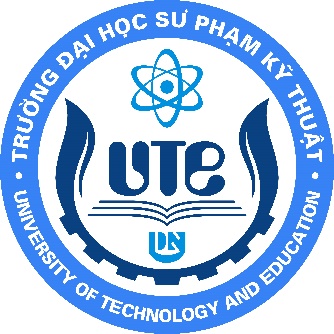
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**KHOA CÔNG NGHỆ SỐ**

****

**BÁO CÁO BÀI TẬP LỚN**

**MÔN CHUYÊN ĐỀ NGÔN NGỮ LẬP TRÌNH**

**Đề tài: Cào dữ liệu từ trang Website Tiki**

**Giảng viên hướng dẫn : Phạm Tuấn**

**Nhóm thực hiện : Nhóm Brand**

**Nhóm sinh viên thực hiện : Phạm Ngọc Tân**

**Đỗ Tấn Từ**

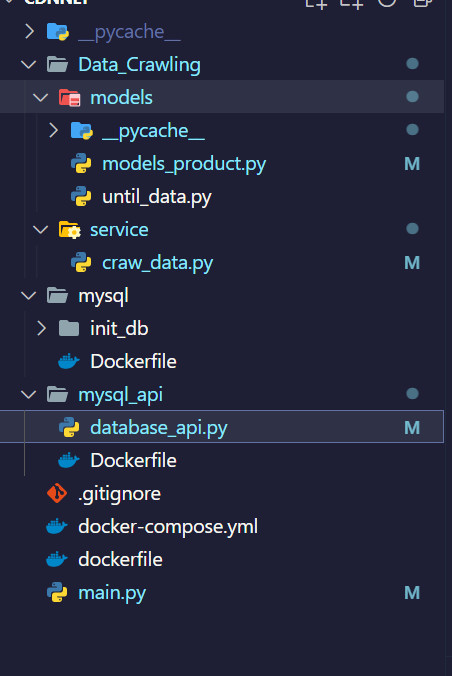
**Lê Văn Thắng**

**Nguyễn Đức Tín**

**Lớp học phần : 222CDNNLT01**

**Đà Nẵng, tháng 04/ 2023**

# Cấu trúc thư mục:



# Source Code:

1. **Model:**

from pydantic import BaseModel

class Product(BaseModel):

id\_product =0

sku = ''

product\_name = ''

url\_key =''

url\_path =''

availability = 0

seller\_id = 0

seller\_name =''

price = 0

original\_price = 0

discount = 0

discount\_rate = 0

review\_count = 0

rating\_average = 0

primary\_category\_path =''

primary\_category\_name =''

productset\_id = 0

seller\_product\_id = 0

thumbnail\_url =''

video\_url =''

class Time\_run(BaseModel):

hour = 0

minute = 0

second = 0

1. **Until data**

host = '127.0.0.1'

user\_name = "root"

password = "root"

database = "tiki"

port = '3306'

link\_api='https://tiki.vn/api/v2/products'

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0',

'Accept': 'application/json, text/plain, \*/\*',

'Accept-Language': 'vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3',

'Referer': 'https://tiki.vn/?src=header\_tiki',

'x-guest-token': '8jWSuIDBb2NGVzr6hsUZXpkP1FRin7lY',

'Connection': 'keep-alive',

'TE': 'Trailers',

}

params = {

'limit': '50',

'include': 'sale-attrs,badges,product\_links,brand,category,stock\_item,advertisement',

'aggregations': '2',

'trackity\_id': '70e316b0-96f2-dbe1-a2ed-43ff60419991',

'category': '1846',

'page': '50',

'src': 'c1846',

'urlKey': 'laptop-may-vi-tinh-linh-kien',

}

1. **Service**

import requests

import time

import schedule

import importlib.util

import sys

sys.path.append("../models")

sys.path.append("./mysql\_api")

from models\_product import Product

import database\_api

import until\_data

def parser\_product(json):

product = Product()

product.id\_product = json.get('id')

product.sku = json.get('sku')

product.product\_name = json.get('name')

product.url\_key = json.get('url\_key')

product.url\_path = json.get('url\_path')

product.availability = json.get('availability')

product.seller\_id = json.get('seller\_id')

product.seller\_name= json.get('seller\_name')

product.price = json.get('price')

product.original\_price = json.get('original\_price')

product.discount = json.get('discount')

product.discount\_rate = json.get('discount\_rate')

product.review\_count = json.get('review\_count')

product.rating\_average = json.get('rating\_average')

product.primary\_category\_path = json.get('primary\_category\_path')

product.primary\_category\_name = json.get('primary\_category\_name')

product.productset\_id= json.get('productset\_id')

product.seller\_product\_id = json.get('seller\_product\_id')

product.thumbnail\_url = json.get('thumbnail\_url')

product.video\_url = json.get('video\_url')

return product

def craw\_data(link\_api,params,headers):

result = list()

for i in range(1,50):

params['page']=i

response = requests.get(link\_api,headers=headers, params=params)

for record in response.json().get('data'):

product = parser\_product(record)

check = database\_api.find\_by\_id(product.id\_product)

if(check != None):

database\_api.update\_product(product)

print("update")

else:

database\_api.insert\_product(product)

print("insert")

result.append(product)

# print(result)

return result

def search\_data(product\_name):

return database\_api.search\_product(product\_name)

def update\_product(product):

return database\_api.update\_product(product)

def delete\_product(product\_id):

return database\_api.delete\_product(product\_id)

def find\_by\_id(product\_id):

return database\_api.find\_by\_id(product\_id)

1. **Database\_api**

import sys

from mysql import connector

sys.path.append("../Data\_Crawling/models")

from models\_product import Product

import until\_data

mydb=connector.connect(user=until\_data.user\_name, password=until\_data.password,port=until\_data.port,host=until\_data.host,database=until\_data.database)

def find\_by\_id(product\_id):

cursor = mydb.cursor()

query = "Select \* FROM product WHERE id\_product=%s"

cursor.execute(query, (product\_id,))

result = cursor.fetchone();

cursor.close()

return result

def insert\_product(product: Product()):

cursor = mydb.cursor()

my\_dict = product.\_\_dict\_\_

list\_key = list(my\_dict.keys())

columns = ", ".join(list\_key)

my\_list = list(my\_dict.values())

values\_placeholders = ", ".join(["%s"] \* len(my\_list))

values\_tuple = tuple(my\_list)

query = (f"INSERT INTO product ({columns}) VALUES ({values\_placeholders})")

try:

cursor.execute(query,values\_tuple)

mydb.commit()

result = cursor.fetchall()

cursor.close()

except:

mydb.rollback()

cursor.close()

return result

def search\_product(product\_name):

cursor = mydb.cursor()

query = "select \* from product where product\_name = %s"

cursor.execute(query, (product\_name,))

results = cursor.fetchall()

return results

def update\_product(product: Product()):

cursor = mydb.cursor()

query = "UPDATE product SET id\_product=%s, sku=%s, product\_name=%s, url\_key=%s ,url\_path=%s, availability=%s ,seller\_id=%s, seller\_name=%s,price=%s, original\_price=%s, discount=%s, discount\_rate=%s, review\_count=%s ,rating\_average=%s ,primary\_category\_path=%s ,primary\_category\_name=%s, productset\_id=%s ,seller\_product\_id=%s ,thumbnail\_url=%s ,video\_url=%s WHERE id\_product=%s"

my\_dict = product.\_\_dict\_\_

my\_list = list(my\_dict.values())

id\_product= my\_list[0]

value = my\_list.append(id\_product)

values\_tuple = tuple(my\_list)

values\_tuple = tuple(my\_list)

try:

cursor.execute(query ,values\_tuple)

mydb.commit()

result = cursor.fetchall()

cursor.close()

except:

mydb.rollback()

cursor.close()

print(result)

return result

def delete\_product(product\_id):

cursor = mydb.cursor()

query = "DELETE FROM product WHERE id\_product=%s"

cursor.execute(query, (product\_id,))

mydb.commit()

cursor.close()

1. **Main**

import sys

import pandas as pd

sys.path.append('Data\_Crawling/models')

sys.path.append('Data\_Crawling/service')

import craw\_data

import until\_data

import schedule

from models\_product import \*

from fastapi import FastAPI, HTTPException

app\_craw = FastAPI()

@app\_craw.get("/craw-product")

async def craw\_product():

try:

result = craw\_data.craw\_data(until\_data.link\_api,

until\_data.params, until\_data.headers)

return {

"status": '200',

'message':'success',

'data' : result

}

except Exception as e:

return {

"status": '500',

'message': str(e),

}

@app\_craw.get("/search/{product\_name}")

async def search\_data(product\_name:str):

if(HTTPException(200)):

result = craw\_data.search\_data(product\_name)

return {

'status' : '200',

'message' : 'success',

'data' : result

}

else:

return{

'status' : '400',

'message' : 'error'

}

@app\_craw.put("/update\_data")

async def update\_product(product : Product):

try:

result = craw\_data.update\_product(product)

return {

"status" : "200",

"message": "User updated",

"data" : result

}

except Exception as e:

return str(e)

@app\_craw.delete("/delete\_data/{product\_id}")

async def delete\_data(product\_id):

try:

craw\_data.delete\_product(product\_id)

return {

"status": "200",

"message": "Data has been deleted successfully!",

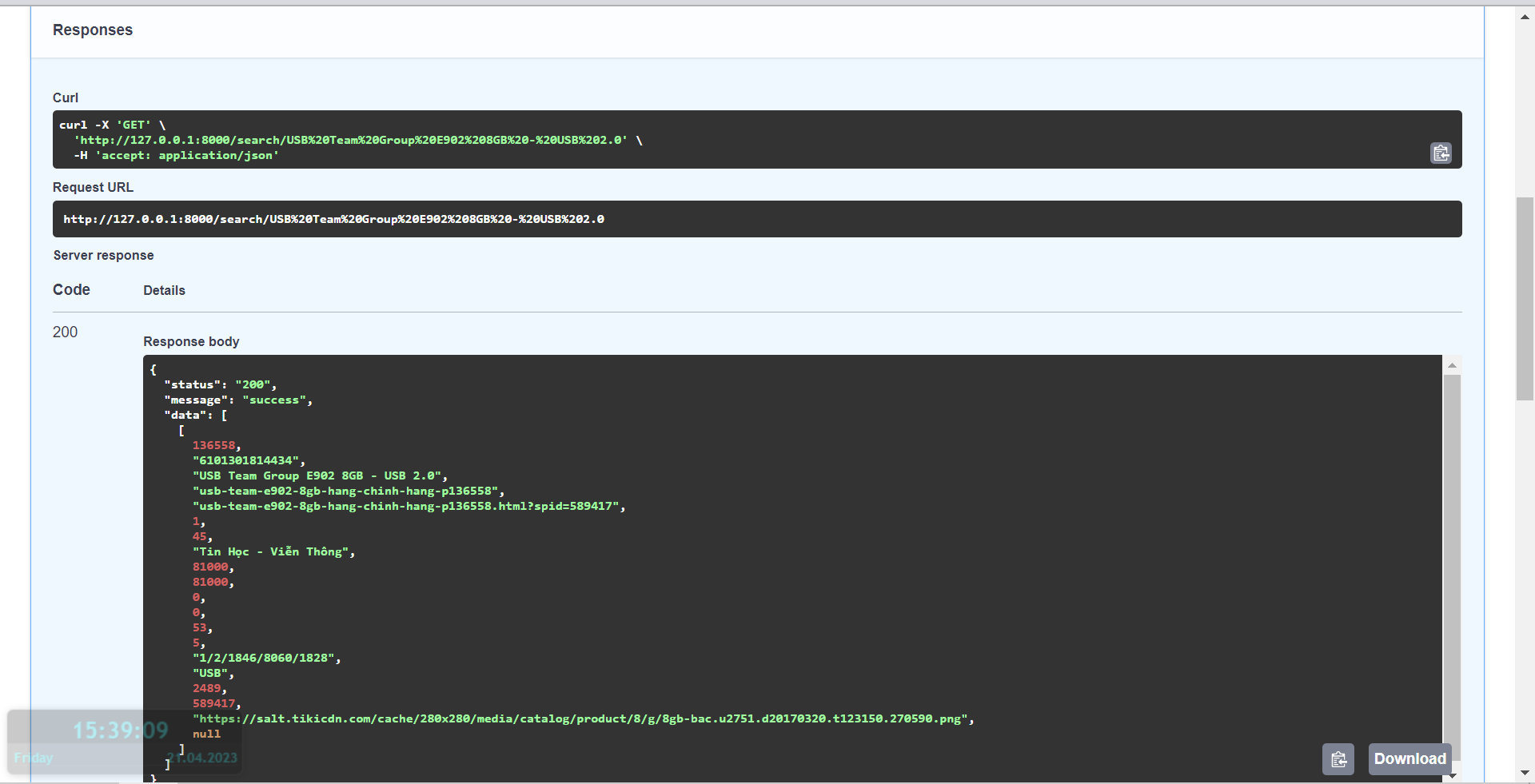
}

except Exception as e:

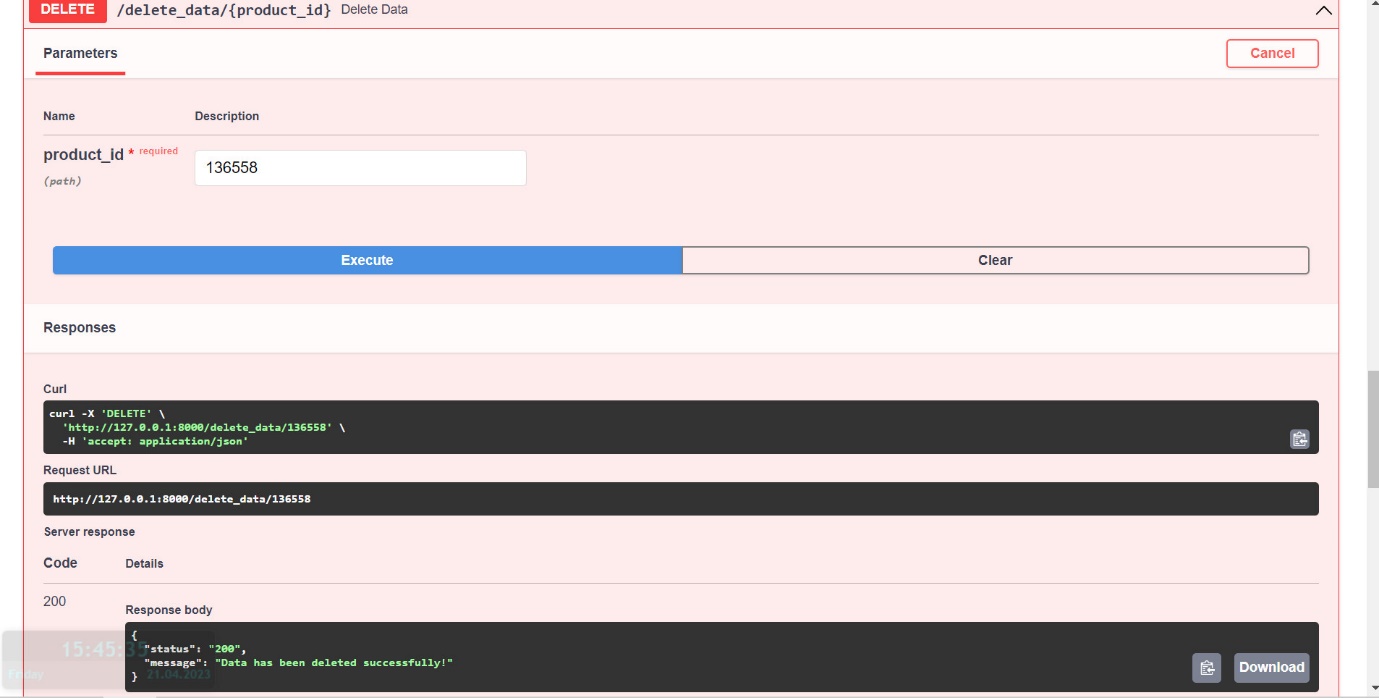
return {"error": str(e)}

# Demo API của DB

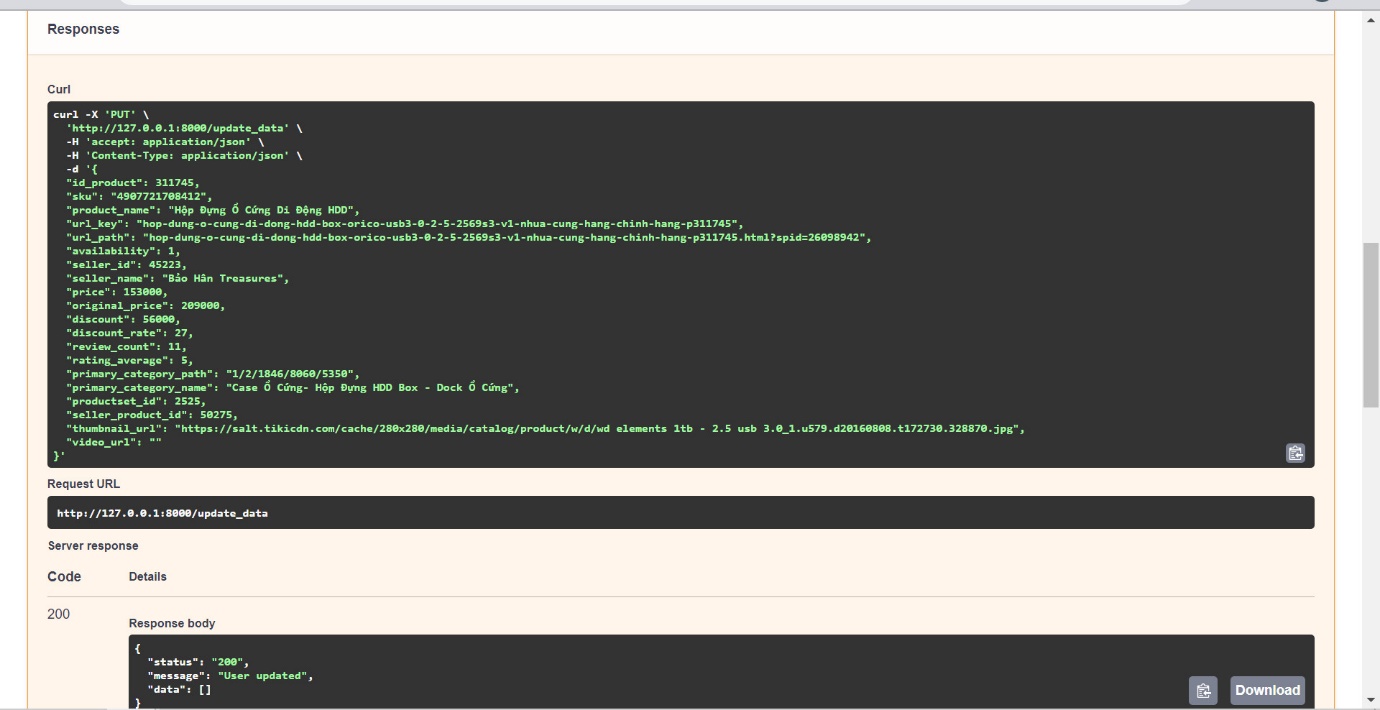
1. Demo search



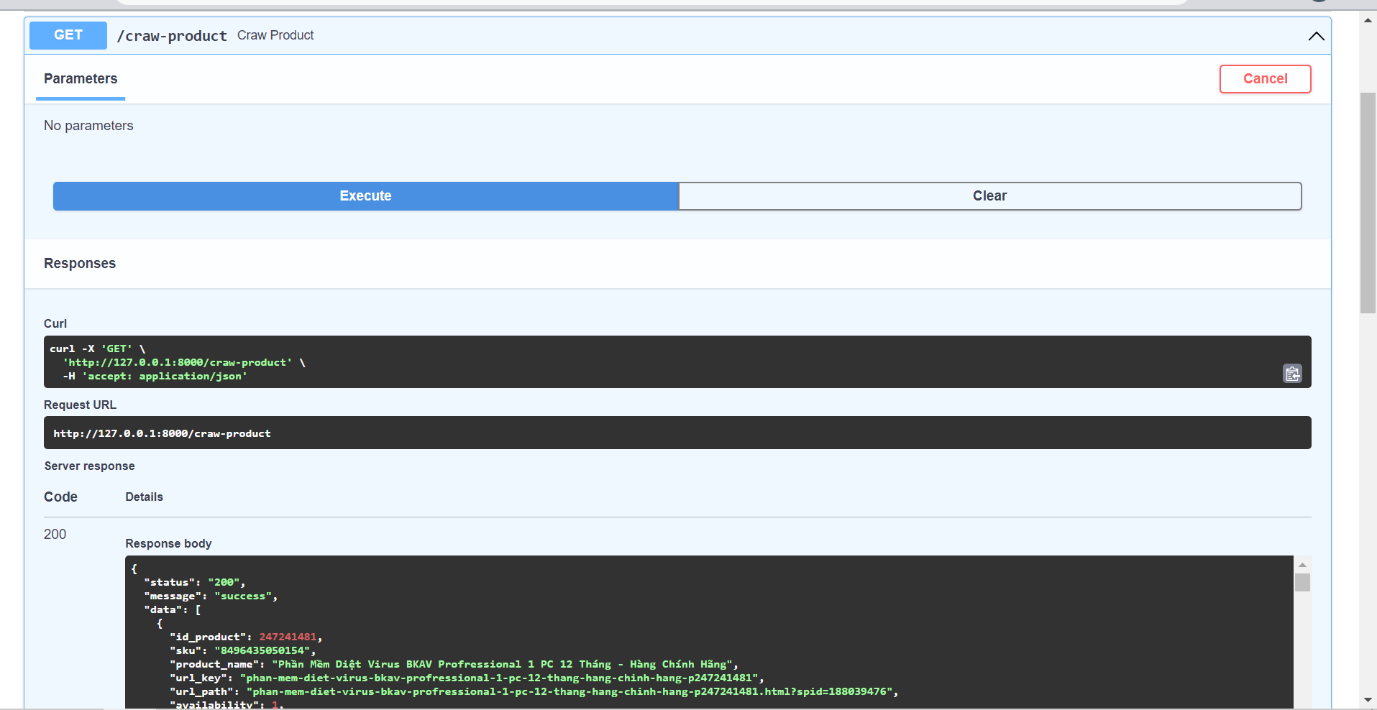
1. Demo delete



1. Update



# Demo API của Data Crawling



# Demo của hàm chính

