Documentation

What is an API?

An API is — in short — a set of dedicated URLs that return pure data responses, in most cases in JSON format — meaning the responses won't contain the kind of presentational overhead that you would expect in a graphical user interface like a website. *I took that great definition from this very understandable article. Please refer to it for more information!*

I'm a total newbie to REST APIs. Is there a beginner-friendly introduction?

Yes, I wrote a blog post that explains in detail how REST APIs work and why you would use them at all. It also covers authentication, JSON handling and a sample React app. This is the link to the blog post and here you will find the according code base for the sample React app.

Which data does the "one API to rule them all" provide?

Well, if you don't know or neither like "The Lord of the Rings", the epic masterpiece epos by J.R.R. Tolkien, then this API is most likely not for you. But if you do, this massive database will provide you with information about the books, the movie trilogy, many characters and quotes. You are welcome to use the data in your own apps, mixups and (fun) projects — like I did with creating this API.

This project is totally non-profit and fan-made!

What about response formats and authentication?

The response format for all datasets is JSON. The API requires an access key for most routes. You can obtain an access token by signing up for an account here. All you need for setting up an account is a valid email address.

You need to send the access key as a bearer token in every request you make to the api. Bearer tokens must be included in the authorization header (More information on authorization headers?) in the following format:

```
Authorization: Bearer your-api-key-123
```

Access for authenticated users to all endpoints is *limited* to 100 requests every 10 minutes. Be fair!

Which routes are available?

All routes must be prefixed with **https://the-one-api.dev/v2**. Only the /book endpoint is available without authentication.

Endpoint	Response	Token required

/book	List of all "The Lord of the Rings" books	no
/book/{id}	Request one specific book	no
/book/{id}/chapter	Request all chapters of one specific book	
/movie	List of all movies, including the "The Lord of the Rings" and the "The Hobbit" trilogies	yes
/movie/{id}	Request one specific movie	yes
/movie/{id}/quote	Request all movie quotes for one specific movie (only working for the LotR trilogy)	yes
/character	List of characters including metadata like name, gender, realm, race and more	yes

/character/{id}	Request one specific character	yes
/character/{id}/quote	Request all movie quotes of one specific character	yes
/quote	List of all movie quotes	yes
/quote/{id}	Request one specific movie quote	yes
/chapter	List of all book chapters	yes
/chapter/{id}	Request one specific book chapter	yes

May I use pagination, sorting and filtering?

Yes, you can add pagination, sorting and filtering options to your API requests.

Pagination

Option	Example
limit	/character?limit=100
page	/character?page=2 (limit default is 10)
offset	/character?offset=3 (limit default is 10)

Sorting

Examples
/character?sort=name:asc
/quote?sort=character:desc

Filtering

The filtering works by casting simple url parameter expressions to mongodb lookup expressions and can be applied to any available key on the data models.

Option	Example	
match, negate match	/character? name=Gandalf	/character? name!=Frodo
include, exclude	/character? race=Hobbit,Human	/character? race!=Orc,Goblin
exists, doesn't exists	/character?name	/character?!name
regex	/character? name=/foot/i	/character? name!=/foot/i
less than, greater than or equal to	/movie?budgetInMillions<100 /movie?academyAwardWins>0 /movie?runtimeInMinutes>=160	

All we have to decide is what to do with the time that is given to us. **Built**