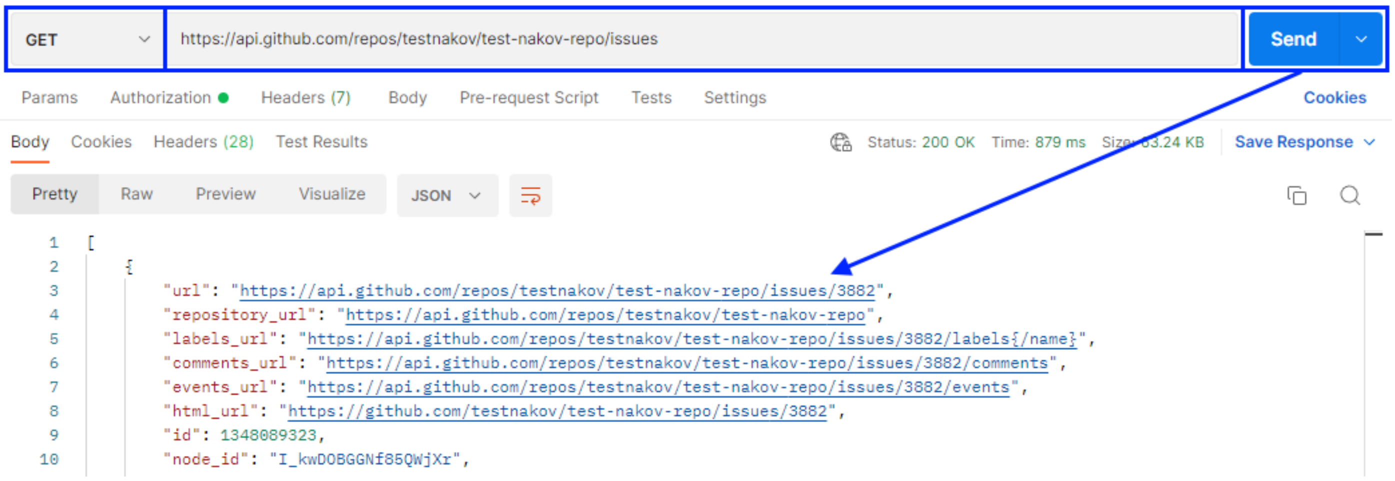
Exercise: HTTP Protocol

This document defines the exercises for the "ASP.NET Fundamentals" Course @ SoftUni.

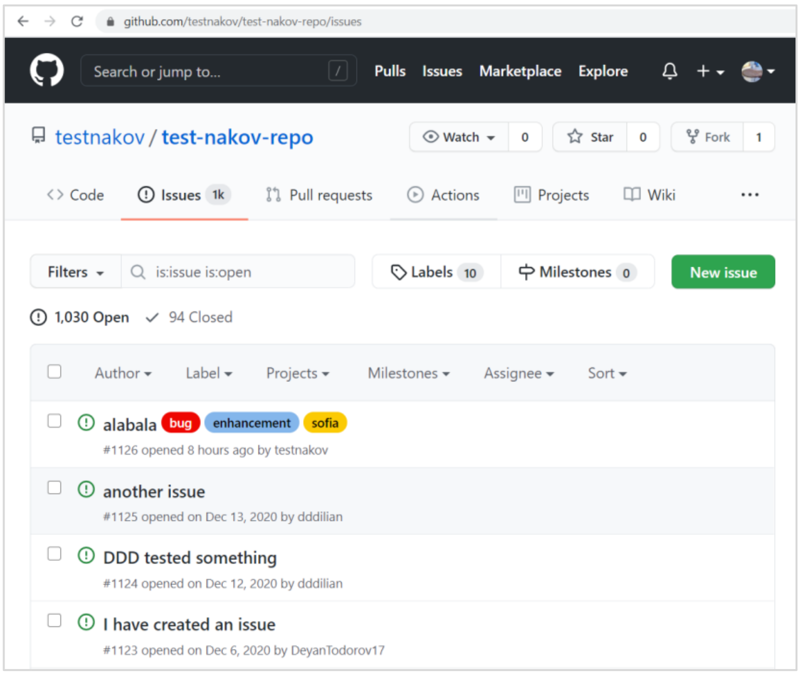
1. The Project for Testing: GitHub Issues API

In this exercise we shall **test the GitHub REST API**, more specifically, the **GitHub Issues API**:



```
[
    {
        "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882",
        "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
        "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/labels{/name}",
        "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/comments",
        "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/events",
        "html_url": "https://github.com/testnakov/test-nakov-repo/issues/3882",
        "id": 1348089323,
        "node_id": "I_kwDOBGGNf85QWjXr",
```

GitHub Issues is a popular issue tracking software, coming with all GitHub repositories. It is available for free, after a free registration in GitHub. This is how **GitHub Issues** user interface looks like:



alabala bug enhancement sofia
#1126 opened 8 hours ago by testnakov

another issue
#1125 opened on Dec 13, 2020 by dddilian

DDD tested something
#1124 opened on Dec 12, 2020 by dddilian

I have created an issue
#1123 opened on Dec 6, 2020 by DeyanTodorov17

The above user interface is publicly accessible from: https://github.com/testnakov/test-nakov-repo/issues.

2. API Endpoints for GitHub Issues

GitHub Issues provides the standard RESTful **API endpoints**, which you can access with Postman HTTP client from https://api.github.com:

- **GET endpoints** – respond with **JSON** object as result.
    - **`GET /repos/{user}/{repo}/issues`** – returns the **issues** in given GitHub repo.
    - **`GET /repos/{user}/{repo}/issues/{num}`** – returns the specified **issue**.

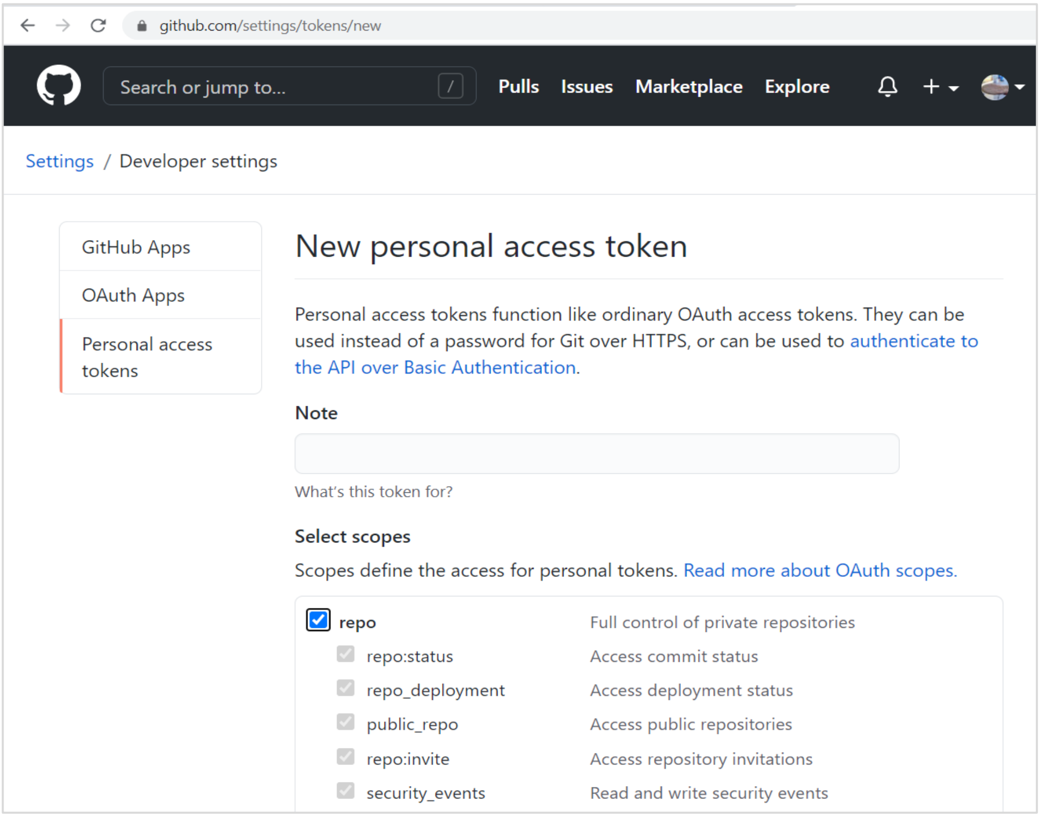
    - `GET /repos/{user}/{repo}/issues/{num}/comments` – returns the **comments** for an issue.
    - `GET /repos/{user}/{repo}/issues/comments/{id}` – returns the specified **comment**.
- **POST / PATCH / DELETE endpoints** – all of them need **authentication**.
    - `POST /repos/{user}/{repo}/issues` – creates a new **issue**.
    - `PATCH /repos/{user}/{repo}/issues/{num}` – modifies the specified **issue**.
    - `POST /repos/{user}/{repo}/issues/{num}/comments` – creates new **comments** for certain issue.
    - `PATCH /repos/{user}/{repo}/issues/comments/{id}` – modifies existing **comment**.
    - `DELETE /repos/{user}/{repo}/issues/comments/{id}` – deletes existing **comment**.

Note that in GitHub API some requests (mostly retrieval requests) identify the resources **by number**. The **issue number** is the sequential number (1, 2, 3, …) inside the project issue tracker. The issue **id** / comment **id** is global identifier (such as 762541045, 843104478).

Authentication in GitHub API

Some GitHub API endpoints need **authentication**. In Postman, you can use **Basic authentication**, using your **GitHub username** + a **password**, created from the "**Personal Access Tokens**" section in the GitHub user profile setting.

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



New personal access token

Personal access tokens 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

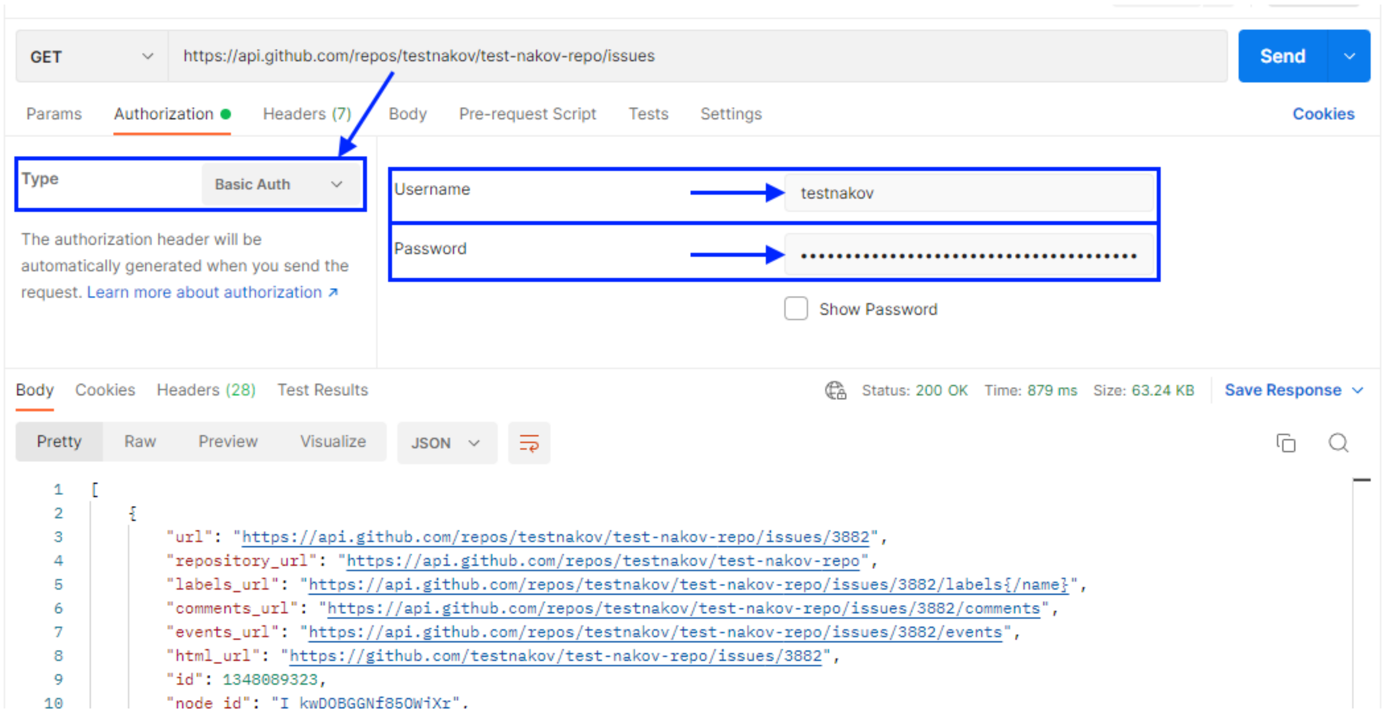
What's this token for?

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

Once, a **personal access token** is created in GitHub, you can use it from **Postman** by adding a "**Basic authentication**" header for the HTTP request. An example is shown below:



Type: Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization ↗

Username: testnakov

Password: ••

Show Password

Status: 200 OK Time: 879 ms Size: 63.24 KB

```
[
    {
        "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882",
        "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
        "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/labels{/name}",
        "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/comments",
        "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/events",
        "html_url": "https://github.com/testnakov/test-nakov-repo/issues/3882",
        "id": 1348089323,
        "node_id": "I_kwDOBGGNf85OWiXr",
```

In this exercise you shall use **HTTP Basic authentication** to authenticate and authorize your GitHub API requests. The **username** is your **GitHub username**. The **password** is your **personal access token**, that you have previously created from the [Developer settings] page in your GitHub profile.

GitHub API: Sample HTTP Request

This is how a typical **HTTP request to GitHub API** looks like:

```
POST /repos/testnakov/test-nakov-repo/issues/6/comments HTTP/1.1
Host: api.github.com
Content-Type: application/json
Authorization: Basic
dGVzdG5ha292OjMzYjQ3MzUzZTE2NGU4YTkxZDlmMDM2MGVjNDdkYmFmNWUzNzJhNg==
Content-Length: 25

{
  "body": "Comment"
}
```

In the above request the **username** and the **password** (the personal access token) used to authorize the request, are encoded in the **"Authorization" header**. This header holds a **base64 string**, which encodes together the **username** and the **password**, separated by '**:**'. This is the decoded base64 string from the above request:

Base64 string	dGVzdG5ha292OjMzYjQ3MzUzZTE2NGU4YTkxZDlmMDM2MGVjNDdkYmFmNWUzNzJhNg==
String value	testnakov:33b47353e164e8a91d9f0360ec47dbaf5e372a6

GitHub API: Sample HTTP Response

A typical **HTTP response** from the GitHub API may look like this:

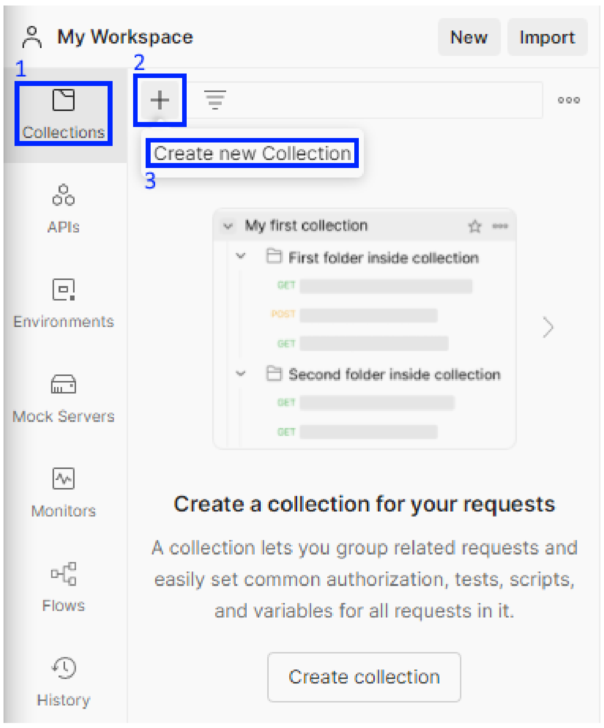
```
HTTP/1.1 201 Created
Date: Tue, 19 Jan 2021 13:20:12 GMT
Content-Type: application/json; charset=utf-8
```

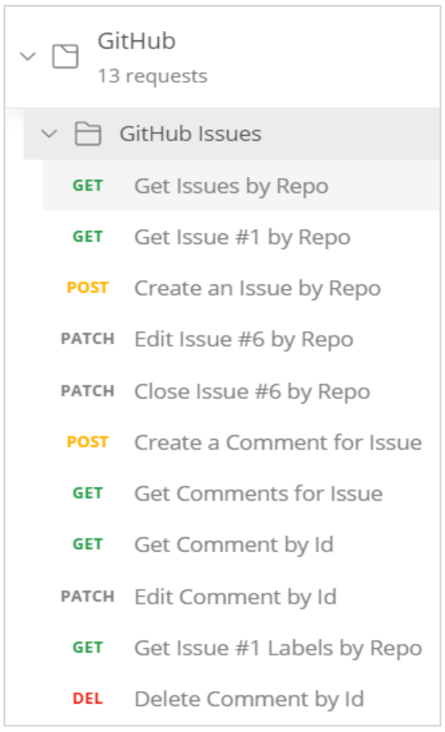
```
Content-Length: 1453
Server: GitHub.com
…

{"url":"https://api.github.com/repos/testnakov/test-nakov-
repo/issues/comments/762834681", "html_url":"https://github.com/testnakov/test-
nakov-repo/issues/6#issuecomment-762834681",
"issue_url":"https://api.github.com/repos/testnakov/test-nakov-repo/issues/6",
"id":762834681,"node_id":"MDEyOklzc3VlQ29tbWVudDc2MjgzNDY4MQ==","user":{"login":"tes
tnakov","id":23406465,"node_id":"MDQ6VXNlcjIzNDA2NDY1","avatar_url":"https://avatars
2.githubusercontent.com/u/23406465?u=b090ea0dc2d6c5cf71bcc39160cda63ab2f28714&v=4","
gravatar_id":"","url":"https://api.github.com/users/testnakov","html_url":"https://g
ithub.com/testnakov","followers_url":"https://api.github.com/users/testnakov/followe
rs","following_url":"https://api.github.com/users/testnakov/following{/other_user}",
"gists_url":"https://api.github.com/users/testnakov/gists{/gist_id}","starred_url":"
https://api.github.com/users/testnakov/starred{/owner}{/repo}","subscriptions_url":"
https://api.github.com/users/testnakov/subscriptions","organizations_url":"https://a
pi.github.com/users/testnakov/orgs","repos_url":"https://api.github.com/users/testna
kov/repos","events_url":"https://api.github.com/users/testnakov/events{/privacy}","r
eceived_events_url":"https://api.github.com/users/testnakov/received_events","type":
"User","site_admin":false},"created_at":"2021-01-19T13:20:11Z","updated_at":"2021-
01-19T13:20:11Z","author_association":"OWNER","body":"This is a
comment","performed_via_github_app":null}
```

3. Create Postman Collection of Requests

Now, you should create a **Postman collection** of HTTP requests for accessing the **GitHub Issues API**. Select **Collections** in the sidebar, then select the **[+]** button and click on **Create new Collection** (see the screenshot).

The new **Postman collection** will hold the **HTTP requests for the GitHub API**, related to **issues** and issue **comments**. The Postman collection may look as shown below. It may be structured in **folders** and the requests should have appropriate names, like it is shown below:


My Workspace
New
Import
1
2
Collections
Create new Collection
3
APIs
Environments
Mock Servers
Monitors
Flows
History
My first collection
First folder inside collection
Second folder inside collection
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create collection



GitHub
13 requests

GitHub Issues

GET Get Issues by Repo
GET Get Issue #1 by Repo
POST Create an Issue by Repo
PATCH Edit Issue #6 by Repo
PATCH Close Issue #6 by Repo
POST Create a Comment for Issue
GET Get Comments for Issue
GET Get Comment by Id
PATCH Edit Comment by Id
GET Get Issue #1 Labels by Repo
DEL Delete Comment by Id

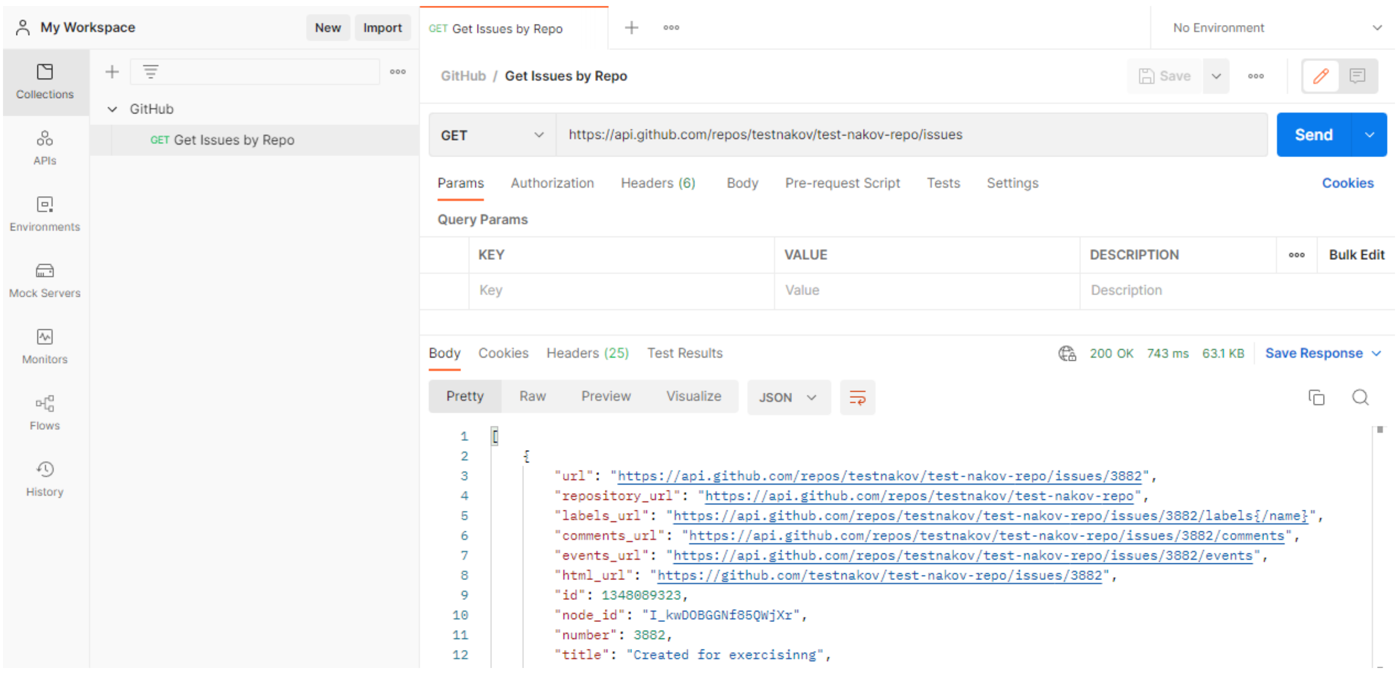
Now it's time to **create a few HTTP requests** in the new Postman collection.

Retrieve All Issues from Repo

Retrieve all issues from repo "**`test-nakov-repo`**" in user "**`testnakov`**". Use the following **HTTP GET request** in Postman:

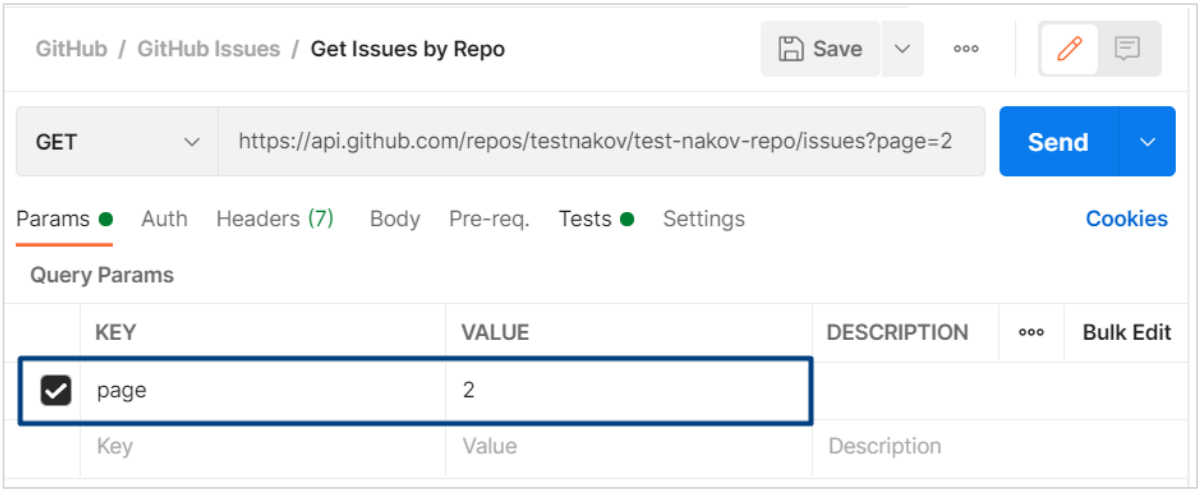
Request	`GET https://api.github.com/repos/testnakov/test-nakov-repo/issues`
Body	*`(empty)`*

This is how the above HTTP request may look in **Postman** after successful execution:



```
[
    {
        "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882",
        "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
        "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/labels{/name}",
        "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/comments",
        "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/3882/events",
        "html_url": "https://github.com/testnakov/test-nakov-repo/issues/3882",
        "id": 1348089323,
        "node_id": "I_kwDOBGGNf85QWjXr",
        "number": 3882,
        "title": "Created for exercisinng",
```

The returned HTTP status code is "**`200 OK`**" and the HTTP **response body** holds the returned issues as **JSON array** of objects. Note that the issues in this repo could be thousands and returning all of them will be too slow and the response will be huge. To optimize the speed, the GitHub API uses **paging**. By default, the above request will return the most recent 30 issues. You can request the others by using a **request parameter "page"**:


GitHub / GitHub Issues / Get Issues by Repo
Save
GET
https://api.github.com/repos/testnakov/test-nakov-repo/issues?page=2
Send
Params
Auth
Headers (7)
Body
Pre-req.
Tests
Settings
Cookies
Query Params
Bulk Edit
Key
Value
Description

	KEY	VALUE	DESCRIPTION
✓	page	2	

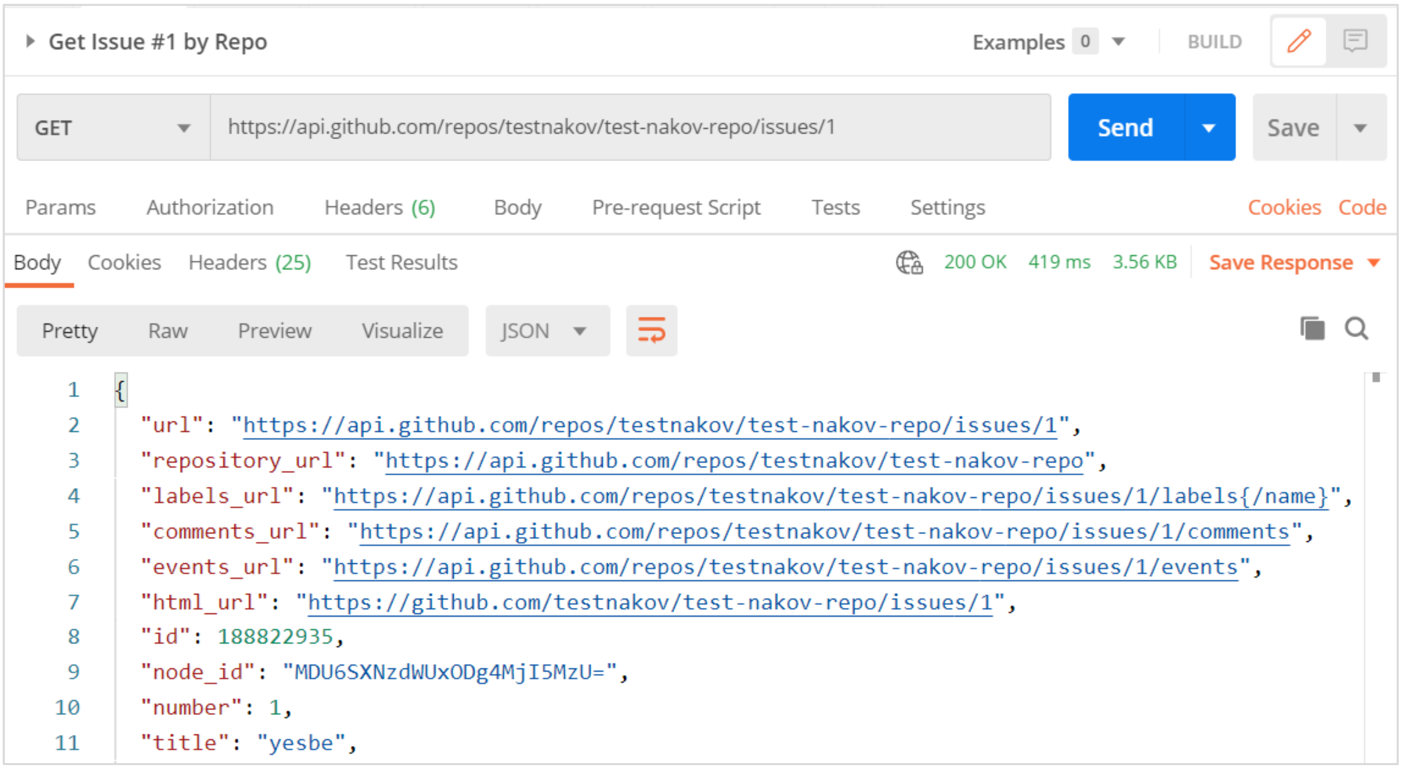
Retrieve Issue by Number

Retrieve issue #1 from repo "**`test-nakov-repo`**" in user "**`testnakov`**":

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

Body	*(empty)*

The returned HTTP status code is "**200 OK**" and the HTTP **response body** holds the requested issue as **JSON** object:

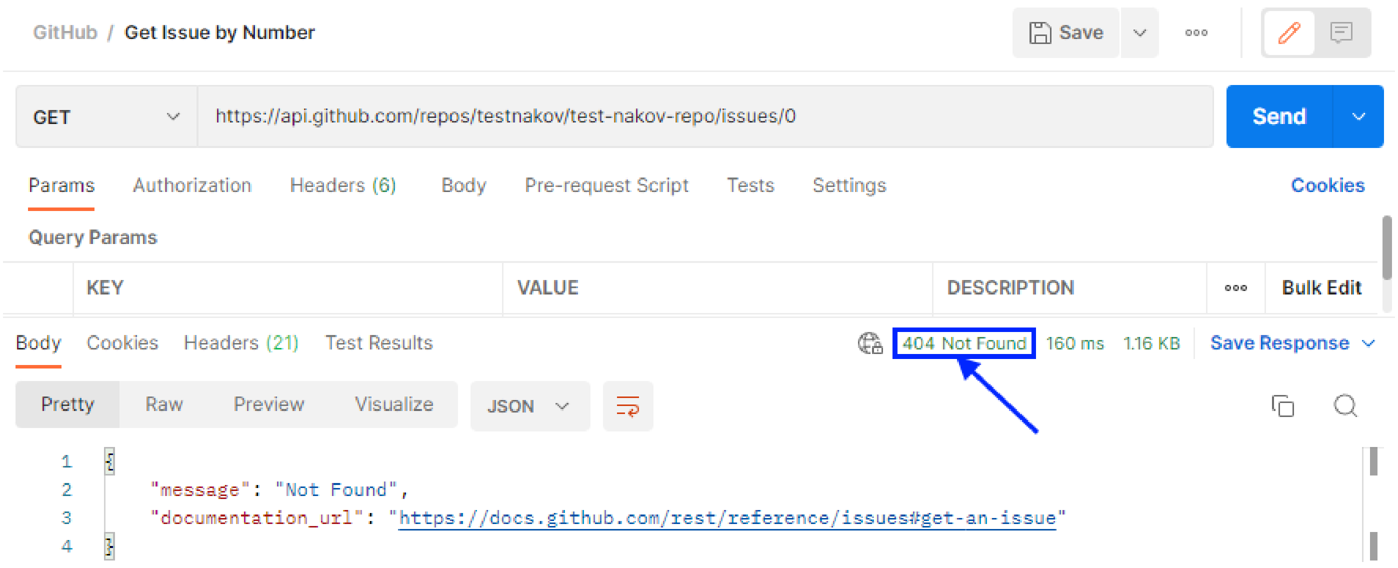


```
{
  "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1",
  "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
  "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/labels{/name}",
  "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/comments",
  "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/events",
  "html_url": "https://github.com/testnakov/test-nakov-repo/issues/1",
  "id": 188822935,
  "node_id": "MDU6SXNzdWUxODg4MjI5MzU=",
  "number": 1,
  "title": "yesbe",
```

Note that "**issue number**" (in this example **1**) and "**issue id**" (in this example **188822935**) are different identifiers of the same issue. The issue **`id`** is a global unique issue identifier in the GitHub issues database. The issue **`number`** is a local identifier within the current project's issue tracker.

You can also view the same **issue #1** from the Web in the GitHub Issues page for the above-mentioned project: https://github.com/testnakov/test-nakov-repo/issues/1.

If you try to retrieve a **non-existing issue** from the GitHub API (for example **issue #0**), you will get **404 Not Found**:



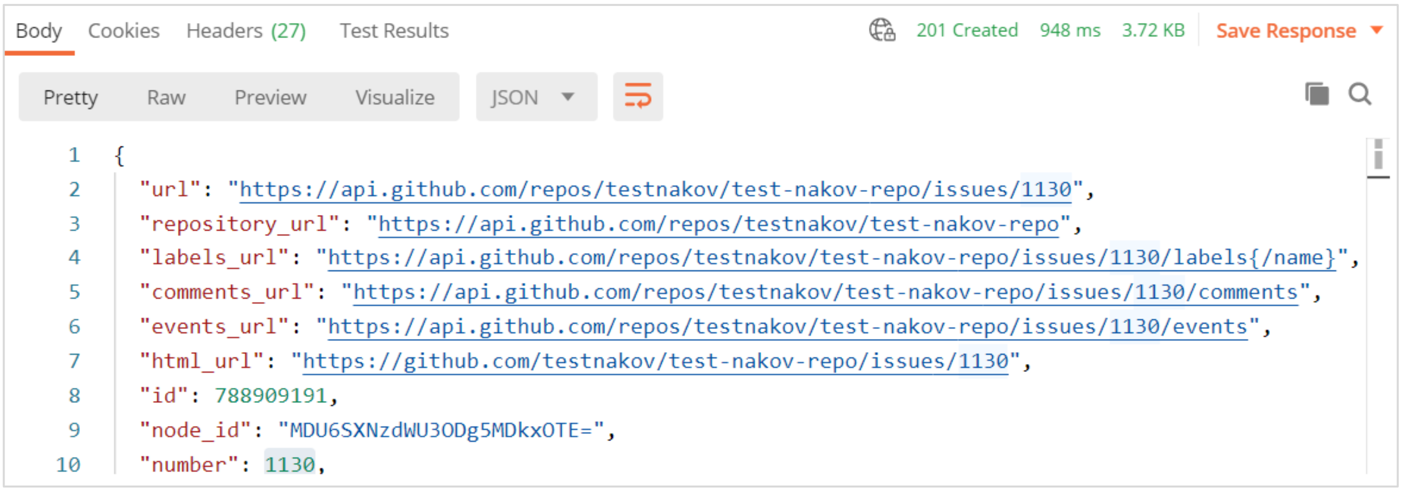
```
{
    "message": "Not Found",
    "documentation_url": "https://docs.github.com/rest/reference/issues#get-an-issue"
}
```

Create a New Issue

Create a new issue in the repo "**`test-nakov-repo`**" of user "**`testnakov`**". You will need a valid GitHub **access token** to authorize the request:

Request	`POST https://api.github.com/repos/testnakov/test-nakov-repo/issues`
Authorization	`Basic` (GitHub username + GitHub personal access token)
Body	`{ "title": "Missing [Submit] button", "body": "I'm having a problem with this." }`

In case of success, the HTTP response should have status **`201 Created`** and should hold in the response body a **JSON** object, holding **the number of the new issue**, together with other issue details:



```
{
  "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1130",
  "repository_url": "https://api.github.com/repos/testnakov/test-nakov-repo",
  "labels_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1130/labels{/name}",
  "comments_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1130/comments",
  "events_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/1130/events",
  "html_url": "https://github.com/testnakov/test-nakov-repo/issues/1130",
  "id": 788909191,
  "node_id": "MDU6SXNzdWU3ODg5MDkxOTE=",
  "number": 1130,
```

The issue number for the above new issue is **#1130**. Note that "**issue id**" and "**issue number**" are different things. The **issue number** is unique for certain GitHub repository. The **issue id** is globally unique at GitHub.

You can view this new issue in the GitHub Issues: https://github.com/testnakov/test-nakov-repo/issues/1130.

In case you don't provide valid authentication for the GitHub API for the HTTP request, you will get an error response: **`401 Unauthorized`** or **`404 Not Found`**.

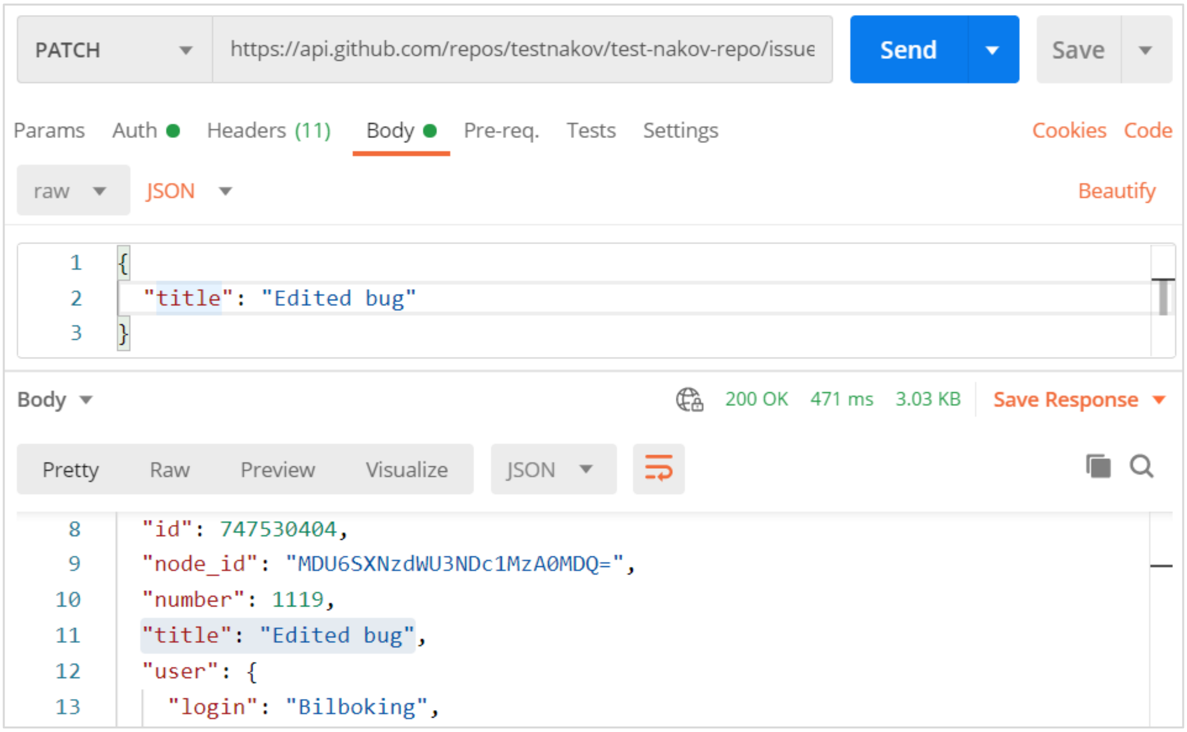
Sometimes GitHub returns **`404 Not Found`** when you request a resource without proper authentication instead of the correct HTTP status code. This is semantically incorrect, but GitHub returns this to avoid "information disclosure". Be warned that if you get **`404 Not Found`** from GitHub, this may mean "**`Unauthorized access`**".

Edit Existing Issue

Edit existing issue #1119 from the repo "**`test-nakov-repo`**" of user "**`testnakov`**":

Request	`PATCH https://api.github.com/repos/testnakov/test-nakov-repo/issues/1119`
Authorization	`Basic` (GitHub username + GitHub personal access token)
Body	`{ "title": "Edited bug (new title)" }`

On success, the HTTP response should have status **`200 OK`** and should hold **the edited issue** as response body:


PATCH https://api.github.com/repos/testnakov/test-nakov-repo/issue

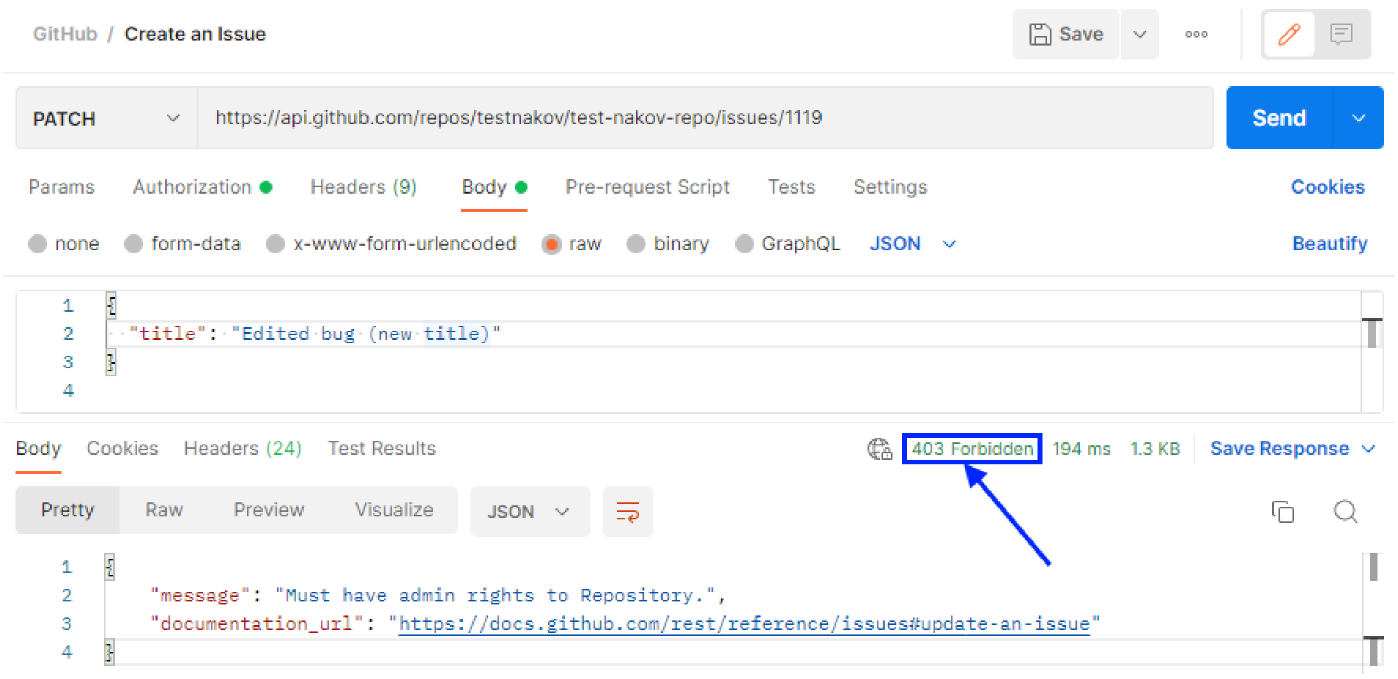
Body: raw JSON

```
{
  "title": "Edited bug"
}
```

200 OK 471 ms 3.03 KB

```
"id": 747530404,
"node_id": "MDU6SXNzdWU3NDc1MzA0MDQ=",
"number": 1119,
"title": "Edited bug",
"user": {
  "login": "Bilboking",
```

Note: you can edit only your own issues. Repo admins can edit also other user's issues. If you try to edit an issue without sufficient privileges, you will get **`403 Forbidden`**:


GitHub / Create an Issue

PATCH https://api.github.com/repos/testnakov/test-nakov-repo/issues/1119

Body: raw JSON

```
{
  "title": "Edited bug (new title)"
}
```

403 Forbidden 194 ms 1.3 KB

```
{
    "message": "Must have admin rights to Repository.",
    "documentation_url": "https://docs.github.com/rest/reference/issues#update-an-issue"
}
```

Close Existing Issue

Close issue #6 from the repo "**`test-nakov-repo`**" of user "**`testnakov`**":

Request	`PATCH https://api.github.com/repos/testnakov/test-nakov-repo/issues/6`
Authorization	`Basic` (GitHub username + GitHub personal access token)
Body	`{ "state": "closed" }`

The HTTP response should have status **200 OK** and should hold **the edited issue** as response body. You can see the closed issue here: https://github.com/testnakov/test-nakov-repo/issues/6.

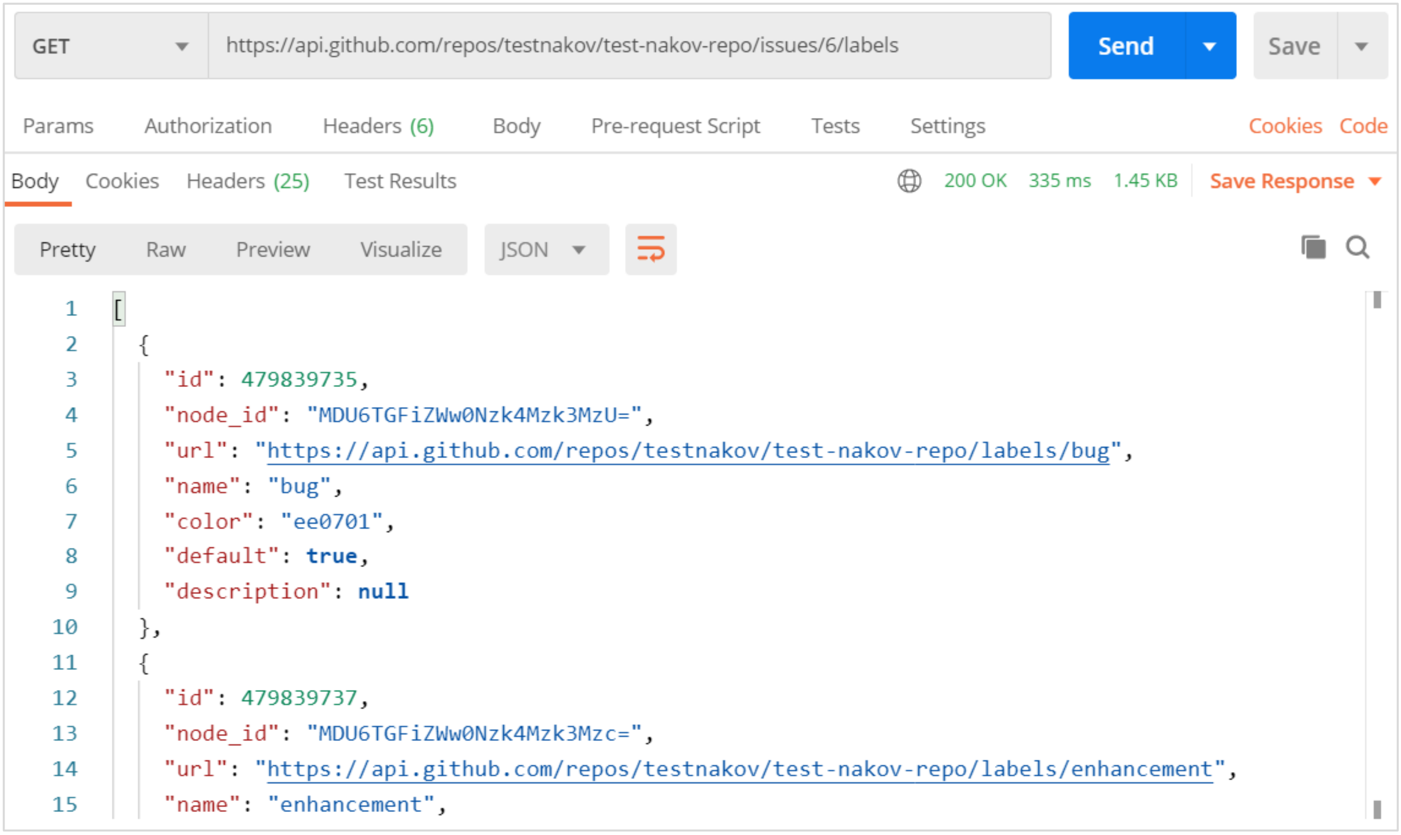
Note: you can edit / close only your own issues. Repo admins can edit or close user's issues. If you try to close an issue without sufficient privileges, you will get **403 Forbidden**.

Retrieve All Labels for Issue

Retrieve all comments for existing **issue #6** from the repo "**`test-nakov-repo`**" of user "**`testnakov`**":

Request	`GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/6/labels`
Body	*(empty)*

The HTTP response should have status **200 OK** and should hold **the issue labels** as response body in JSON format (array of labels):



```
[
  {
    "id": 479839735,
    "node_id": "MDU6TGFiZWw0Nzk4Mzk3MzU=",
    "url": "https://api.github.com/repos/testnakov/test-nakov-repo/labels/bug",
    "name": "bug",
    "color": "ee0701",
    "default": true,
    "description": null
  },
  {
    "id": 479839737,
    "node_id": "MDU6TGFiZWw0Nzk4Mzk3Mzc=",
    "url": "https://api.github.com/repos/testnakov/test-nakov-repo/labels/enhancement",
    "name": "enhancement",
```

You can see these labels here: https://github.com/testnakov/test-nakov-repo/issues/6. All available labels for this repos can be seen here: https://github.com/testnakov/test-nakov-repo/labels.

In case of **no labels** available for the specified issue, the HTTP response body will hold and **empty JSON array**: **`[]`**.

Create a Comment for Issue

Create a new comment for existing **issue #6** from the repo "**`test-nakov-repo`**" of user "**`testnakov`**":

Request	`POST https://api.github.com/repos/testnakov/test-nakov-repo/issues/6/comments`
Authorization	`Basic` (GitHub username + GitHub personal access token)
Body	`{`

	`"body": "This is a comment"`
	`}`

The HTTP response should have status **`201 Created`** and should hold **the new comment** as response body in JSON format. You can see the new comment here: https://github.com/testnakov/test-nakov-repo/issues/6.

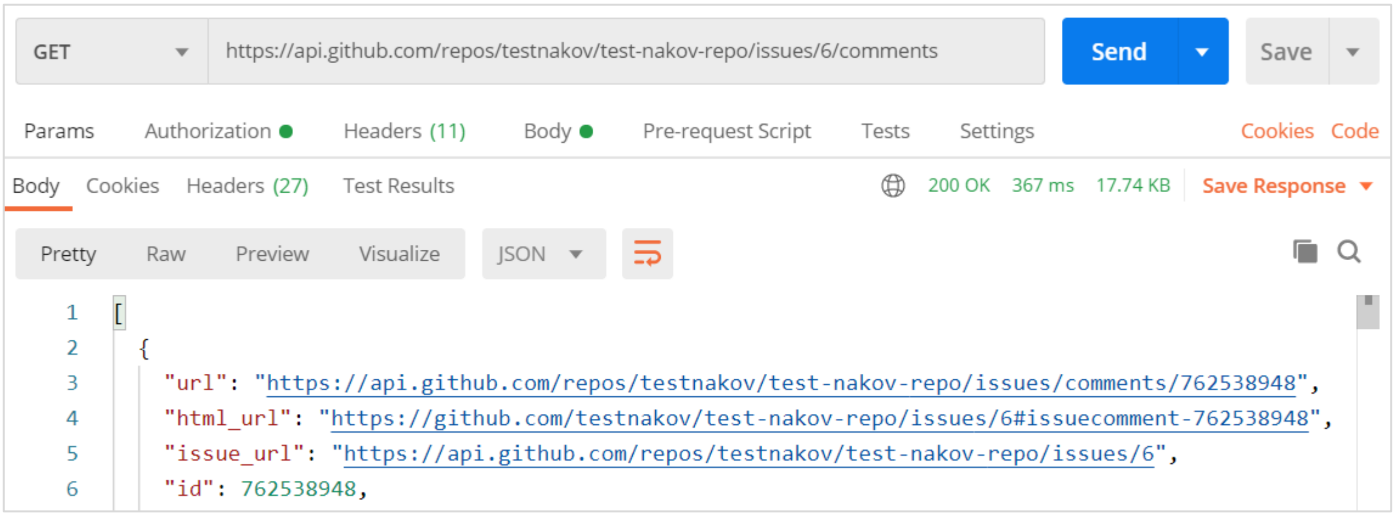
If you try to create a comment in a repo, where you don't have sufficient privileges, you will get **`403 Forbidden`**.

Retrieve All Comments for Issue

Retrieve all comments for existing **issue #6** from the repo "**`test-nakov-repo`**" of user "**`testnakov`**":

Request	`GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/6/comments`
Body	*(empty)*

The HTTP response should have status **`200` OK** and should hold **the issue comments** as response body in JSON format (array of comments):

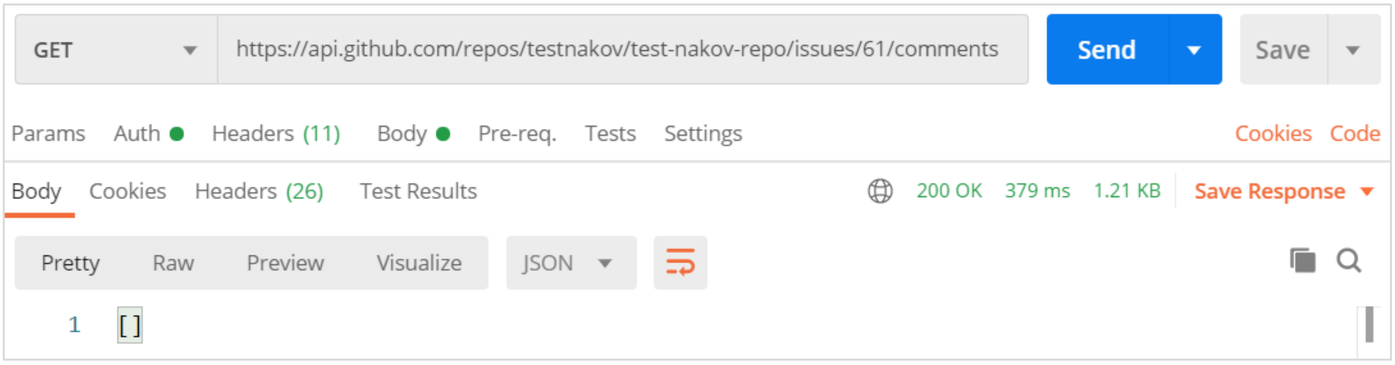

GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/6/comments

200 OK 367 ms 17.74 KB

```
[
  {
    "url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/comments/762538948",
    "html_url": "https://github.com/testnakov/test-nakov-repo/issues/6#issuecomment-762538948",
    "issue_url": "https://api.github.com/repos/testnakov/test-nakov-repo/issues/6",
    "id": 762538948,
```

You can see these comments here: https://github.com/testnakov/test-nakov-repo/issues/6.

In case of **no comments** available for the specified issue, the HTTP response body will hold an **empty JSON array**:


GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/61/comments

200 OK 379 ms 1.21 KB

```
[]
```

Retrieve Comment by Id

Retrieve a comment by id. The comment **`id`** is global for the entire GitHub (in this example **#762538948**), but still, the **user** and **repo** for the comment are required in the request URL:

Request	`GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/comments/762538948`

Body	*(empty)*

The HTTP response should have status **200 OK** and should hold **the new issue comment** in JSON format. You can see this comment here: https://github.com/testnakov/test-nakov-repo/issues/6#issuecomment-762538948.

Edit Existing Comment

Edit existing comment by id. The comment **id** is global for the entire GitHub (in this example **#762541976**), but still, the **user** and **repo** for the comment are required in the request URL:

Request	`PATCH https://api.github.com/repos/testnakov/test-nakov-repo/issues/comments/762541976`
Body	`{ "body": "Edited Comment" }`

The HTTP response should have status **200 OK** and should hold **the modified issue comment** in JSON format. You can see this comment here: https://github.com/testnakov/test-nakov-repo/issues/6#issuecomment-762541976.

Note: you can edit only your own comments. Repo admins can edit also other user's comments. If you try to edit a comment without sufficient privileges, you will get **401 Unauthorized** or **403 Forbidden**.

Delete Existing Comment

Delete existing comment by id. The comment **id** is global for the entire GitHub. First **create a new comment** and put its **id** in the request below:

Request	`DELETE https://api.github.com/repos/testnakov/test-nakov-repo/issues/comments/{id}`
Body	*(empty)*

The HTTP response should have status **204 No Content** and should hold empty body.

In case of non-existing comment, the above request will return **404 Not Found**.

Note: you can delete only your own comments. Repo admins can delete also other user's comments. If you try to delete a comment without sufficient privileges, you will get **401 Unauthorized** or **403 Forbidden**.