

Working with Postgres. Create a book stock management page.

1. List of all books

Working locally in Scratch Pad. Switch to a Workspace

Overview

POST http://127.0.0.1:3000/book

GET http://127.0.0.1:3000/book

GET http://127.0.0.1:3000/book

DEL http://127.0.0.1:3000/book

+

...

No Environment

http://127.0.0.1:3000/book

Save

GET

http://127.0.0.1:3000/book

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (14)

Test Results

200 OK 397 ms 990 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "id": 1,
4     "bookName": "Harry Potter",
5     "bookAuthor": "JK Rowling",
6     "bookQuantity": 4,
7     "created_at": "2023-01-13T13:40:01.200Z",
8     "updated_at": "2023-01-13T13:40:01.200Z"
9   },
10  {
11    "id": 3,
12    "bookName": "The Girl with the Dragon Tattoo",
13    "bookAuthor": "Stieg Larsson",
```

Runner

Trash

2. Details of individual book

http://127.0.0.1:3000/book/1

Save

GET

http://127.0.0.1:3000/book/1

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (14)

Test Results

200 OK 408 ms 807 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 1,
3   "bookName": "Harry Potter",
4   "bookAuthor": "JK Rowling",
5   "bookQuantity": 4,
6   "created_at": "2023-01-13T13:40:01.200Z",
7   "updated_at": "2023-01-13T13:40:01.200Z"
8 }
```

Runner

Trash

Working with Postgres. Create a book stock management page.

3. Add a new book

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:3000/book`
- Method:** `POST`
- Body (JSON):**

```
{
  "bookName": "The Book Thief ",
  "bookAuthor": "Markus Zusak",
  "bookQuantity": 20
}
```
- Status:** `200 OK`, `357 ms`, `864 B`
- Response Body (JSON):**

```
{
  "message": "Book Addition successful!",
  "newBook": {
    "id": 11,
    "bookName": "The Book Thief ",
    "bookAuthor": "Markus Zusak",
    "bookQuantity": 20,
    "created_at": "2023-01-14T10:37:49.952Z"
  }
}
```

4. Update the new book

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:3000/book/11`
- Method:** `PUT`
- Body (JSON):**

```
{
  "bookName": "The Book Thief ",
  "bookAuthor": "Markus Zusak",
  "bookQuantity": 25
}
```
- Status:** `200 OK`, `199 ms`, `683 B`
- Response Body:** `Book Details Updated Successfully`

Working with Postgres. Create a book stock management page.

5. Delete the new book

http://127.0.0.1:3000/book/11

DELETE http://127.0.0.1:3000/book/11

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

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (14) Test Results 200 OK 353 ms 675 B Save Response

Pretty Raw Preview Visualize JSON

```
1 Book Deleted Successfully
```

6. Search the book by name of the Author

http://127.0.0.1:3000/searchauthor

POST http://127.0.0.1:3000/searchauthor

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "bookAuthor": "JK Rowling"
3 }
4
```

Body Cookies Headers (14) Test Results 200 OK 174 ms 815 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "id": 1,
4     "bookName": "Harry Potter",
5     "bookAuthor": "JK Rowling",
6     "bookQuantity": 4,
7     "created_at": "2023-01-13T13:40:01.200Z",
8     "updated_at": "2023-01-13T13:40:01.200Z"
9   }
10 }
```

7. Search book by name of the book name

