Facultad de Informática
MUEI

Generación de un comparador de precios de artículos de supermercado a partir de la extracción e indexació de información pública

- Recuperación de la Información y Web Semántica -

Ramiro Serantes Garrido
Juan Ramil Díaz
Javier Cancela Mato

Índice general

Capítulo 1

Contexto, dominio y datos

En este capítulo inicial definiremos el contexto de la aplicación desarrollada, así como el dominio del problema y explicaremos con qué datos trabajaremos.

1.1 Contexto y dominio

El contexto para esta aplicación es el de un agregador de productos de supermercados. Pretendemos recoger información sobre la oferta de cada uno e incorporarla en nuestra aplicación, incluyendo prestaciones adicionales como el filtrado por categoría, oferta y/o precio de producto.

1.2 Datos

Los datos con los que trabajaremos son productos de supermercado. Desde alimentación hasta parafarmacia, pasando por las categorías habituales de cualquier tienda de estas características.

Nos enfrentamos a la posibilidad de que cada supermercado incluya información diferente sobre cada producto, o la estructure de distintas maneras. Para ello, hemos decidido definir un tipo de producto con características comunes. El formato para dicho producto, explicado con un ejemplo, es el siguiente:

- **Category**: Categoría a la que pertenece el producto. Ej: Bebidas
- **Subcategory**: Subcategoría dentro de la categoría a la que pertenece el producto. Ej: Refrescos
- **Branch**: Grupos de elementos del mismo tipo. Ej: Gaseosa
- **Name**: Nombre asociado al producto.

- **Shop**: Tienda a la que pertenece el producto.
- **Badges**: Ofertas sobre el producto. Ej: 2x1.
- **Features**: Características del producto. Ej: Sin glúten.
- **Prices**: Conjunto de características que definen el precio del producto, como descuentos, precio antes, precio después, etc

Para más detalles, consultar la clase Product dentro de backend > crawler > items .

Capítulo 2

Tecnologías utilizadas

En este capítulo se enunciarán y explicarán las diferentes tecnologías utilizadas.

2.1 Python

Python es un lenguaje de programación interpretado y dinámico. Es también multiparadigma, ya que se puede utilizar tanto para programación imperativa como orientada a objetos. Es uno de los lenguajes de programación más populares y es de código abierto.

2.2 Scrapy

Scrapy es un framework de crwaling y scraping escrito en Python y de código abierto. En concreto, nosotros lo usaremos en su formato de librería para Python.

2.3 Elasticsearch

Elasticsearch es una herramienta de indexación basada en Lucene. Se basa en una API RESTful y documentos JSON para implementar un motor de búsqueda de texto distribuido.

2.4 Angular

Angular es un marco de desarrollo de interfaces basadas en componentes para aplicaciones web de una sola página. Funciona con Typescript en sus últimas versiones y es de código abierto..

2.5 Material

Material es un sistema de diseño de interfaces y experiencia de usuario de código abierto que propone componentes adaptables a distintos marcos de desarrollo.

2.6 Tailwind

Tailwind es un marco de desarrollo para CSS, premitiendo aplicar estilos de manera ágil y rápida.

Capítulo 3

Casos de búsqueda solucionados

Con la solución implementada, podemos resolver la necesidad de información de cualquiera que desee reducir el precio de sus futuras compras, buscando productos más asequibles. Para poder realizar esto, hemos diseñado los siguientes fitros para los casos de búsqueda de productos.

- **Filtrado por Precios**: Permite catalogar los productos del supermercado poniendo franjas de precios finales deseados, seleccionándolos desde 1€ hasta mayores de 50€.
- **Filtrado por Categoría**: Al utilizar este filtro, se pueden seleccionar diferentes categorías a las que pertenecerán los productos encontrados en la búsqueda.
- **Filtrado por Oferta**: Al seleccionar esta opción, solo se mostrarán en la búsqueda aquellos productos que estén actualmente bajo algún tipo de oferta.
- **Filtrado por Nombre**: Se permite introducir el nombre del producto que se desea buscar.

Todos estos filtros son complementarios, es decir, se pueden usar varios al mismo tiempo para mostrar resultados de búsqueda más precisos.

Capítulo 4

Como usar la aplicación

Para emplear esta aplicación es necesario tener instaladas las siguientes herramientas:

- Python 3.10 y su gestor de paquetes Pip
- Docker y Docker-compose
- Npm 8.18 o superior
- Node 14.15.5 o superior
- Navegador Google Chrome y ChromeDriver (https://chromedriver.chromium.org/downloads)

Primero, es neceasrio levantar el cluster de ElasticSearch. Para ello, debe dirigirse con el terminal al directorio */backend/elastic* y ejecutar *docker-compose up*. Esto levantará una instancia de ElasticSearch en *localhost:9200*.

NOTA: En caso de que fuese necesario modificar la información de esta instancia, los datos de acceso son los siguientes.

- Usuario: elastic
- Contraseña: riws

A continuación será necesario instalar las dependencias que usará Python. Estas vienen indicadas en el fichero *requirements.txt*, por lo que es necesario dirigirse a */backend* y ejecutar *pip install -r requirements.txt*.

Ahora debe ejecutarse el script *main.py* mediante *python main.py*. Esto se encargará de comprobar si los productos ya están descargados, y descargarlos en caso de que no lo estén; luego creará un índice en ElasticSearch e indexará los productos obtenidos en él.

AVISO: el proceso de crawling necesita mucho tiempo, aproximadamente 20-25 minutos a causa de que necesita esperar a que el contenido generado dinámicamente se cargue en el navegador.

Finalmente, para arrancar el frontend es necesario dirigirse a la carpeta */frontend/app* con el terminal y ejecutar *npm install* para descargar las dependencias del frontend. Para ejecutar el frontend se ejecutará *npm start*, el cual arrancará un servidor que cargará el frontend y será accesible con el navegador desde *localhost:4200*

Capítulo 5

Errores conocidos

5.1 Backend

- Es posible que el crawler se quede colgado. Para ello es necesario pulsar Ctrl + C, esperar a que finalice el proceso y volver a empezar.

5.2 Frontend

- El frontend no es capaz de completar las solicitudes a Elastic. Se cree que la causa es que faltan los certificados .cert de Elastic en las cabeceras de las peticiones que realiza Angular.