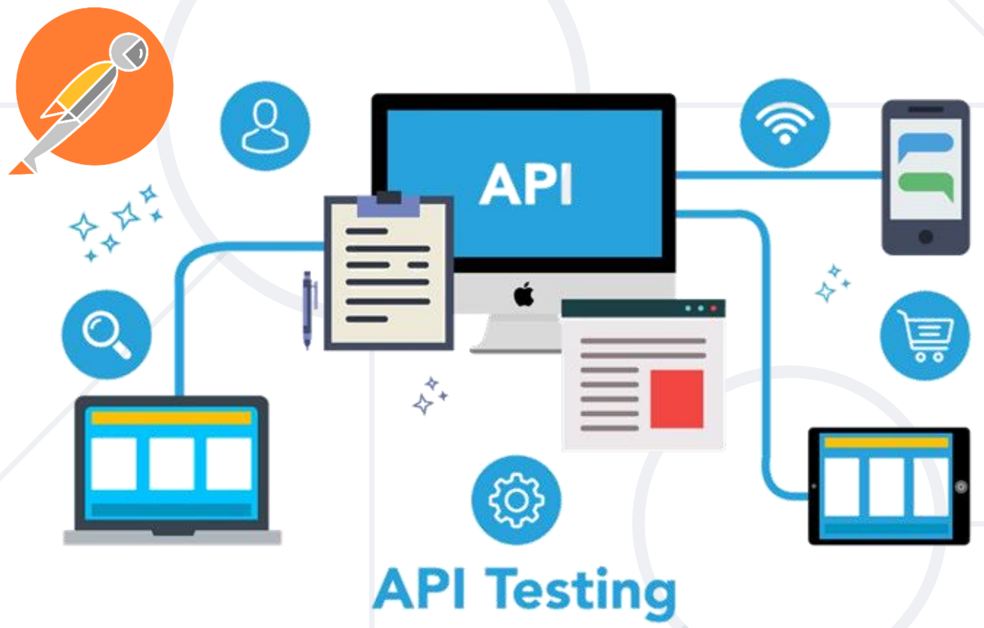
Web API and Postman

Web APIs, HTTP Protocol, REST and Postman



SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg/

Have a Question?

Software University

sli.do

#qa-fund

Table of Contents

Web Services

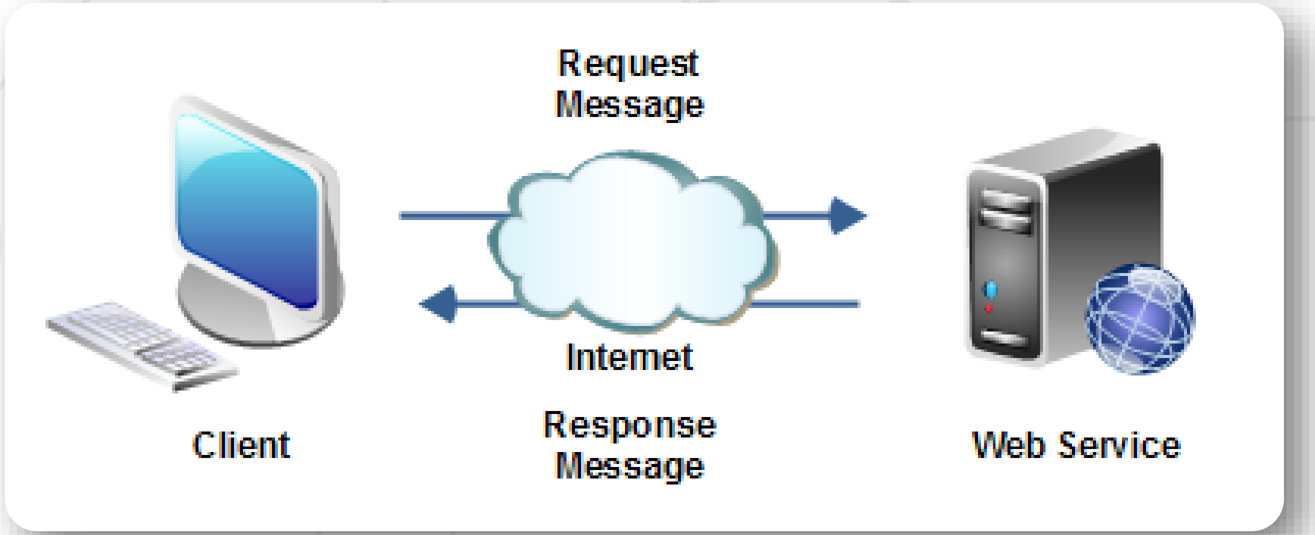
Communication between Systems and Components

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 and APIs

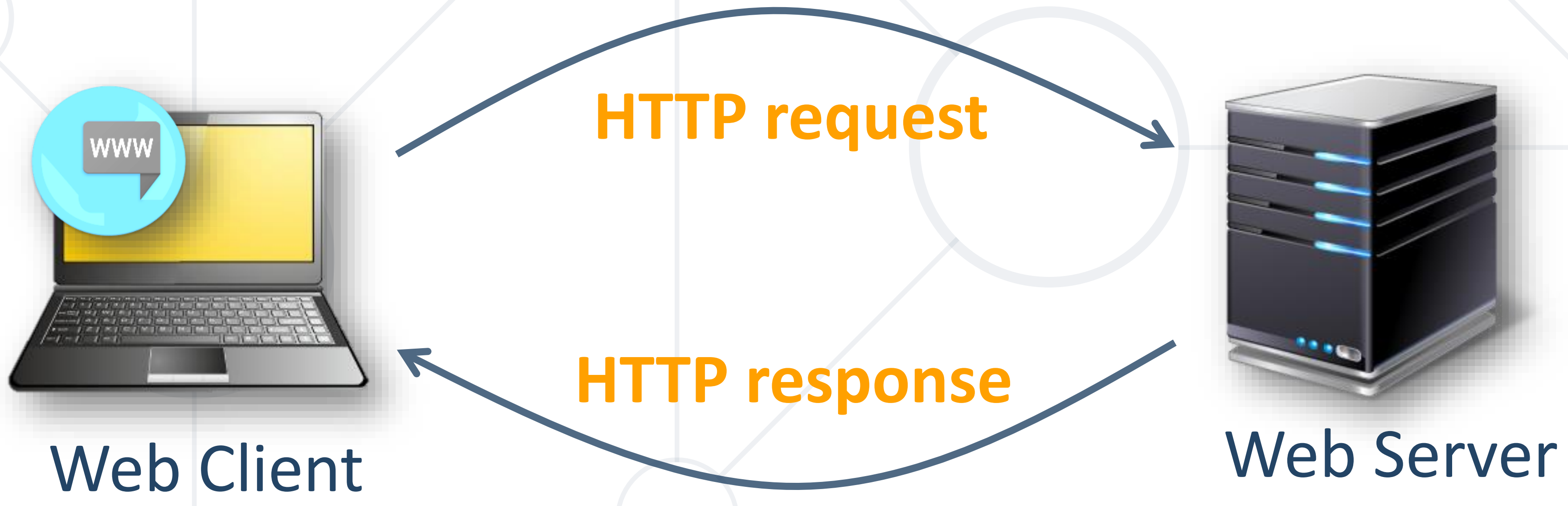
- **Web services** expose **back-end APIs** over the **network**
  - May use different **protocols** and **data formats**: **HTTP**, **REST**, **GraphQL**, **gRPC**, JSON-RPC, JSON, BSON, XML, YML, SOAP…
- **Web services** are hosted on a Web server (HTTP server)
  - Provide a set of functions, invokable 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

http://

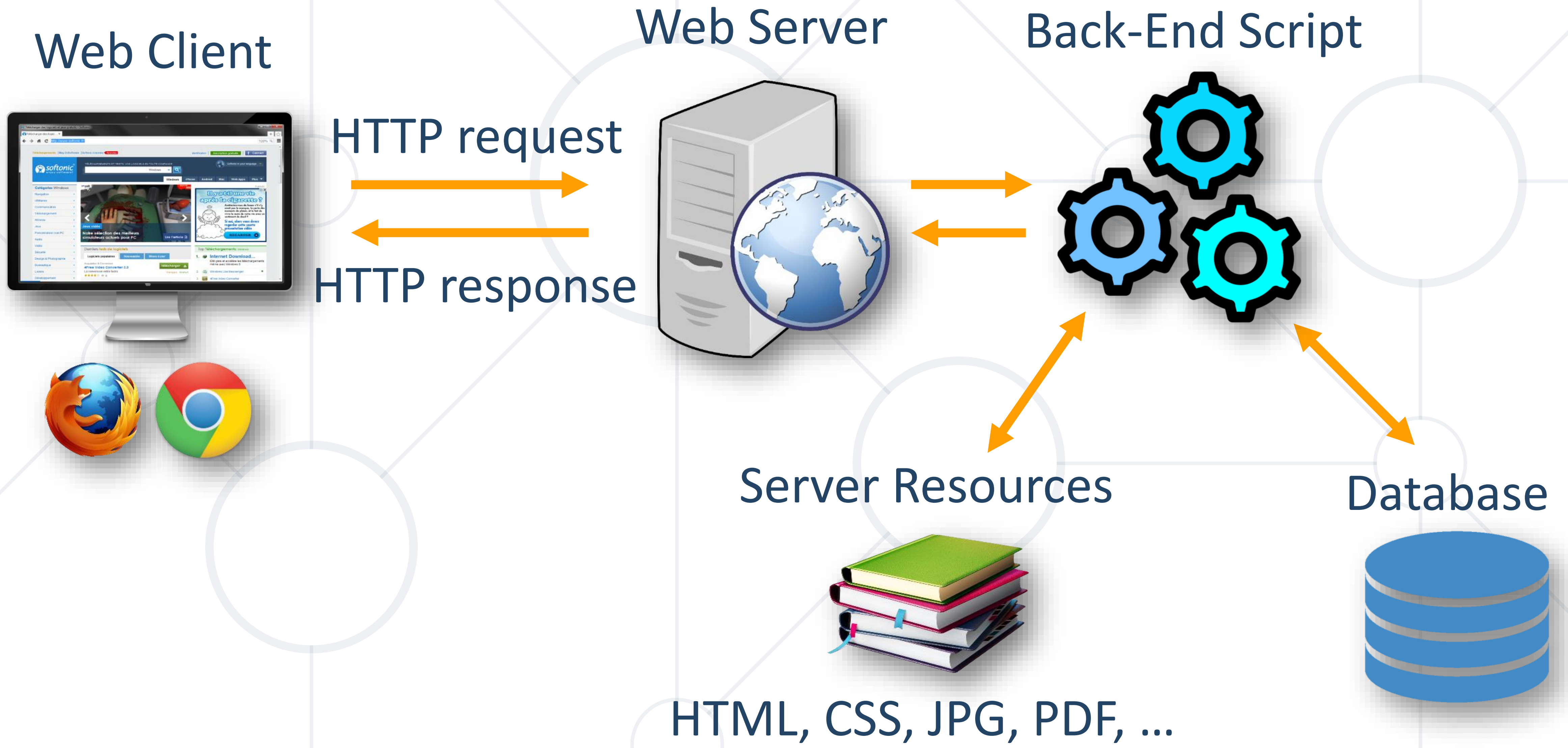
HTTP Protocol – Basics

HTTP 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
GET	**R**etrieve a resource
POST	**C**reate / store a resource
PUT	**U**pdate (replace) a resource
DELETE	**D**elete (remove) a resource
PATCH	**U**pdate resource partially (modify)
HEAD	**R**etrieve the resource's headers

Other Methods
CONNECT
OPTIONS
TRACE

CRUD == the four main functions of persistent storage

HTTP GET Request – Example

```
GET /users/SoftUni-Tech-Module/repos HTTP/1.1
Host: api.github.com
Accept: */*
Accept-Language: en
Accept-Encoding: gzip, deflate, br
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64; rv:103.0)
 AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/103.0.5060.134
Connection: keep-alive
Cache-Control: no-cache

<CRLF>
```

Relative URI, not full URL

HTTP request line

The host is part of the URL

HTTP headers

The request body is empty

HTTP Response – Example

```
HTTP/1.1 200 OK

Date: Fri, 04 Feb 2023 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Accept-Ranges: bytes
Content-Length: 80
Content-Type: application/json
<CRLF>

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

HTTP response status line

HTTP response headers

Describes the returned content

HTTP response body

XML

Specifics and Structure

XML Specifics

- **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

XML Markup and Content

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

person.xml

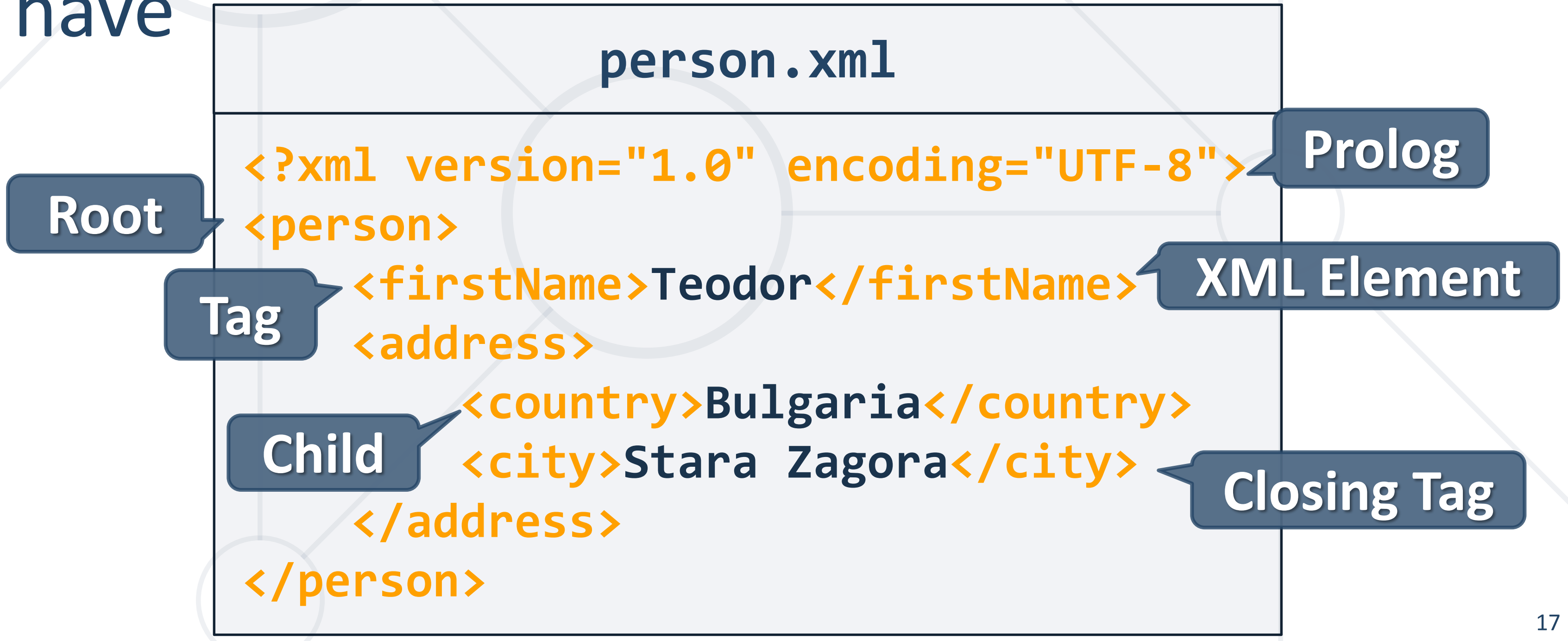
```
<?xml version="1.0" encoding="UTF-8">
<person>
    <firstName>Teodor</firstName>
</person>
```

Markup tags for Person Object

Content (Person Name)

XML Structure

- 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:



person.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <firstName>Teodor</firstName>
    <address>
        <country>Bulgaria</country>
        <city>Stara Zagora</city>
    </address>
</person>
```

JSON

Specifics and Structure

JSON Explained

- 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**

JSON Example

- **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"
}
```

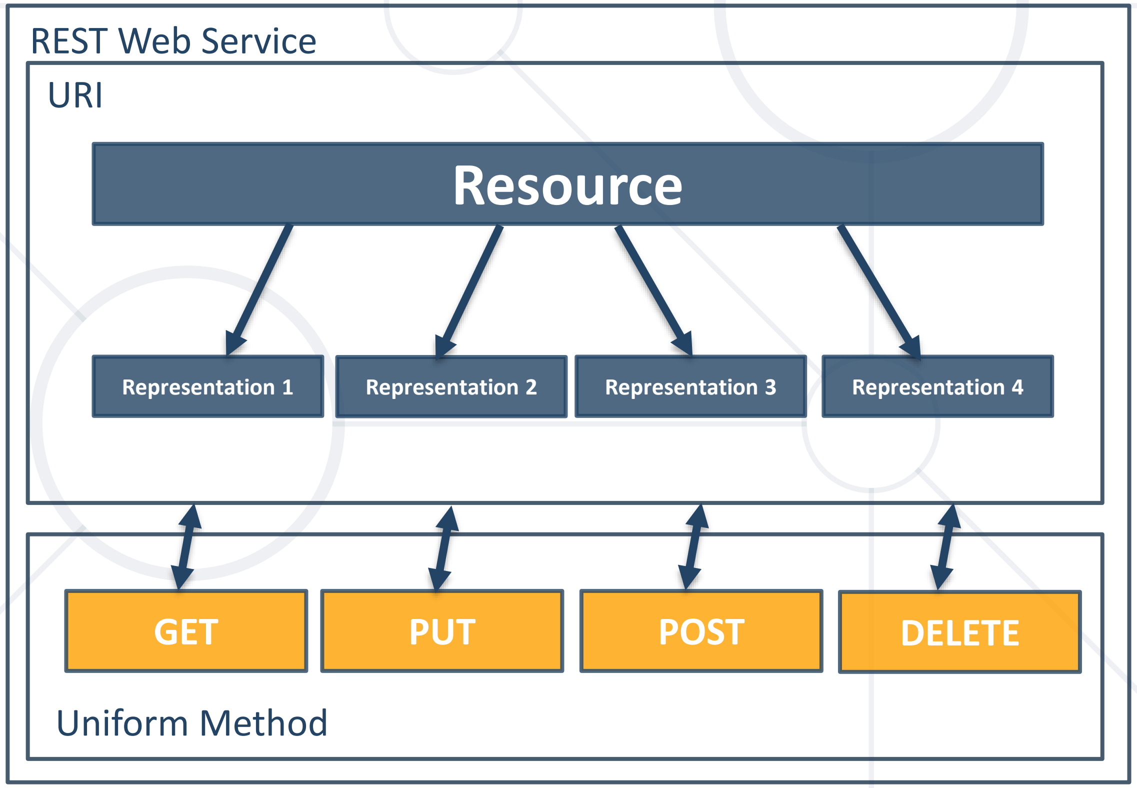
RESTful APIs

REST

- **REST** (**Re**presentational **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.

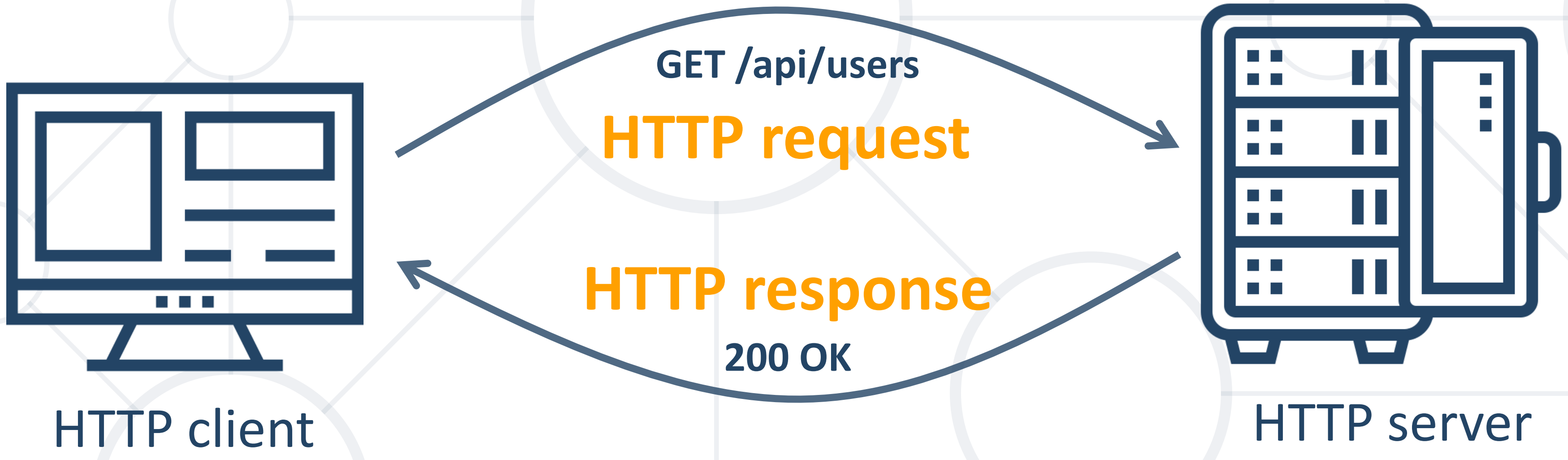
REST and RESTful Services

- **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**



RESTful APIs

- **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

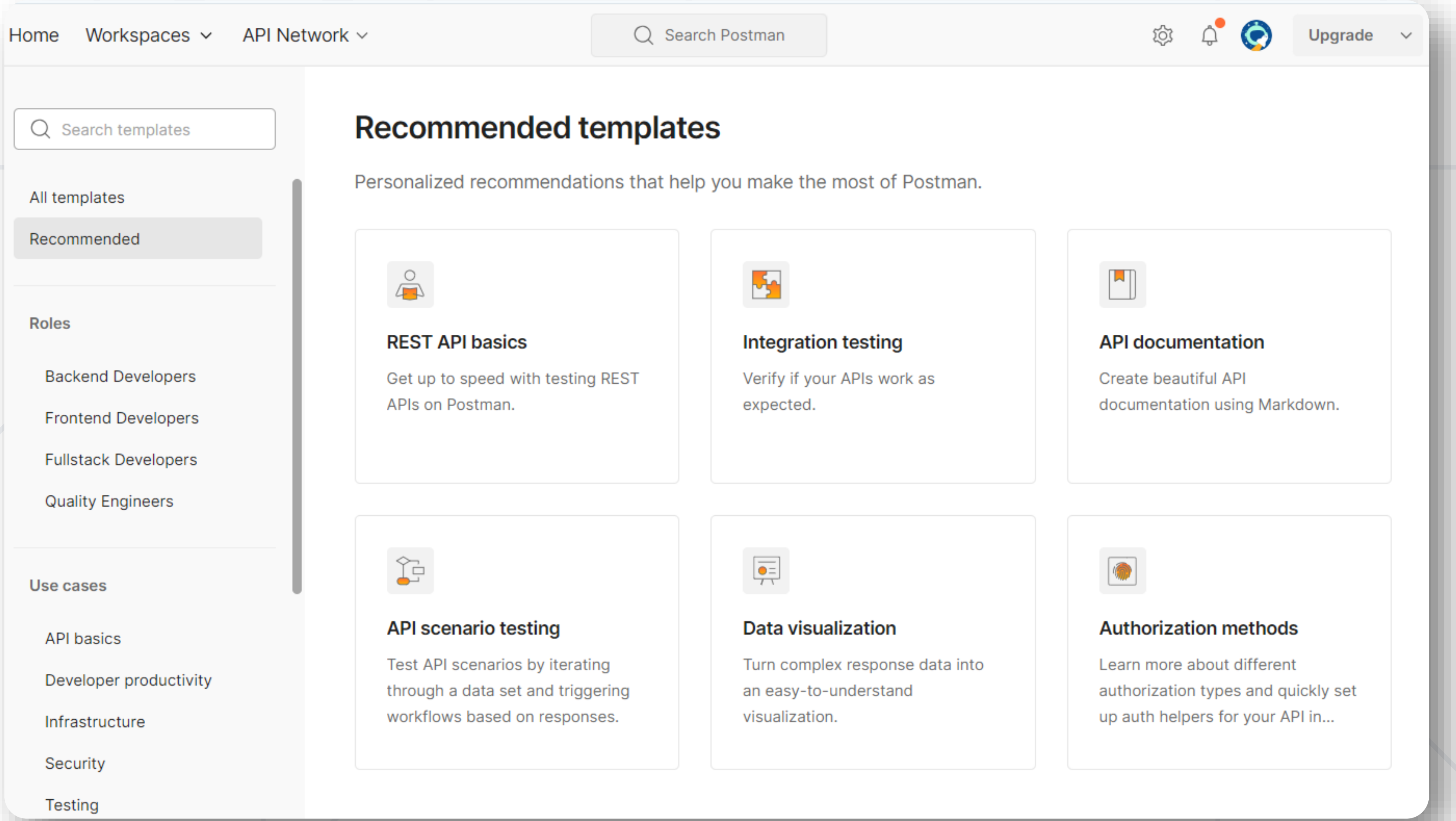
Testing Tool for RESTful APIs

Postman

Software University

Postman

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



Recommended templates

Personalized recommendations that help you make the most of Postman.

REST API basics
Get up to speed with testing REST APIs on Postman.

Integration testing
Verify if your APIs work as expected.

API documentation
Create beautiful API documentation using Markdown.

API scenario testing
Test API scenarios by iterating through a data set and triggering workflows based on responses.

Data visualization
Turn complex response data into an easy-to-understand visualization.

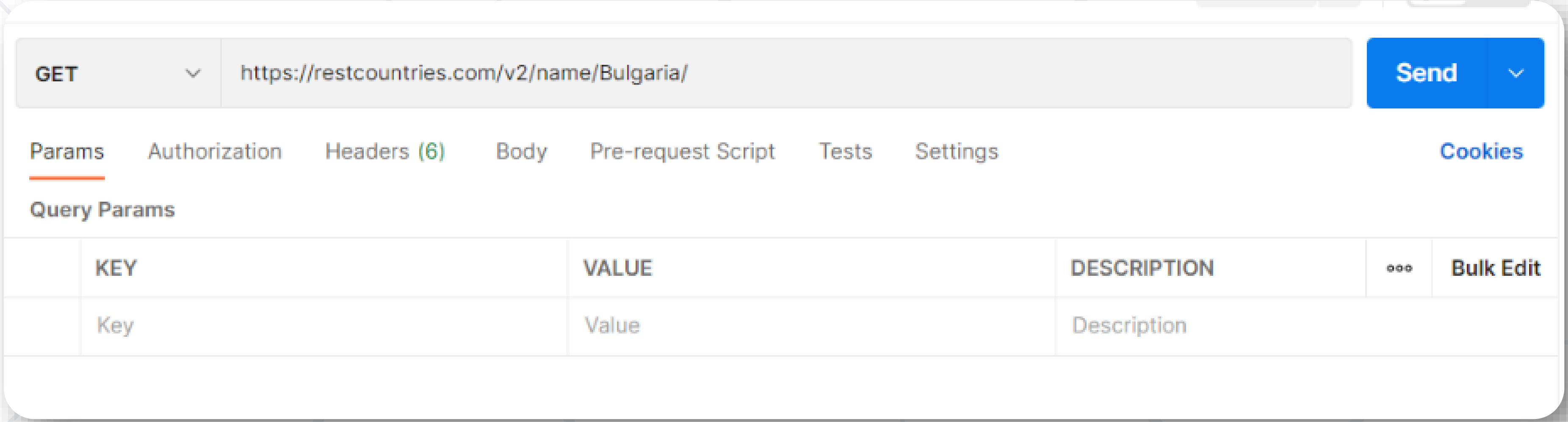
Authorization methods
Learn more about different authorization types and quickly set up auth helpers for your API in...

Postman Overview

- **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/

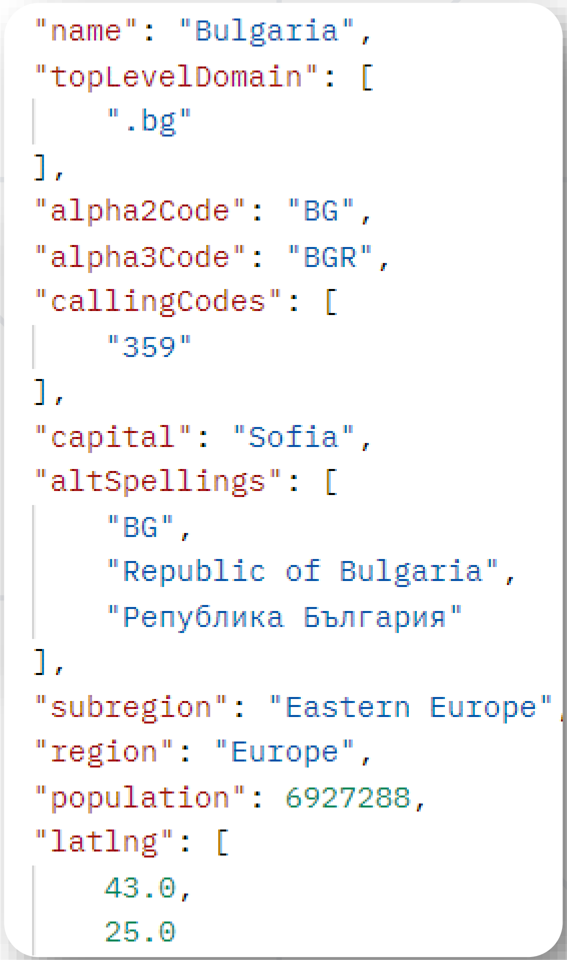


GET https://restcountries.com/v2/name/Bulgaria/ Send

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

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

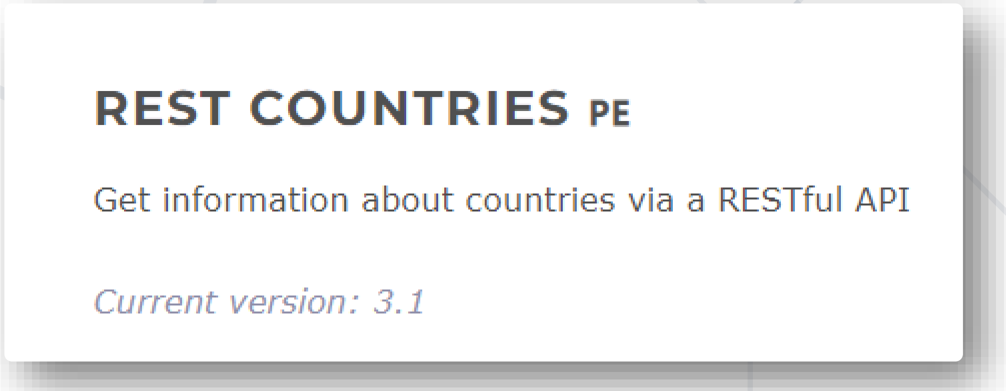


```
"name": "Bulgaria",
"topLevelDomain": [
    ".bg"
],
"alpha2Code": "BG",
"alpha3Code": "BGR",
"callingCodes": [
    "359"
],
"capital": "Sofia",
"altSpellings": [
    "BG",
    "Republic of Bulgaria",
    "Република България"
],
"subregion": "Eastern Europe"
"region": "Europe",
"population": 6927288,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

- You should receive detailed information about Bulgaria in JSON format

Postman – Practice Another Request

- 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



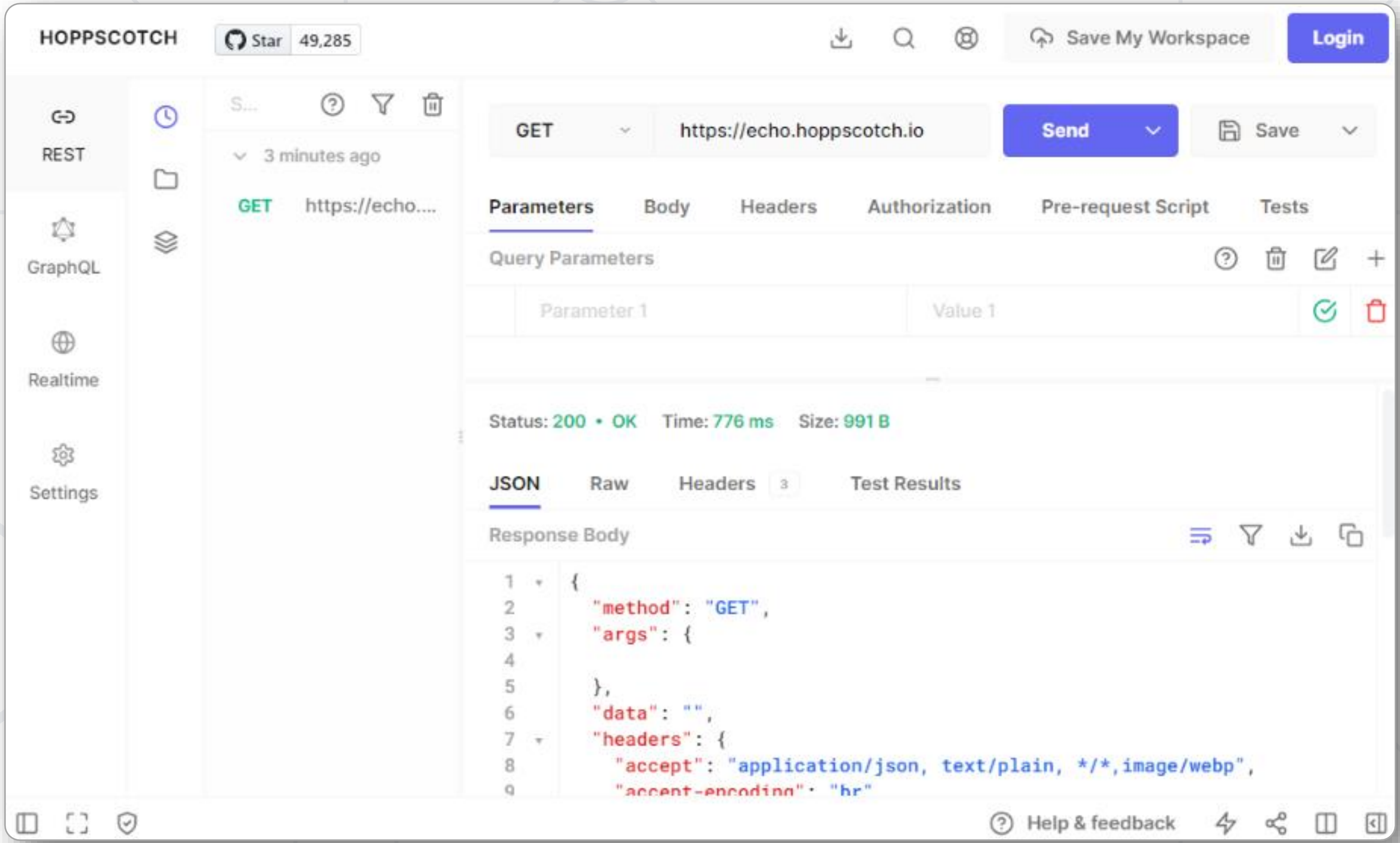
REST COUNTRIES PE

Get information about countries via a RESTful API

Current version: 3.1

Hoppscotch

- Hoppscotch.io
- Postman alternative



```
{
  "method": "GET",
  "args": {

  },
  "data": "",
  "headers": {
    "accept": "application/json, text/plain, */*,image/webp",
    "accept-encoding": "br"
```

The GitHub API

Accessing the RESTful API for GitHub Issues

GitHub API Overview

- **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

GitHub Provides Public API for External Apps

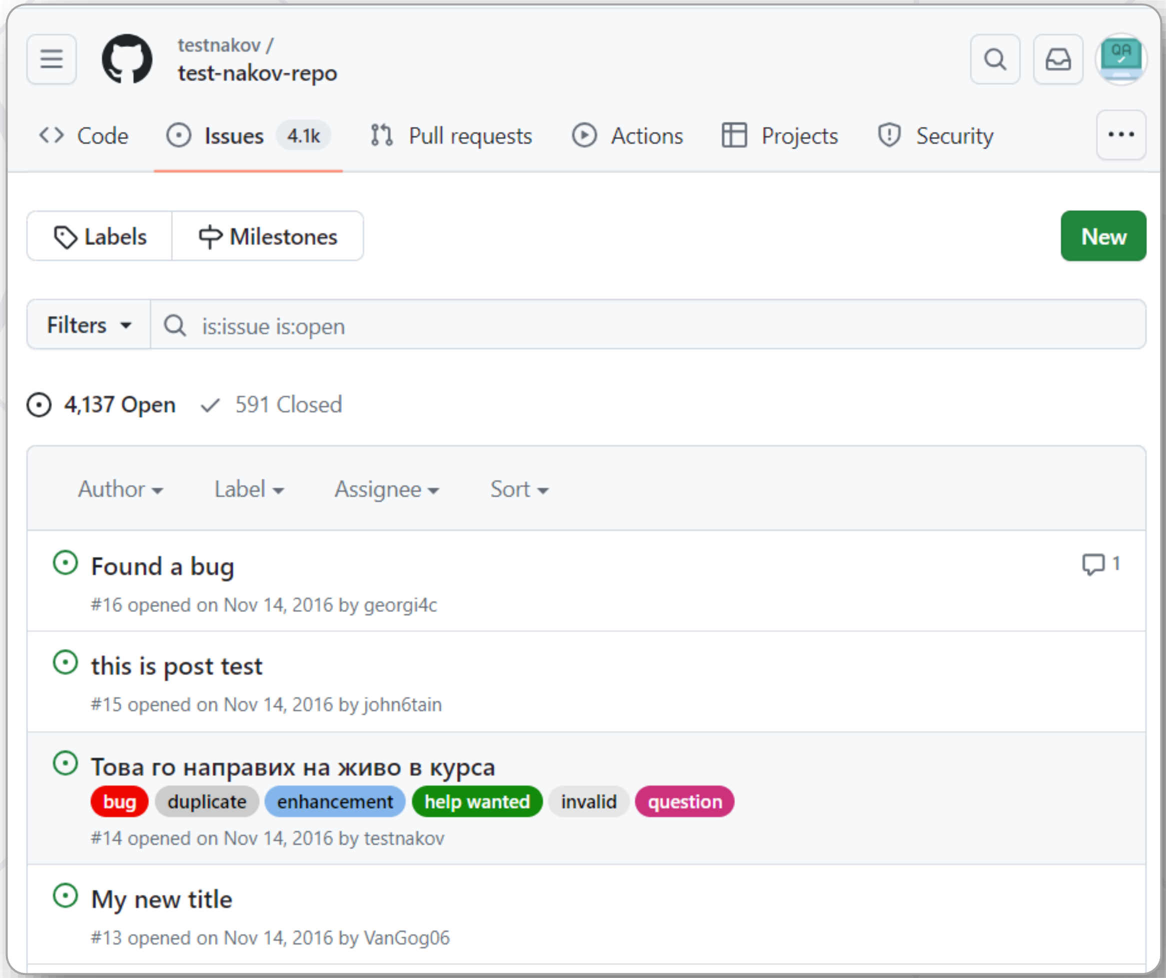
- **Reading** from a public GitHub project is open to everyone

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



4,137 Open ✓ 591 Closed

Found a bug 1
#16 opened on Nov 14, 2016 by georgi4c

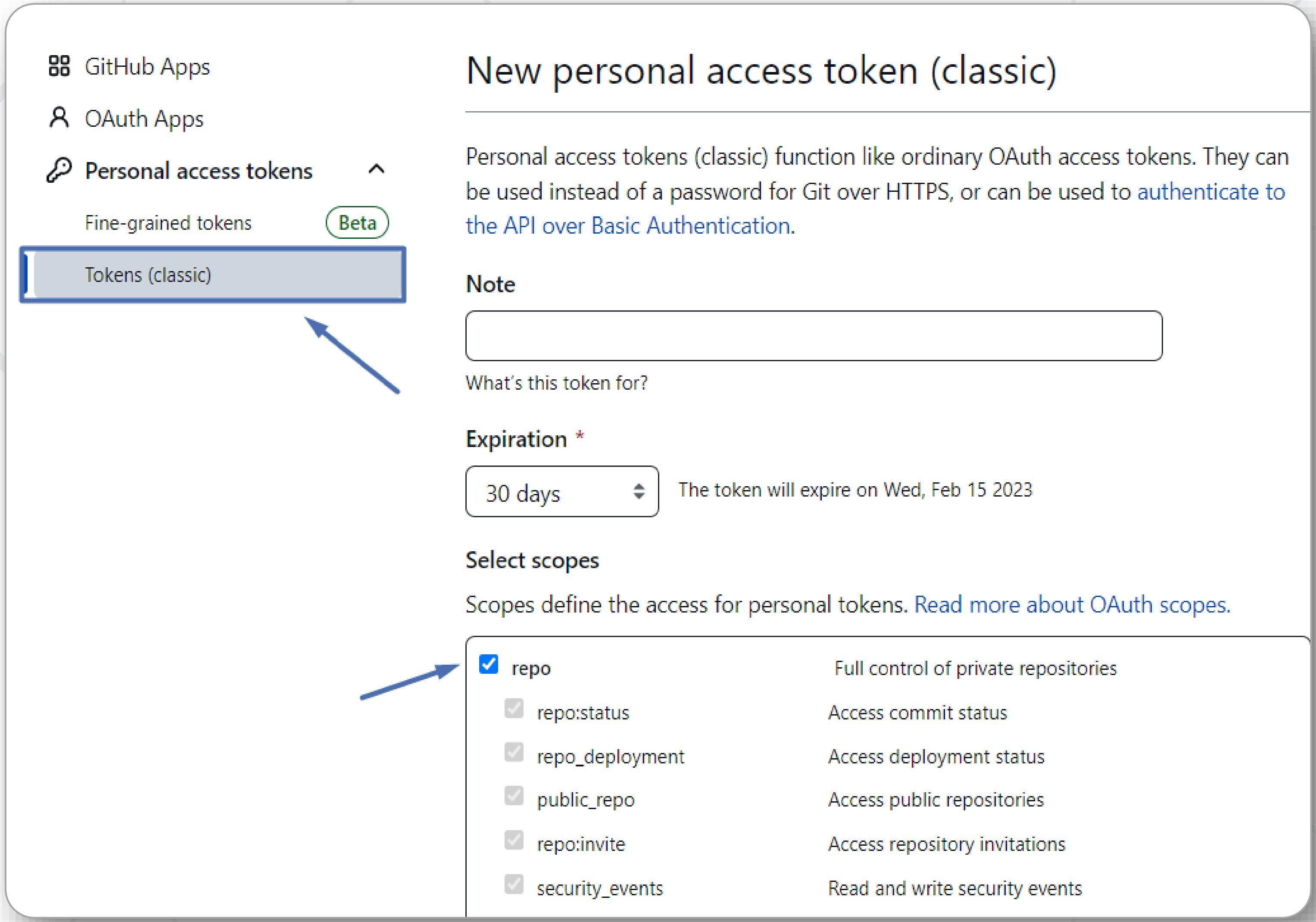
this is post test
#15 opened on Nov 14, 2016 by john6tain

Това го направих на живо в курса
bug duplicate enhancement help wanted invalid question
#14 opened on Nov 14, 2016 by testnakov

My new title
#13 opened on Nov 14, 2016 by VanGog06

GitHub Token

- GitHub API endpoints need **authentication**
- Create new **personal access token** for the GitHub API from your profile: https://github.com/settings/tokens/new



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

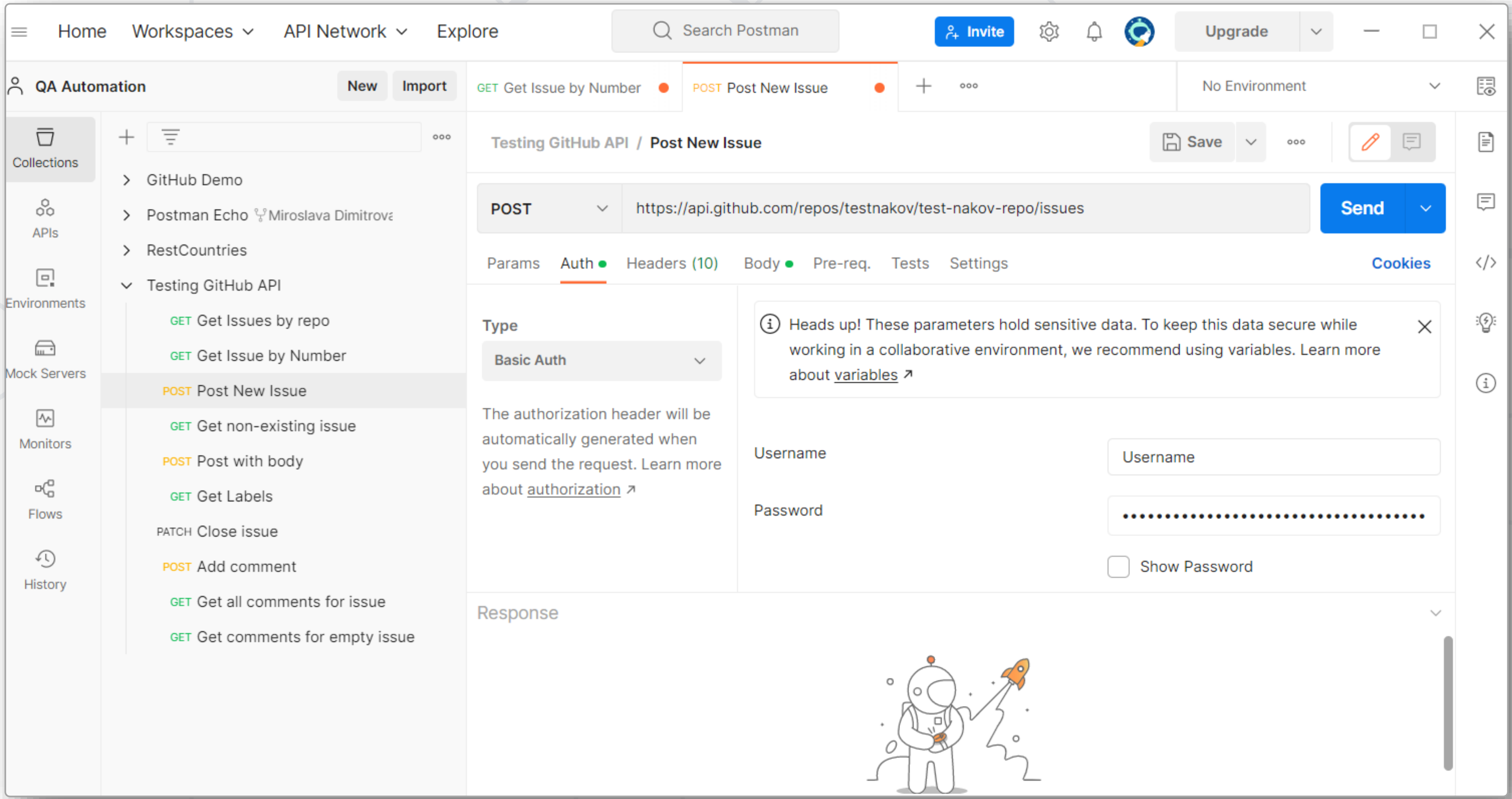
Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events

Authorization

- Add your GitHub Username and the Token



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

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 ↗

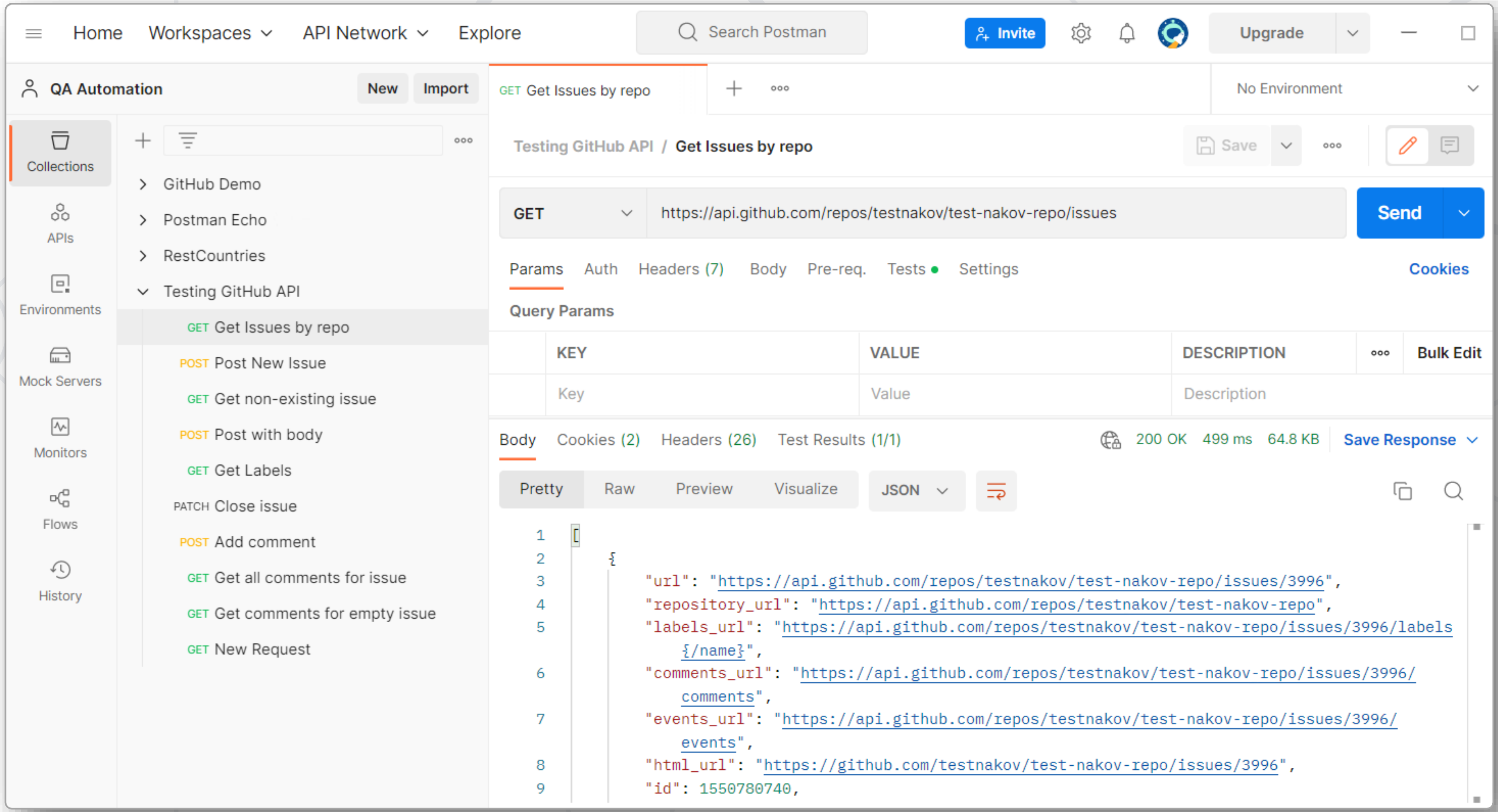
Username: Username

Password: ••

Show Password

Postman Examples: Get Issues from GitHub

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

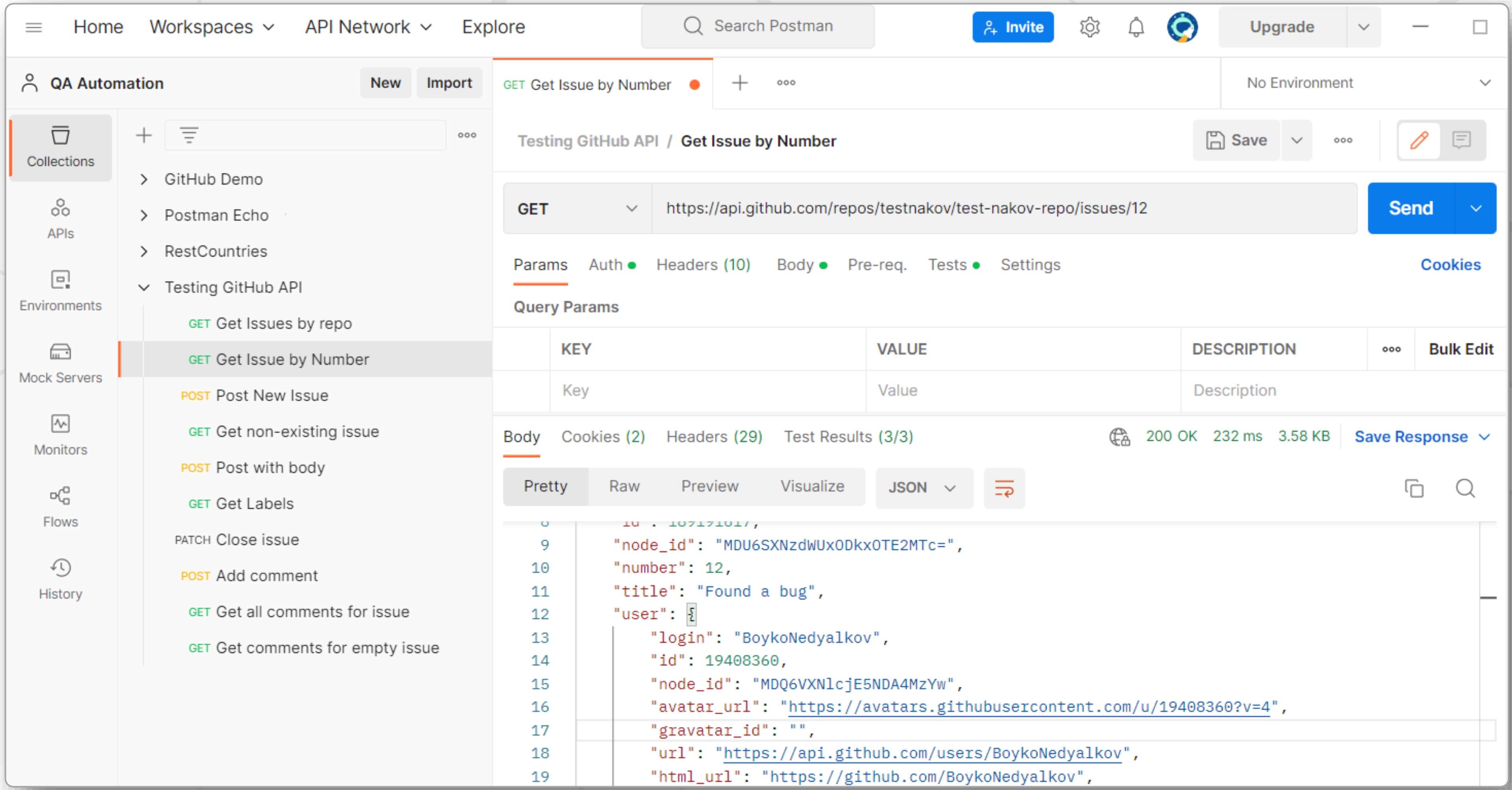


GET https://api.github.com/repos/testnakov/test-nakov-repo/issues

```
[
    {
        "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

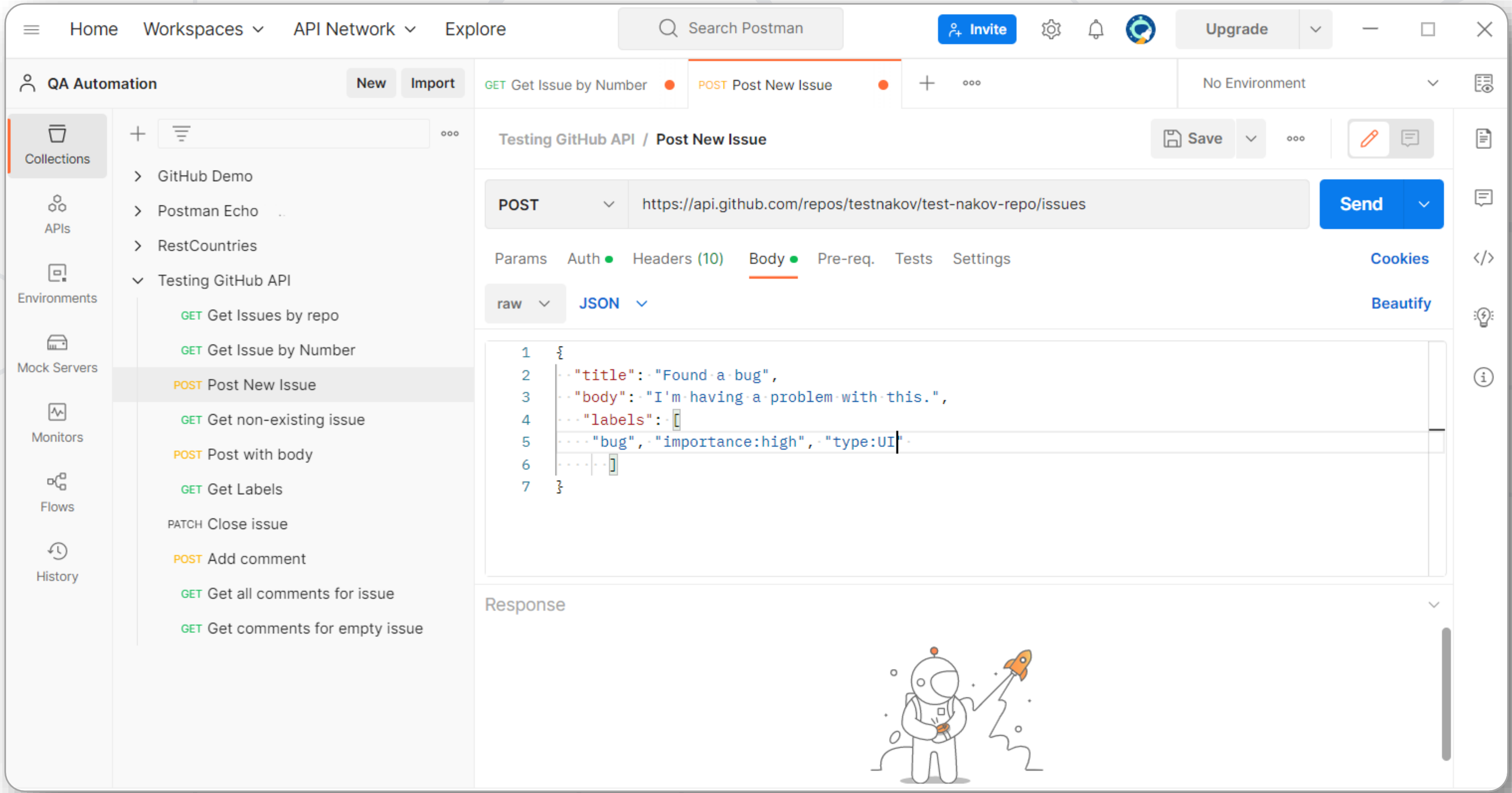


GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/12

```
    "node_id": "MDU6SXNzdWUxODkxOTE2MTc=",
    "number": 12,
    "title": "Found a bug",
    "user": {
        "login": "BoykoNedyalkov",
        "id": 19408360,
        "node_id": "MDQ6VXNlcjE5NDA4MzYw",
        "avatar_url": "https://avatars.githubusercontent.com/u/19408360?v=4",
        "gravatar_id": "",
        "url": "https://api.github.com/users/BoykoNedyalkov",
        "html_url": "https://github.com/BoykoNedyalkov",
```

Postman Examples: Create New Issue

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

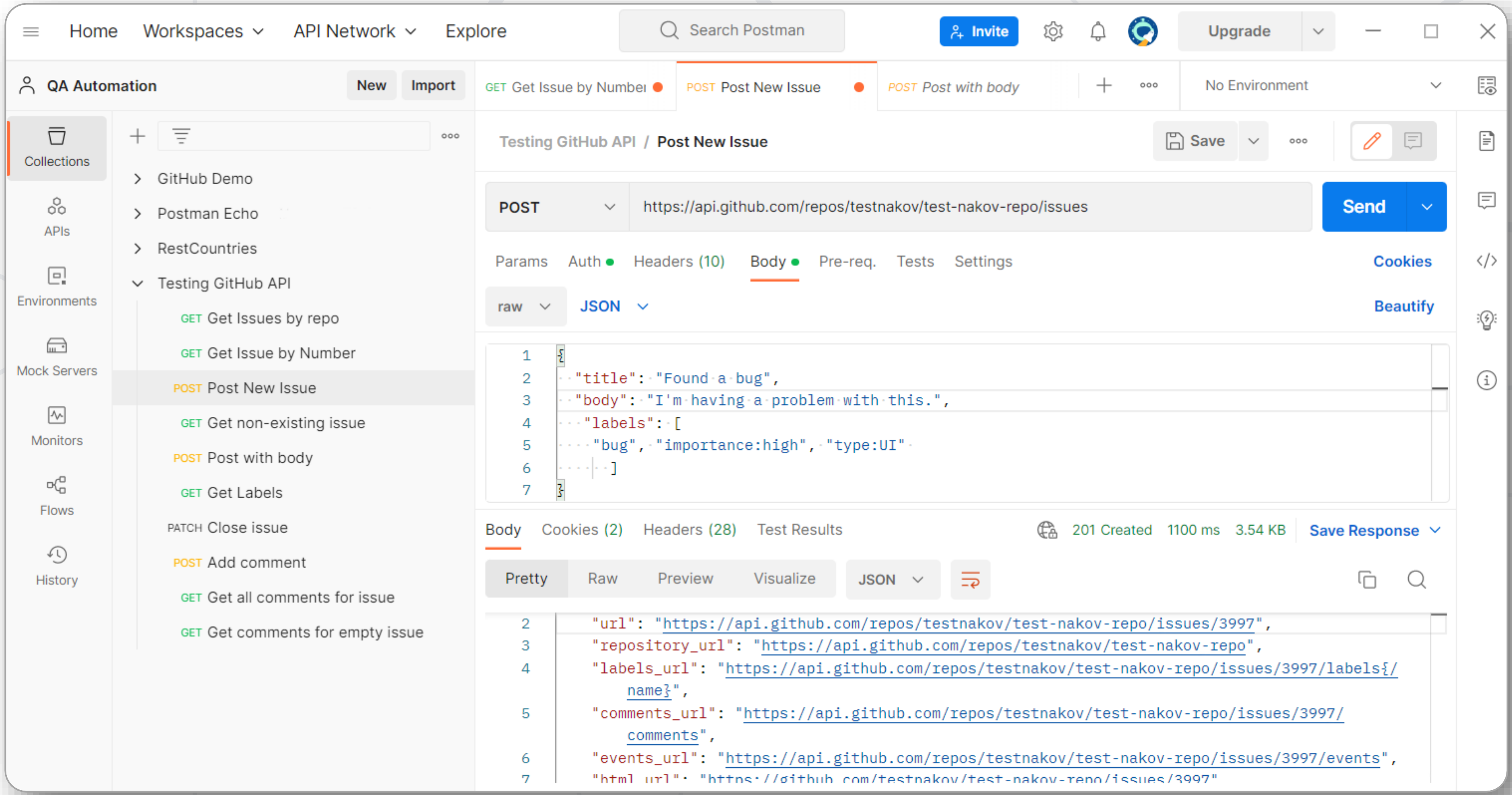


POST https://api.github.com/repos/testnakov/test-nakov-repo/issues

```
{
  "title": "Found a bug",
  "body": "I'm having a problem with this.",
   "labels": [
    "bug", "importance:high", "type:UI"
      ]
}
```

Postman Examples: Response

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



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

```
{
  "title": "Found a bug",
  "body": "I'm having a problem with this.",
   "labels": [
    "bug", "importance:high", "type:UI"
    ]
}
```

201 Created 1100 ms 3.54 KB

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

GitHub API

- 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
```

GitHub: Labels Issue

- 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

GitHub API

- 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)`

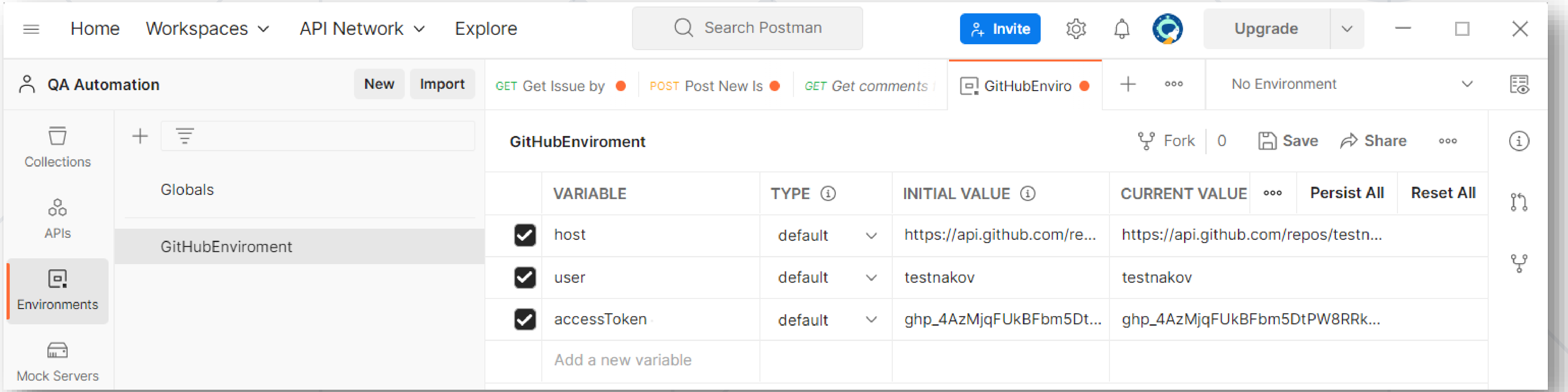
Body	`{"title":"Found a bug",` `"body": "I'm having a problem with this."}`

Variables

Parameterize requests

Variables in Postman

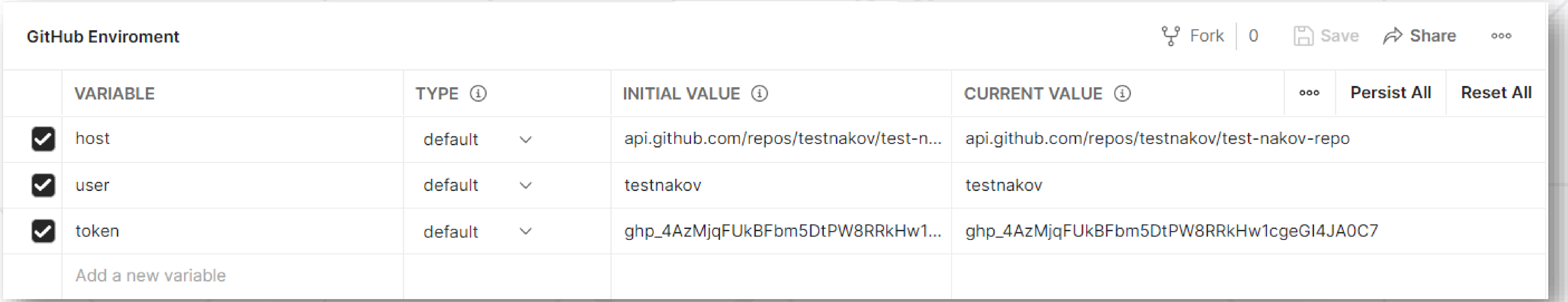
- Postman supports **environment variables**
- Can be used to parameterize the requests
- Can be edited by the test scripts



GitHubEnviroment

	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...
	Add a new variable			

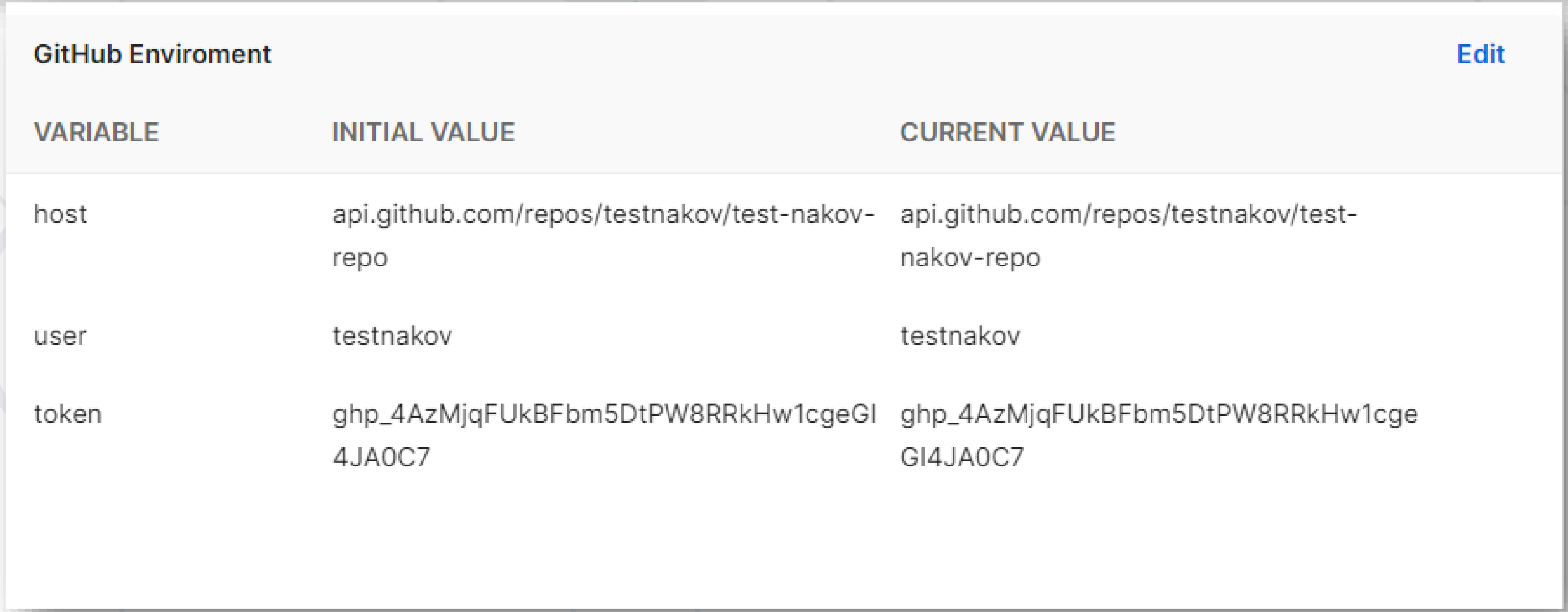
View / Edit Variables



GitHub Enviroment

Fork 0 Save Share ooo

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	ooo	Persist All	Reset All
✓	host	default	api.github.com/repos/testnakov/test-n...	api.github.com/repos/testnakov/test-nakov-repo			
✓	user	default	testnakov	testnakov			
✓	token	default	ghp_4AzMjqFUkBFbm5DtPW8RRkHw1...	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			
	Add a new variable						

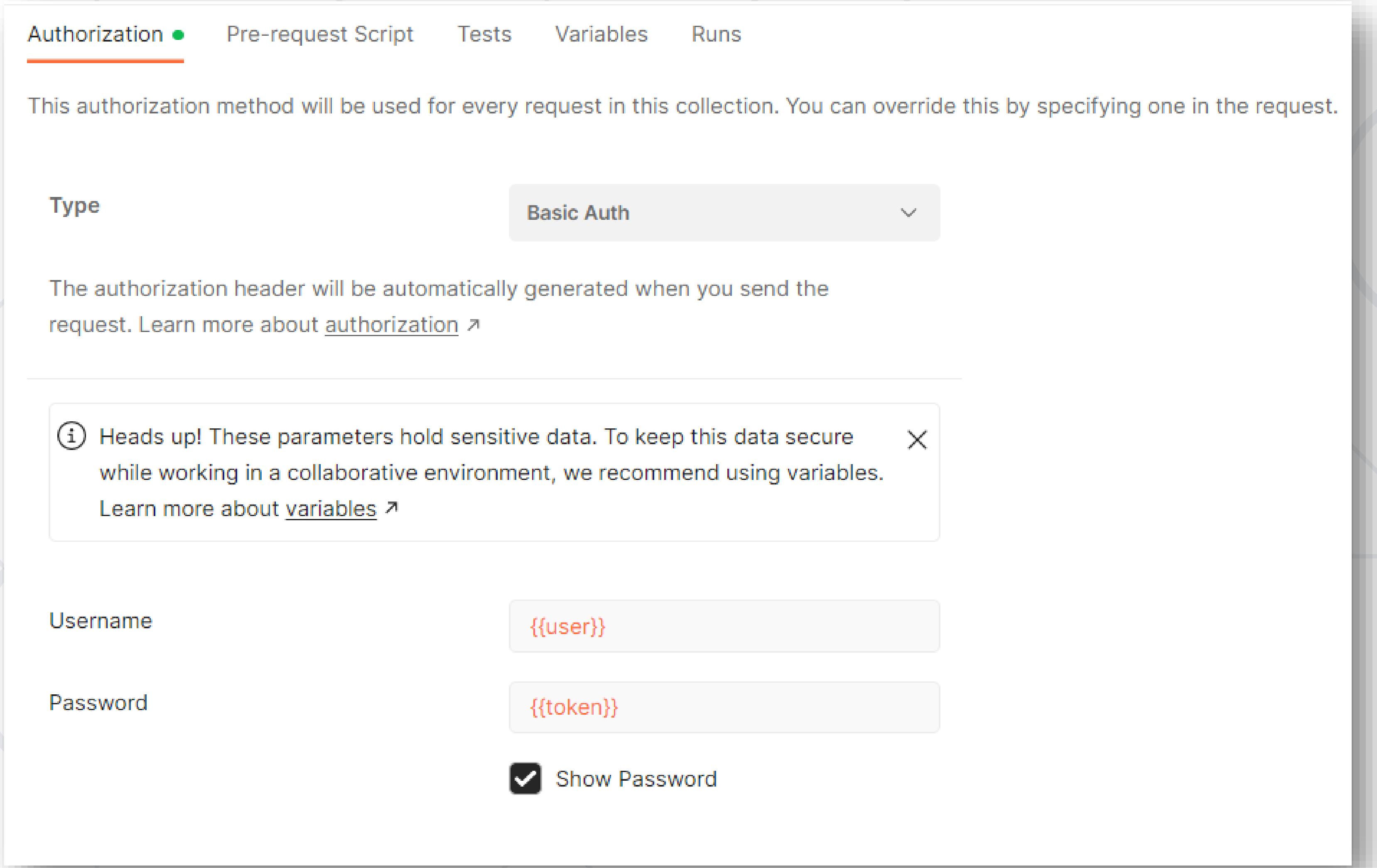


GitHub Enviroment

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	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	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

Using Variables for Authentication



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 ↗

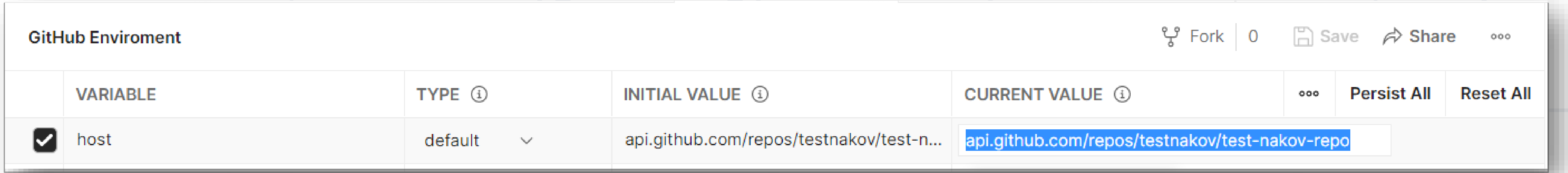
Username {{user}}

Password {{token}}

☑ Show Password

Using Variables in Requests

- Define an environment variable

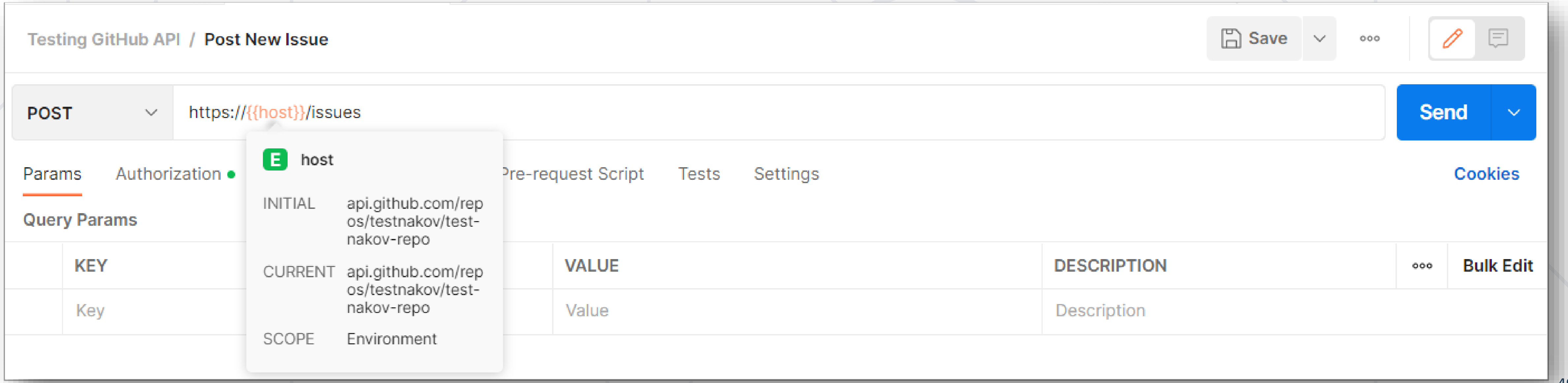


GitHub Enviroment

Fork 0 Save Share

	VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
✓	host	default	api.github.com/repos/testnakov/test-n...	api.github.com/repos/testnakov/test-nakov-repo

- Use the variable:

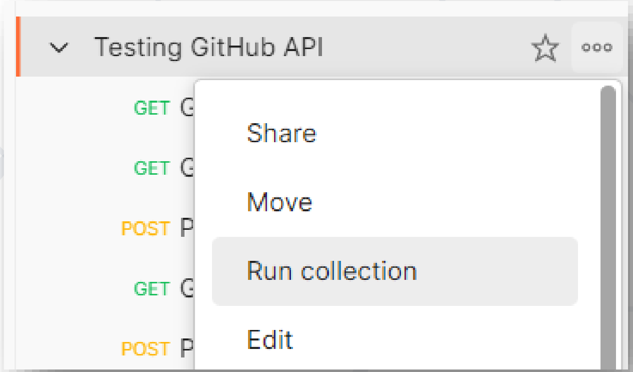


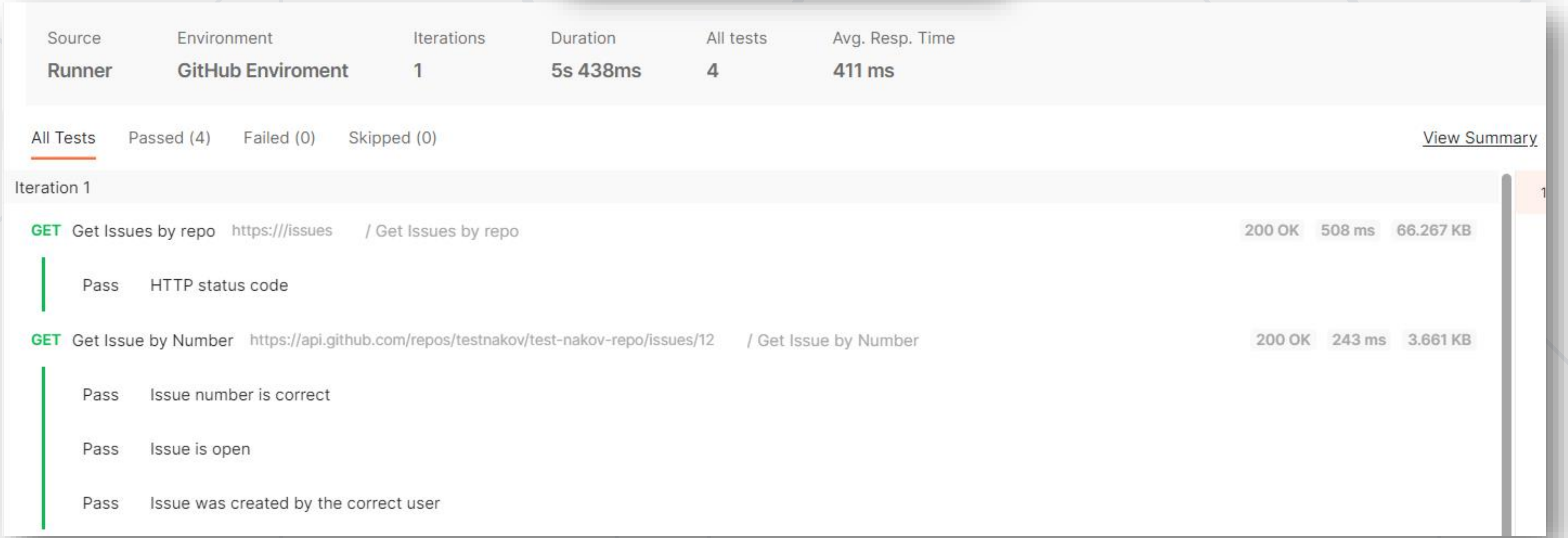
Testing GitHub API / Post New Issue

```
POST https://{{host}}/issues
```

E	host
INITIAL	api.github.com/repos/testnakov/test-nakov-repo
CURRENT	api.github.com/repos/testnakov/test-nakov-repo
SCOPE	Environment

Running Postman Collections and Tests





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

Summary

- **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?

Software University

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank

Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- 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

License