Asteroids - NeoWs Rest Api

1. What does this rest api?
2. Frequently Asked Questions (FAQ)
3. Installation instructions
4. Explain what does the endpoint
5. Project’s license

1. What does this Rest API ?

NASA has many open API. One of them is called NeoWs.

Its job is to provide information about objects approaching the world.

This is where the rest API I wrote comes into play.

Shredding the data returned by NASA's API provides more meaningful and more accessible data.

2. Frequently Asked Questions (FAQ)

- Why this API works slowly if someone requests more than 2 days of data?

  It seems that NASA's API takes 8 seconds to respond to a data period of 4 days.

  There's not much I can do about NASA's API being slow.

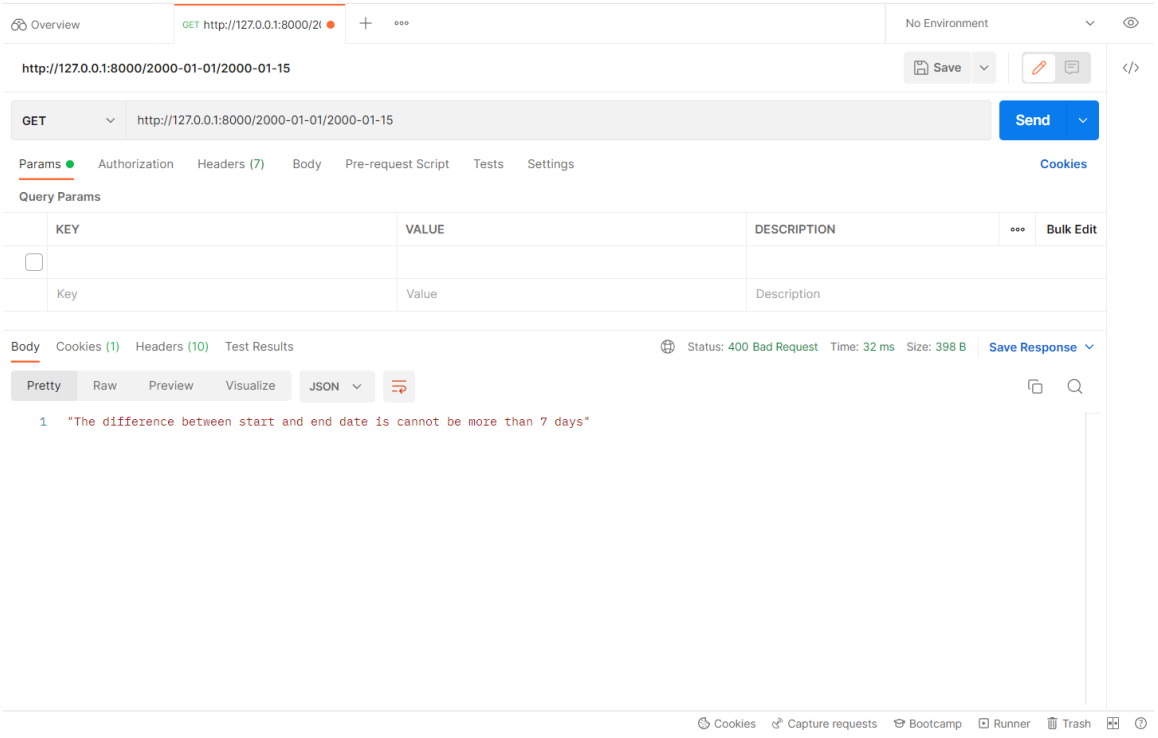
  But I'm caching data from the same date range for an hour after the request comes in.

  So subsequent requests for the same range would then be near-instant.

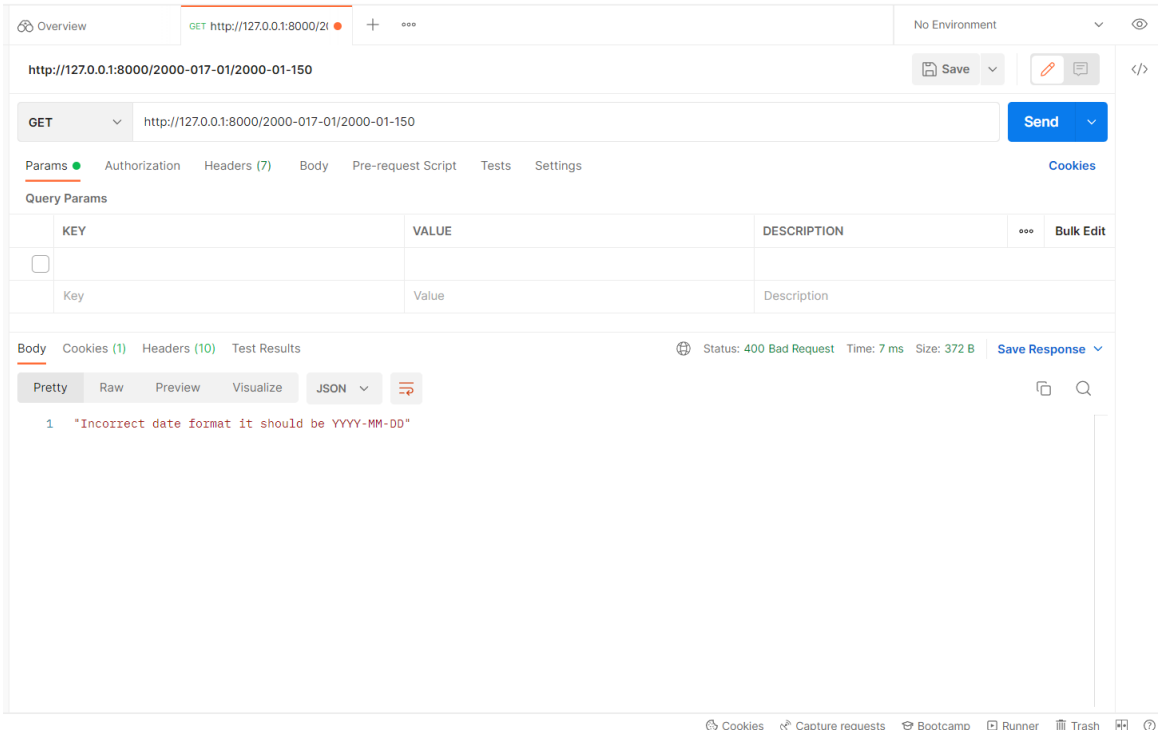
- Why I am getting a HTTP 400 Bad Request Error?

So there is a couple of reason that you are getting HTTP 400 Bad Request.

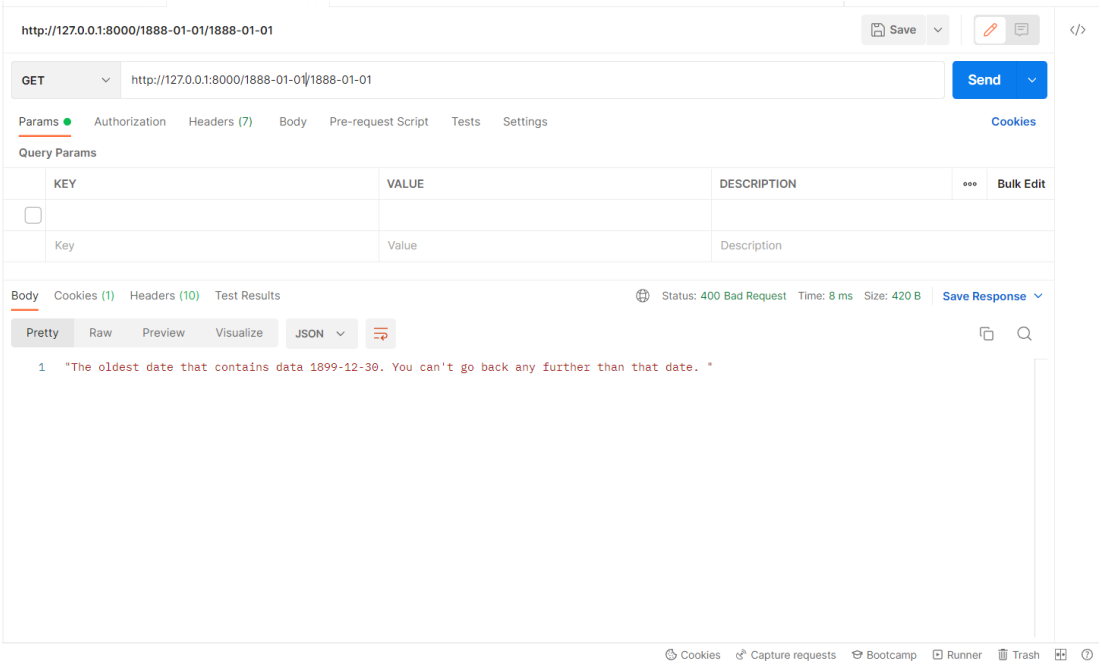
- So maybe you are requesting more than 7 days of data.



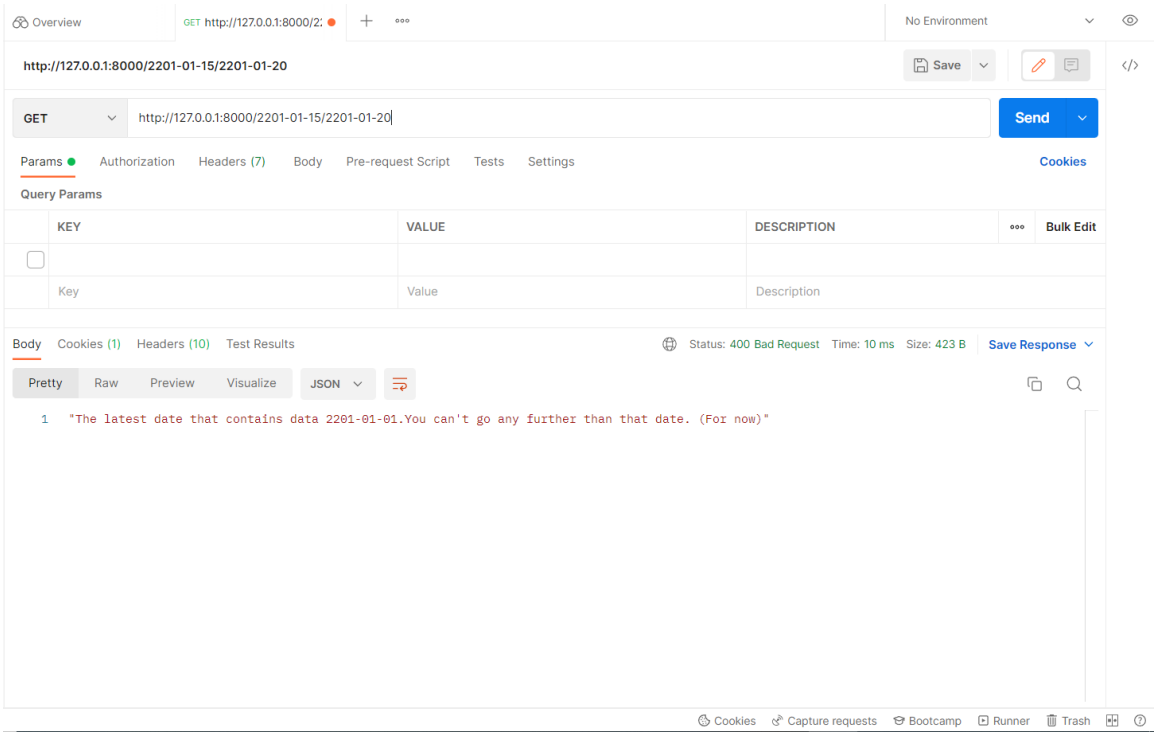
- Or maybe you entering the date format wrongly it should be (YYYY-MM-DD)



- Or maybe the date you are requesting date very old date that Nasa doesn't have data about that date.

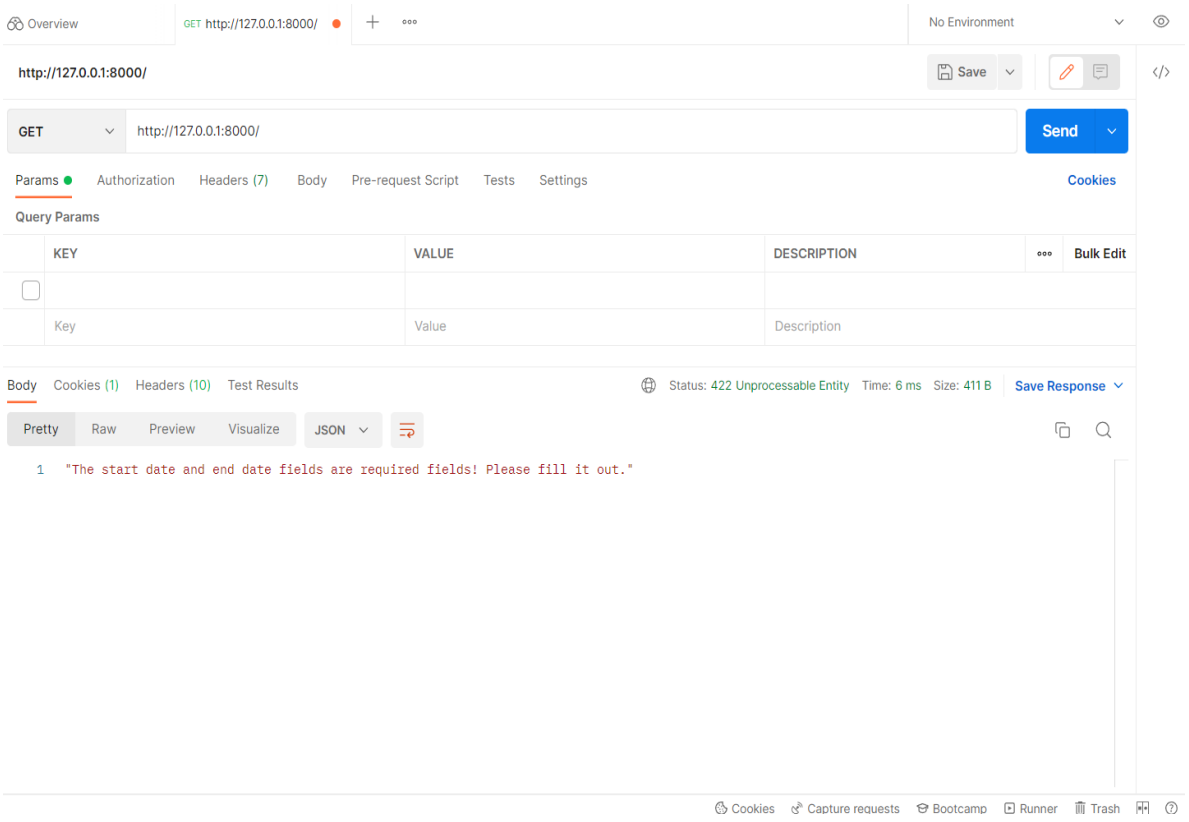


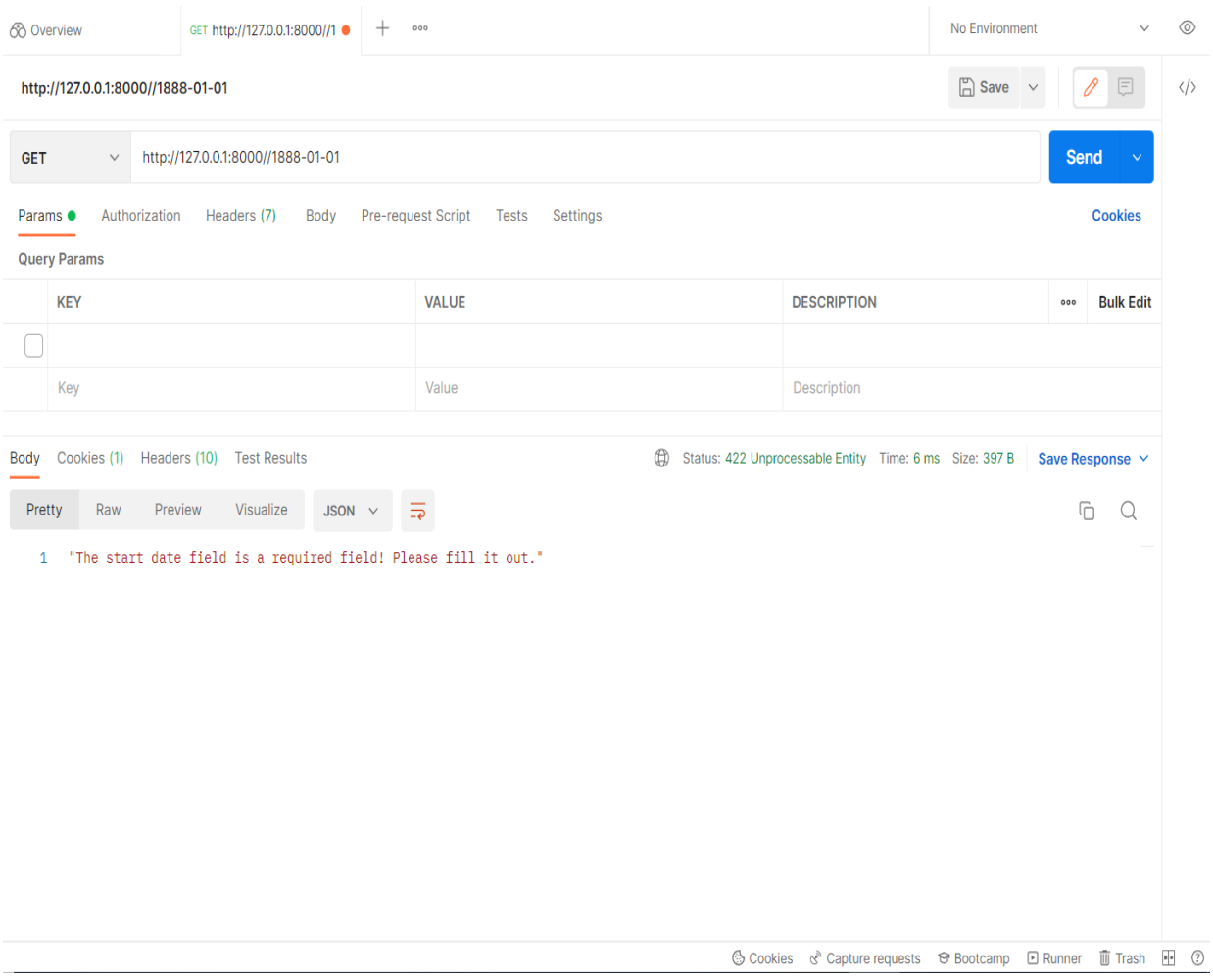
The same thing counts also for a very late date that Nasa doesn't have data about this date.

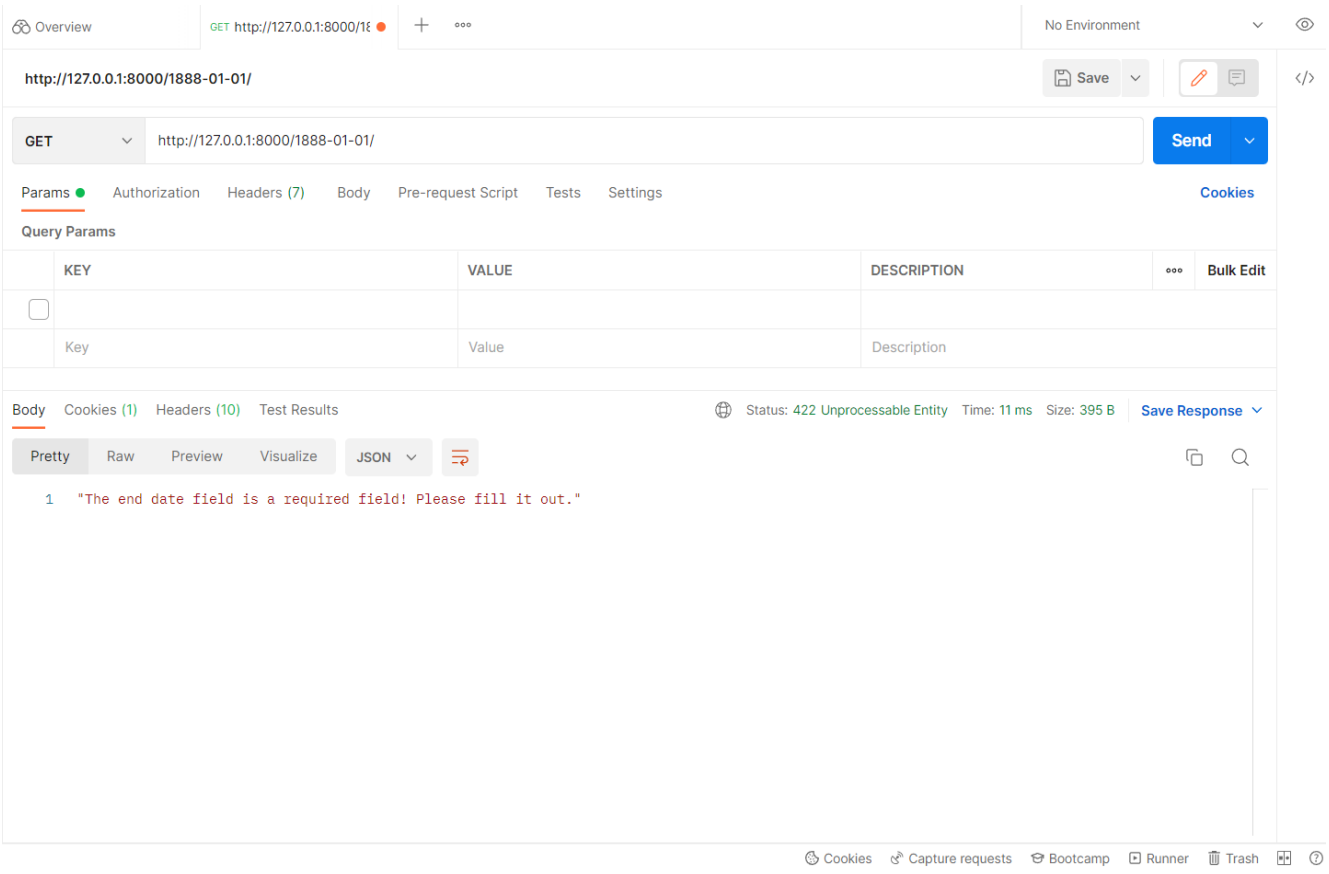


- Why I am getting a HTPP 422 UNPROCESSABLE ENTITY Error?

Because you probably didn't enter the start and end date. Or one of them.







3. Installation instructions

- Virtualenv environment

1.Clone the repo

```
git clone https://github.com/Muhammed-Kahraman/Nasa_near_earth_objects_api.git
```

2. cd into the new directory

```
cd Nasa_near_earth_objects_api
```

3.Create a new virtual environment env in the directory

```
python -m virtualenv env
```

4.Activate the new environment

```
.\env\Scripts\activate
```

5.Install dependencies in new environment

```
pip install -r requirements.txt
```

6.Create a .env file same directory with the project.

Set up secret_key and nasa apikey inside this file (secret_key = "", apiKey = "")

7.Run the server locally

```
python manage.py migrate
```

```
python manage.py makemigrations
```

```
python manage.py runserver
```

Docker build

```
docker compose up –build
```

4. Explain what does the endpoint ?

This endpoint requires 3 parameters. 2 of them are given by the user And these parameters are required parameters (start_date and end_date).

One of them is arranged by the Django framework (request) ,

And returns a JSON array.

```
@api_view(['GET'])
def getDates(request, start_date="", end_date="")
```

Inside of the endpoints, I am checking the wrong date format, 7 seven days limit, Missing date parameter, and Date range.

```
# If start_date or end_date is not provided, we are returning the error message.
if start_date == "" or end_date == "":
    # splitting  the errors more specific way.
    if start_date == "" and end_date != "":
        return Response("The start date field is a required field! Please fill it out.",
                        status=status.HTTP_422_UNPROCESSABLE_ENTITY)
    elif start_date != "" and end_date == "":
        return Response("The end date field is a required field! Please fill it out.",
                        status=status.HTTP_422_UNPROCESSABLE_ENTITY)
else:
    return Response("The start date and end date fields are required fields! Please fill it out.",
                    status=status.HTTP_422_UNPROCESSABLE_ENTITY)

date_format = "%Y-%m-%d"
oldest_date_contains_data = datetime.datetime.strptime('1899-12-30', date_format)
latest_date_contains_data = datetime.datetime.strptime('2201-01-01', date_format)
# Checking if the start and end date are in valid format.
start_date = datetime.datetime.strptime(start_date, date_format)
end_date = datetime.datetime.strptime(end_date, date_format)
if start_date.day - end_date.day > 7 or end_date.day - start_date.day > 7:
    return Response('The difference between start and end date is cannot be more than 7 '
                    'days',
                    status=status.HTTP_400_BAD_REQUEST)
    # Checking if the start and end date are in valid format.
```

```
    # Checking start date or end date not equals the oldest date.
if start_date != oldest_date_contains_data or end_date != oldest_date_contains_data:

    # Checking Is the start date older than the oldest date.
    if start_date.year <= oldest_date_contains_data.year \
            and start_date.month <= oldest_date_contains_data.month \
            and start_date.day <= oldest_date_contains_data.day:

        return Response("The oldest date that contains data 1899-12-30."
                        " You can't go back any further than that date. ",
                        status=status.HTTP_400_BAD_REQUEST)

    # Checking Is the end date older than the oldest date.
    elif end_date.year <= oldest_date_contains_data.year \
            and end_date.month <= oldest_date_contains_data.month \
            and end_date.day <= oldest_date_contains_data.day:

        return Response("The oldest date that contains data 1899-12-30."
                        " You can't go back any further than that date. ",
                        status=status.HTTP_400_BAD_REQUEST)

    # Checking Is the start date later than the latest date.
    if start_date != latest_date_contains_data or end_date != latest_date_contains_data:
        if start_date.year >= latest_date_contains_data.year \
                and start_date.month >= latest_date_contains_data.month \
```

```
            and start_date.month >= latest_date_contains_data.month \
            and start_date.day >= latest_date_contains_data.day:
        return Response("The latest date that contains data 2201-01-01."
                        "You can't go any further than that date. (For now)",
                        status=status.HTTP_400_BAD_REQUEST)

    # Checking Is the end date later than the latest date.
    elif end_date.year >= latest_date_contains_data.year \
            and end_date.month >= latest_date_contains_data.month \
            and end_date.day >= latest_date_contains_data.day:
        return Response("The latest date that contains data 2201-01-01."
                        "You can't go any further than that date. (For now)",
                        status=status.HTTP_400_BAD_REQUEST)

    # Checking if the start and end date are in valid format.
    if start_date.day - end_date.day > 7 or end_date.day - start_date.day > 7:
        return Response('The difference between start and end date is cannot be more than 7 '
                        'days',
                        status=status.HTTP_400_BAD_REQUEST)
    # Checking if the start and end date are in valid format.
    # Checking start date or end date not equals the oldest date.
    if start_date != oldest_date_contains_data or end_date != oldest_date_contains_data:

        # Checking Is the start date older than the oldest date.
        if start_date.year <= oldest_date_contains_data.year \
```

```
if start_date.year <= oldest_date_contains_data.year \
        and start_date.month <= oldest_date_contains_data.month \
        and start_date.day <= oldest_date_contains_data.day:

    return Response("The oldest date that contains data 1899-12-30."
                    " You can't go back any further than that date. ",
                    status=status.HTTP_400_BAD_REQUEST)

# Checking Is the end date older than the oldest date.
elif end_date.year <= oldest_date_contains_data.year \
        and end_date.month <= oldest_date_contains_data.month \
        and end_date.day <= oldest_date_contains_data.day:

    return Response("The oldest date that contains data 1899-12-30."
                    " You can't go back any further than that date. ",
                    status=status.HTTP_400_BAD_REQUEST)

# Checking Is the start date later than the latest date.
if start_date != latest_date_contains_data or end_date != latest_date_contains_data:
    if start_date.year >= latest_date_contains_data.year \
            and start_date.month >= latest_date_contains_data.month \
            and start_date.day >= latest_date_contains_data.day:
        return Response("The latest date that contains data 2201-01-01."
                        "You can't go any further than that date. (For now)",
                        status=status.HTTP_400_BAD_REQUEST)
```

```
 # Checking Is the start date later than the latest date.
 if start_date != latest_date_contains_data or end_date != latest_date_contains_data:
     if start_date.year >= latest_date_contains_data.year \
             and start_date.month >= latest_date_contains_data.month \
             and start_date.day >= latest_date_contains_data.day:
         return Response("The latest date that contains data 2201-01-01."
                         "You can't go any further than that date. (For now)",
                         status=status.HTTP_400_BAD_REQUEST)

     # Checking Is the end date later than the latest date.
     elif end_date.year >= latest_date_contains_data.year \
             and end_date.month >= latest_date_contains_data.month \
             and end_date.day >= latest_date_contains_data.day:
         return Response("The latest date that contains data 2201-01-01."
                         "You can't go any further than that date. (For now)",
                         status=status.HTTP_400_BAD_REQUEST)
```

And if parameters passed every validation. First I am checking whether this date range is inside the cache memory or not.

```
# if the both date valid then we can proceed.
if is_valid_date:
    url_neo_feed = "https://api.nasa.gov/neo/rest/v1/feed?"
    # Creating a cache key and looking for data inside cache memory.
    cache_key = f'nasa_neo_{start_date.strftime("%Y-%m-%d")}_{end_date.strftime("%Y-%m-%d")}'
    json_data = cache.get(cache_key)
```

If is not I am making a new request and caching the coming data (for one hour).

```
# If data is not found in cache then we are making a new request to the API.
if not json_data:
    response = requests.get(url_neo_feed, params={
        'api_key': api_key,
        'start_date': start_date,
        'end_date': end_date
    })
    response.raise_for_status()
    json_data = orjson.loads(response.text)

# After getting the data from the API we are storing it in cache memory.
cache.set(cache_key, json_data, timeout=4000)
```

After that, I am shredding data for getting more understandable and more accessible data. And adding a dictionary

```
date_asteroids = json_data['near_earth_objects']

# Iterating over the dates and add to them a list one by one.
for date in date_asteroids:
    dates.append(date)

# Getting the data for each date.
for date in dates:
    collection = json_data.get('near_earth_objects')
    unsplit_data = collection.get('{}'.format(date))

    # Iterating over the data for getting specific data each asteroid.
    for list_data in unsplit_data:
        name = list_data.get('name')
        closes_date = list_data.get('close_approach_data')
        estimated_diameter = list_data.get('estimated_diameter')
        estimated_diameter_kilometers = estimated_diameter.get('kilometers')
        json_dict = {
            'name': name,
            'closest_date': closes_date[0].get('close_approach_date_full'),
            'miss_distance_km': closes_date[0].get('miss_distance').get('kilometers'),
            'estimated_diameter_km': estimated_diameter_kilometers
        }
        all_data.append(json_dict)
```

and for the last checking is the dictionary empty or not if is empty I am returning a 404 not found error. İf is not empty I am sorting the data according to the miss distance and I am returning the data to the user.

```
# Checking if the data is empty or not. If empty then return HTTP_404_NOT_FOUND.
# İf not empty then return HTTP_200_OK
if all_data is not None:
    sorted_all_data = Sort(all_data)
    return Response(sorted_all_data, status=status.HTTP_200_OK)

# If data is empty then return HTTP_404_NOT_FOUND.
else:
    return Response("Oops! Something went wrong. But don't worry we are working on it.",
                    status=status.HTTP_404_NOT_FOUND)
```

5. Project’s license

```
The project is licensed under the MIT license.
```