1 Choix BDD

Mon choix s'est porté sur les données suivantes:

https ://data.world/datafiniti/electronic-products-and-pricing-data

Après pré-processing détaillé dans le rapport ci-joint, toutes les lignes de ce jeu de données ont pu être intégrées avec succès à l'aide d'un index. A l'aide d'ElasticSearch des requêtes ont pu être testées. La rapidité des réponses et leur pertinence ont contribuées au choix de cette BDD. De plus, la possibilité de faire une API à l'aide de fastApi pour requêter, insérer et peupler la BDD ont conforté ce choix. De plus, Elasticsearch fonctionne à l'aide de docker, le déploiement est donc possible avec docker-compose.

2 FastApi

2.1 Lancement

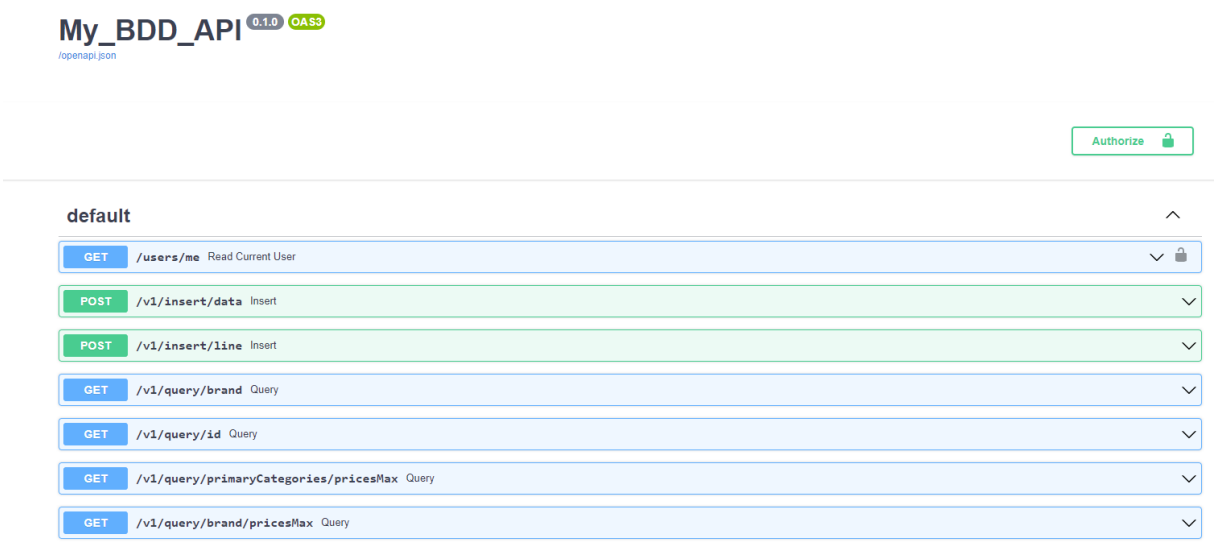
En se plaçant dans le répertoire « fastApi_SNICOLE » ci-joint , il faut lancer ElasticSearch par la commande : « docker-compose up -d ». Il faut patienter un peu et lancer la commande pour s'assurer que tout fonctionne.
« curl -X GET "localhost:9200" »

En restant dans le répertoire « fastApi_SNICOLE » il faut ensuite lancer l'api par la commande :

« uvicorn app.main:api --reload »:

```
ubuntu@ip-172-31-33-27:~/fastApi_SNICOLE$ docker-compose up -d
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "fastapi_snicole_default" with the default driver
Creating elasticsearch ... done
ubuntu@ip-172-31-33-27:~/fastApi_SNICOLE$ curl -X GET "localhost:9200"
{
  "name" : "9187d9f9a93f",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "H2CU4GVyQuaQfwgQBNcc6Q",
  "version" : {
    "number" : "7.10.1",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "1c34507e66d7db1211f66f3513706fdf548736aa",
    "build_date" : "2020-12-05T01:00:33.671820Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
ubuntu@ip-172-31-33-27:~/fastApi_SNICOLE$ uvicorn app.main:api --reload
INFO:     Will watch for changes in these directories: ['/home/ubuntu/fastApi_SNICOLE']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [518184] using statreload
INFO:     Started server process [518186]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60144 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:60144 - "GET /openapi.json HTTP/1.1" 200 OK
```

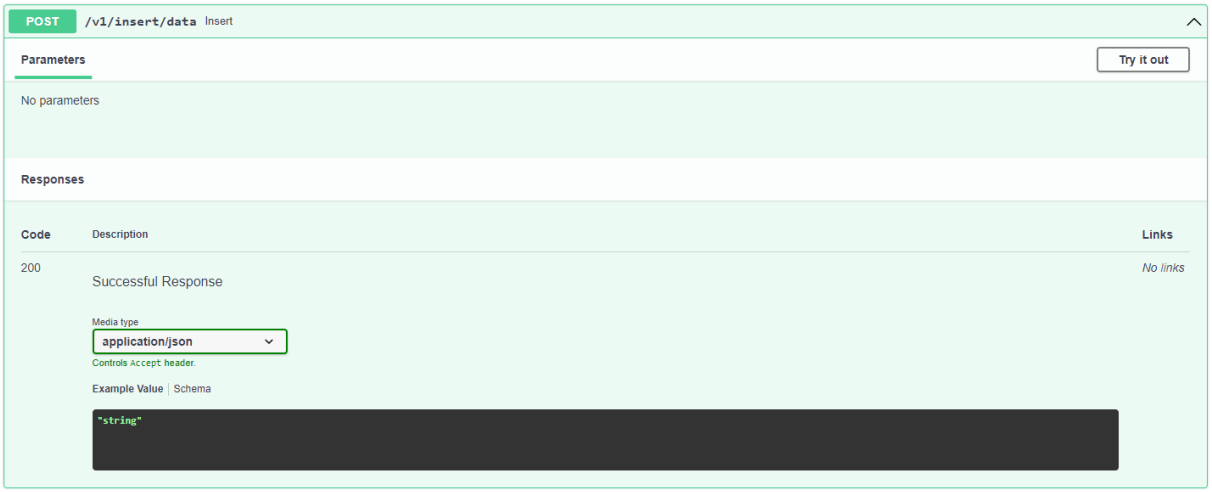
Ouvrez l'Api grâce à ce lien : http://localhost:8000/docs :



2.2 Présentation des routes

2.2.1 "/v1/insert/data

Cette route permet de peupler la base de données à l'aide du jeu de données.



2.2.2 « /v1/insert/line

Cette route permet d'insérer une ligne où les champs sont séparés par une virgule (voir dossier ElasticSearch ci-joint pour plus de détail sur l'index à respecter) à l'aide du script modifiable « test.sh » , un exemple ci-dessous :

```
curl -X POST "localhost:9200/test1_ingest/_doc" \
 -H "Content-Type: application/json" \
 -d "{\"csv_line\": \"AVphzgbJlJeJML391983|1500.0|15000.0|Yes|New|USD|False|Bestbuy.com|800C78VIUE|Sony|Sony VLF410B1 10-Inch Super Slim Full-Motion Mount for 37 - 84 Inches TV's|Electronics|7.94E+11\"}" | jq
```

2.2.3 « /v1/insert/brand »

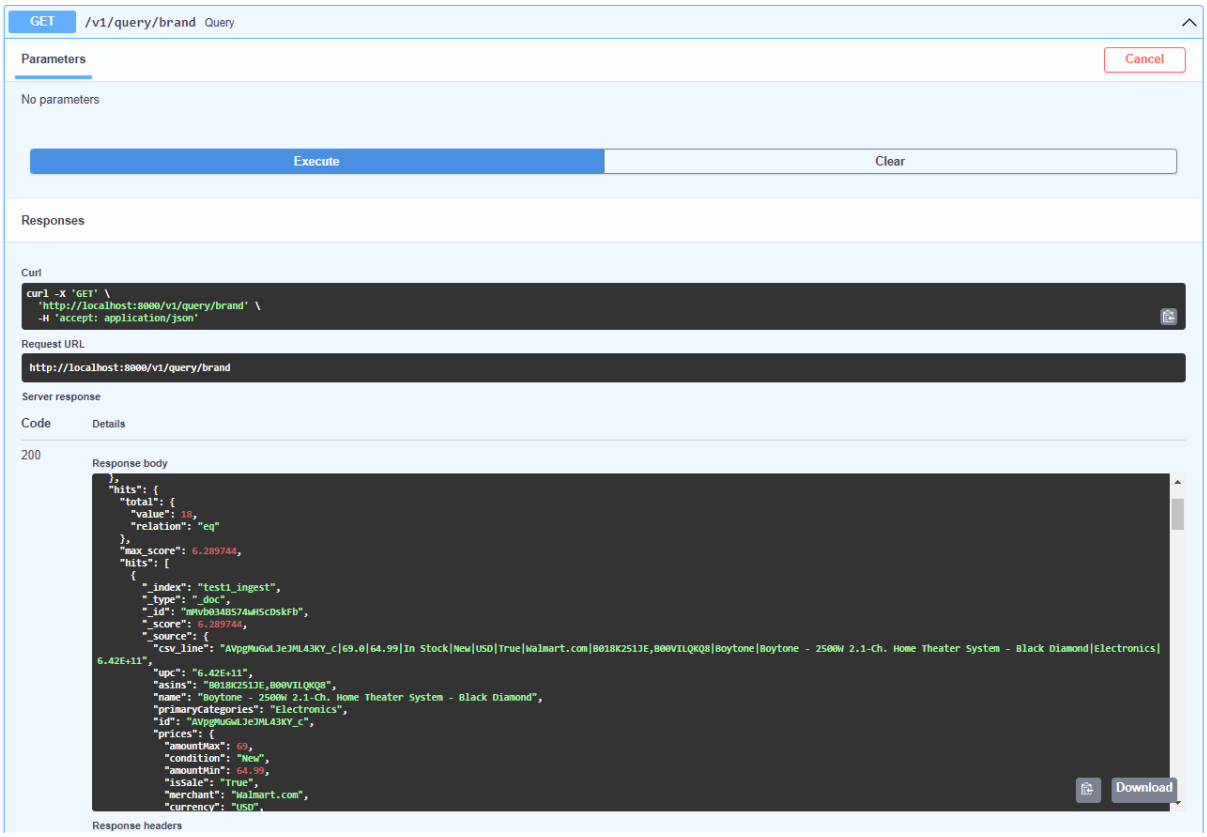
Cette route permet de faire une recherche par marque. La marque « Boytone » est rentrée par défaut mais il suffit de modifier le fichier « main.py » dans le répertoire « app » ci-joint pour requête suivant la marque de votre choix :

```python
# création d'une route pour requêter la base par marque
@api.get("/v1/query/brand")
def query():

    s = Search(using=client, index="test1_ingest") \
        .query("match", brand= "Boytone")

    response = s.execute()

    return response.to_dict()
```



2.2.4 « /v1/query/id »

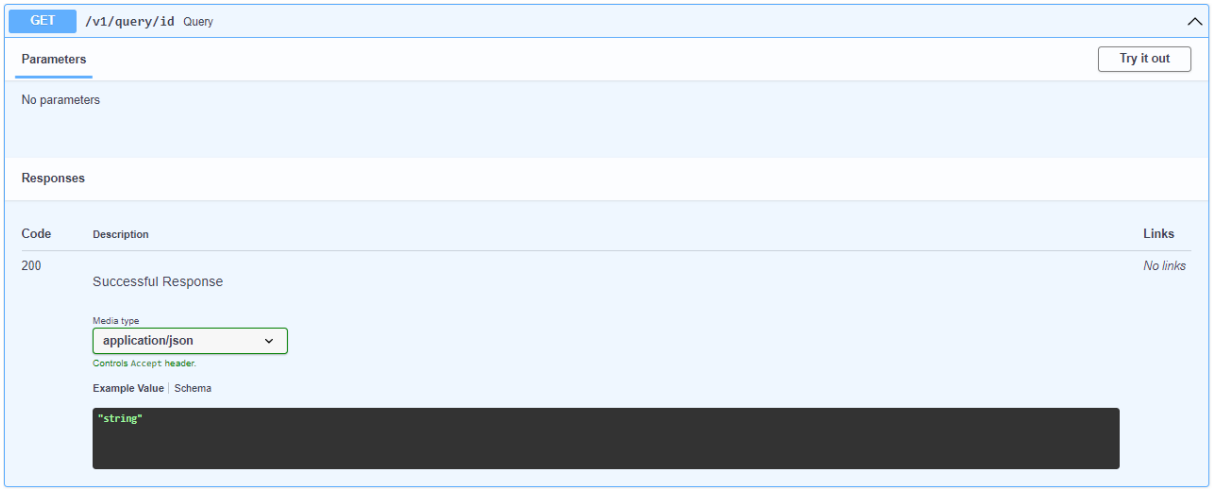
Cette route permet de faire des rechercher par « id ». Un id est rentré par défaut mais peut être modifié dans le fichier « main.py ».

```python
# création d'une route pour requêter la base par id
@api.get("/v1/query/id")
def query():

    s = Search(using=client, index="test1_ingest") \
        .query("match", id="AVphzgbJLJeJML43fA0o")

    response = s.execute()

    return response.to_dict()
```



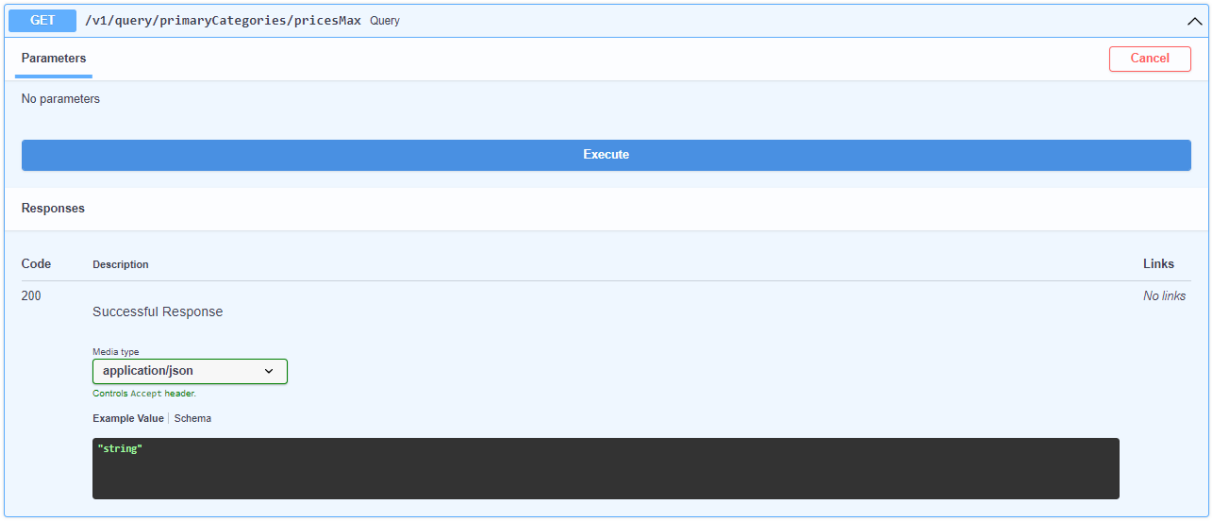
2.2.5 « /v1/query/primaryCategories/pricesMax »

Cette route permet de faire des recherches par catégorie dans une fourchette de prix. Une recherche est rentrée par défaut, il est possible de la modifier dans le fichier main.py :

```python
# création d'une route pour requêter la base par catégorie dans une fourchette de prix
@api.get("/v1/query/primaryCategories/pricesMax")
def query():

    s = Search(using=client, index="test1_ingest") \
        .query("match", primaryCategories= "Electronics") \
        .filter("range", **{ "prices.amountMax": { "gte": 20, "lte": 500 }})
    response = s.execute()


    return response.to_dict()
```



2.2.6 « /v1/query/brand/pricesMax »

Cette route permet de faire des recherches par marque dans une fourchette prix. Une recherche est rentrée par défaut, il est possible de la modifier dans le fichier main.py :

```python
  # création d'une route pour requêter la base par marque dans une fourchette de prix
@api.get("/v1/query/brand/pricesMax")
def query():

    s = Search(using=client, index="test1_ingest") \
        .query("match", brand= "Sanus") \
        .filter("range", **{ "prices.amountMax": { "gte": 10, "lte": 300 }})
    response = s.execute()

    return response.to_dict()
```