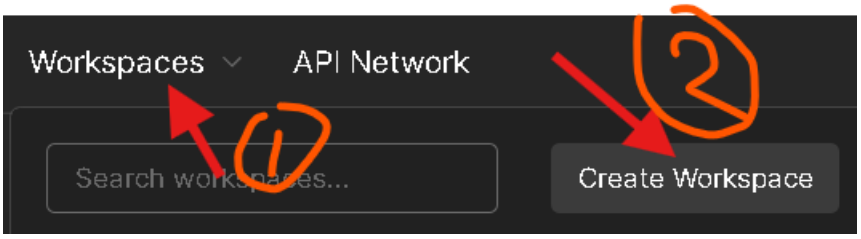
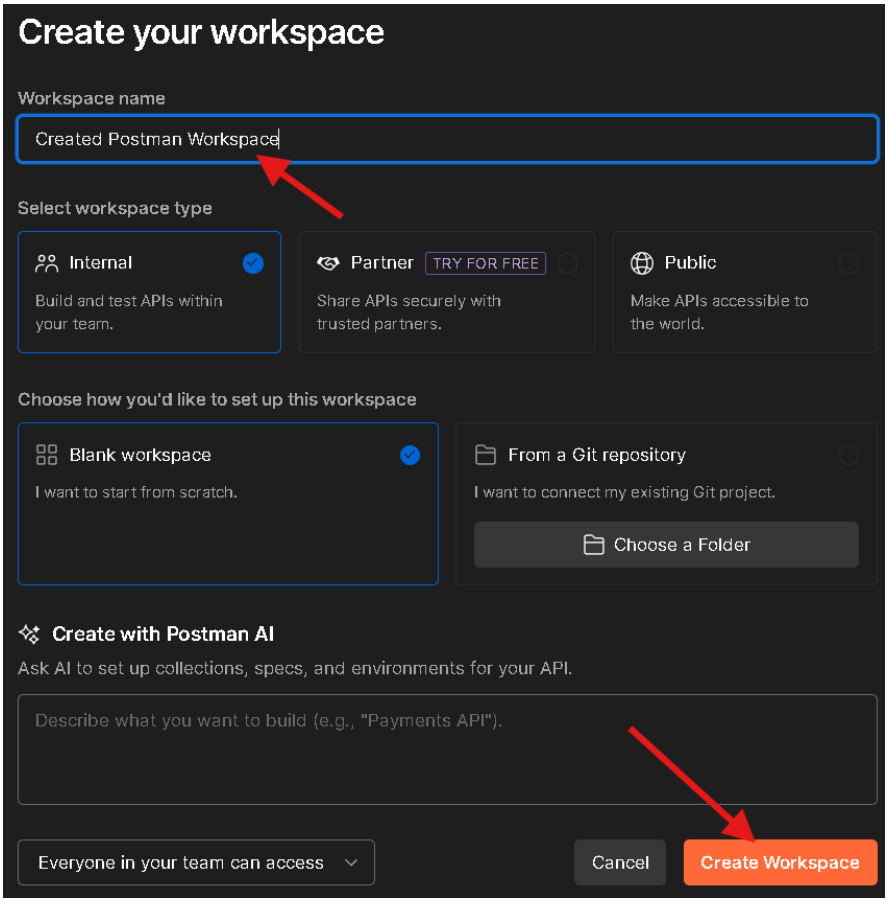
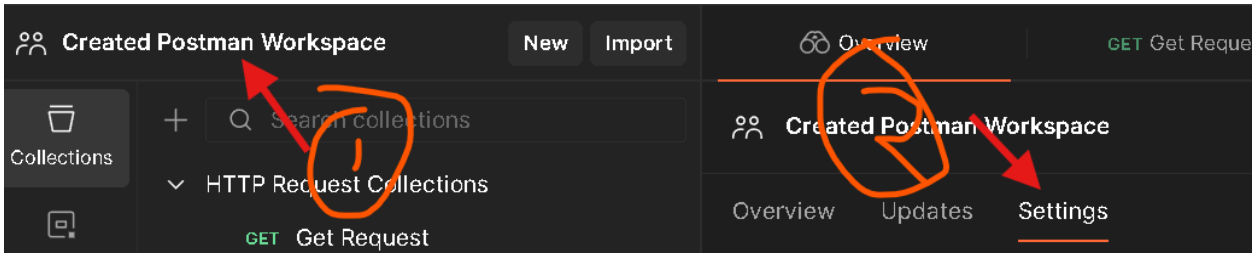
Create workspace - To work on APIs

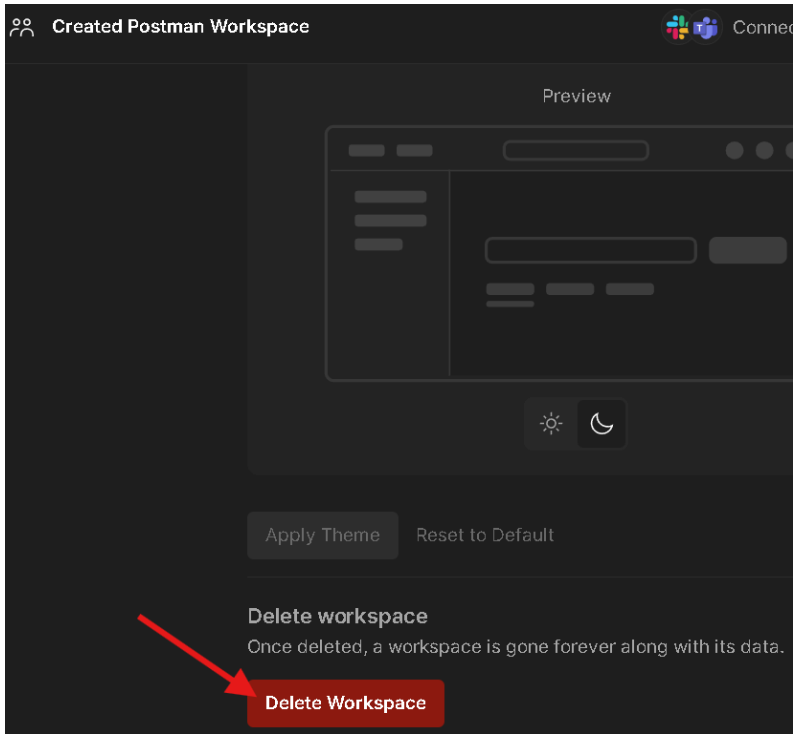


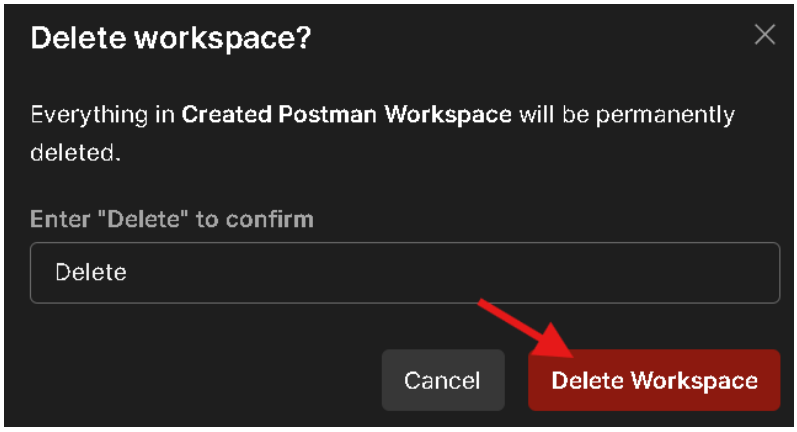


Delete WorkSpace

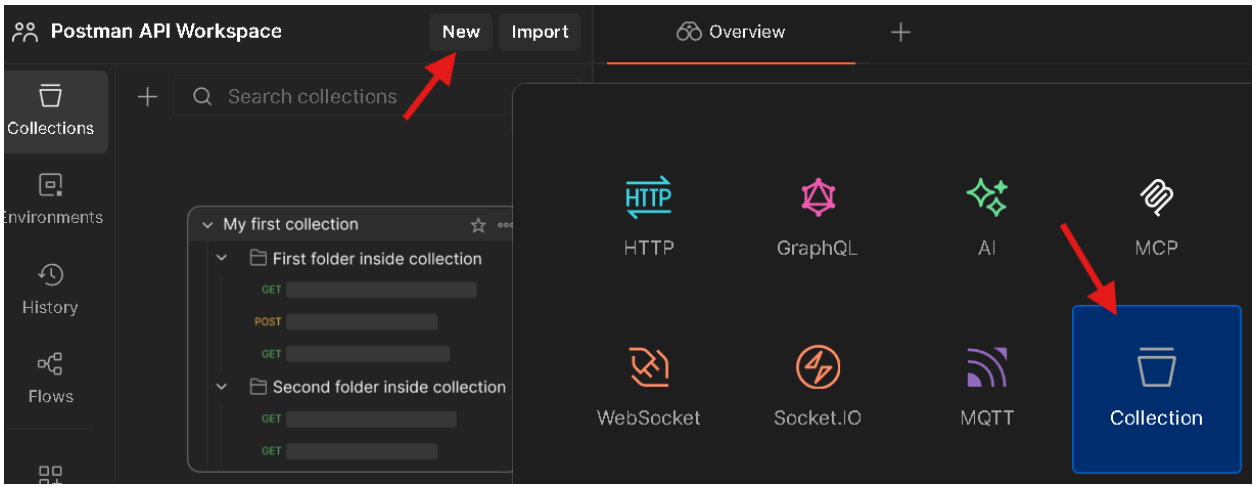


Scroll down


Created Postman Workspace
Preview
Apply Theme
Reset to Default
Delete workspace
Once deleted, a workspace is gone forever along with its data.
Delete Workspace

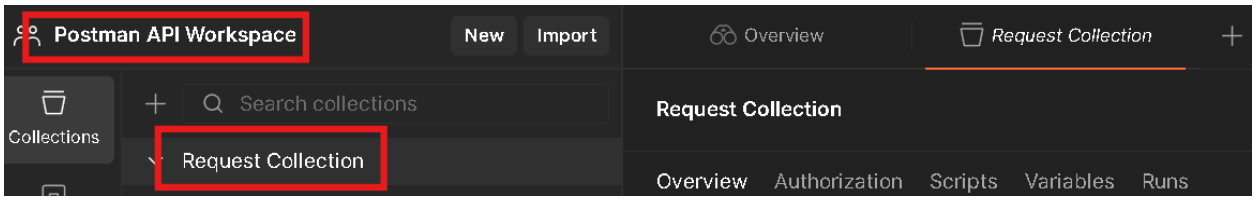

Delete workspace?
Everything in Created Postman Workspace will be permanently deleted.
Enter "Delete" to confirm
Delete
Cancel
Delete Workspace

Create Collection - To have HTTP Request



Name the collection

Finally Workspace and Collection to keep request is created



CRUD API Testing

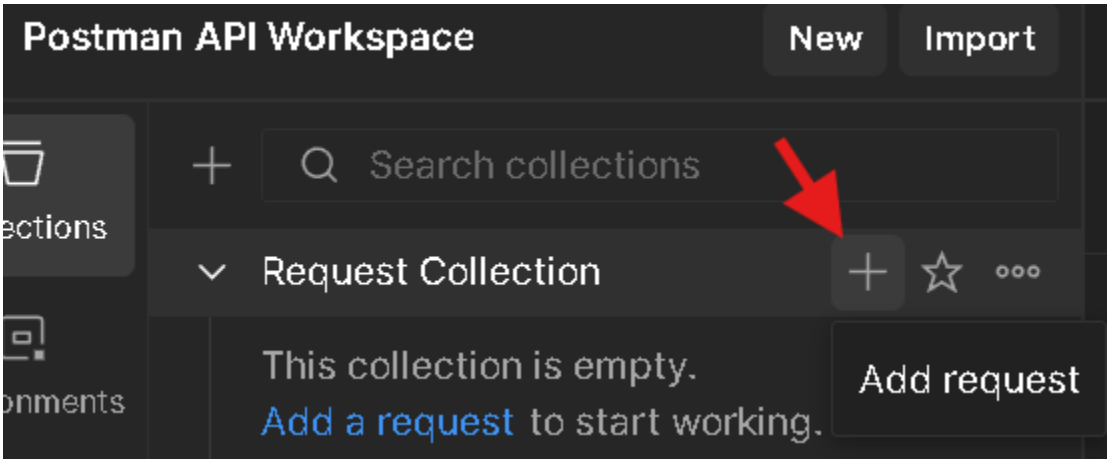
CRUD = Create, Read, Update, Delete

It means testing all basic operations of an API.

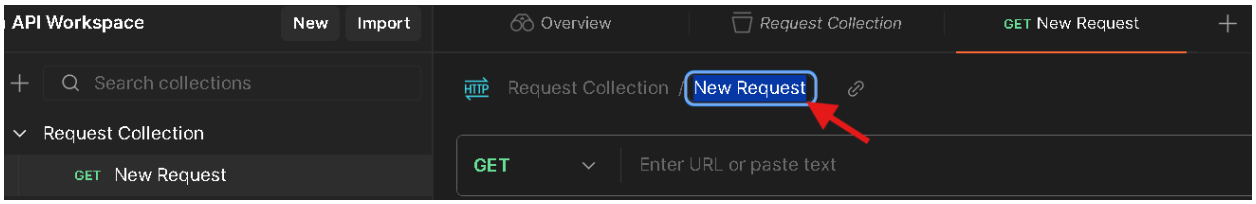
Operation	HTTP Method	Purpose
Create	POST	Add new data
Read	GET	Fetch data
Update	PUT / PATCH	Modify data
Delete	DELETE	Remove data

GET Request

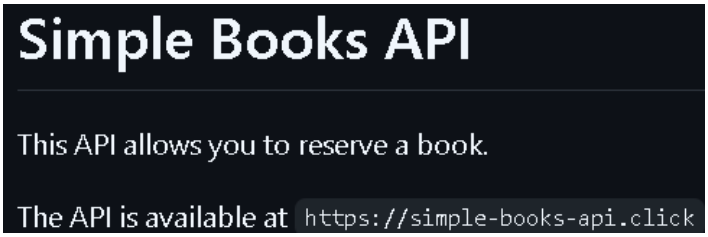
GET Status

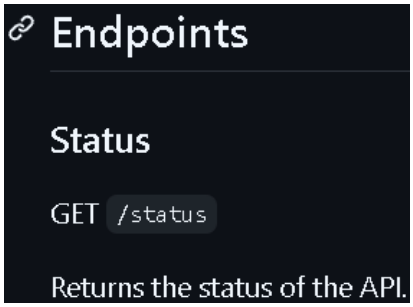


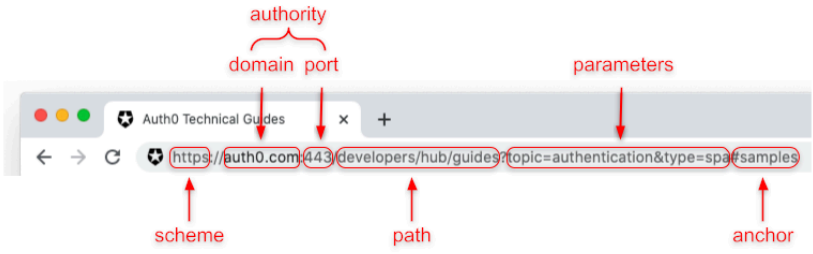
Rename the HTTP(Hypertext Transfer Protocol) - GET Request



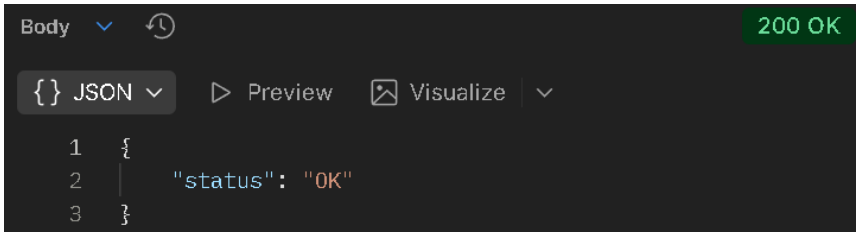
Notes





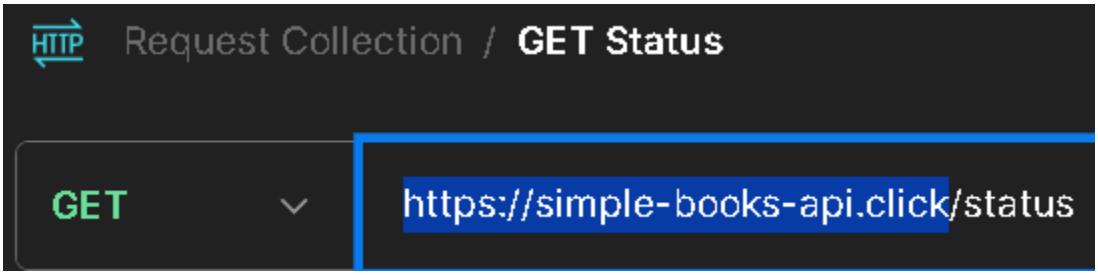


Write -> Save -> Send

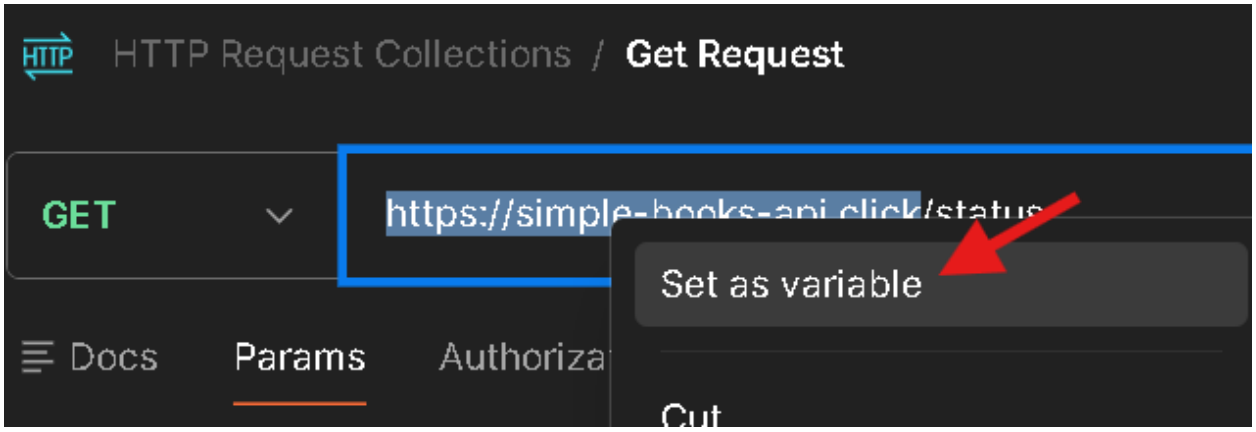


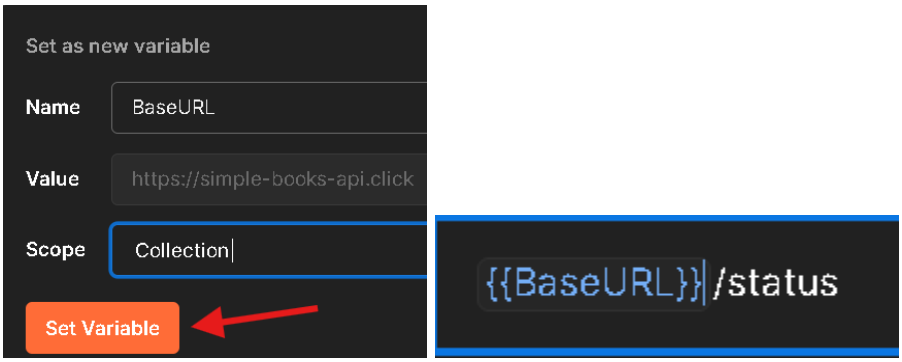
Set a Variable

Select the API

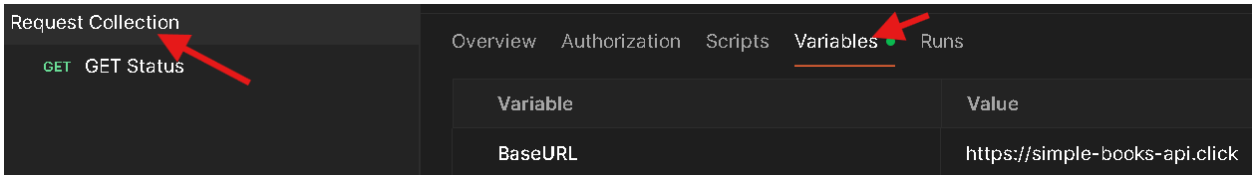


Right click on it



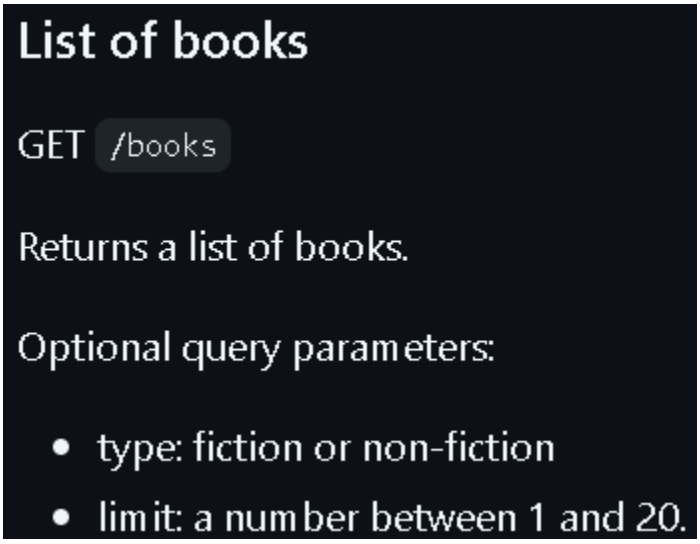


To Look collection Variables: Click target collection -> Variables

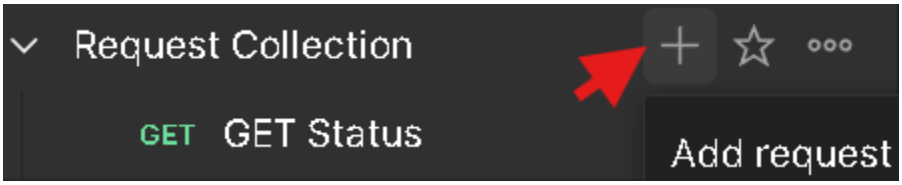


GET List of Books

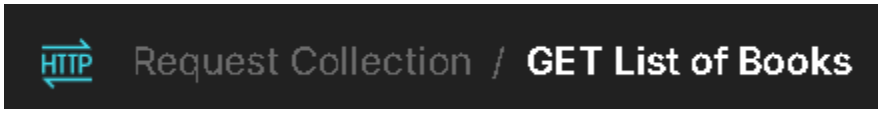
Notes



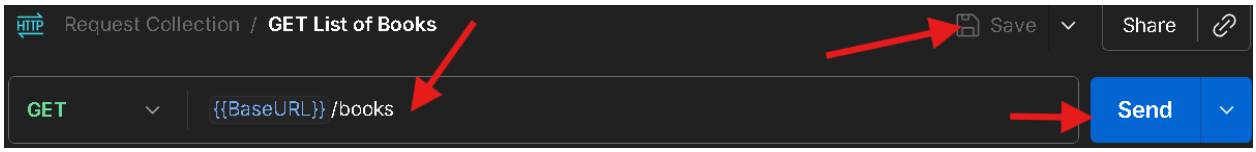
Let's get list of books
Add a request to colleciton



Rename it

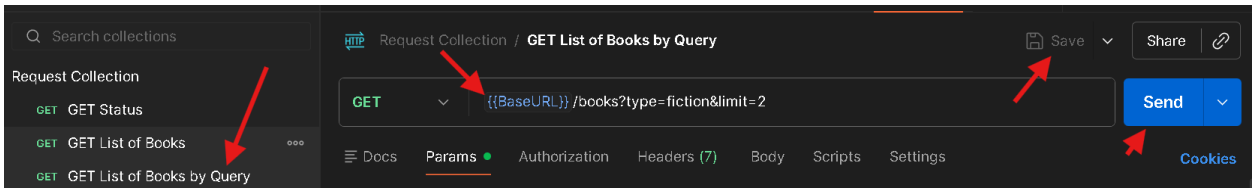


Provide -> save -> Send

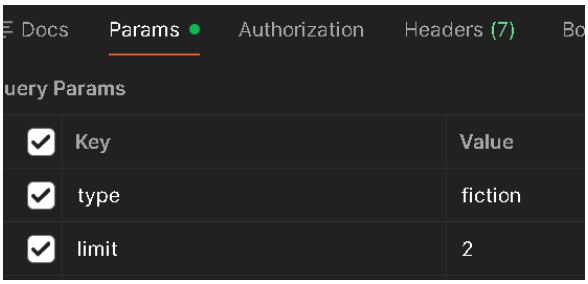


GET List of Books By Query Parameters

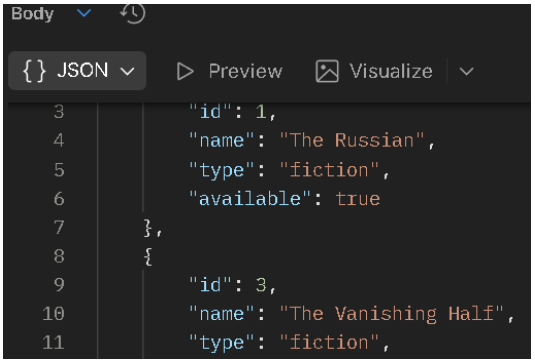
Create request -> Write -> save -> Send



During Write it automatically adds



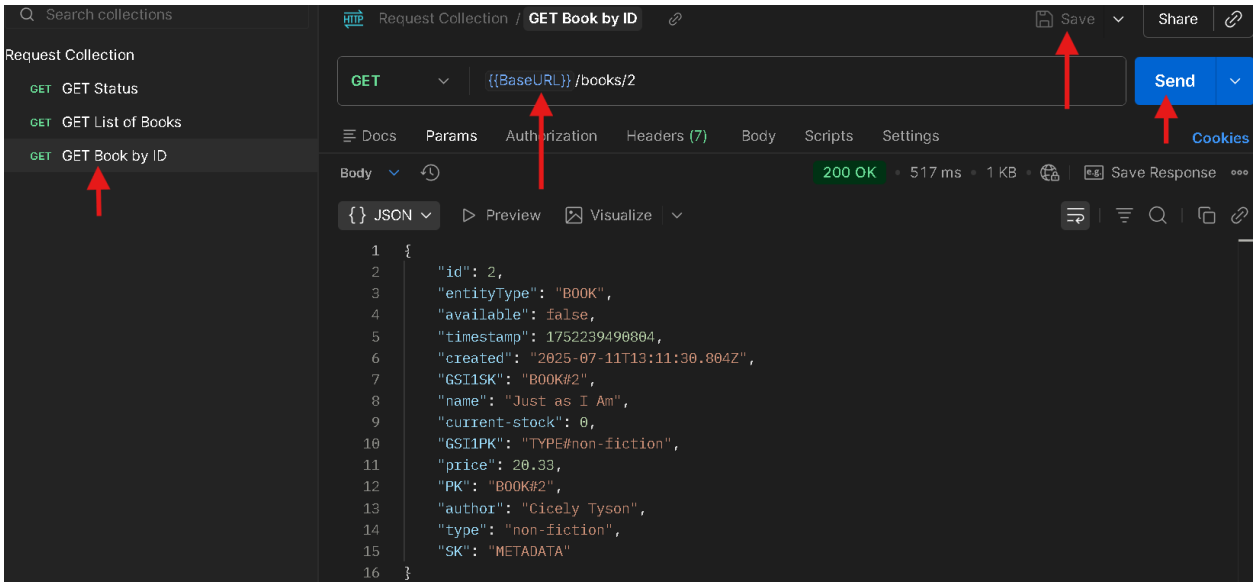
Where type if type of book, limit is number of books to watch



GET Book By ID

Similarly, let's GET Book by ID

Add request -> Write -> Save -> Send



POST Request - Submit Order without Access Token

Notes

Submit an order

POST `/orders`

Allows you to submit a new order. Requires authentication.

The request body needs to be in JSON format and include the following properties:

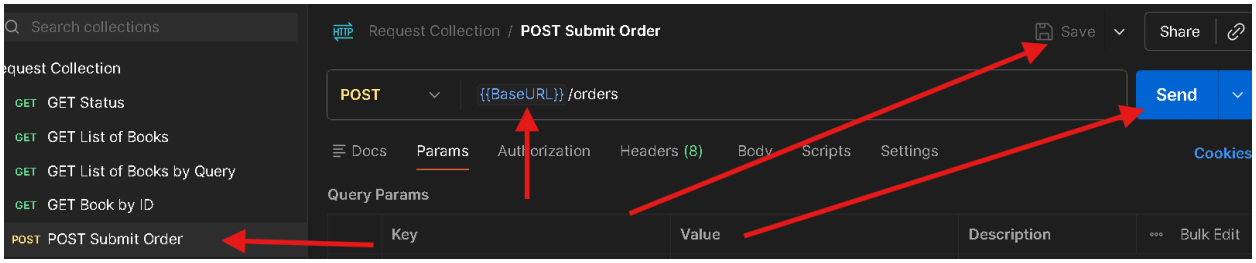
- `bookId` - Integer - Required
- `customerName` - String - Required

Example

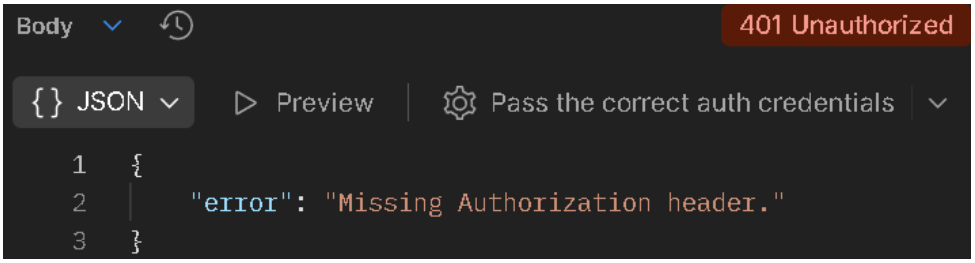
```
POST /orders/
Authorization: Bearer <YOUR TOKEN>

{
  "bookId": 1,
  "customerName": "John"
}
```

The response body will contain the order Id.

Create -> Write -> Save -> Send



Results



HTTP Status Codes Reference

🔵 1XX – Informational Responses

Code	Name	Description
100	*Continue*	*Client should continue with the request*
101	*Switching Protocols*	*Server agrees to switch protocols*
102	*Processing*	*Server is processing the request (WebDAV)*

🟢 2XX – Success Responses

Code	Name	Description
200	*OK*	*Request succeeded and data returned*
201	*Created*	*Resource successfully created*
202	*Accepted*	*Request accepted but processing not complete*
203	*Non-Authoritative Information*	*Meta-information from third party*
204	*No Content*	*Request succeeded, no response body*

205	*Reset Content*	*Client should reset the document view*
206	*Partial Content*	*Partial GET successful (range request)*

🟡 3XX – Redirection Responses

Code	Name	Description
300	*Multiple Choices*	*Multiple options for the resource*
301	*Moved Permanently*	*Resource permanently moved to new URL*
302	*Found*	*Temporarily moved to another URL*
303	*See Other*	*Response found at different URL*
304	*Not Modified*	*Cached version should be used*
307	*Temporary Redirect*	*Temporary redirect (same method)*
308	*Permanent Redirect*	*Permanent redirect (same method)*

🔴 4XX – Client Errors

Code	Name	Description
400	*Bad Request*	*Server cannot understand request*
401	*Unauthorized*	*Authentication required or failed*
403	*Forbidden*	*Server refuses to authorize*
404	*Not Found*	*Resource not found*
405	*Method Not Allowed*	*Method not allowed for URL*
406	*Not Acceptable*	*Resource not acceptable per headers*
407	*Proxy Authentication Required*	*Client must authenticate with proxy*
408	*Request Timeout*	*Server timed out waiting*
409	*Conflict*	*Conflict with current server state*

410	*Gone*	*Resource no longer available*
411	*Length Required*	*Content-Length missing*
412	*Precondition Failed*	*Request header conditions failed*
413	*Payload Too Large*	*Request entity too large*
414	*URI Too Long*	*URI too long*
415	*Unsupported Media Type*	*Media format not supported*
417	*Expectation Failed*	*Server cannot meet request expectation*
429	*Too Many Requests*	*Too many requests in short time*

⚫ 5XX – Server Errors

Code	***Name***	***Description***
500	*Internal Server Error*	*Unexpected server condition*
501	*Not Implemented*	*Method not supported*
502	*Bad Gateway*	*Invalid response from upstream server*
503	*Service Unavailable*	*Server overloaded or down*
504	*Gateway Timeout*	*No timely response from upstream*
505	*HTTP Version Not Supported*	*HTTP version not supported*

API Authentication

To submit or view an order, you need to register your API client.

POST `/api-clients/`

The request body needs to be in JSON format and include the following properties:

- `clientName` - String
- `clientEmail` - String

Example

```
{
   "clientName": "Postman",
   "clientEmail": "valentin@example.com"
}
```

The response body will contain the access token. The access token is valid for 7 days.

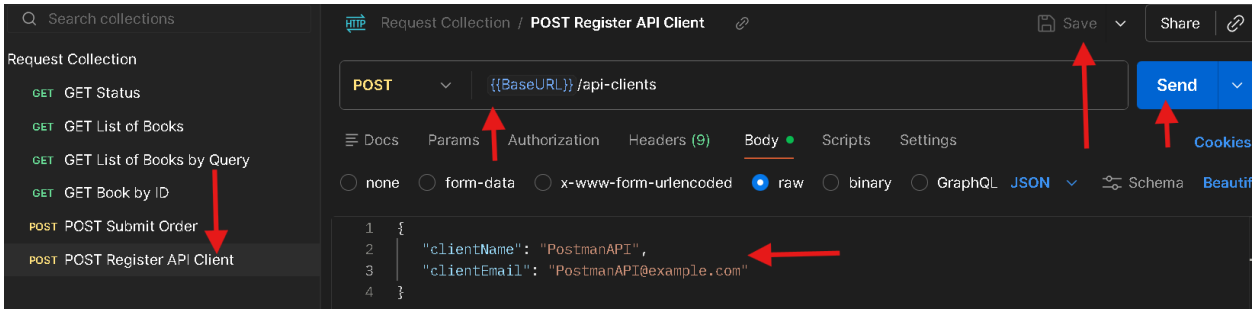
Possible errors

Status code 409 - "API client already registered." Try changing the values for `clientEmail` and `clientName` to something else.

Access token is like a temporary password, and using for the request

POST Request[Register API Client] - To Access Token

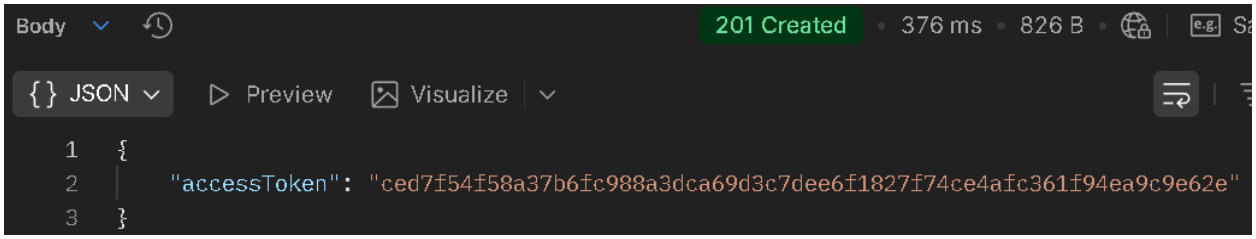
Create -> Name it -> Write URL & Body raw -> Save -> Send



Body raw Json =

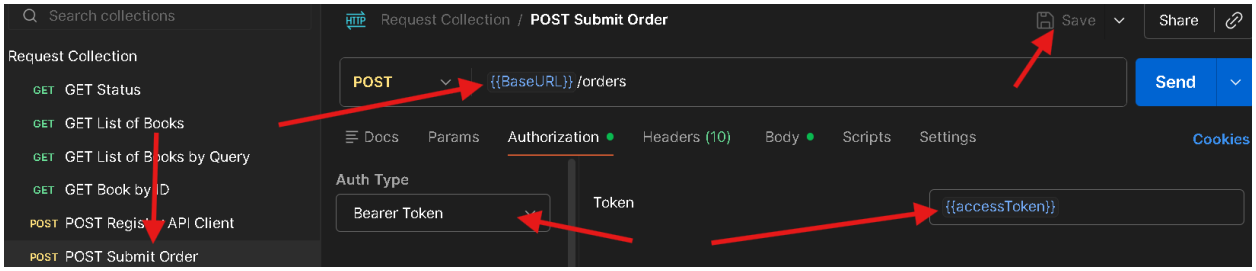
```
"clientName": "PostmanAPI",
"clientEmail": "PostmanAPI@example.com"
}
```

Results:

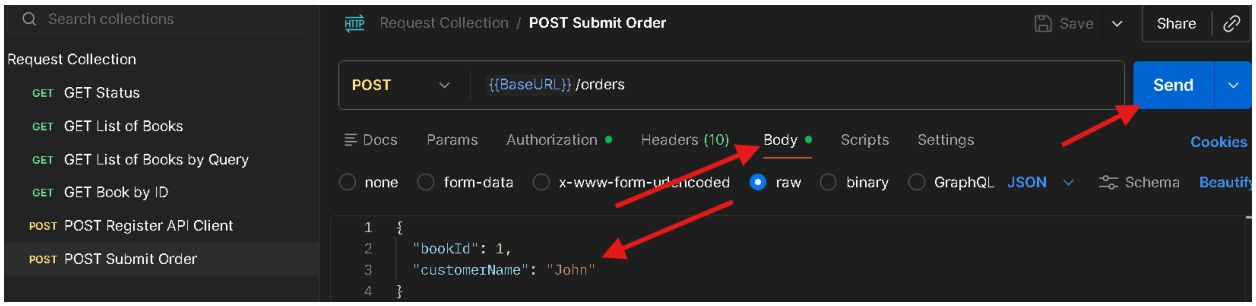


Save token as Collection Variable

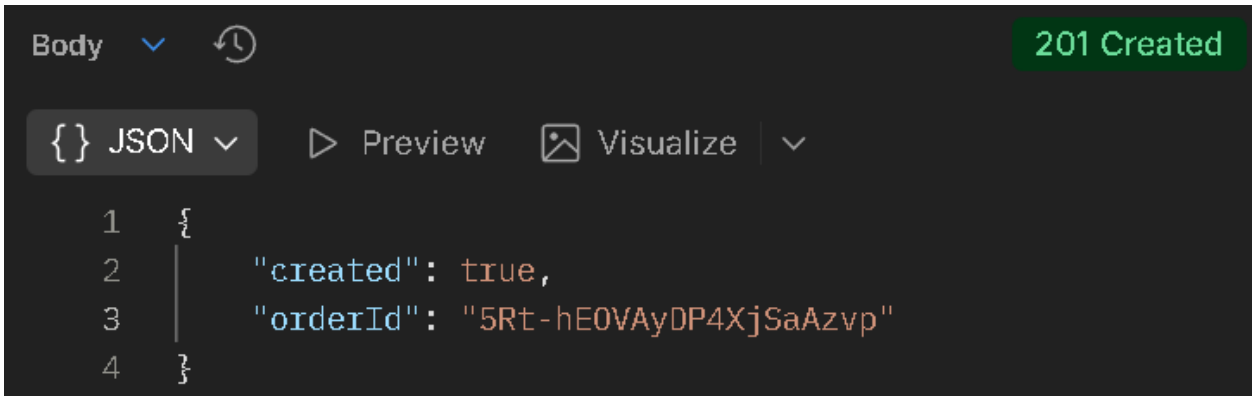
POST Request - Submit Order with Access Token



*Body includes **Make sure JSON is choosen***

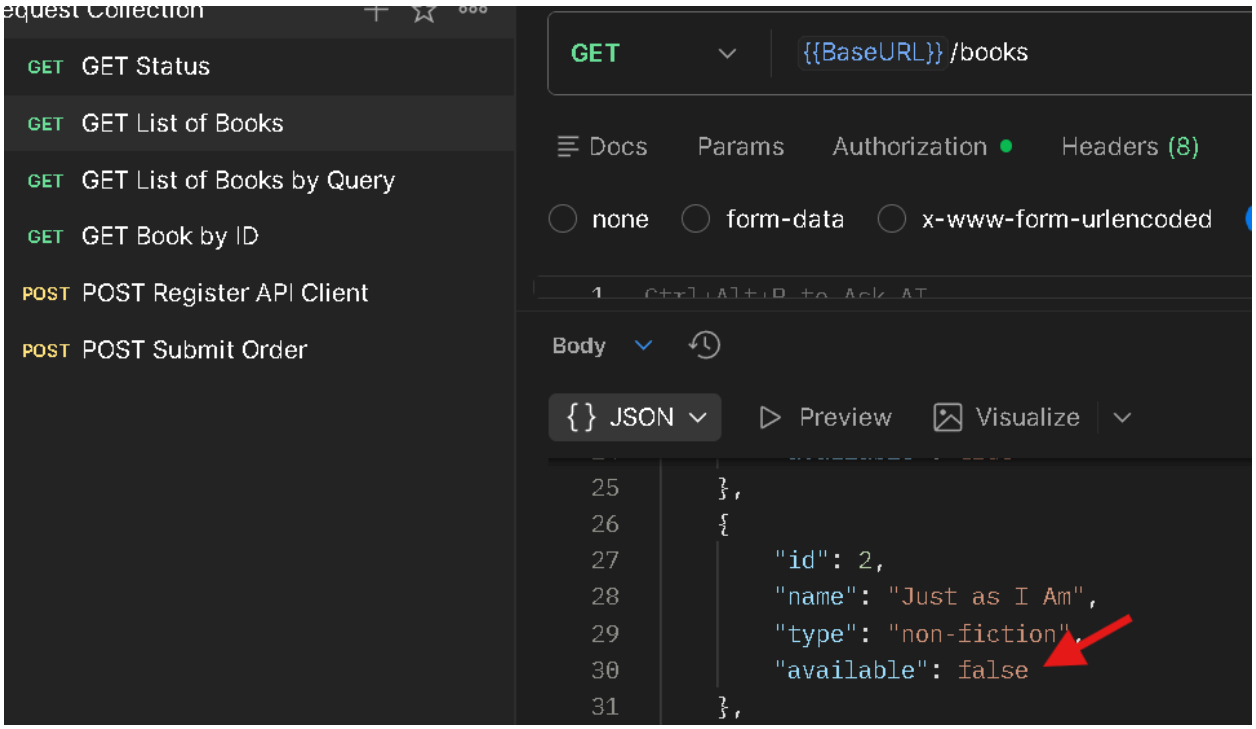


Results



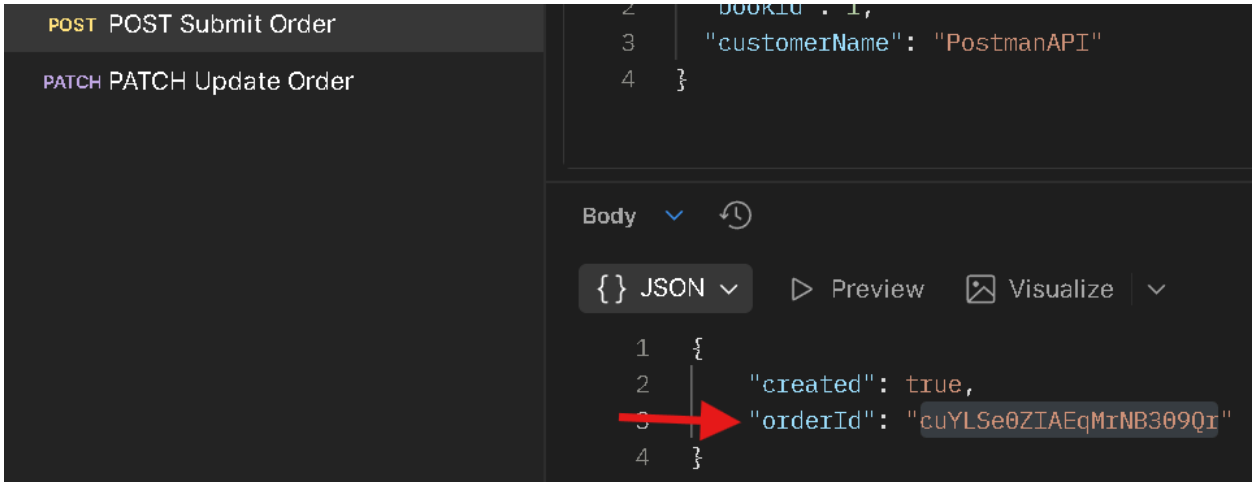
Order is created if its in the stock.
For ID = 2

Proof

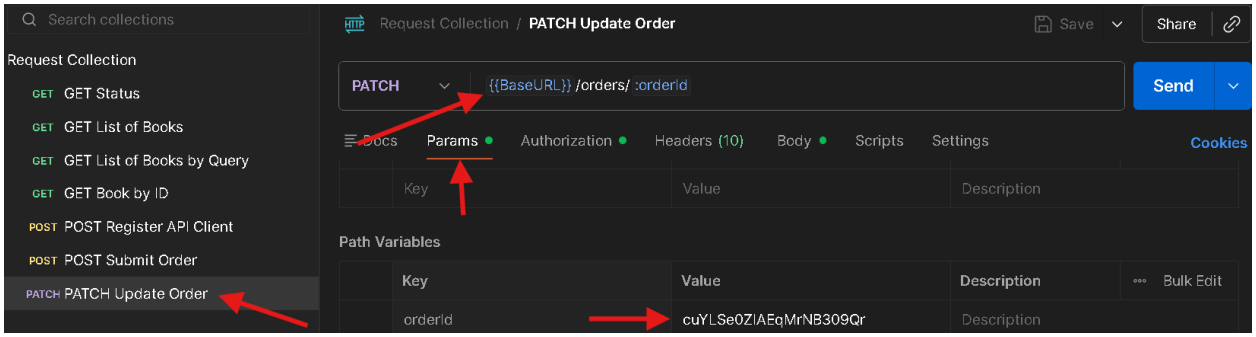


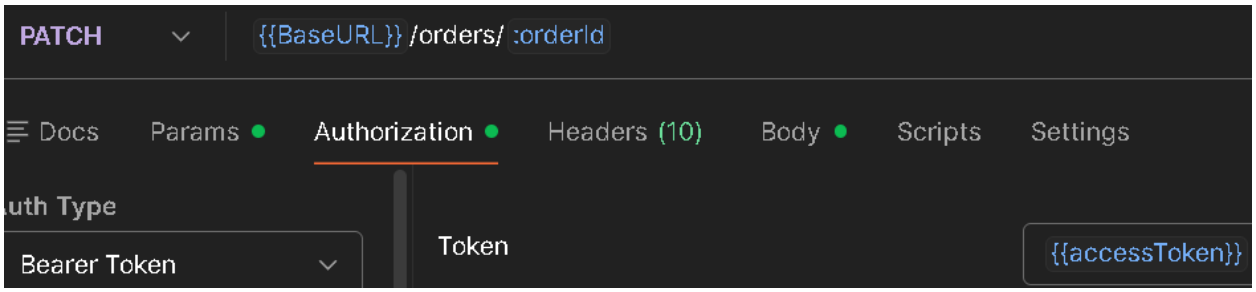
PATCH Request - Update the Order

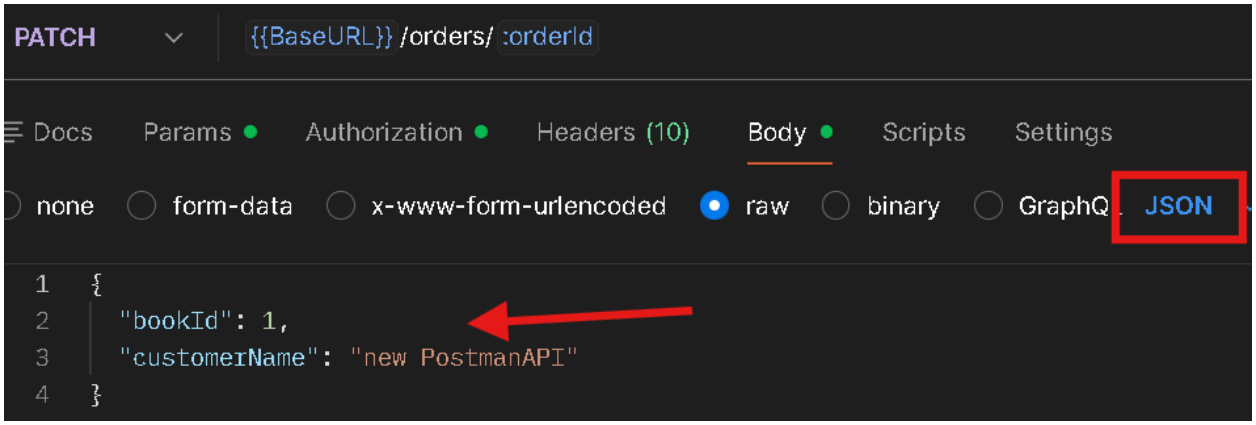
Get Token from submit order

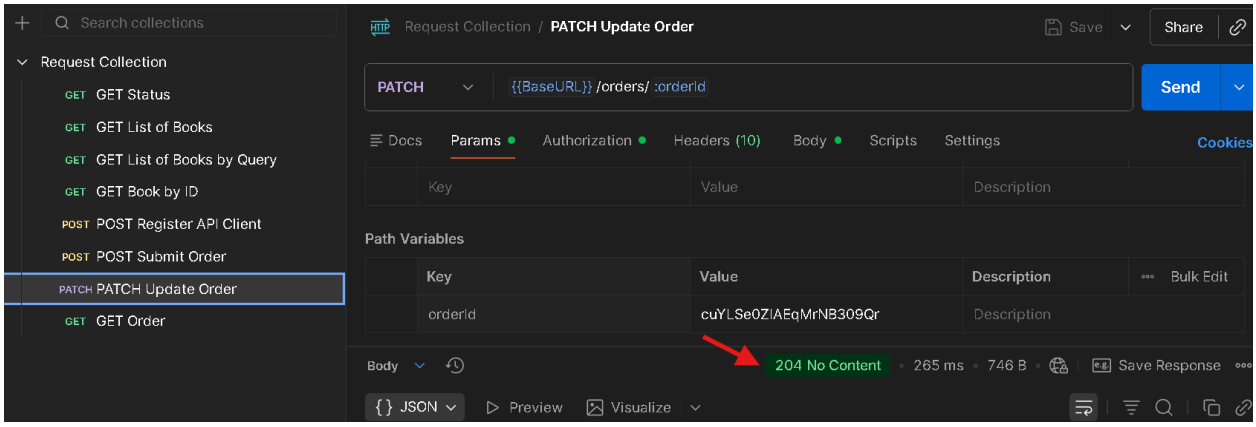


Create -> Write -> Params -> Write token

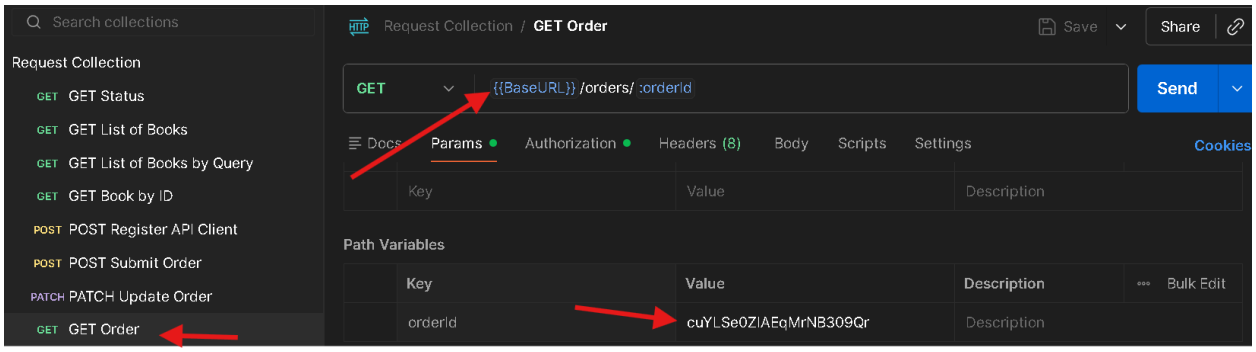


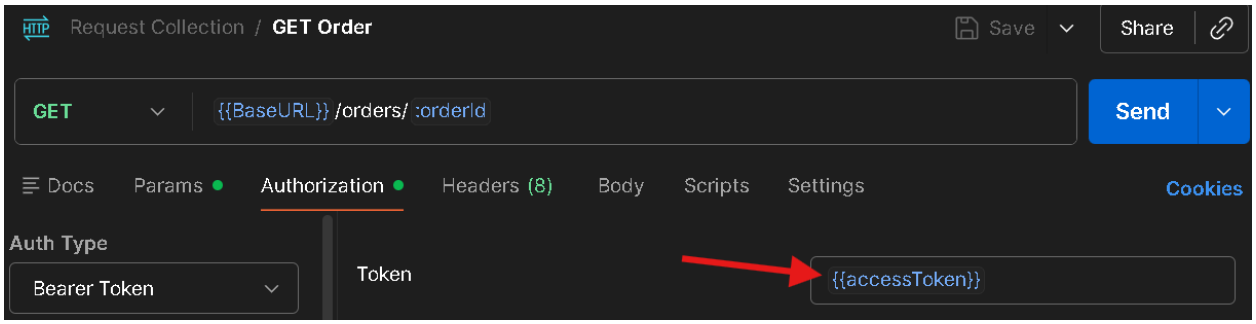




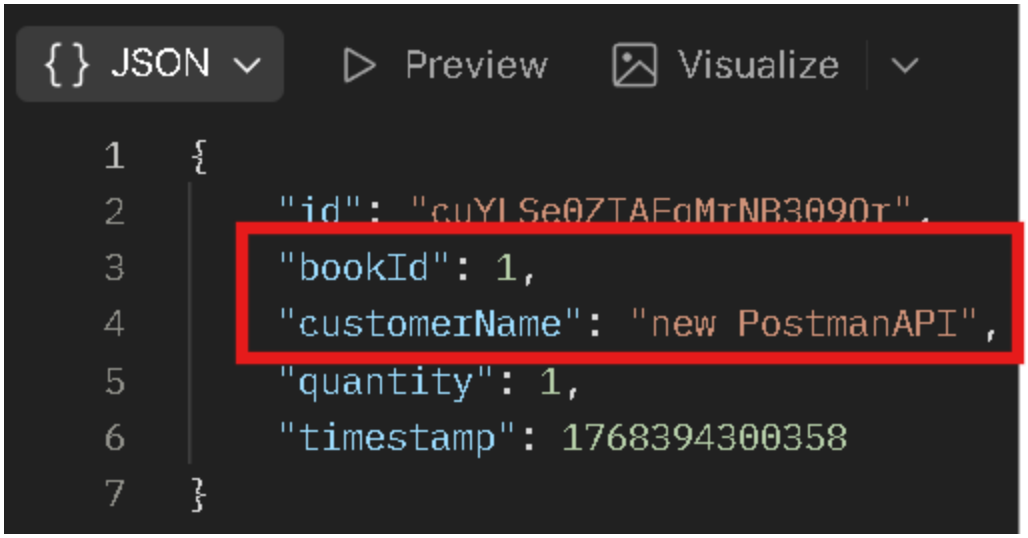


To check the order

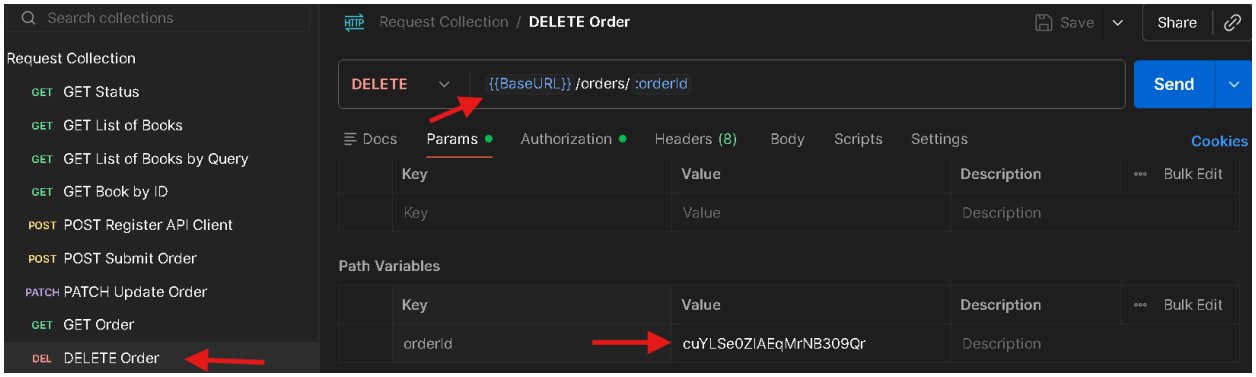


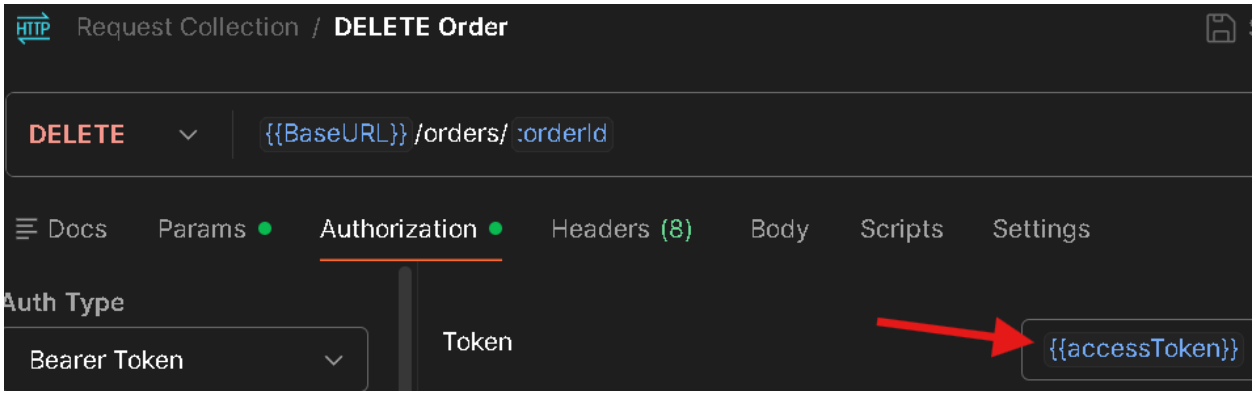


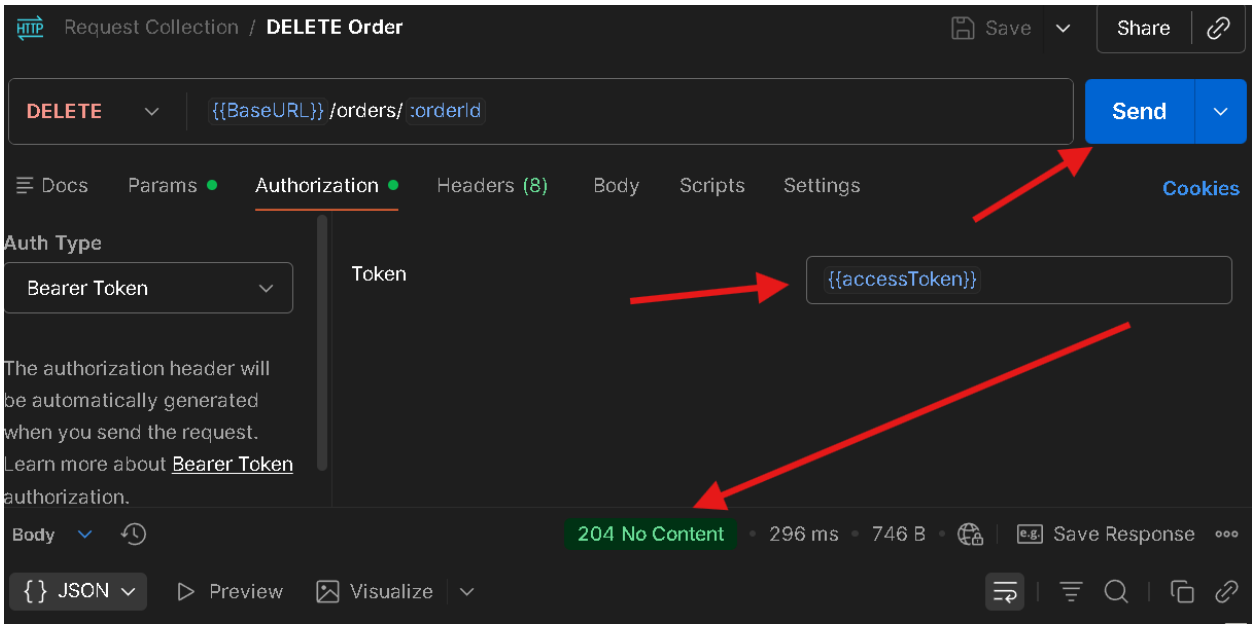
Then it actually changes



DELETE Request - Delete an Order







Check Is it deleted