**API Exercise**

**Part 1: SWAPI**

1. the height of Darth Vader ***(cm)***

"height": "202",

1. the population of the planet Alderaan

"population": "2000000000",

1. the name of the manufacturer of the Millennium Falcon

"manufacturer": "Corellian Engineering Corporation",

1. the name of the species that C-3PO belongs to (multiple URLs)

"species": [

"<https://swapi.dev/api/species/2/>"

]

1. the title of each film that Obi-Wan Kenobi is in (multiple URLs)

"films": [

"<https://swapi.dev/api/films/1/>",

"<https://swapi.dev/api/films/2/>",

"<https://swapi.dev/api/films/3/>",

"<https://swapi.dev/api/films/4/>",

"<https://swapi.dev/api/films/5/>",

"<https://swapi.dev/api/films/6/>"

],

1. use the search query (the how to on the search query is at the bottom of the Getting Started section of the documentation) to get the information about the Millennium Falcon, it’s a starship

**GET** /api/starships/?search=millennium

**HTTP 200 OK**

**Content-Type:** application/json

**Vary:** Accept

**Allow:** GET, HEAD, OPTIONS

{

"count": 1,

"next": null,

"previous": null,

"results": [

{

"name": "Millennium Falcon",

"model": "YT-1300 light freighter",

"manufacturer": "Corellian Engineering Corporation",

"cost\_in\_credits": "100000",

"length": "34.37",

"max\_atmosphering\_speed": "1050",

"crew": "4",

"passengers": "6",

"cargo\_capacity": "100000",

"consumables": "2 months",

"hyperdrive\_rating": "0.5",

"MGLT": "75",

"starship\_class": "Light freighter",

"pilots": [

"<https://swapi.dev/api/people/13/>",

"<https://swapi.dev/api/people/14/>",

"<https://swapi.dev/api/people/25/>",

"<https://swapi.dev/api/people/31/>"

],

"films": [

"<https://swapi.dev/api/films/1/>",

"<https://swapi.dev/api/films/2/>",

"<https://swapi.dev/api/films/3/>"

],

"created": "2014-12-10T16:59:45.094000Z",

"edited": "2014-12-20T21:23:49.880000Z",

"url": "<https://swapi.dev/api/starships/10/>"

}

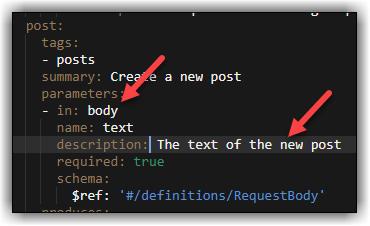
]

}

**Part 2: Social Mountain**

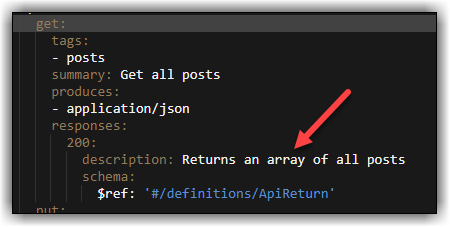
1. Check if the POST request accept params, queries, and/or a body. Which one(s) and what information is it expecting to be sent?

-The post request accepts a body with a description with the text of the new post



1. What data type does the GET request return?

-The GET request returns an array data type.



1. What would the URL look like for deleting the post with the id 555? (This post does not exist anymore, but the syntax is the same for existing posts, )



1. List the possible response codes from the GET request at ‘/posts/filter’

-Two response codes of 200 and 409.

Graphical user interface, application

Description automatically generated

1. Create a post whose text is your name, record the URL and body here:

-URL: https://practiceapi.devmountain.com/api/posts

-Body: {

"id": 980,

"text": "JBS",

"date": "15 Aug 2022"

}

Text

Description automatically generated

1. What would the URL and body object be to update the post you just made to contain your faovrite color instead of your name?

-URL: https://practiceapi.devmountain.com/api/posts

-Body: {

"id": 980,

"text": "JBS",

"date": "15 Aug 2022"

}

Graphical user interface, text, application

Description automatically generated

1. What is the URL to get posts that contain the text “blue”?

-The URL is: <https://practiceapi.devmountain.com/api/posts/filter?text=blue>. But there were none at the time, see screenshot.

Graphical user interface, application

Description automatically generated

1. Make a request to GET all the posts. What are the content type and charset of the response? (Hint: look on the Headers)

The content-type is application/json; the charset is utf-8

Text

Description automatically generated

1. What would cause a PUT request to return a 409 status?

A missing req.query.id or req.body.text

A picture containing text

Description automatically generated

1. What happens if you try to send a query in the GET request URL? Why do you get that response?

You will get a filtered response with whatever contains that query. Because its filtered.