Guide

May 25, 2020

1 API Documentation Practice

In this workspace, your task is to practice writing documentation for the bookstore app we created earlier.

You'll soon be writing documentation for your final project (the Trivia API), after which you'll get feedback from a reviewer. You can think of this as some rudimentary practice to prepare for that.

At each step, you can compare what you've written with our own version. Of course, there isn't a single correct way to write a piece of documentation, so your version may look quite different. However, there are principles and practices you should follow in order to produce quality documentation, and we'll point this out so you can check whether you've incorporated them in what you wrote.

To get started, go ahead and move to the next page in this workspace.

1.1 Getting started

Now, add a Getting Started section to your documentation. Remember, this should include at least your base URL and an explanation of authentication. Feel free to provide other information that is relevant for your API

...

When you're ready, you can compare it with our version.

1.1.1 Getting Started

- Base URL: At present this app can only be run locally and is not hosted as a base URL. The backend app is hosted at the default, `http://127.0.0.1:5000/`, which is set as a proxy in the frontend configuration.
- Authentication: This version of the application does not require authentication or API keys.

1.2 Error Handling

Now, add an Error Handling section to your documentation. It should include the format of the error responses the client can expect as well as which status codes you use.

...

When you're ready, you can compare it with our version.

1.2.1 Error Handling

Errors are returned as JSON objects in the following format:

```
{
    "success": False,
    "error": 400,
    "message": "bad request"
}
```

The API will return three error types when requests fail: - 400: Bad Request - 404: Resource Not Found - 422: Not Processable

1.3 Endpoint Library

Now, add an Endpoint Library section to your documentation. Make sure that endpoints, methods and returned data are all clear. Consider including sample requests for clarity

...
When you're ready, you can compare it with our version.

GET /books

- General:
  - Returns a list of book objects, success value, and total number of books
  - Results are paginated in groups of 8. Include a request argument to choose page number, starting from 1.
- Sample: `curl http://127.0.0.1:5000/books`

```
"books": [
  {
    "author": "Stephen King",
    "id": 1,
    "rating": 5,
    "title": "The Outsider: A Novel"
  },
  {
    "author": "Lisa Halliday",
    "id": 2,
    "rating": 5,
    "title": "Asymmetry: A Novel"
  },
  {
    "author": "Kristin Hannah",
    "id": 3,
    "rating": 5,
    "title": "The Great Alone"
  },
  {
```

```
      "author": "Tara Westover",
      "id": 4,
      "rating": 5,
      "title": "Educated: A Memoir"
    },
    {
      "author": "Jojo Moyes",
      "id": 5,
      "rating": 5,
      "title": "Still Me: A Novel"
    },
    {
      "author": "Leila Slimani",
      "id": 6,
      "rating": 5,
      "title": "Lullaby"
    },
    {
      "author": "Amitava Kumar",
      "id": 7,
      "rating": 5,
      "title": "Immigrant, Montana"
    },
    {
      "author": "Madeline Miller",
      "id": 8,
      "rating": 5,
      "title": "CIRCE"
    }
  ],
"success": true,
"total_books": 18
}
```


POST /books

- General:
  - Creates a new book using the submitted title, author and rating. Returns the id of the created book, success value, total books, and book list based on current page number to update the frontend.

- `curl http://127.0.0.1:5000/books?page=3 -X POST -H "Content-Type: application/json" -d '{"title":"Neverwhere", "author":"Neil Gaiman", "rating":"5"}'`


```
{
  "books": [
```

```
{
  "author": "Neil Gaiman",
  "id": 24,
  "rating": 5,
  "title": "Neverwhere"
}
  ],
  "created": 24,
  "success": true,
  "total_books": 17
}
```


- General:

  - Deletes the book of the given ID if it exists. Returns the id of the deleted book, success value, total books, and book list based on current page number to update the frontend.

- `curl -X DELETE http://127.0.0.1:5000/books/16?page=2`


```
{
  "books": [
{
  "author": "Gina Apostol",
  "id": 9,
  "rating": 5,
  "title": "Insurrecto: A Novel"
},
{
  "author": "Tayari Jones",
  "id": 10,
  "rating": 5,
  "title": "An American Marriage"
},
{
  "author": "Jordan B. Peterson",
  "id": 11,
  "rating": 5,
  "title": "12 Rules for Life: An Antidote to Chaos"
},
{
  "author": "Kiese Laymon",
  "id": 12,
  "rating": 1,
  "title": "Heavy: An American Memoir"
},
```

```
{
  "author": "Emily Giffin",
  "id": 13,
  "rating": 4,
  "title": "All We Ever Wanted"
},
{
  "author": "Jose Andres",
  "id": 14,
  "rating": 4,
  "title": "We Fed an Island"
},
{
  "author": "Rachel Kushner",
  "id": 15,
  "rating": 1,
  "title": "The Mars Room"
}
  ],
  "deleted": 16,
  "success": true,
  "total_books": 15
}
```


- General:

  - If provided, updates the rating of the specified book. Returns the success value and id of the modified book.

- `curl http://127.0.0.1:5000/books/15 -X PATCH -H "Content-Type: application/json" -d '{"rating":"1"}'` “‘ { "id": 15, "success": true }