TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Võ Trung Hoàng Hưng - 21120011
Nguyễn Phúc Tân - 21120028
Hồ Lê Minh Quân - 21120314

XÂY DỰNG WEB CRAWLER SO SÁNH GIÁ THIẾT BỊ TIN HỌC

ĐỒ ÁN MÔN HỌC: MẠNG MÁY TÍNH
CHƯƠNG TRÌNH CỬ NHÂN TÀI NĂNG

GIÁO VIÊN HƯỚNG DẪN
Thầy Đỗ Hoàng Cường
Cô Huỳnh Thụy Bảo Trân
Cô Chung Thùy Linh

TP. Hồ Chí Minh, tháng 05/2023

Lời cảm ơn

Chúng em xin trân trọng cảm ơn các thầy cô:

- Thầy Đỗ Hoàng Cường
- Cô Huỳnh Thụy Bảo Trân
- Cô Chung Thùy Linh

đã tận tình hỗ trợ chúng em qua bài giảng trên lớp, tài liệu, và trả lời các thắc mắc của chúng em trong quá trình thực hiện đồ án này.

Mục lục

Danh sách hình

Danh sách bảng

Chương 1

Giới thiệu

Trong đồ án này, chúng tôi đã tạo ra một website có chức năng tra cứu thông số và so sánh giá bán các sản phẩm công nghệ (bao gồm laptop, máy tính để bàn, màn hình, bàn phím và chuột máy tính) trên thị trường. Website hoạt động theo cơ chế thu thập (crawler) và cập nhật dữ liệu định kỳ từ 4 trang bán hàng lớn:

- `https://www.thegioididong.com/`
- `https://fptshop.com.vn/`
- `https://phongvu.vn/`
- `https://thinkpro.vn/`

Khi người dùng tìm được sản phẩm mong muốn, website sẽ kết nối người dùng tới trang bán hàng gốc, ngoài ra còn hiển thị giá của sản phẩm đó tại các cửa hàng khác nhau. Từ đó, website giúp khách hàng tiếp cận sản phẩm nhanh, đầy đủ hơn và đưa ra lựa chọn mua sắm thuận tiện, tiết kiệm hơn mà không cần tốn thời gian truy cập nhiều website khác nhau.

Link website

`https://thegioilaptop.vercel.app/`

Link video demo

`https://youtu.be/vdWq4sdTs1E`

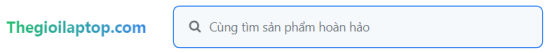
Link source code GitHub

`https://github.com/minhquanBr2/HCMUS-web-scraper.git`

Chương 2

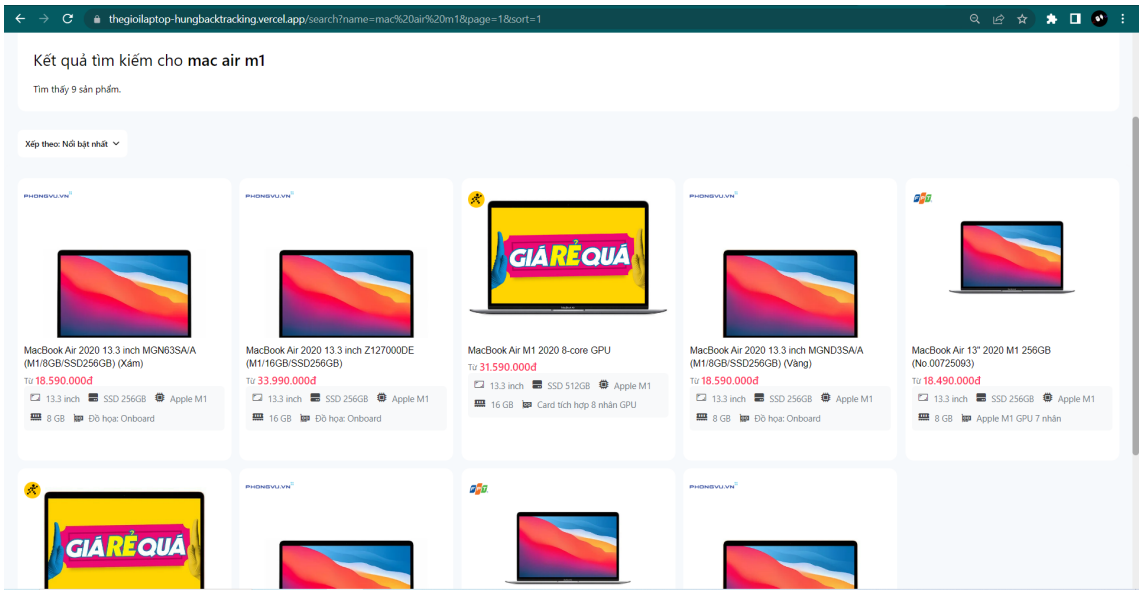
Tính năng chính

2.1 Tìm sản phẩm theo tên



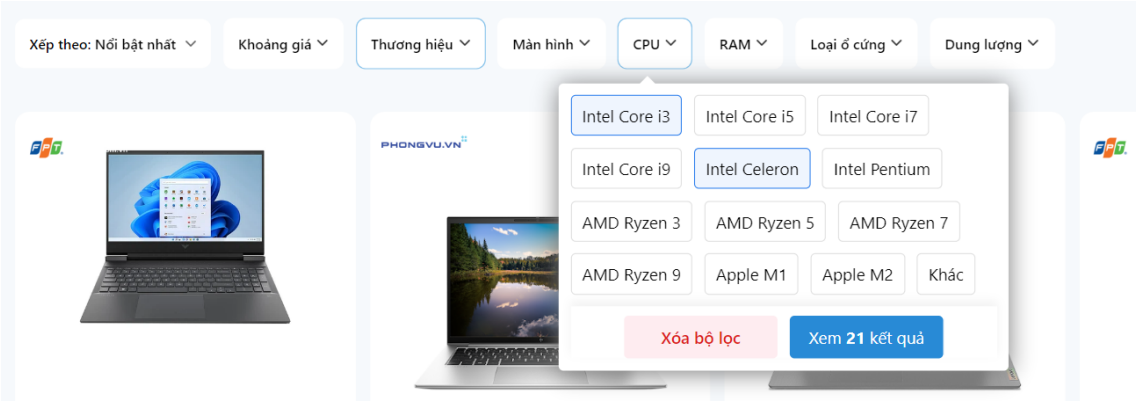
Hình 2.1: Thanh tìm kiếm

Nếu người dùng đang tìm kiếm một sản phẩm cụ thể, có thể nhập tên sản phẩm đó vào thanh tìm kiếm, website sẽ đề xuất kết quả bao gồm chính sản phẩm đó tại các cửa hảng khác nhau và các sản phẩm liên quan kèm theo thông số và giá bán.



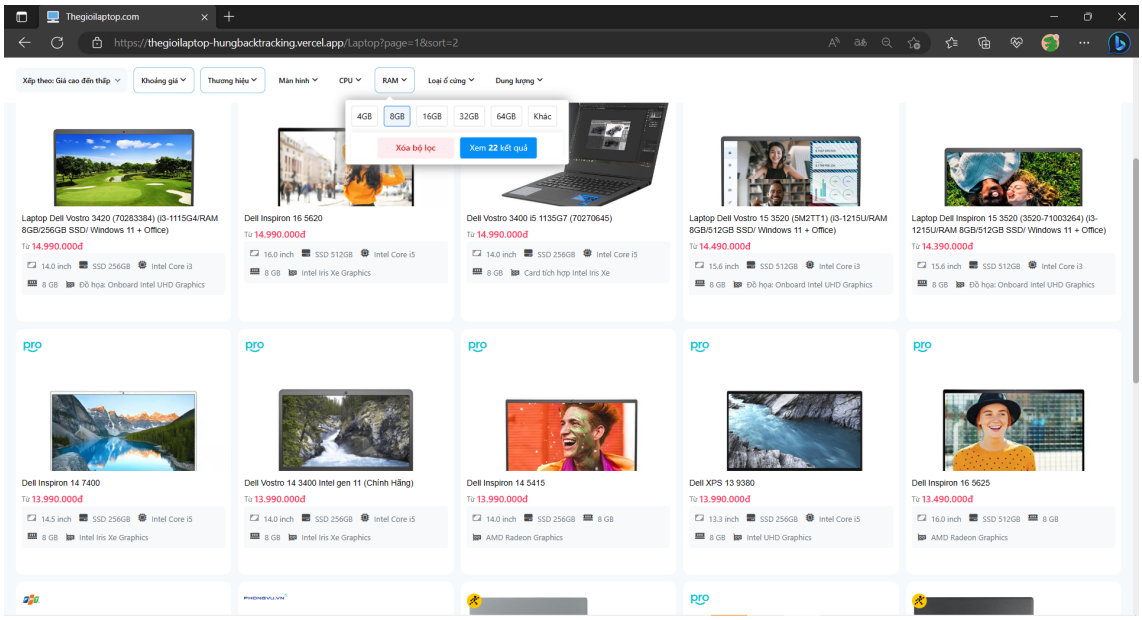
Hình 2.2: Ví dụ kết quả tìm kiếm 'mac air m1'

2.2 Tìm sản phẩm theo bộ lọc



Hình 2.3: Hình ảnh bộ lọc của website

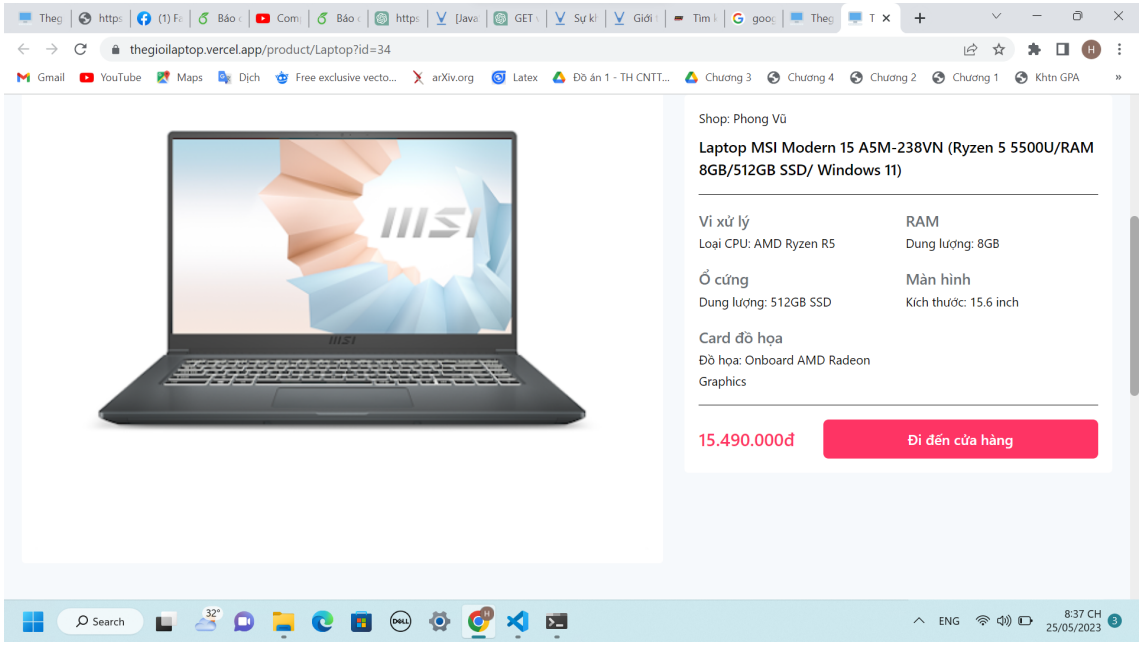
Nếu người dùng muốn tìm sản phẩm dựa trên nhu cầu về thông số hay giá cả thì có thể sử dụng bộ lọc. Từ đó website sẽ đề xuất tất cả các sản phẩm thỏa điều kiện bộ lọc kèm thông số và giá bán.



Hình 2.4: Kết quả lọc các laptop của hãng Dell có RAM 8GB, giá tiền từ 10-15 triệu đồng, sắp xếp theo giá tiền giảm dần.

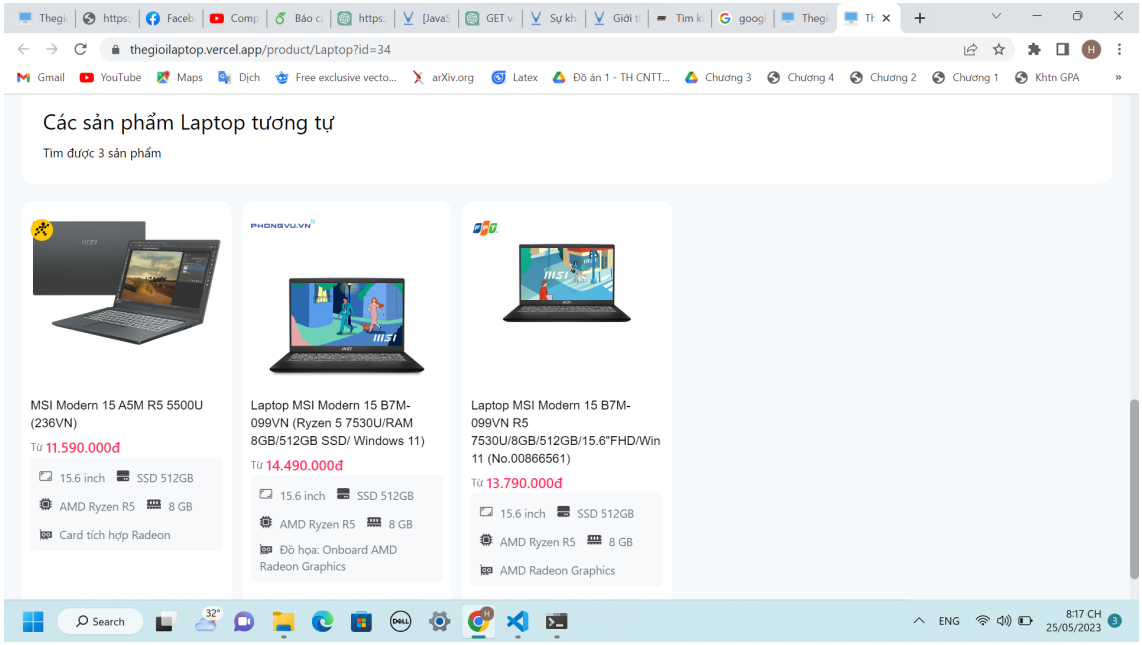
2.3 Trang thông tin chi tiết và so sánh với các sản phẩm tương tự

Khi nhấn vào đường dẫn của một sản phẩm sẽ được dẫn đến trang thông tin chi tiết của sản phẩm đó.



Hình 2.5: Trang hiển thị thông tin chi tiết của một Laptop.

Với mỗi sản phẩm tìm kiếm được, trang website thông tin chi tiết sẽ giúp người dùng so sánh giá bán của sản phẩm đó so với các mặt hàng gần giống hoặc giống nhau ở các trang bán hàng. Điều này có thể giúp người dùng biết được mặt hàng đó và những mặt hàng gần giống có giá cả thế nào thuận tiện cho việc lựa chọn một cách hợp lý.



Hình 2.6: Các laptop tương tự của một sản phẩm thương hiệu MSI.

Chương 3

Quá trình thực hiện

3.1 Cào và lưu trữ dữ liệu

3.1.1 Cào dữ liệu

Công cụ

1. **BeautifulSoup** Đây là một thư viện của **Python** được sử dụng để phân tích cú pháp HTML và XML, từ đó giúp tìm kiếm, trích xuất dữ liệu từ các thẻ, thuộc tính thẻ trên các trang web và lưu trữ trong các định dạng khác nhau.

2. **Selenium** Trên thực tế, chỉ dùng mỗi **BeautifulSoup** là chưa đủ cào hết tất cả sản phẩm, vì các trang web chỉ hiển thị một vài sản phẩm đầu tiên. Vì thế, chúng tôi sử dụng Selenium - một công cụ cho phép tự động hóa các thao tác của người dùng như access một trang, nhấn nút, điền form, v.v. Các nút Xem thêm hoặc Chuyển trang sẽ được tự động nhấn liên tục, nhờ vậy tất cả sản phẩm sẽ hiện ra trên webdriver và ta có thể cào đầy đủ dữ liệu mong muốn.

Phương pháp

Để lấy được dữ liệu từ các trang web, chúng tôi sử dụng 2 phương pháp:

1. Đối với những loại sản phẩm cần thu thập ít thông tin (như chuột, bàn phím), thông tin cần thiết đã hiển thị đầy đủ ở webpage của loại sản phẩm chứa tất cả các sản phẩm (chẳng hạn `https://www.thegioididong.com/laptop`, ta chỉ cần trích xuất dữ liệu đã hiển thị sẵn trên html của các webpage này.

2. Đối với những loại sản phẩm cần thu thập nhiều thông tin (như laptop, PC, màn hình), các thông tin cần thiết không nằm trong trang chung mà nằm sâu trong trang của từng sản phẩm (chẳng hạn `https://www.thegioididong.com/laptop/apple-macbook-air-2020-mgn63saa`. Với các loại sản phẩm này, đầu tiên ta cần trích xuất link đến các trang của từng sản phẩm thông qua url của trang chính, sau đó truyền url vào `webdriver` để lấy thông tin chi tiết của từng sản phẩm.

Url sẽ được truyền vào `webdriver` để lấy ra `page_source` ở dạng html.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
driver.get(url)
src = driver.page_source
soup = BeautifulSoup(src, "html.parser")

```

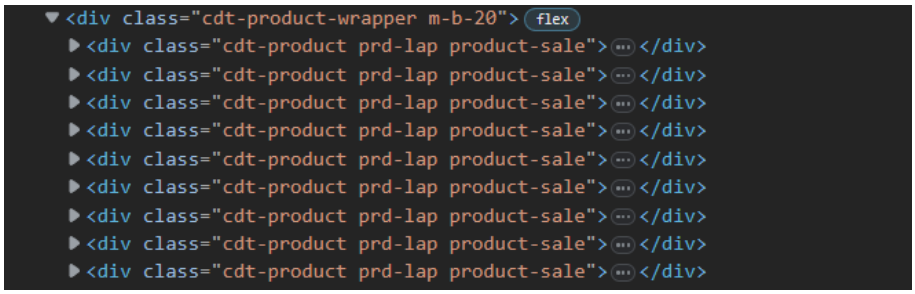
Sau khi có được page source của các webpage, các element trên sẽ dần được rút trích bằng các phương thức của **BeautifulSoup**. Có 3 phương thức được sử dụng chính là

`find()`, `select_one()` và `select()`. Điểm giống nhau của các phương thức này là đều sử dụng **CSS Selector** (quy tắc xác định hàng loạt các element có cùng định dạng trên webpage), từ đó rút ra những thông tin cần thiết. Bên cạnh đó, đối tượng `driver` cũng có phương thức `find_element` được sử dụng để rút trích các element cần thiết, thông qua 2 phương pháp là `CSS_SELECTOR` và `CLASS_NAME`. Một số ví dụ:

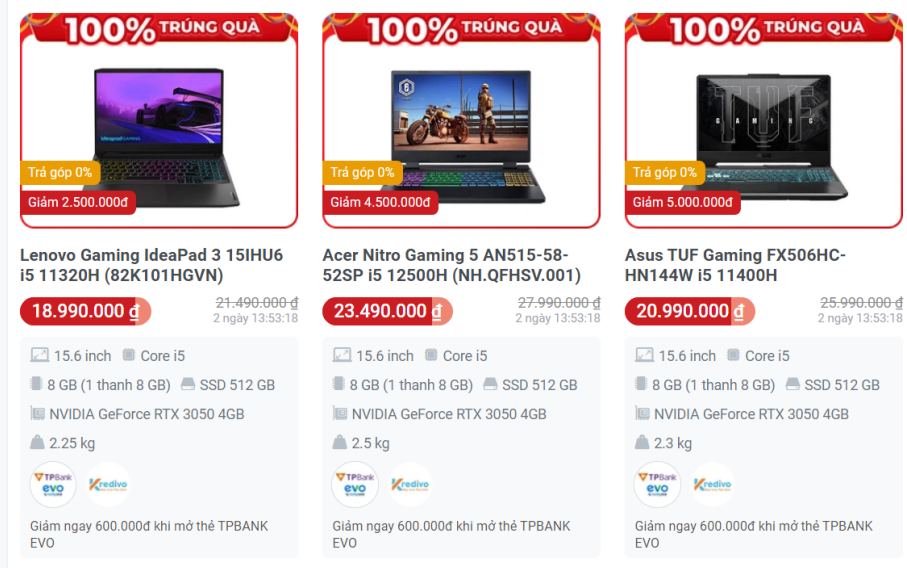
```
tiles = soup.select("div.container-productbox > ul > li")
name = tile.find('h3').get_text().strip()
info_link = 'https://www.thegioididong.com' + tile.find('a')
    .get("href")
ram = soup.select_one('div.parameter ul li:nth-child(2) div'
    ).get_text()
button = driver.find_element(By.CLASS_NAME, "btn-detail.btn-
    short-spec.not-have-instruction")

```

Chú ý trong đoạn mã nguồn trên, `tiles` là biến dùng để lưu các phần tử HTML ứng với các ô thông tin sản phẩm trên một trang web bất kì. Do phương thức `select()` trả về một danh sách, ta tuần tự duyệt từng phần tử của danh sách và lấy ra các thông tin cần thiết bên trong phần tử đó (chẳng hạn `name`, `info_link`, `ram`, v.v).



Hình 3.1: Đoạn mã cho các phần tử chứa thông tin từng sản phẩm trên webpage `https://fptshop.com.vn/may-tinh-xach-tay`



Hình 3.2: Danh sách các phần tử chứa thông tin từng sản phẩm trên webpage `https://fptshop.com.vn/may-tinh-xach-tay`

Bên cạnh đó, các trang web thương mại điện tử thường có nút "Xem thêm" (như `https://www.thegioididong.com/`) hoặc các nút đánh số trang ở cuối trang. Các nút này khi được click sẽ được kích hoạt để load thêm các sản phẩm khác từ cơ sở dữ liệu. Để thực hiện điều này, chúng ta cần một công cụ giúp điều khiển trang web tự động, chẳng hạn tự động click vào các nút trên, và may mắn thay `webdriver` của **Selenium** cung cấp đầy đủ tính năng này. Khi click vào các nút liên tục, nội dung trang (vốn bị ẩn) sẽ hiện ra đầy đủ, từ đó có thể lấy được dữ liệu của tất cả các sản phẩm của webpage thay vì phải dò các link, một công việc không phải website nào cũng cho phép.

Bên dưới là cách tìm một nút và tự động click vào nút đó. Cần lưu ý, nếu các nút được bấm vào liên tục, trang web có thể nhận ra đây không phải là các thao tác do con người điều khiển một cách tự nhiên, thậm chí block IP. Vì vậy, chúng tôi sử dụng phương thức `time.sleep` với những khoảng thời gian ngẫu nhiên để loại bỏ nguy cơ này.

```
button = driver.find_element(By.CLASS_NAME, "view-more")
button.click()
s = randint(1, 2)
time.sleep(0.5)

```

Trên thực tế, một số website sử dụng kỹ thuật **lazy loading** để tối ưu tốc độ tải trang và tiết kiệm băng thông. Các phần tử **lazy-loaded** không được tải ngay khi trang web được truy cập mà chỉ hiển thị ra khi người dùng cuộn (scroll) trang web đến đúng vị trí của phần tử đó trên webpage. Điều này gây ra khó khăn vì một số webpage không tự động cuộn xuống sau khi bấm các nút "Xem thêm", làm cho các phần tử **lazy-loaded** không hiển thị, ta không thể cào các dữ liệu này. Đoạn mã nguồn sau sẽ khắc phục vấn đề trên, bằng cách sử dụng `driver` để cuộn webpage về cuối trang và đợi một khoảng thời gian nhất định để đợi các phần tử hiện ra.

```
SCROLL_PAUSE_TIME = 1.5
last_height = driver.execute_script("return document.body.
    scrollHeight")
while True:
    driver.execute_script("window.scrollTo(0, document.body.
    scrollHeight);")
    time.sleep(SCROLL_PAUSE_TIME)
    new_height = driver.execute_script("return document.body
    .scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height
driver.implicitly_wait(2)

```

Dữ liệu sau khi được cào về sẽ được làm sạch (trình bày ở phần sau) và lưu trong cấu trúc dữ liệu `tuple` của **Python**, và được nạp vào câu lệnh SQL để cuối cùng nạp lên database.

3.1.2 Làm sạch dữ liệu

Công cụ

Sử dụng **regular expression** để rút trích dữ liệu từ những đoạn text có nhiều thông tin, biến đổi thành dạng chuẩn và đưa về kiểu dữ liệu phù hợp.

Ví dụ

- Hàm `extract_screen`: loại bỏ các kí tự ', ", " hay từ "inch" để trả về kích thước màn hình.

```
def extract_screen(screen):
    screen = str(screen)
    if screen == None or screen == '':
        return None
    pattern = r"""\d+(\.\d+)?(?=\s*(inch|\'|\"|(\'\')))
    """
    result = re.search(pattern, screen)
    if result:
        return float(result.group())
    return None

```

- Hàm `extract_ram`: rút trích ra kích thước RAM bằng cách tìm ra các số không quá 2 chữ số nằm trước chuỗi "GB", ví dụ "8GB", "64GB".

```
def extract_ram(data):
    tmp = " " + str(data)
    if data == None or data == '':
        return None
    pattern = r'(?<=\s)((\d\d)|(\d))(?=(\s*)?GB)'
    amount = re.search(pattern, tmp.upper())
    if amount:
        return int(amount.group())
    return None

```

- Hàm `price_to_int`: thay thế các kí tự không phải chữ số trong chuỗi thành ", sau đó chuyển kiểu dữ liệu từ string về int.

```
def price_to_int(price):
    price = str(price)
    if price:
        price = re.sub(r"\D", "", price)
        if price == '':
            price = None
        else:
            price = int(price)
        return price
    return None

```

3.1.3 Lưu trữ dữ liệu

Database

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở hoạt động theo mô hình client-server. **MySQL** có tốc độ xử lý khá cao, ổn định và khá dễ sử dụng, có thể hoạt động được trên khá nhiều hệ điều hành. Với tính bảo mật cao, **MySQL** rất thích hợp cho các ứng dụng có truy cập CSDL trên Internet khi sở hữu nhiều nhiều tính năng bảo mật cấp cao.

Thay vì sử dụng máy cá nhân làm máy chủ, trong đồ án này, chúng tôi sử dụng nền tảng **PlanetScale**. **PlanetScale** là một nền tảng cơ sở dữ liệu đám mây mã nguồn mở,

sử dụng một loại cơ sở dữ liệu phân tán được gọi là Vitess, được tạo ra bởi Google và được sử dụng bởi nhiều công ty lớn như YouTube và Slack.

PlanetScale cung cấp cho các nhà phát triển một nền tảng đáng tin cậy để quản lý các cơ sở dữ liệu phân tán, đồng thời cung cấp khả năng mở rộng linh hoạt và khả năng xử lý tải trọng lớn. Nền tảng này cũng cung cấp các tính năng quản lý và giám sát cơ sở dữ liệu, bao gồm quản lý khả năng sẵn sàng và khả năng sao lưu dữ liệu. Để sử dụng cơ sở dữ liệu trên **PlanetScale**, ta có thể đăng ký bằng tài khoản **GitHub**. Sau khi đăng ký, tạo 1 database, **PlanetScale** sẽ gửi chúng ta `user` và `password` của database, từ đó ta mới có thể sử dụng database này vào nền tảng của chúng ta.

Để kết nối chương trình cào dữ liệu với database của **PlanetScale** trong **Python**, ta sử dụng module `mysql.connector`. Thao tác kết nối được thực hiện như sau:

```
import mysql.connector
db = mysql.connector.connect(
        host="aws.connect.psdb.cloud",
        user=config.USER,
        password=config.PASSWORD,
        database="products"
    )

```

Cấu trúc database

1. Bảng **Laptop**

	Field	Type
▶	id	int
	name	varchar(200)
	price	int
	info_link	varchar(300)
	img_link	varchar(300)
	brand	varchar(20)
	screen	double
	cpu	varchar(20)
	ram	int
	disk_type	varchar(20)
	disk_stor...	varchar(20)
	gpu	varchar(200)
	shop	char(3)
	type	varchar(20)

Hình 3.3: Lược đồ bảng **Laptop**

2. Bảng **Screen**

Field	Type
id	int
name	varchar(200)
price	int
info_link	varchar(300)
img_link	varchar(300)
brand	varchar(20)
screen	double
refresh_rate	int
shop	char(3)
type	varchar(20)

Hình 3.4: Lược đồ bảng **Screen**

3. Bảng **Mouse**

Field	Type
id	int
name	varchar(200)
price	int
info_link	varchar(300)
img_link	varchar(300)
brand	varchar(20)
shop	char(3)
type	varchar(20)

Hình 3.5: Lược đồ bảng **Mouse**

4. Các bảng **PC**, **Keyboard** tương tự.

Lưu dữ liệu vào database

Đầu tiên, ta cần tạo 1 đối tượng **cursor** nhằm lưu kết quả mỗi khi thực hiện xong một query của **MySQL**. Lưu ý, việc gán tham số **buffed** bằng True giúp tất cả dữ liệu được lưu vào cache, sau đó mới được truy xuất ra, điều này rất tiện dụng khi xử lý với bảng dữ liệu kích thước lớn.

```
mycursor = db.cursor(buffered=True)

```

Sau đó, xóa dữ liệu trên bảng cũ và thêm dữ liệu mới vào bằng các câu lệnh **DELETE**, **INSERT**. Đoạn code sau mô tả quá trình nạp dữ liệu vào bảng Laptop:

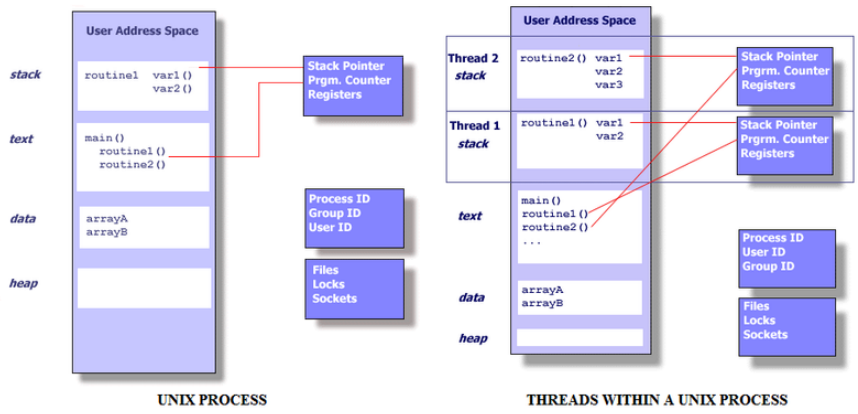
```
sql_insert = """INSERT INTO Laptop VALUES (%s, %s, %s, %s, %
    s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"""
mycursor.execute("DELETE FROM {}".format(table))
for data in dataset:
    try:
        mycursor.execute(sql_insert, data)
        db.commit()
    except:
        continue

```

3.1.4 Ứng dụng đa luồng (Multithreading) khi cào dữ liệu

Các khái niệm

- **Process**: là quá trình hoạt động của một ứng dụng. Tiến trình (process) chứa đựng thông tin tài nguyên, trạng thái thực hiện của chương trình.
- **Thread**: là một khối các câu lệnh (instructions) độc lập trong một tiến trình và có thể được lập lịch bởi hệ điều hành. Nói một cách đơn giản, thread là các hàm hay thủ tục chạy độc lập đối với chương trình chính. Một tiến trình có thể có nhiều luồng và phải có ít nhất một luồng, ta gọi đó là luồng chính (ví dụ hàm main() trong ngôn ngữ C). Ngoài các tài nguyên riêng của mình (các biến cục bộ trong hàm), các luồng chia sẻ tài nguyên chung của tiến trình.
- **Chương trình đa luồng**: là chương trình chứa hai hoặc nhiều phần mà có thể chạy đồng thời và mỗi phần có thể xử lý tác vụ khác nhau tại cùng một thời điểm, để sử dụng tốt nhất các nguồn có sẵn, đặc biệt khi máy tính của bạn có nhiều CPU.



Hình 3.6: Minh họa sự khác nhau giữa luồng và tiến trình.
Nguồn: `https://computing.llnl.gov/`

Công cụ

Trong đồ án này, chúng tôi sử dụng module `threading`, cụ thể là class `Thread`. Nó cung cấp các phương thức cơ bản để làm việc với luồng:

- `start()`: Method dùng để kích hoạt (chạy) một thread object, mặc định sẽ gọi `run()` method.
- `run()`: Đây là phương thức chính chúng ta cần cài đặt, mô tả các công việc mà luồng thực hiện. Mặc định phương thức này sẽ gọi hàm liên kết với đối số target lúc khởi tạo luồng.
- `join()`: Khi được gọi, method này sẽ block thread gọi nó (calling thread) cho đến khi thread được gọi (called thread - tức là thread có method join() vừa gọi) kết thúc. Method này thường được dùng trong luồng chính để đợi các thread khác kết thúc công việc của mình và xử lý tiếp kết quả.

Các bước cài đặt

1. Tạo danh sách các thread và lưu trong cấu trúc dữ liệu `list` của **Python**.

```
def scrape0():
    TGD.LaptopScraper(driver[0], dataset['Laptop'], '
    https://www.thegioididong.com/laptop/')

# create threads
t = ['' for _ in range(NUMBER_OF_THREADS)]
t[0] = threading.Thread(target = scrape0)
t[1] = threading.Thread(target = scrape1)
...
t[8] = threading.Thread(target = scrape8)

```

   Ở đây, chúng tôi sử dụng 9 threads để cào dữ liệu từ 4 trang web. Các hàm `scrape0` đến `scrape8` gọi các hàm dùng để cào dữ liệu các sản phẩm khác nhau từ các trang web khác nhau (ví dụ hàm `scrape0` cào thông tin laptop của trang web `https://www.thegioididong.com/`). Mỗi thread được khởi tạo sẽ ứng với một hàm `scrape_` mà ta truyền tên vào tham số `target`, để sau đó khi các thread chạy song song, các hàm `scrape0`, `scrape1`, v.v. sẽ được chạy đồng thời.

2. Lần lượt gọi phương thức `start()` cho từng thread.

3. Lần lượt gọi phương thức `join()` cho từng thread.

4. Lần lượt gọi phương thức `quit()` cho từng driver đã được sử dụng.

3.1.5 Ứng dụng lập lịch (Scheduling) khi cào dữ liệu

Module `schedule` của **Python** hỗ trợ việc lập lịch để chạy các chương trình định kỳ. Để lên lịch cào dữ liệu mỗi ngày một lần, ta có thể sử dụng đoạn mã sau:

```
schedule.every(1).day.do(UPDATE)
while True:
    schedule.run_pending()
    time.sleep(1)
```

Trong đó `UPDATE` là hàm thực hiện tất cả công việc từ cào dữ liệu đến nhập dữ liệu lên database. Trong vòng lặp vô tận, nó sẽ kiểm tra xem có nhiệm vụ lập lịch nào đang chờ thực hiện, nếu có thì sẽ thực hiện chúng. Lưu ý cần có thời gian nghỉ 1 giây giữa mỗi lần lặp để tránh việc chương trình ngốn quá nhiều tài nguyên CPU.

3.2 Thiết kế giao diện và cải thiện trải nghiệm người dùng

Trang web bao gồm ba đường dẫn chính tương ứng với ba chức năng chính là tìm kiếm theo bộ lọc và sắp xếp, tìm kiếm theo từ khóa và so sánh với sản phẩm tương tự.

Phần lớn các trang đều đã được thiết kế để phù hợp với cả những thiết bị màn hình nhỏ như điện thoại. Có thể kể đến như thanh bộ lọc có thể lướt sang chiều ngang khi quá dài so với kích cỡ màn hình hay số lượng sản phẩm hiện thị trên một hàng được tùy biến phù hợp với mọi loại thiết bị.

Ngoài ra còn khá nhiều hiệu ứng khác đều được xử lý bằng Javascript nhằm giúp người dùng dễ dàng sử dụng nhất có thể.

Vì số lượng tổng sản phẩm khá lớn, chúng tôi đã sử dụng kỹ thuật **xử lý bất đồng bộ** cho trang web để tăng tốc thời gian vận hành. Trang web sẽ ưu tiên tải và hiển thị những phần quan trọng nhất sau đó mới tiếp tục tải những phần chưa cần thiết.

```
let response = await fetch(`/searchProduct?name=${name}&page=1&sort=1`, {
    method: "POST",
    headers: {
        "Content-Type": "application/json",
    },
    body: JSON.stringify(name),
}).then((res) => res.json());
```

3.3 Xử lý các truy vấn từ người dùng

3.3.1 Các phương thức GET và POST

POST và GET là hai phương thức HTTP phổ biến được sử dụng trong giao tiếp giữa server(máy chủ) và client(trình duyệt web).

- Phương thức POST được sử dụng để gửi dữ liệu từ trình duyệt lên máy chủ để xử lý. Thông thường phương thức POST được sử dụng khi có yêu cầu thay đổi trạng thái trên server.
- Phương thức GET được sử dụng để yêu cầu dữ liệu từ máy chủ. Thường được sử dụng để lấy thông tin từ máy chủ mà không làm thay đổi dữ liệu trên máy chủ.

3.3.2 Tìm sản phẩm theo từ khóa

Dùng phương thức POST để lấy chuỗi từ khóa người dùng nhập vào từ thanh tìm kiếm. Tiếp theo gọi truy vấn SQL đến cơ sở dữ liệu thực hiện:

- Gộp các bảng sản phẩm khác nhau lại bằng từ khóa `UNION`.

```
cursor.execute("SHOW TABLES")
rows = cursor.fetchall()
basic_info_sql_cmd = []
for row in rows:
    if row[0] not in allow_path:
        continue
    basic_info_sql_cmd.append(f"""
    (SELECT '{row[0]}' AS table_name, name, price, info_link, img_link, shop """ + selectFilterAll[row[0]] + f""", id
    FROM {row[0]})
    """)
search_table_sql = " UNION ".join(basic_info_sql_cmd)

search_sql = f"""
SELECT *
FROM ({search_table_sql}) AS R
WHERE name LIKE %s AND price IS NOT NULL AND img_link IS NOT NULL AND info_link IS NOT NULL
"""
```

- Tìm kiếm sản phẩm trên bảng theo thuộc tính `name` (thỏa nếu tất cả các từ trong tên sản phẩm người dùng nhập là chuỗi con của giá trị thuộc tính `name`).

```
cursor.execute(search_sql, ['%' + name.replace(' ', '%')
    + '%'])
rows = cursor.fetchall()
```

Sau đó ứng dụng web sẽ xử lý và đưa dữ liệu lên website. Danh sách sản phẩm được trả về không chỉ là từ một loại sản phẩm mà được tìm kiếm từ toàn bộ năm loại sản phẩm chính của website.

3.3.3 Tìm sản phẩm theo bộ lọc

Dùng phương thức POST để thu thập các bộ lọc mà người dùng đã chọn.

Đầu tiên, chương trình sẽ gọi hàm `createSQL` để chuyển các bộ lọc thành điều kiện `WHERE` trong câu truy vấn của SQL. Cụ thể, ứng với mỗi bộ lọc, trang web sẽ chuyển tên bộ lọc được hiển thị trên màn hình thành tên của thuộc tính tương ứng với bộ lọc đó trong bảng dữ liệu (ví dụ `Màn hình` -> `screen`). Giá trị của bộ lọc cũng được chuẩn hóa để match với dữ liệu trong bảng, đặc biệt với những giá trị bộ lọc là khoảng số như "Từ 10 đến 15 triệu", các số sẽ được trích xuất ra để sử dụng với từ khóa `BETWEEN`. Các điều kiện trên được gắn với nhau bằng từ khóa `AND`, ta sẽ được một tập hợp các điều kiện lọc hoàn chỉnh. Kết hợp với `SELECT` và `FROM`, ta có câu truy vấn SQL.

Sau khi có câu truy vấn SQL và thực hiện thông qua **cursor** của module `mysql.connector`, dữ liệu được đưa lên webpage bằng hàm `filter_product`

3.3.4 So sánh giá giữa các trang bán hàng

Mỗi sản phẩm hiển thị trên trang web đều có đường dẫn href chứa thông tin về id và loại mặt hàng của sản phẩm. Từ đó server có thể dễ dàng lấy dữ liệu từ URL và đưa trình duyệt truy cập trang thông tin chi tiết của sản phẩm đó bằng phương thức GET.

Những sản phẩm tương tự được hiển thị bên trong trang sản phẩm chi tiết được tìm kiếm bằng Tìm kiếm toàn văn (Full Text Search). Tìm kiếm toàn văn là một chức năng có trong MySQL cho phép người dùng tìm kiếm các mẫu thông tin khớp với một chuỗi trên một hay một số bảng nhất định.

Đối với chức năng so sánh giá các sản phẩm tương tự từ các trang bán hàng khác nhau, chúng tôi quyết định sử dụng Tìm kiếm toàn văn thay vì sự so khớp dạng LIKE 'NAME' vì nó không chính xác và cũng không nhanh bằng Tìm kiếm toàn văn.

```
select = "SELECT name, price, info_link, img_link, shop" +
    selectFilter[table_name]
sql = select + f"""
FROM {table_name}
WHERE name IS NOT NULL
AND price IS NOT NULL
AND info_link IS NOT NULL
AND img_link IS NOT NULL
AND id <> {index}
AND brand LIKE  '%{product['brand']}%' """ + conditional + f
    """AND MATCH(name) AGAINST('{against}')
LIMIT 15
"""
cursor.execute(sql)
```

3.4 Triển khai lên Internet

Trong đồ án này, chúng tôi đã sử dụng `Vercel`, một nền tảng web hosting giúp triển khai trang web một cách nhanh chóng, dễ dàng.

Các bước thực hiện:

1. Tạo tài khoản Vercel: Truy cập trang web Vercel (`https://vercel.com`) và tạo một tài khoản mới nếu chưa có. Đăng nhập vào tài khoản.

2. Vì ở đồ án này chúng tôi xây dựng website trên nền tảng backend là `Flask` nên cần phải tạo file `vercel.json` với nội dung tùy thuộc vào cách đặt vị trí mã nguồn.

```
{
    "version": 2,
    "builds": [
        {
            "src": "app.py",
            "use": "@vercel/python"
        }
    ],
    "routes": [
        {
            "src": "/(.*)",
            "dest": "app.py"
        }
    ]
}
```

3. Chạy lệnh `vercel .` tại thư mục chứa mã nguồn, sau đó đăng nhập, xác nhận đồng ý triển khai lên `Vercel`.