Proyecto de Prácticas

Ingeniería de Sistemas de Información

UNIVERSIDAD DE GRANADA

GROCERY GURU

Comparador de Precios de Productos de Supermercados

Grocery Guru

Comparador Precios de Productos de Supermercados

- Rafael Luque Framit - A1
- Guillermo López Jiménez - A3

ÍNDICE

1. Descripción

Enlace a la web: https://learned-vault-417715.oa.r.appspot.com/

En esta práctica final hemos realizado un comparador de precios de productos de supermercados, nuestra aplicación se llama Grocery Guru. Es una aplicación web que permite a los usuarios comparar precios de productos de diferentes supermercados de manera eficiente y conveniente en una única aplicación web.

Nuestra aplicación es una herramienta muy útil para todas aquellas personas de cualquier clase que pretenden ahorrar a la hora de realizar su compra. Además, nuestra aplicación ayuda a satisfacer una necesidad real en la sociedad, promoviendo transparencia y eficiencia en el proceso de compra de alimentos y productos básicos de primera necesidad.

En nuestra aplicación hemos tenido en cuenta los siguientes aspectos principales:

- Comparar los precios según el supermercado.
- Ordenar los productos de menor a mayor precio.
- Ordenar los productos de mayor a menor precio.

Para nuestra aplicación hemos realizado la búsqueda de productos de 3 diferentes supermercados. En primer lugar, a la hora de hacer web scraping hemos extraído datos de los supermercados Dia y Amazon (Hemos buscado otros supermercados, pero estos eran los únicos con los que logramos que todo funcionara correctamente, exceptuando Hipercor, del que hablaremos ahora). En segundo lugar, nuestra tercera fuente de datos la hemos obtenido de la API Open Food la cual está basada en JSON.

En cuanto a la API que hemos usado debemos destacar que es una API que se centra en la información nutricional de los productos y de la capacidad de estos para ser reciclados. Por lo que no podemos obtener el precio de los productos sacados de esta API. No obstante, si se proporciona los supermercados en los que se encuentran estos productos lo cual es una información destacable.

Por otro lado, intentamos realizar web scraping de distintos supermercados como Carrefour, Alcampo y Aldi pero ninguno de ellos con éxito ya que todas ellas detectaban el comportamiento de web scraping y nos rechazaban la petición HTTP.

Cabe destacar que también hemos realizado web scraping del supermercado Hipercor, esta implementación funciona perfectamente en nuestro sistema local pero a la hora de ejecutarlo en Google Cloud da problemas (relacionados con chromium de la librería playwright). Por lo que hemos descartado la extracción de datos de este supermercado, quedándonos con las 3 fuentes mencionadas anteriormente.

Por otro lado, también intentamos acceder a la API del supermercado Carrefour pero a la hora de registrarnos en la web no nos dejó por problemas de credenciales, por lo que nos tuvimos que quedar con la de Open Food.

2. Desarrollo

En primer lugar, el archivo *main.py* contiene el código para realizar la búsqueda, ordenación e impresión de los productos. Este archivo es el encargado de lanzar la aplicación mediante Flask. Todo esto es llamado desde *index.html* y *styles.css* los cuales son archivos encargados de la creación de la página web, el primero de ellos contiene el contenido y estructura básica además de llamar a los procedimientos del main.py, el segundo de ellos define cómo se deben mostrar los elementos del HTML.

En segundo lugar, tenemos el archivo *Producto.py,* el cual contiene la información de los productos: nombre, precio, enlace al producto, supermercado de origen, logo del supermercado y url de la imagen del producto. Además, definimos las funciones get y set para cada atributo.

En tercer lugar, tenemos el archivo *scraper.py*, que contiene las funciones para hacer web scraping sobre los supermercados que hemos elegido extraer la información. Estas funciones se encargan de entrar en la web del supermercado y adquirir la información que queremos guardar de cada producto.

Por otro lado, tenemos el archivo *apiOF.py* el cual contiene lo necesario para obtener la información de la API Open Food. La función *search_OpenFood* se encarga de realizar la petición HTTP a la web, analizar la respuesta dada JSON y extraer la información relevante de cada producto.

Por último, el archivo *busquedaCompleta.py* se encarga de llamar a las funciones del scraper y de la API respectivamente y juntar en una misma lista los productos encontrados.

3. Lenguajes y librerías utilizadas

Para el desarrollo de la página web hemos utilizado *Html* y *Css*.
El proyecto ha sido desarrollado en **Python**, hemos seguido el tutorial para desplegar aplicaciones web utilizando *Flask* y *Jinja* proporcionado en la página de la asignatura.

Además, hemos utilizado *BeautifulSoup* para hacer el scrapping en las páginas que mostramos (Dia y Amazon). También hemos utilizado *Playwright* para hacer el scrapping en Hipercor, pero como hemos explicado anteriormente esto solo nos funciona al correr la aplicación localmente, por lo que en la aplicación que se verá (la del enlace), en realidad esta biblioteca no está siendo utilizada. También hemos usado la biblioteca *Requests* principalmente, así como el resto de cosas que hay incluidas en *requirements.txt*.

4. Modelado de la integración de datos

Cada fuente de datos que utilizamos en nuestro proyecto tiene distintas formas de representar los datos, por lo que hemos tenido que unificarlas en una clase, que será nuestra GCS; *Producto*.
Esta clase será una clase sencilla, con los setters y getters correspondientes, y que contendrá:

- **Precio**: El precio del producto en el supermercado donde se ha obtenido el resultado.
- **Nombre**: El nombre que da cada supermercado al producto.
- **Url al producto**: la url a la página origen donde sacamos el producto.
- **Supermercado**: Nombre del supermercado donde se ha obtenido la información del producto.
- **Logo**: Imagen con el logo del supermercado del producto.
- **Imagen**: Imagen del producto, obtenida del supermercado en cuestión.

A partir de esta clase trabajaremos con los datos en nuestra página, los ordenaremos, los mostraremos, etc.

Por otro lado tenemos las fuentes de datos locales, en este caso son 3 (4 si contamos Hipercor), con sus equivalencias correspondientes.

Clase Producto	**Hipercor**	**Dia**	**Amazon**	**Api Open Food**
Nombre	`product_tile-description`	`search-product-card__product-name`	`title-recipe`	`product_name`
Precio	`prices-price_current`	`search-product-card-unit-price`	`a-price`	
Imagen	`event js-product-link`	`search-product-card__product-image`	`s-image s-image-optimized-rendering`	`image_front_url`

5. Posibles mejoras y actualizaciones

En primer lugar, podríamos considerar la extracción de información de más fuentes de datos ya sea mediante web scraping o de las APIs de los supermercados. De esta manera nuestro abanico de información sería mucho más amplio y completo.

Por otro lado, se podría implementar la extracción de otro tipo de información de los productos como por ejemplo alérgenos, información nutricional, o reseñas y opiniones de

usuarios. Esta información es complicada de extraer ya que no todas las páginas web de los supermercados incorporan esta información.

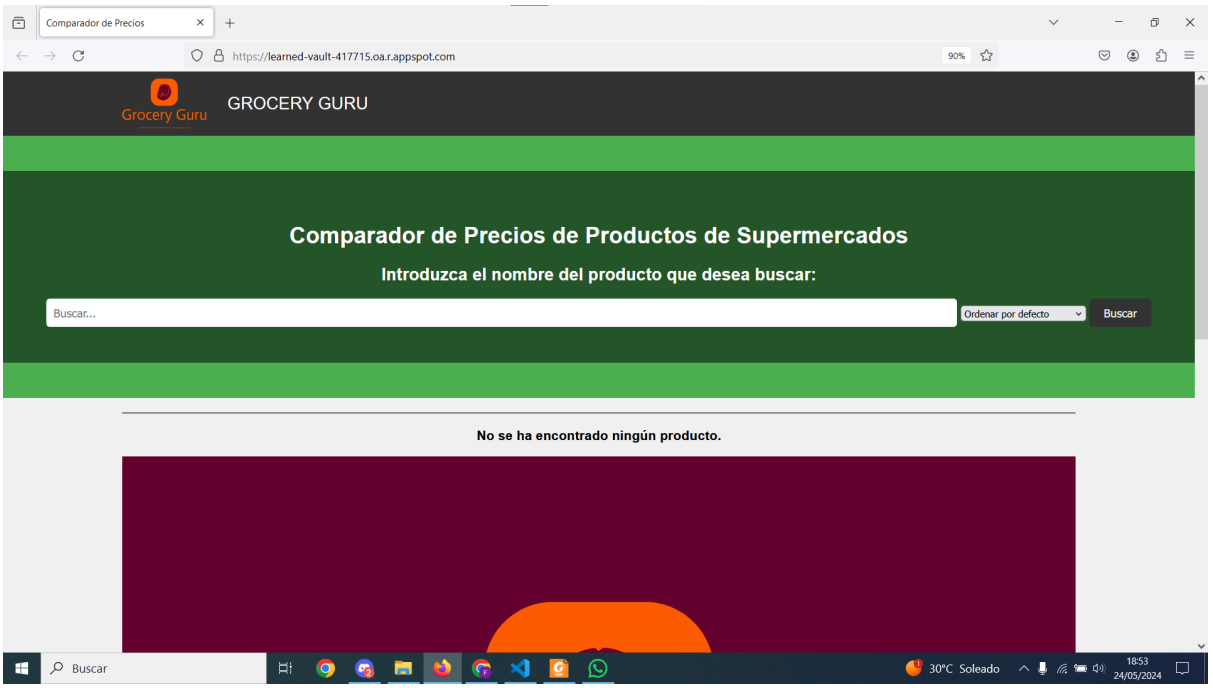
Por último, se podría desarrollar aún más nuestra página web consiguiendo una interfaz de usuario mucho más profesional y acabada dando así al usuario una experiencia de uso mucho más reconfortante e integra de la que hay ahora.

6. Funcionamiento

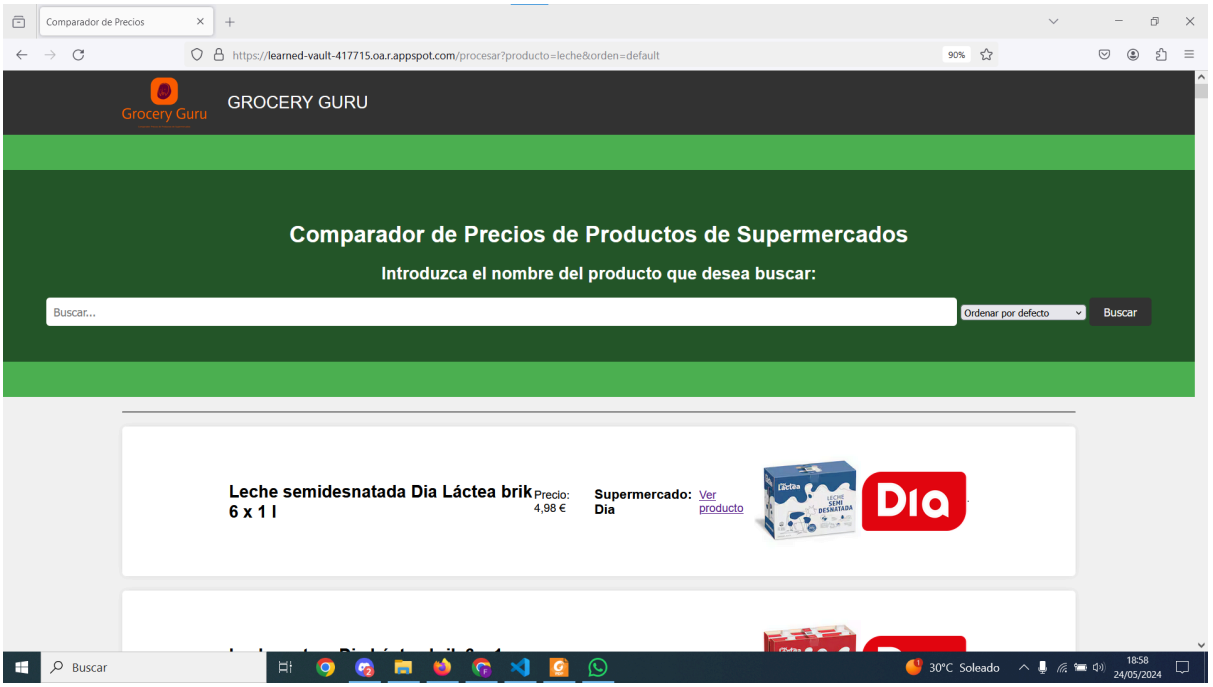
Enlace a la web: https://learned-vault-417715.oa.r.appspot.com/

Al entrar a nuestra aplicación encontramos en primera instancia esta interfaz de usuario. En el nav tenemos el logo de la web junto con el nombre al pinchar en ellos nos lleva a esta misma página web.

En la barra de búsqueda escribiremos el producto que queremos buscar, a la derecha de la barra de búsqueda seleccionamos el orden que queremos que nos aparezcan los productos siendo por defecto primero los productos del supermercado Dia, después los de Amazon y por último los de la API Open Food. Al pinchar en el botón Buscar realizaremos la búsqueda.



Una vez completada la búsqueda en este caso “Leche con orden por defecto” obtenemos el siguiente resultado:



Para cada producto obtenemos a la izquierda el nombre, seguidamente su precio, el supermercado del que proviene, un enlace que accede a la página original del supermercado donde está el producto sacado, la imagen del producto y por último el logo del supermercado donde hemos sacado ese producto.

7. Bibliografía

https://elvex.ugr.es/decsai/information-systems/slides/40%20Data%20Integration.pdf

https://elvex.ugr.es/decsai/information-systems/slides/42%20Data%20Integration%20-%20Schema%20Matching%20&%20Mapping.pdf

https://elvex.ugr.es/decsai/information-systems/slides/P1%20Python%20Flask.pdf

https://stackoverflow.com/questions/61933492/playwright-error-target-closed-after-navigation

https://stackoverflow.com/questions/74111323/python-memory-limit-exceeded-during-google-app-engine-deployment

https://stackoverflow.com/questions/73171905/chromium-executable-doesnt-exist-for-running-playwright-within-a-deployed-googl