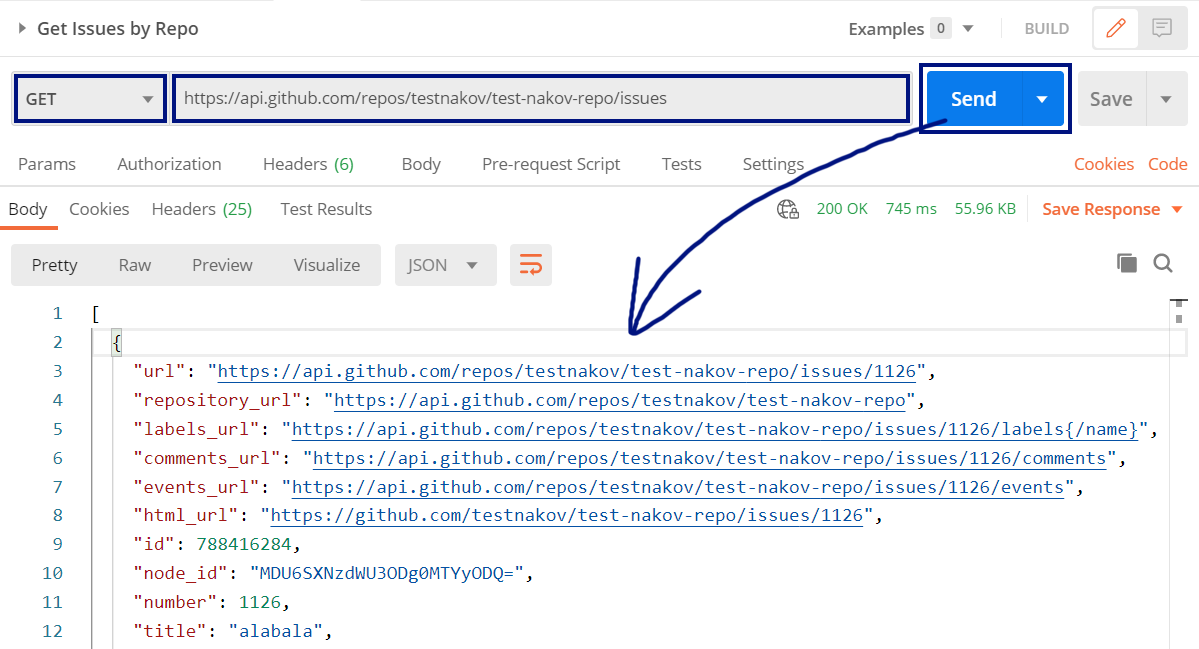
# Exercise: API Testing

This document defines the exercises and homework assignments for the ["QA Automation" Course @ SoftUni](https://softuni.bg/trainings/2550/qa-automation-may-2020).

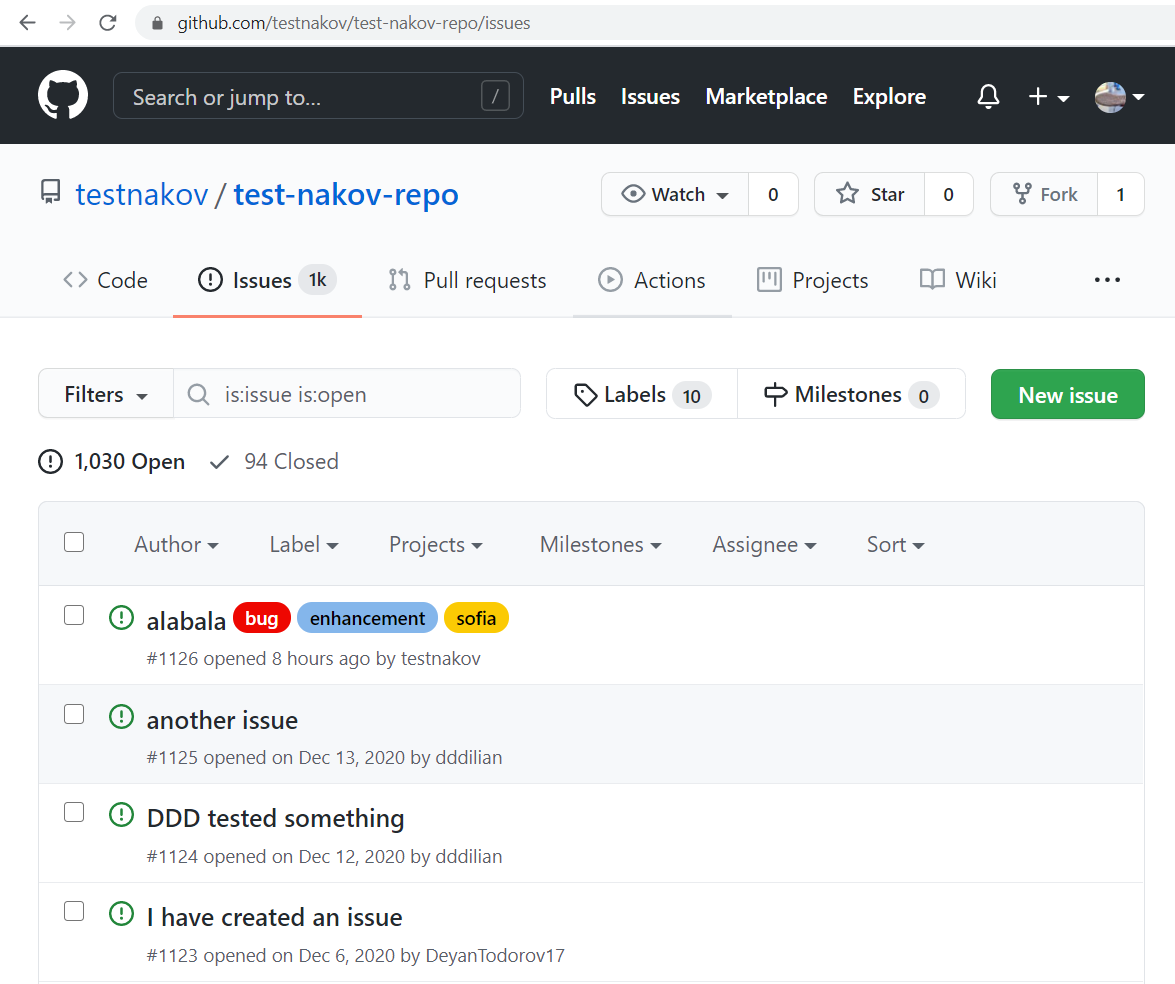
Please submit the homework a single zip / rar / 7z archive holding the source code and any other project assets.

## The Project for Testing: GitHub Issues API

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



**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:



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

## API Endpoints for GitHub Issues

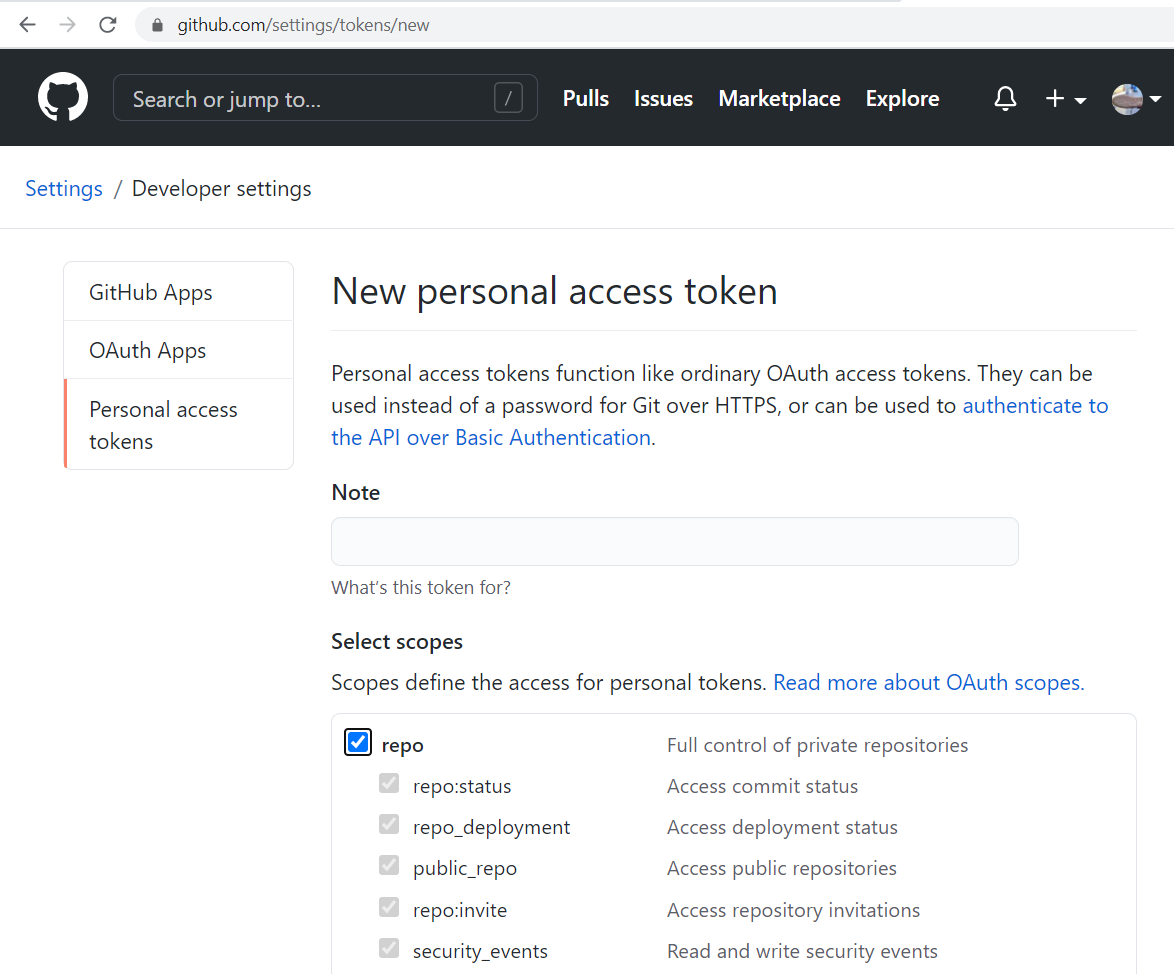
GitHub Issues provides the standard RESTful **API endpoints**, which you can access with Postman HTTP client from <https://api.github.com>:

* **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 specified **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/{id}** –modifies the specified **issue**.
  + **POST /repos/{user}/{repo}/issues/{num}/comments** –creates a 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**.

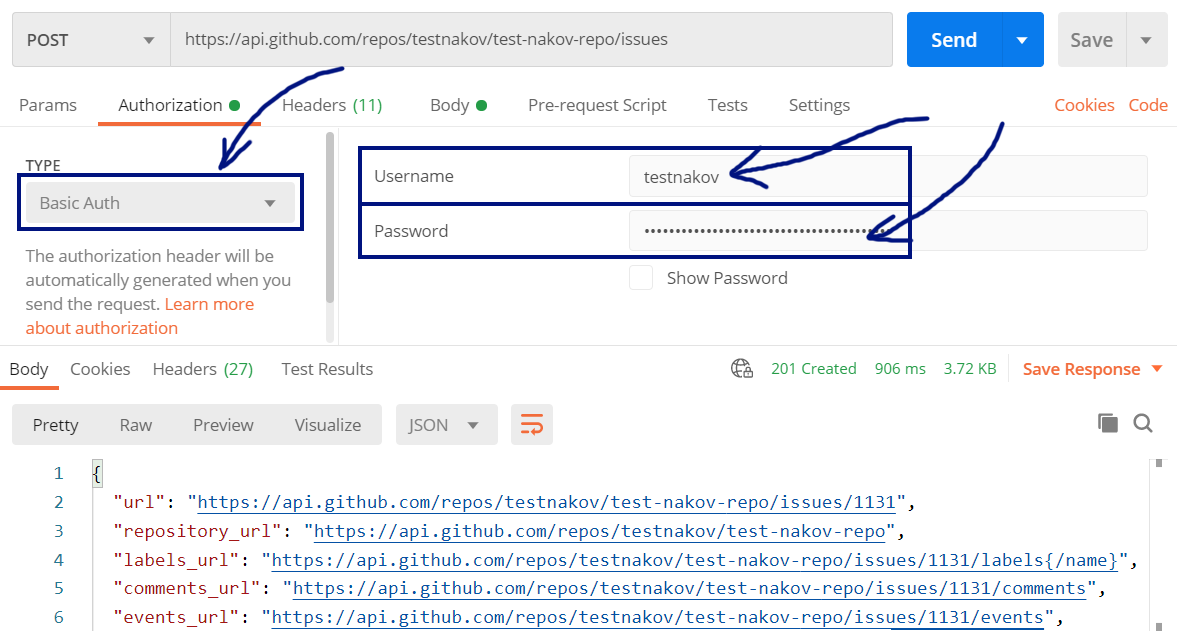
### Authentication in GitHub API

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



Once, a **personal access token** is created in GitHub, you can use it from **Postman** using the “**Basic authentication**” settings for the HTTP request. An example is shown below:



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 a **personal access token**, that you have previously created from the Developer Settings 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**. This is the decoded base64 string from the above request:

|  |  |
| --- | --- |
| Base64 string | dGVzdG5ha292OjMzYjQ3MzUzZTE2NGU4YTkxZDlmMDM2MGVjNDdkYmFmNWUzNzJhNg== |
| String value | testnakov:33b47353e164e8a91d9f0360ec47dbaf5e372a6 |

### GitHub API: Sample HTTP Response

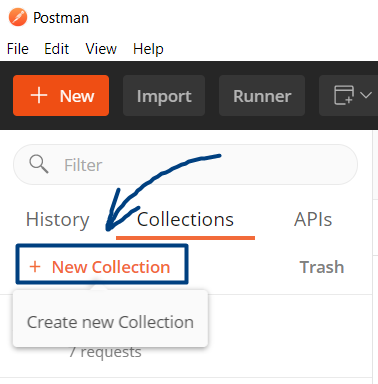
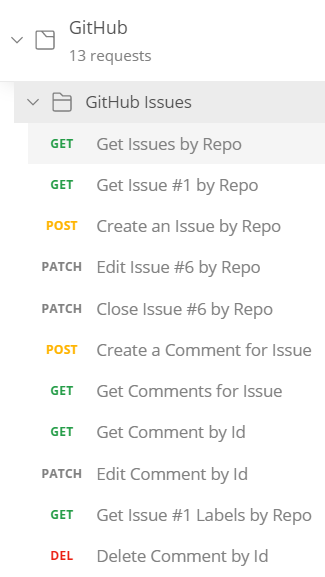
A typical **HTTP response** from the GitHub API may look like this:

|  |
| --- |
| 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":"testnakov","id":23406465,"node\_id":"MDQ6VXNlcjIzNDA2NDY1","avatar\_url":"https://avatars2.githubusercontent.com/u/23406465?u=b090ea0dc2d6c5cf71bcc39160cda63ab2f28714&v=4","gravatar\_id":"","url":"https://api.github.com/users/testnakov","html\_url":"https://github.com/testnakov","followers\_url":"https://api.github.com/users/testnakov/followers","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://api.github.com/users/testnakov/orgs","repos\_url":"https://api.github.com/users/testnakov/repos","events\_url":"https://api.github.com/users/testnakov/events{/privacy}","received\_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} |

## Create Postman Collection of Requests

Now, you should create a **Postman collection** of HTTP requests for accessing the **GitHub Issues API**. Use the **[+ New Collection]** button on the left sidebar in Postman (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 looks as shown below. It may be structured in **folders** and the requests should have appropriate names, like it is shown below:

 🡪 

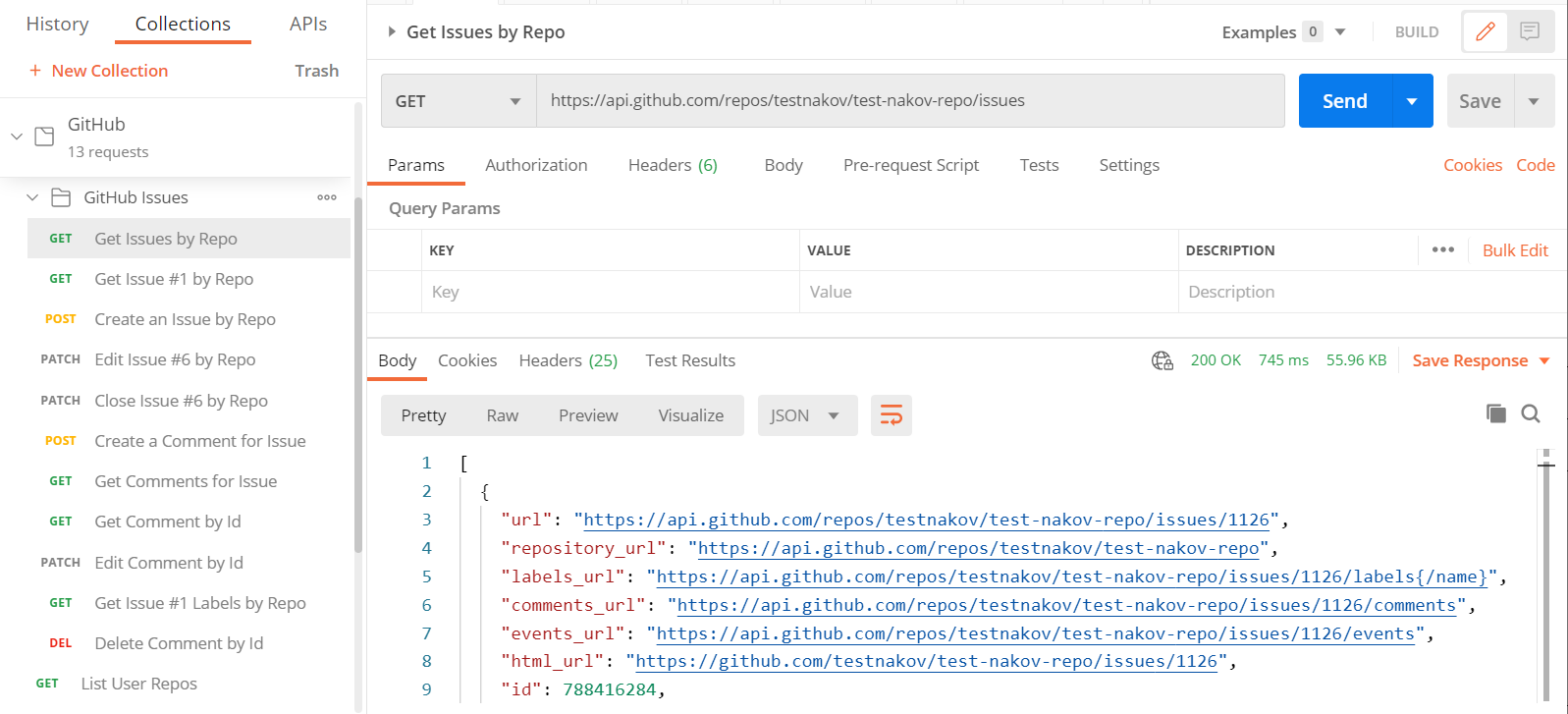
Now it’s time to **create a few HTTP requests** in the new Postman collection.

### 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 looks in **Postman** after successful execution:



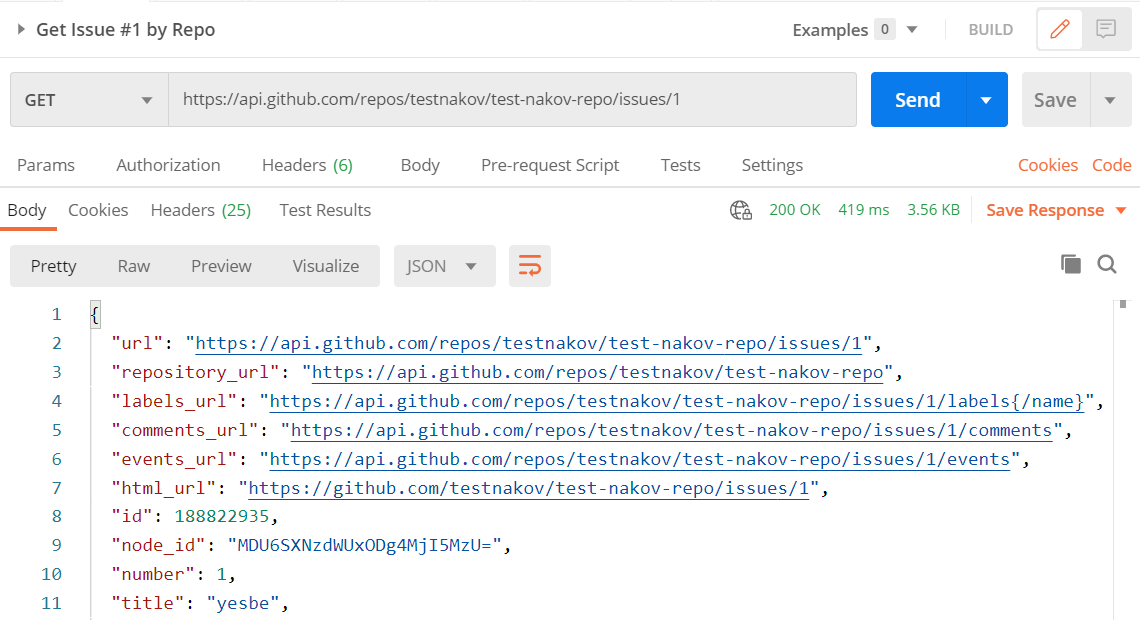
The returned HTTP status code is “**200 OK**” and the HTTP **response body** holds the issues as **JSON array** of objects.

### Retrieve Issue by Number

Retrieve issue #1 from repo “test-nakov-repo” in user “testnakov”:

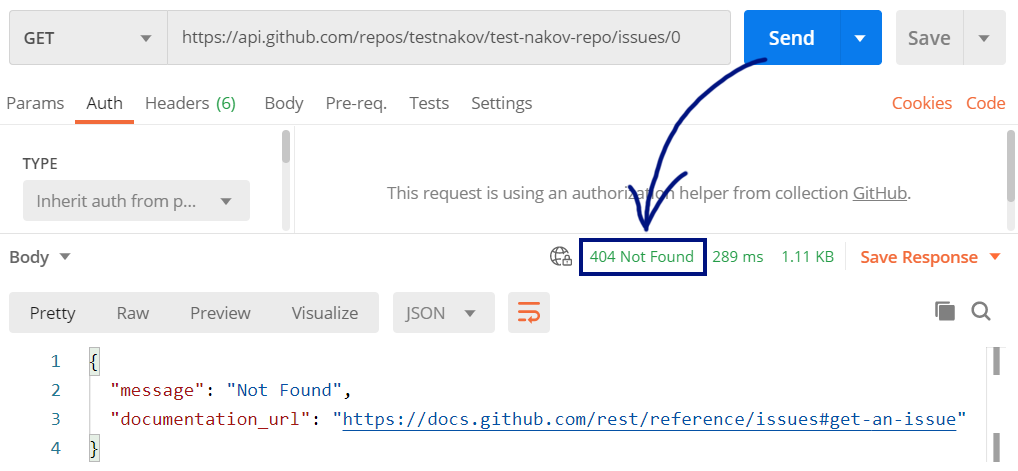
|  |  |
| --- | --- |
| Request | GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/1 |
| Body | *(empty)* |

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



You can also view the same **issue #1** in the GitHub Issues: <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**:

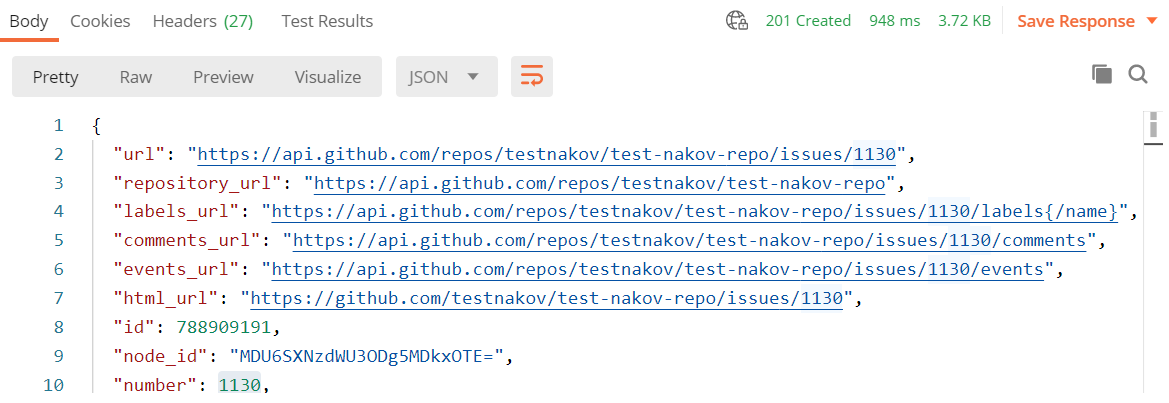


### Create a New Issue

Create a new issue in the repo “test-nakov-repo” of user “testnakov”:

|  |  |
| --- | --- |
| 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 theresponsebody a **JSON** object, holding **the number of the new issue**, together with other issue details:



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>.

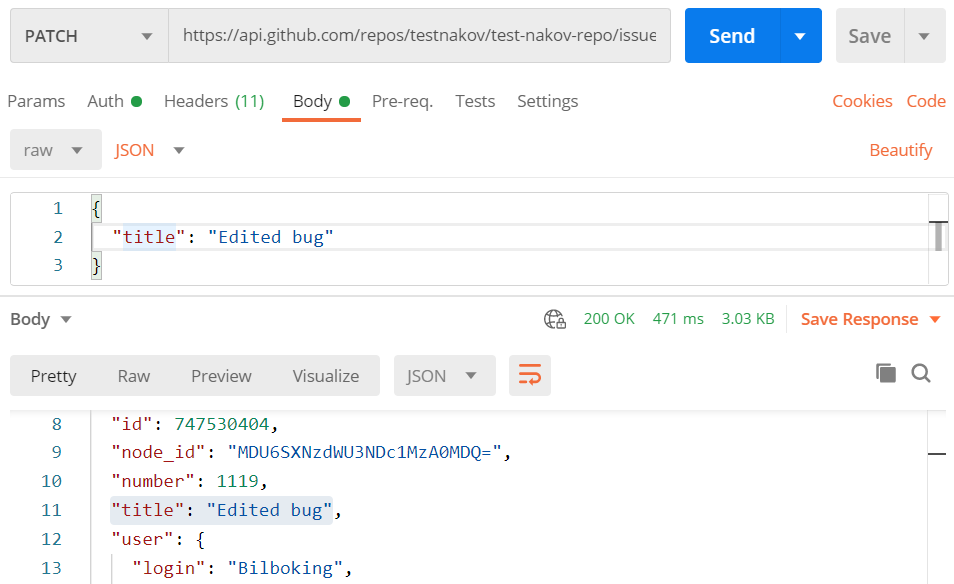
In case you don’t provide valid authentication for the GitHub API for the HTTP request, you will get an error response: **401 Unauthorized**.

### Edit Existing Issue

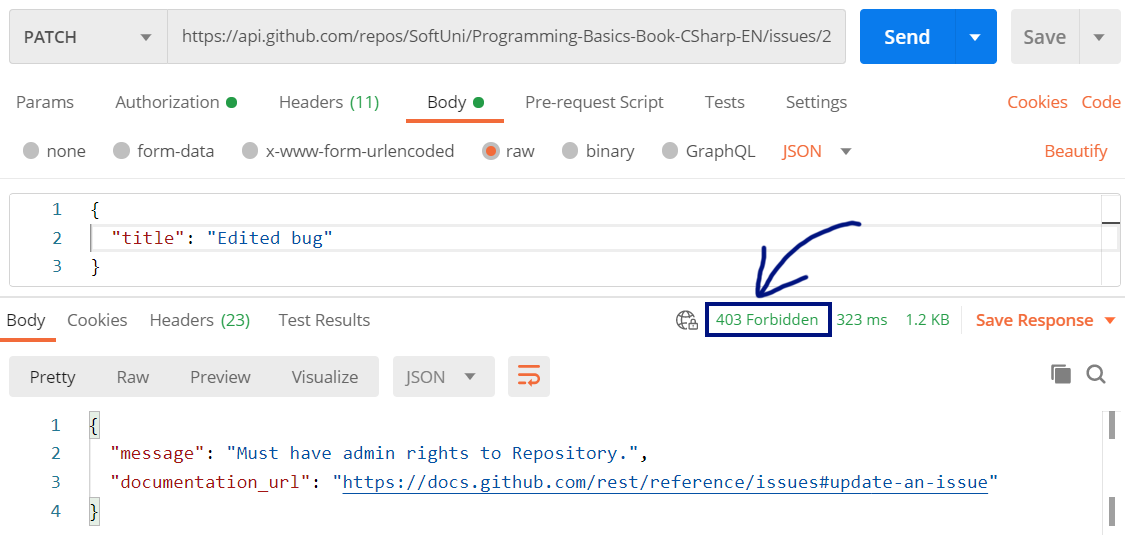
Edit existing issue #1119 from the repo “test-nakov-repo” of user “testnakov”:

|  |  |
| --- | --- |
| Request | PATCH https 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** asresponsebody:



**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**:



### Close Existing Issue

Close issue #6 from the repo “test-nakov-repo” of user “testnakov”:

|  |  |
| --- | --- |
| Request | PATCH https 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** asresponsebody. You can see the closed issue here: <https://github.com/testnakov/test-nakov-repo/issues/6>.

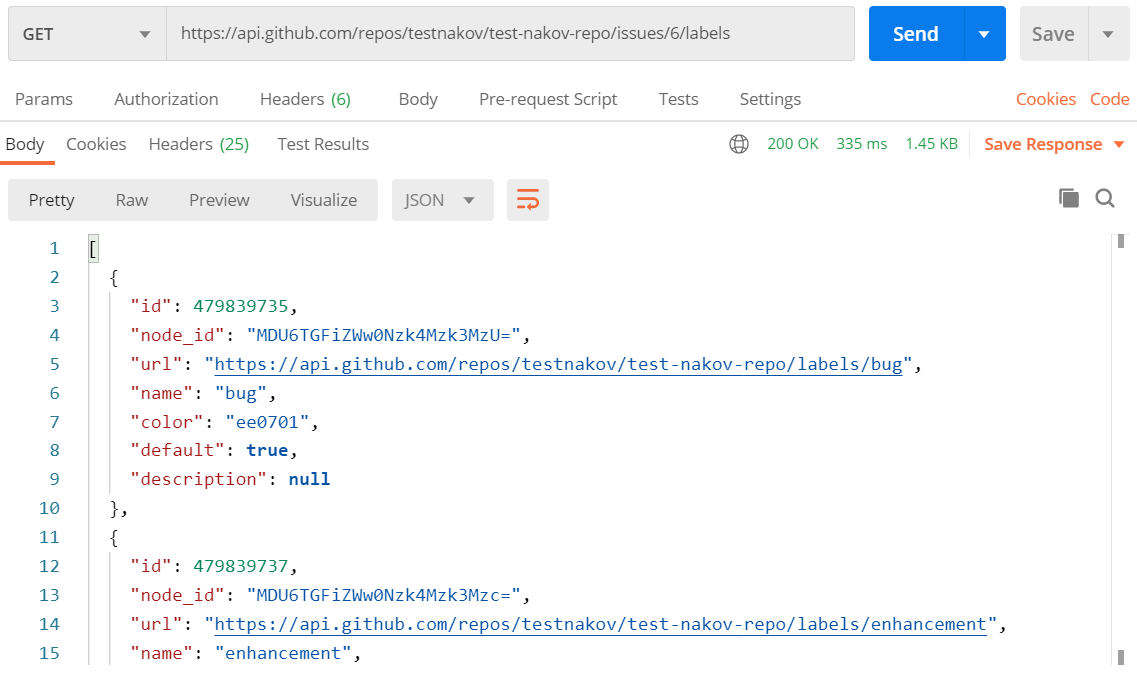
**Note**: you can edit / close only your own issues. Repo admins can edit or close user’s issues. If you try to close an issue without sufficient privileges, you will get **403 Forbidden**.

### 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** asresponsebody in JSON format (array of labels):



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** asresponsebody in JSON format. You can see the new comment here: <https://github.com/testnakov/test-nakov-repo/issues/6>.

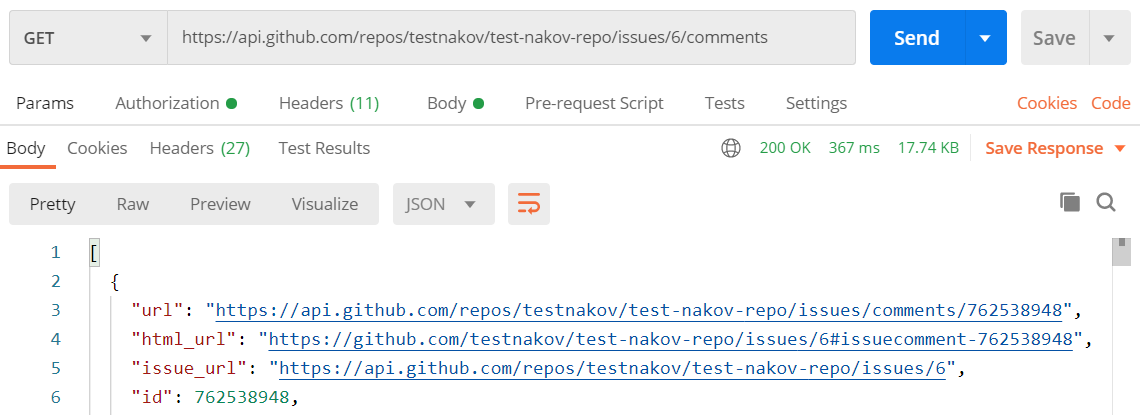
If you try to create a comment in a repo, where you don’t have sufficient privileges, you will get **403 Forbidden**.

### Retrieve All Comments for Issue

Retrieve all comments for existing issue #6 from the repo “test-nakov-repo” of user “testnakov”:

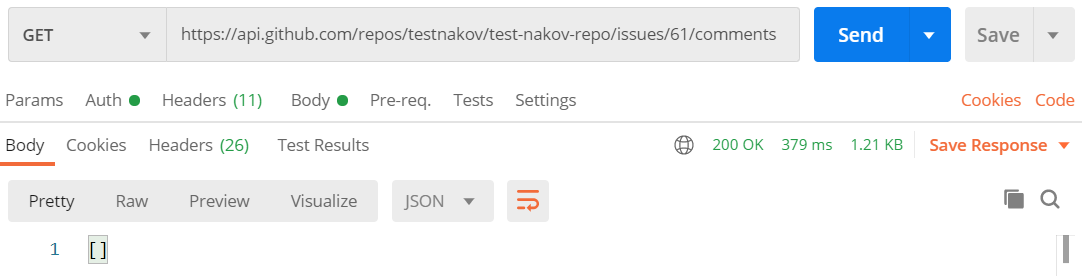
|  |  |
| --- | --- |
| Request | GET https://api.github.com/repos/testnakov/test-nakov-repo/issues/6/comments |
| Body | *(empty)* |

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



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 and **empty JSON array**:



### 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 take its id for 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**.

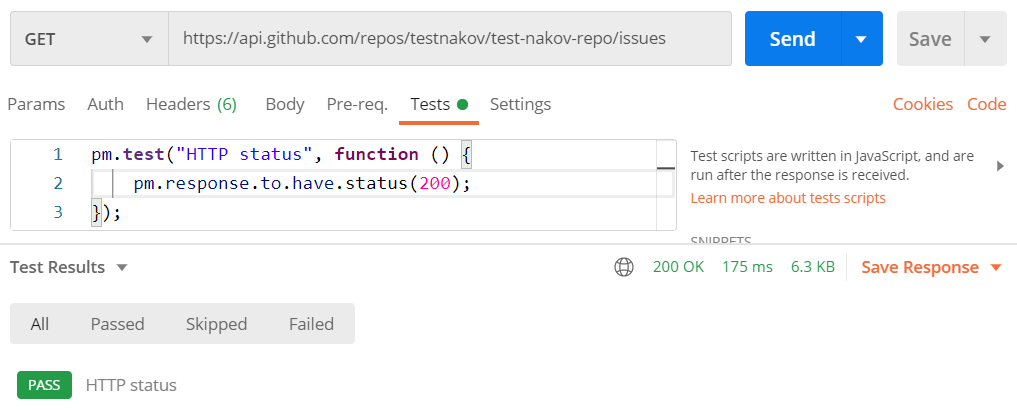
## Write Postman Tests

Write **Postman tests** for the requests from your GitHub Issues API collection.

### About Postman Tests

**Postman** allows writing **simple automated tests** for each of your **HTTP requests**. Learn more at the Postman documentation page: [https://learning.postman.com/docs/writing-scripts/test-scripts](https://learning.postman.com/docs/writing-scripts/test-scripts/).

The **simplest test** for a RESTful API endpoint is to check its **HTTP status code**:



The test itself is a **JavaScript code**, which uses the pm object in Postman. Postman HTTP request-level tests are based on the **Chai assertion library** (<https://chaijs.com>) and use the **BDD** (behavior-driven development) syntax of assertion. This is how we can assert that the expected **HTTP status code** in the HTTP response is correct:

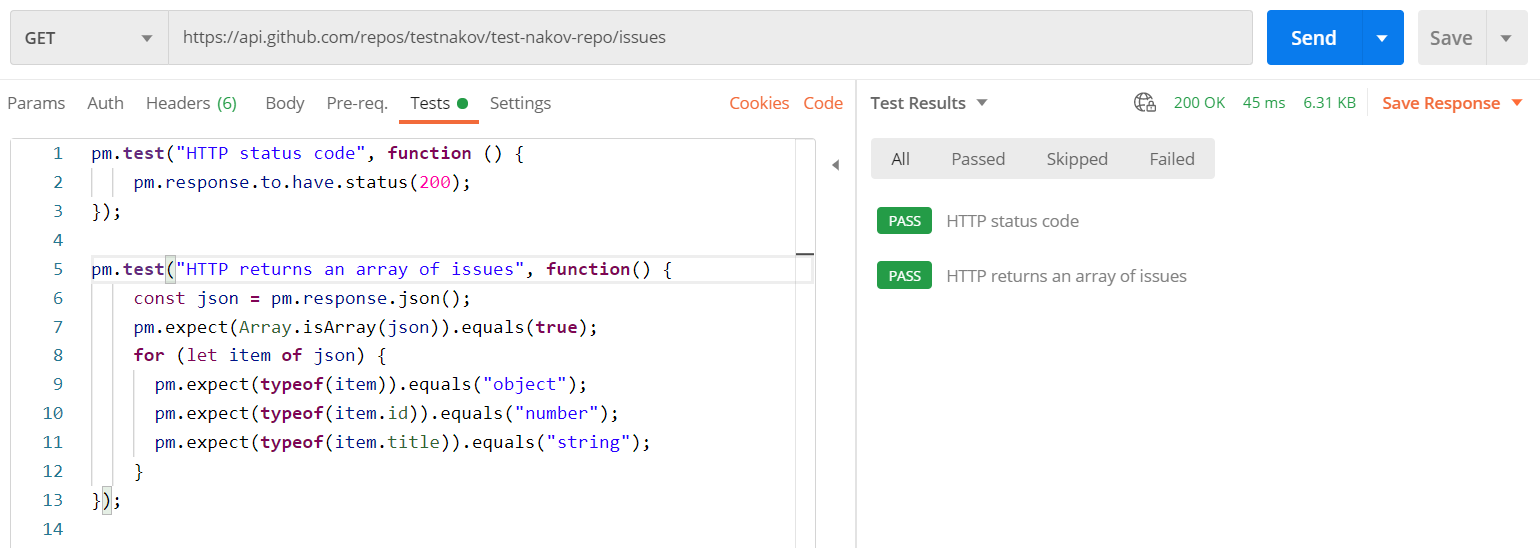
|  |
| --- |
| pm.**test**("HTTP status code", **function** () {      pm.response.to.have.status(200);  }); |

This is how we can assert that certain HTTP request has returned a **JSON object**, holding an **array of objects**, and each object has property “id”, “title” and “body” of types **number**, **string** and **string**:

|  |
| --- |
| pm.**test**("Response is arrays of issues", **function**() {    pm.expect(pm.response.headers.**get**('Content-Type'))  .to.eql('application/json; charset=utf-8');    const json **=** pm.response.json();    pm.expect(Array.isArray(json)).equals(**true**);  **for** (let item **of** json) {      pm.expect(**typeof**(item)).equals("object");      pm.expect(Number.isInteger(item.id)).equals(**true**);      pm.expect(**typeof**(item.title)).equals("string");      pm.expect(**typeof**(item.body)).equals("string");    }  }); |

### Test “All Issues” GET Request

Now, you are ready to write a Postman API test for the “**GET all issues**” HTTP request:



The above code asserts that the invoked **HTTP GET request** returns stats code **200 OK** and a **JSON array of objects** and each object has properties: id (number) and title (string).

### Test “Get Issue by Number” GET Request

Write a Postman API test for the “**Get Issue by Number**” HTTP request. We have several **test scenarios**:

* GET request for **valid issue** returns **200 OK** + the requested issue as **JSON** object.
  + Assert that the returned data is JSON object, with “id” and “number” properties, which are **integers**.
* GET request for **invalid issue** (#631631265) returns status code **404 Not Found**.

### Test “Create New Issue” POST Request

Write a Postman API test for the “**Get Issue by Number**” HTTP request. We have several **test scenarios**:

* POST request with **valid auth data and JSON body** returns **201 Created** + the new issue as **JSON** object.
  + Assert that the returned data is JSON object, with “id” and “number” properties, which are **integers**.
* Authenticated POST request (**without** Authorization **header**) returns **401 Unauthorized**.
* Invalid POST request (with **empty body**) returns an error status code (**422 Unprocessable Entity**).
* Invalid POST request (with **missing title**) returns an error status code (**422 Unprocessable Entity**).

### Test “Edit Existing Issue” PATCH Request

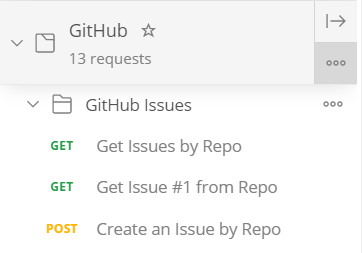
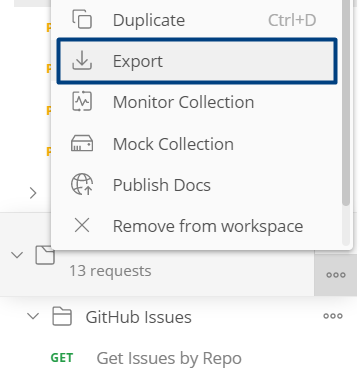
Write a Postman API test for the “**Edit Issue by ID**” HTTP request. We have several **test scenarios**:

* PATCH request with **valid auth data and JSON body** returns **200 ОК** + the modified issue as **JSON** object.
  + Assert that the returned data is JSON object, with “id” and “number” properties, which are **integers**.
* Invalid issue (**with non-existing ID**) returns **404 Not Found**.
* Authenticated PATCH request (**without** Authorization **header**) returns **401 Unauthorized**.

## Run the Postman Tests from the Console

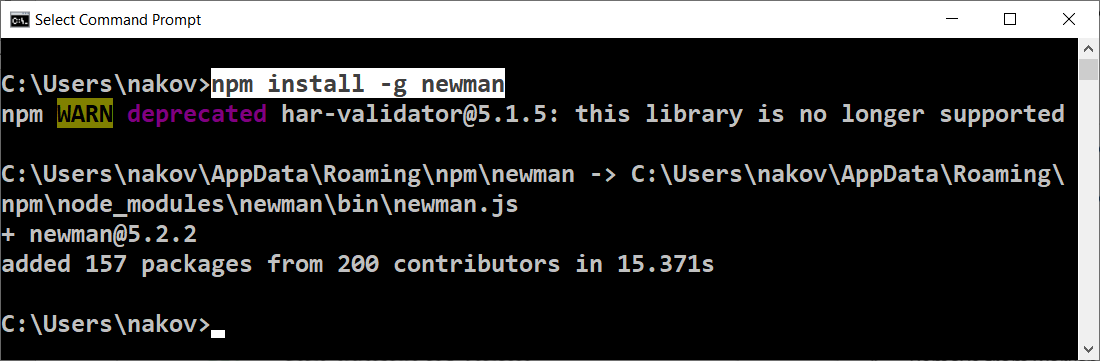
**Export** the Postman collection of requests, together with the tests for each request to external file:

* Postman\_GitHub\_Issues.postman\_collection.json

 🡪 

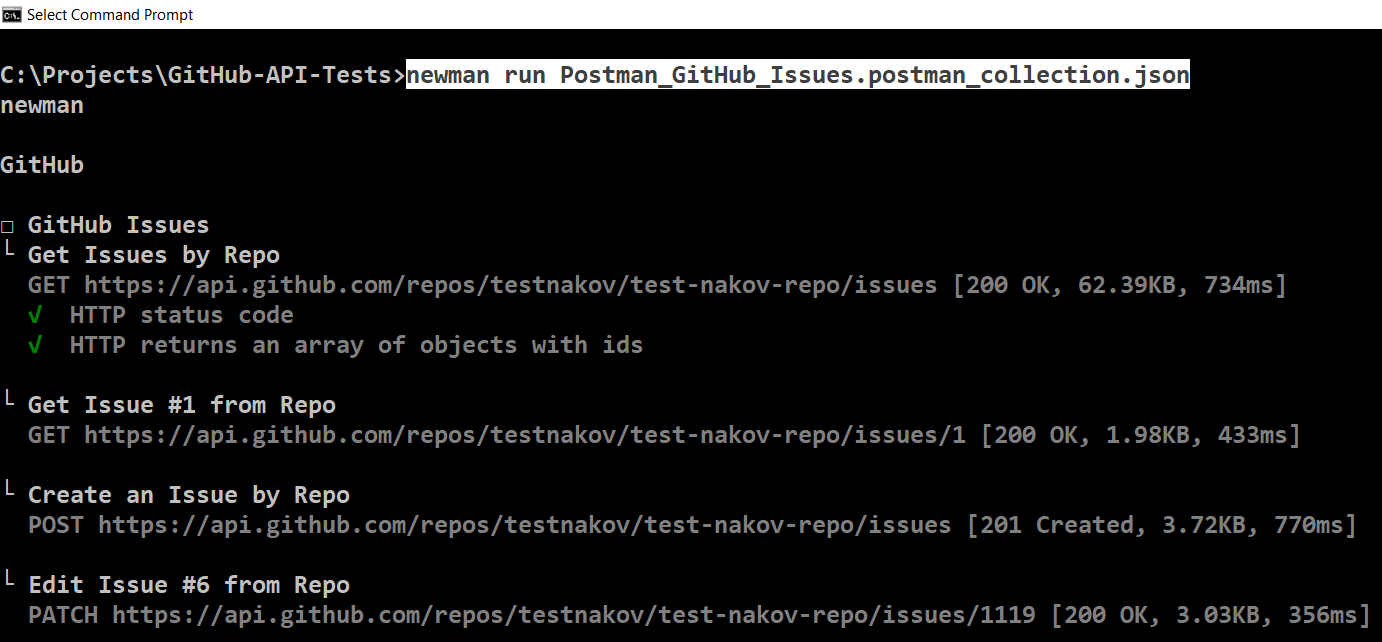
Install the “newman” tool (a command-line collection runner for Postman):

|  |
| --- |
| npm install -g newman |



**Run the Postman tests** from the console, using the following command:

|  |
| --- |
| newman run Postman\_GitHub\_Issues.postman\_collection.json |

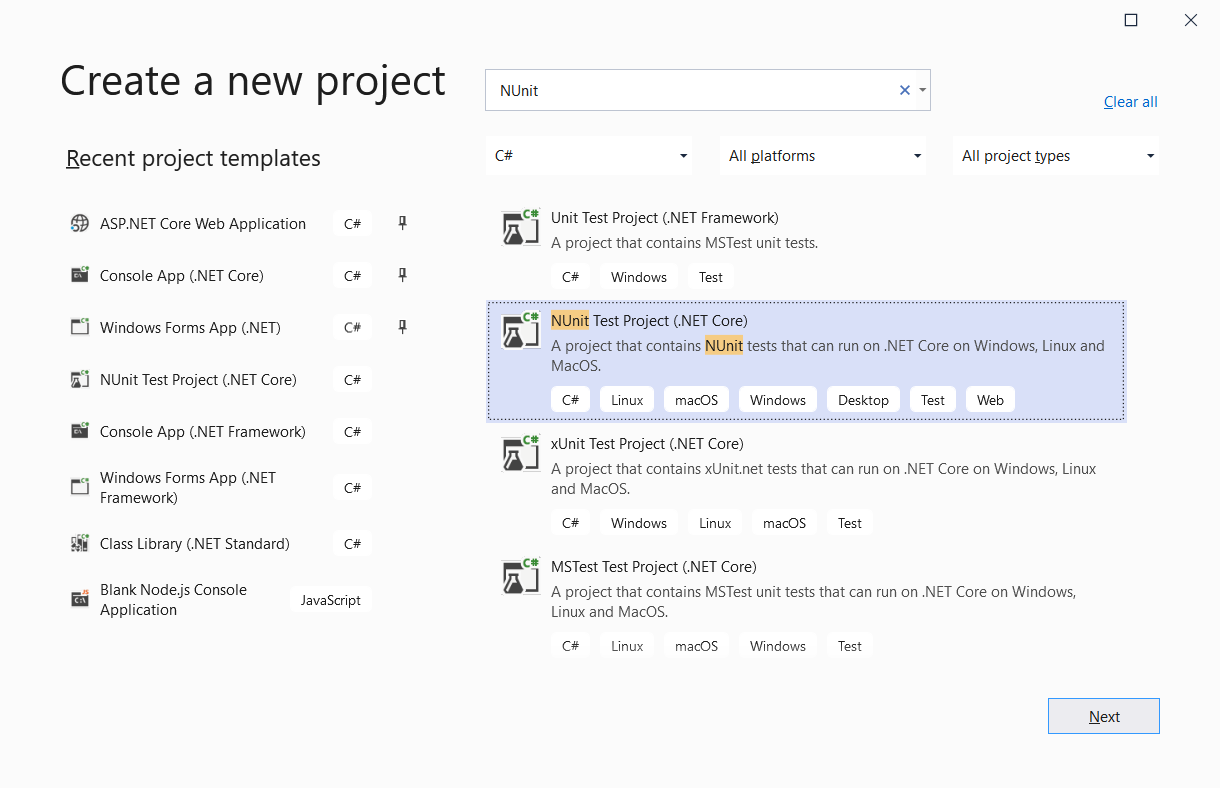


## API Tests in C#

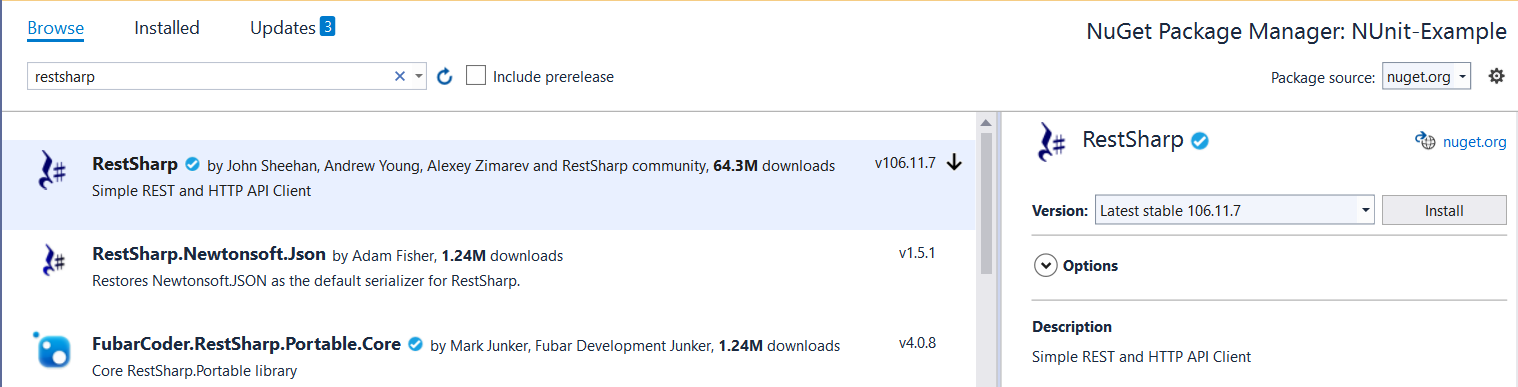
Implement API tests for GitHub Issues in C#.

### How to Write API Tests in C#?

Create **new C# project** in Visual Studio:



Add the “**RestSharp**” library from NuGet packages:



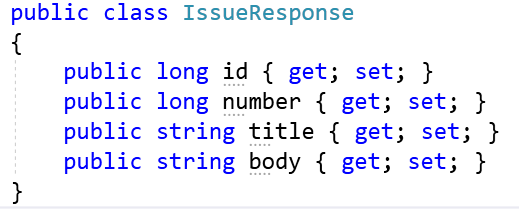
Use the following code as a reference:

* <https://gist.github.com/nakov/ba92d9e995635eefc8a9b0aba9d37211>

### Test “All Issues” GET Request

To test the “all issues for a repo” GET endpoint, use the following code as reference:





### Test “Create Issue” POST Request

To test the “create new issue” POST endpoint, use the following code as reference:

