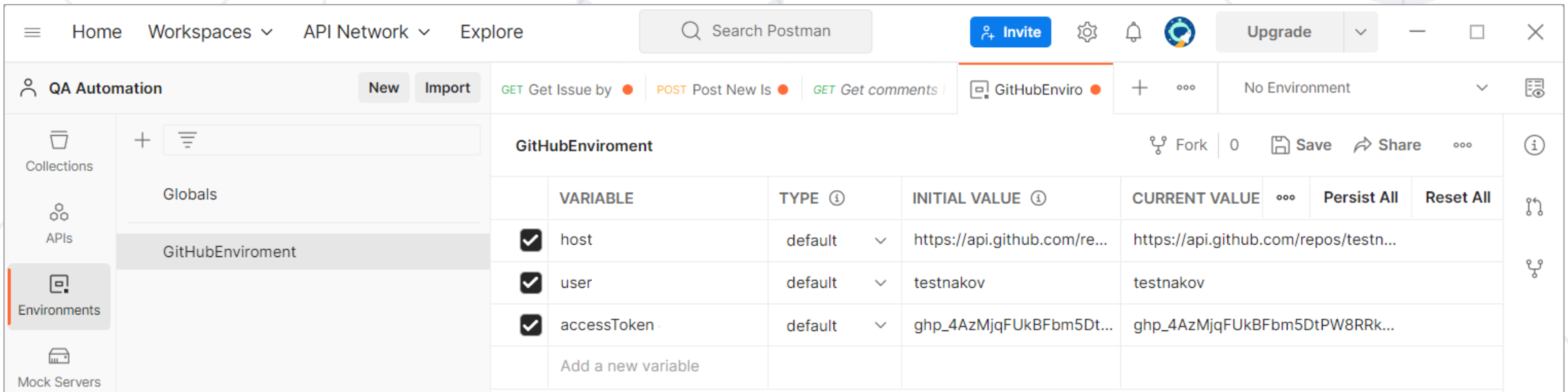


Variables

Parameterize requests

Variables in Postman

- Postman supports **environment variables**
- Can be used to parameterize the requests
- Can be edited by the test scripts



The screenshot shows the Postman interface with the 'Environments' section active. The 'GitHubEnviroment' is selected, displaying a table of variables. The table has columns for 'VARIABLE', 'TYPE', 'INITIAL VALUE', and 'CURRENT VALUE'. The variables listed are 'host', 'user', and 'accessToken', all of type 'default'. The 'CURRENT VALUE' column shows the values being used in the environment.

VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
host	default	https://api.github.com/re...	https://api.github.com/repos/testn...
user	default	testnakov	testnakov
accessToken	default	ghp_4AzMjqFUKBFbm5Dt...	ghp_4AzMjqFUKBFbm5DtPW8RRk...

View / Edit Variables

GitHub Enviroment

Fork

0

Save

Share

...

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	host	default ▾	api.github.com/repos/testnakov/test-n...	api.github.com/repos/testnakov/test-nakov-repo			
<input checked="" type="checkbox"/>	user	default ▾	testnakov	testnakov			
<input checked="" type="checkbox"/>	token	default ▾	ghp_4AzMjqFukBFbm5DtPW8RRkHw1...	ghp_4AzMjqFukBFbm5DtPW8RRkHw1cgeGI4JA0C7			
	Add a new variable						

GitHub Enviroment			Edit
VARIABLE	INITIAL VALUE	CURRENT VALUE	
host	api.github.com/repos/testnakov/test-nakov-repo	api.github.com/repos/testnakov/test-nakov-repo	
user	testnakov	testnakov	
token	ghp_4AzMjqFukBFbm5DtPW8RRkHw1cgeGI4JA0C7	ghp_4AzMjqFukBFbm5DtPW8RRkHw1cgeGI4JA0C7	

Using Variables for Authentication

Authorization ● Pre-request Script Tests Variables Runs

This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Type Basic Auth ▾

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#) ↗

ⓘ Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#) ↗

✕

Username {{user}}

Password {{token}}

☒ Show Password

Using Variables in Requests

- Define an environment variable

GitHub Enviroment Fork 0 Save Share ...

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	host	default ▾	api.github.com/repos/testnakov/test-n...	api.github.com/repos/testnakov/test-nakov-repo			

- Use the variable:

Testing GitHub API / Post New Issue Save ▾ ... ✎ 💬

POST ▾ https://{{host}}/issues Send ▾

Params Authorization • Pre-request Script Tests Settings Cookies

Query Params

E host

INITIAL api.github.com/repos/testnakov/test-nakov-repo

CURRENT api.github.com/repos/testnakov/test-nakov-repo

SCOPE Environment

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Running Postman Collections and Tests

Testing GitHub API

GET

GET

POST

GET

POST

Share

Move

Run collection

Edit

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	GitHub Enviroment	1	5s 438ms	4	411 ms

All Tests

Passed (4)

Failed (0)

Skipped (0)

View Summary

Iteration 1

GET

Get Issues by repo

https://issues

/ Get Issues by repo

200 OK

508 ms

66.267 KB

Pass

HTTP status code

GET

Get Issue by Number

https://api.github.com/repos/testnakov/test-nakov-repo/issues/12

/ Get Issue by Number

200 OK

243 ms

3.661 KB

Pass

Issue number is correct

Pass

Issue is open

Pass

Issue was created by the correct user



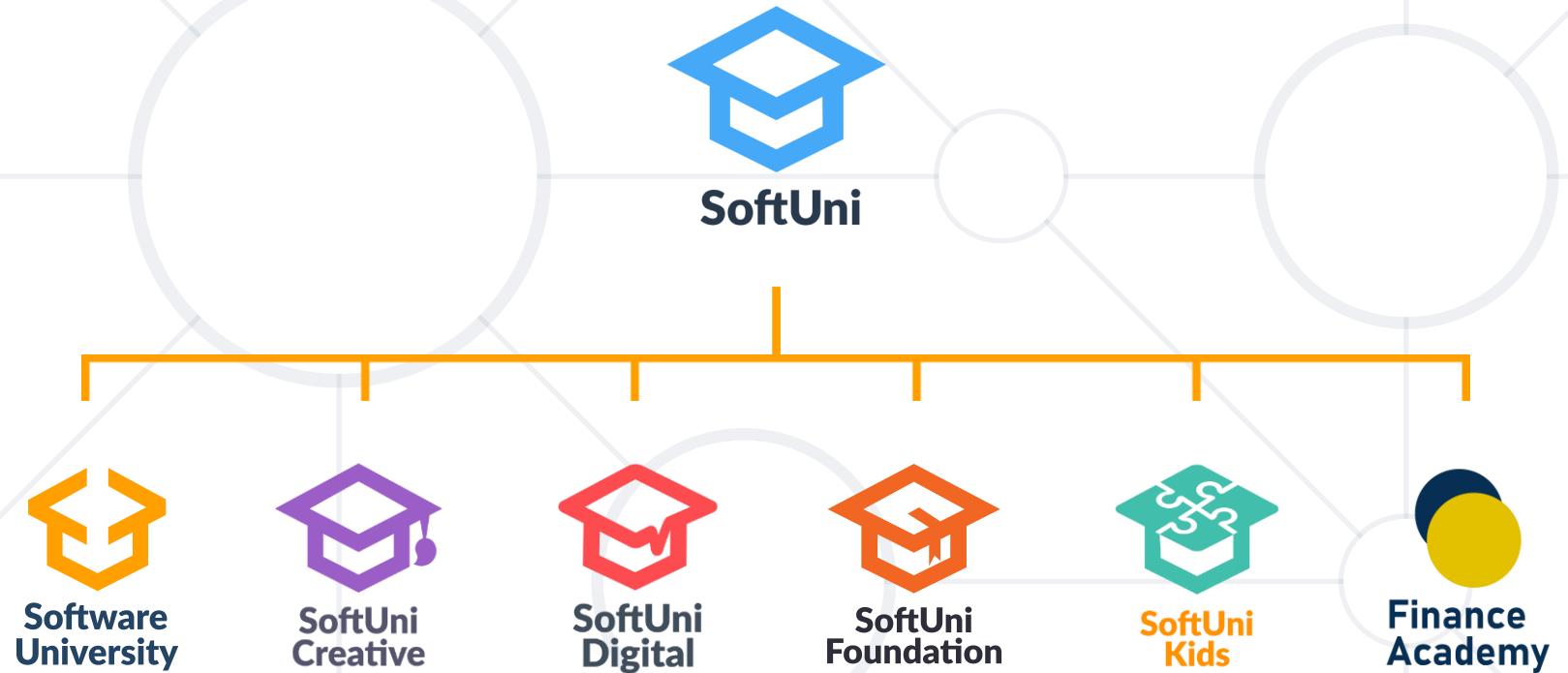
What is API Testing?

Live Demo

- **Web Services** and **APIs**
- **HTTP** used for data transfer between clients and servers on the web.
- **XML** for structuring and exchanging data
- **JSON** commonly used in web **APIs**
- **REST** and **RESTful Services**
- **Postman Overview**, creating and sending **HTTP** requests
- **Accessing** the **RESTful API** for GitHub Issues



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**



POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**



**SOFTWARE
GROUP**

createX



Postbank
Решения за твоето утре

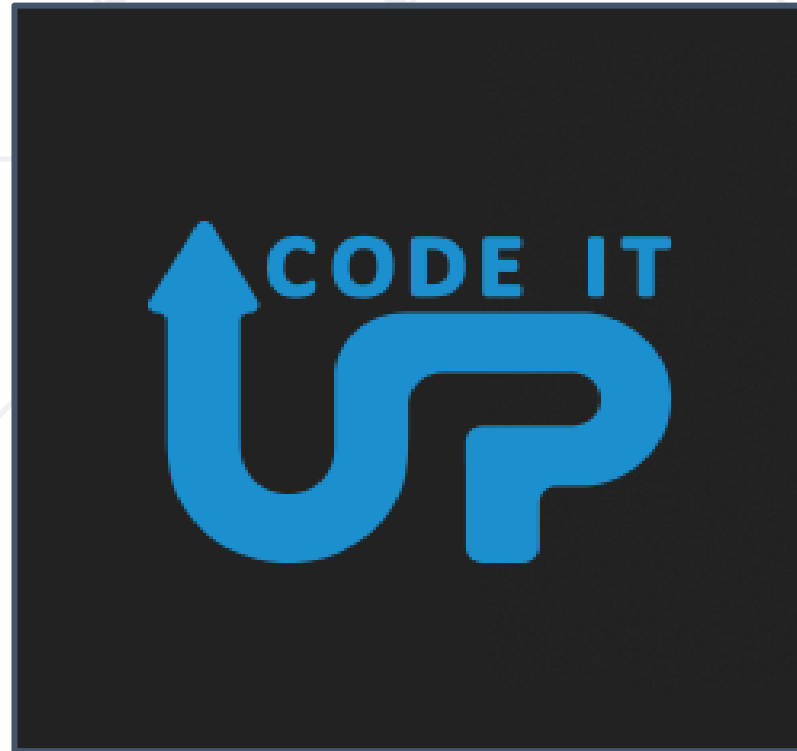


BOSCH

DXC
TECHNOLOGY



SmartIT



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

