RECUPERACIÓN DE INFORMACIÓN Y RECOMENDACIONES EN LA WEB

Informe Final - Grupo 4

Facultad de Ingeniería, Udelar - 2017

Damián Piccini	5.470.192-7
Nicolás Luraschi	4.898.239-5
Isabel Ivagnes	4.649.675-6
Daniel Porzio	4.623.009-5

Índice

1- Introducción

El siguiente documento describe el diseño y la implementación de un sistema como solución frente al problema que se enfrentan los usuarios a la hora de buscar o filtrar artículos de compra desde la web debido al gran número de oferta existente. Este sistema consiste en una aplicación web dirigida a productos tecnológicos, el cual facilita el trabajo y reduce el tiempo de los usuarios a la hora de buscar un producto, proporcionando los precios del producto deseado asociado a diversos proveedores.

2- Descripción del Problema

Desde que la economía de los países se ha globalizado y la industria de los productos tecnológicos ha aumentado sus ventas de forma exponencial en los últimos años, el consumidor final se encuentra cada vez más en una situación de indecisión a la hora de elegir qué producto comprar y en donde hacerlo ante la amplia oferta existente. Es entonces donde entra nuestra propuesta para solucionar el problema planteado, desarrollando una aplicación web cuyo principal objetivo es centralizar la información que se encuentra en las páginas web de venta de artículos de tecnología y brindarla a los usuarios de manera unificada y amigable. De esta forma la búsqueda de productos por parte de dichos usuarios se vería simplificada, permitiéndoles filtrar y clasificar por precio, calidad, y diversas categorías que van a variar de acuerdo al tipo de producto.

Supóngase que uno desea comprar un producto, por ejemplo un celular, pero desconoce el modelo que desea y que vendedores lo ofrecen, por lo que debería realizar una búsqueda en varios sitios web y consultar por la disponibilidad del artículo. En cambio utilizando nuestro servicio, los clientes solamente deberán acceder a un sitio e ingresando el modelo del producto, la página les proporcionará los diferentes proveedores que lo venden. El usuario podrá también ordenar los productos resultantes en las búsquedas

según precio, stock, y otras variables acordes de manera de encontrar la opción que se adecue mejores a sus pretensiones.

3- Enfoque de la solución

La solución planteada apunta a productos de tecnología en sitios web de compra de Uruguay. Debido a la gran variedad de productos que ofrece el mercado se resolvió por comenzar a trabajar únicamente con celulares. La razón por la cual optamos por los dispositivos móviles es que son de los productos más demandados por los usuarios y a su vez la información se presenta, en general, de forma uniforme y homogénea en la mayoría de las páginas web. Esto quiere decir que podemos encontrar los mismos datos en la mayoría de los sitios, cosa que no pasaba con otros artículos, como por ejemplo: computadoras portátiles o tablets.
Las fuentes para obtener los datos fueron elegidas acorde a la cantidad de productos a la venta disponibles, popularidad y uso entre los compradores uruguayos. Luego de hacer una investigación sobre diferentes sitios web de productos tecnológicos se resolvió trabajar con los siguientes:

- www.motociclo.com.uy
- www.zonalaptop.com.uy
- www.zonatecno.com.uy

Para recabar toda la información necesaria sobre los productos y sus proveedores se decidió llevar a cabo técnicas de web scraping en los sitios ya mencionados.
Para que el usuario pueda realizar la búsqueda de productos es necesario implementar un buscador donde se pueda introducir texto y ofrecer diferentes criterios de filtrado. Para esto último definimos los siguientes filtros: modelo o marca, precio, etc, los cuales para el grupo son los más utilizados para buscar celulares.

4- Diseño e Implementación

4.1- Herramientas Utilizadas

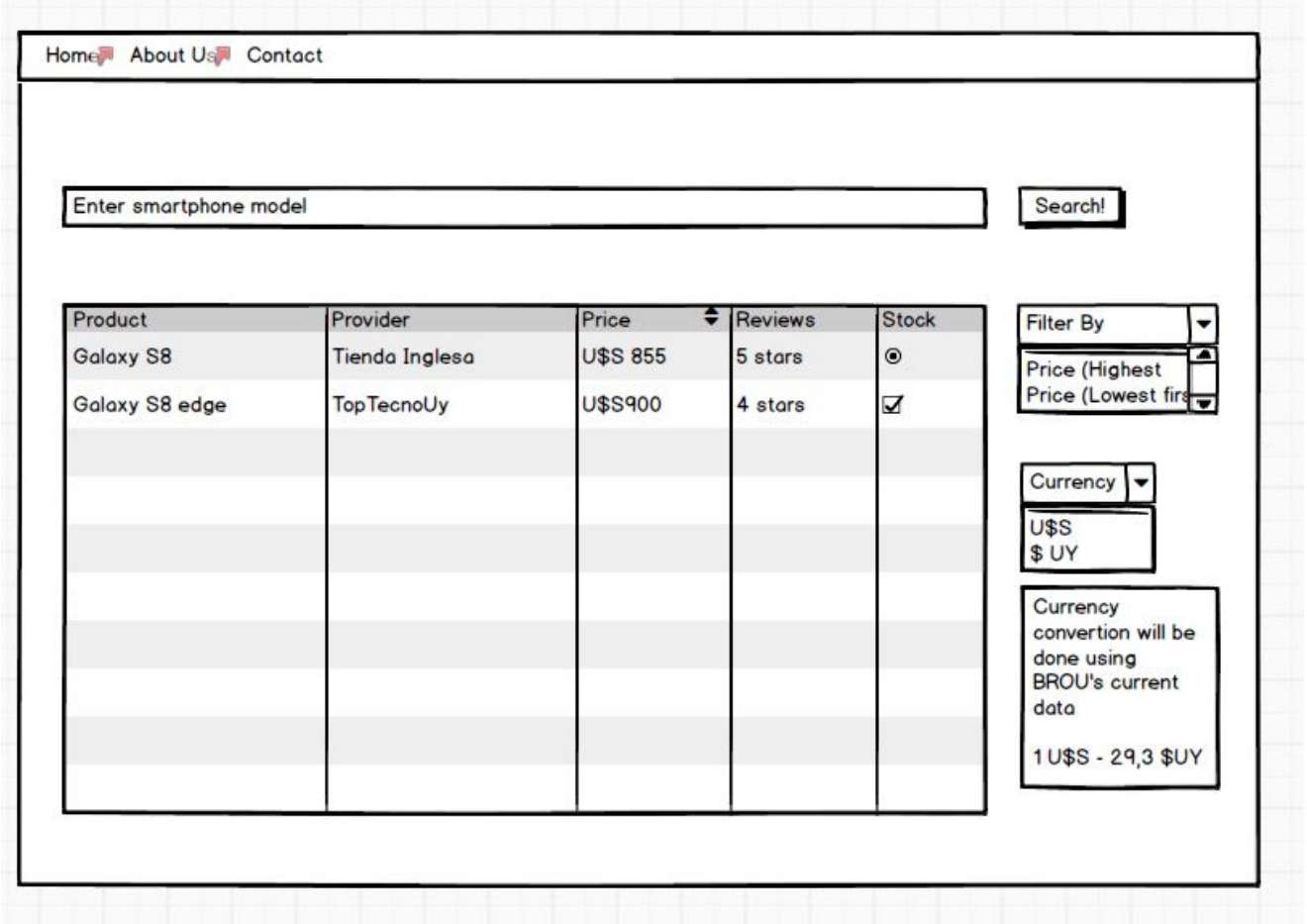
Para implementar la aplicación web se utilizó Ruby on Rails, tanto para el backend como para el frontend. Para las funcionalidades de recuperación de información se utilizaron las siguientes herramientas, las cuales pueden integrarse fácilmente con Rails:

- Nokogiri: gema que ofrece diferentes funcionalidades para web scraping.
- Mechanize: librería que permite llenar formularios e interactuar con otros elementos de una página web.
- CasperJS: herramienta de Javascript que se suele utilizar para testear frontend. Presenta la ventaja con respecto a Mechanize de que puede ejecutar código Javascript (Mechanize no puede), lo cual es útil en el contexto de algunos sitios que utilizan frameworks SPA que interactúan asincrónicamente con sus servidores de datos.
- MongoDB: como manejador del sistema de base de datos. Se optó por un manejador no relacional para poder persistir los datos en formato JSON. Se almacenaron los objetos más solicitados con el fin de incrementar la performance de la aplicación ahorrandonos de hacer consultas contra los sitios web externos.
- ElasticSearch: este servidor de búsqueda nos permitió indexar y acceder a los datos de manera rápida y eficiente.

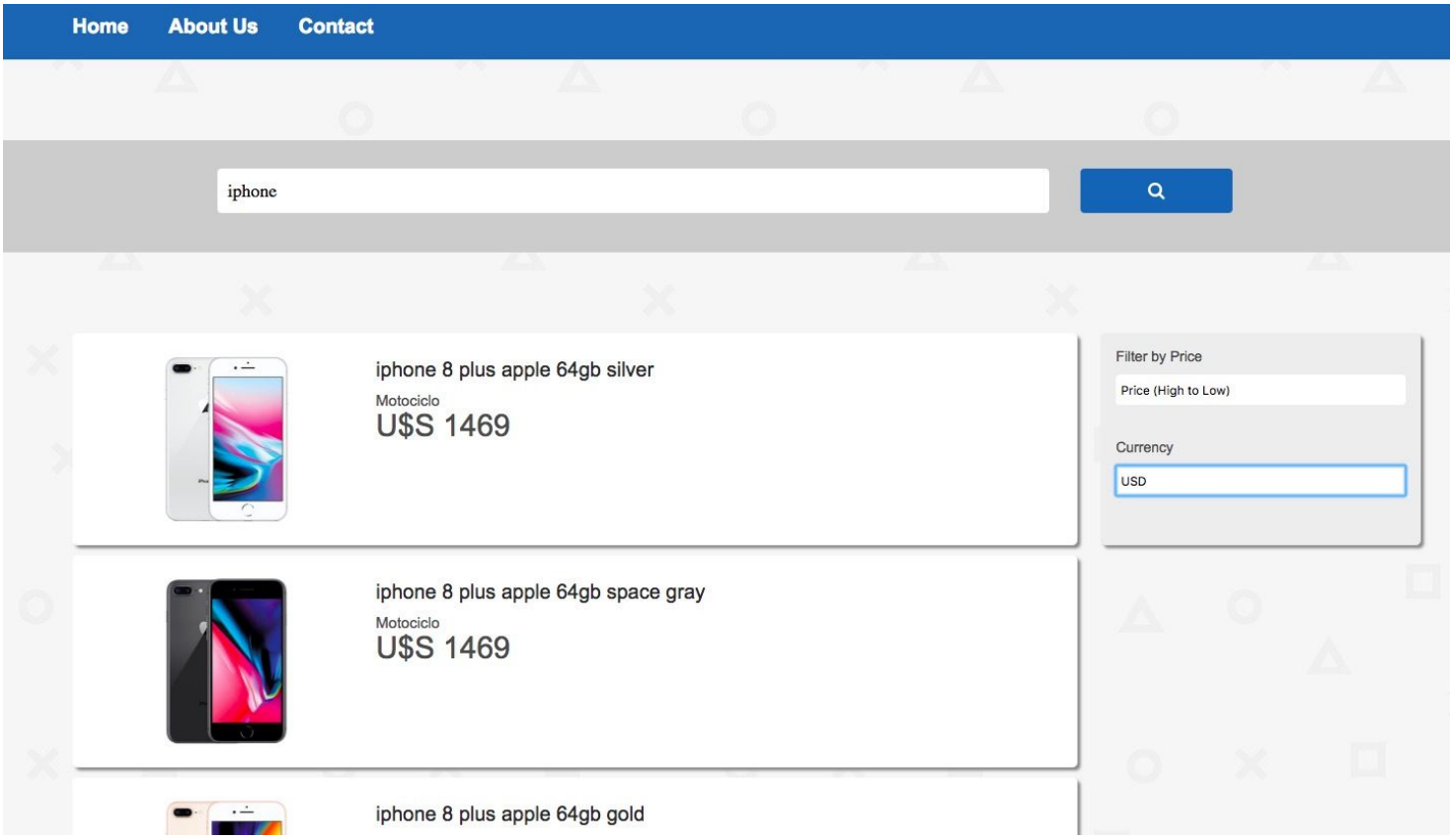
En una etapa inicial, se realizó un maquetado del diseño de la aplicación, en el cual se definieron los componentes básicos de diseño a implementar. Para realizar dicho diseño, se utilizó la herramienta *Balsamiq*.

4.2- Front-end

A continuación se muestra el maquetado mencionado en la sección anterior que se usó como base para el desarrollo del frontend de la aplicación. Ciertos componentes fueron re-diseñados para mejorar la usabilidad y el flujo de la aplicación. Por ejemplo, la tabla de productos fue sustituida por "cartas" de productos, las cuales presentan de una forma más actual o moderna, para mostrar en cada una los datos e imágenes correspondientes.
Dada la simplicidad del diseño propuesto, no se utilizó ningún css framework para realizar la implementación.

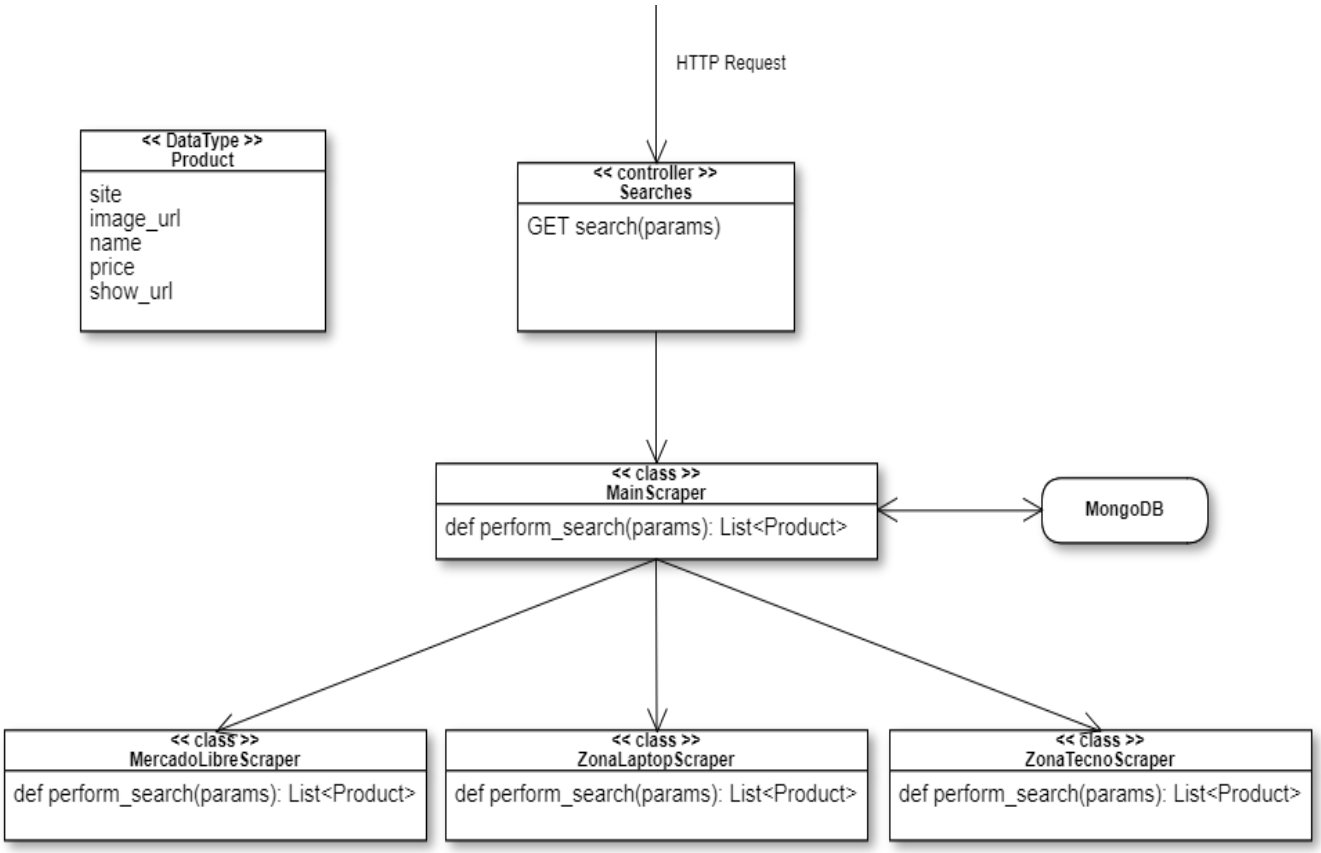


Diseño del sitio actualizado:



4.3- Arquitectura

El sistema tiene una arquitectura cliente-servidor, donde el cliente está basado en Javascript y HTML. Se cuenta con una base de datos MongoDB para almacenar los productos recuperados, incluyendo su imagen en caso de que exista .El sistema cuenta además con un servidor ElasticSearch para poder acceder de forma rápida a los datos. El módulo del servidor es el encargado de realizar el web scraping para obtener la información, guardarla en la base de datos y comunicarse con ElasticSearch para realizar las consultas y comunicarse con el cliente para devolver la información de dichas consultas.



Para la lógica de scrapeo se propuso la siguiente solución: se implementó una clase abstracta llamada "*MainScraper*" que define un método llamado "*perform_search*", y se construyó una clase por scraper por cada sitio específico que heredan de esta clase abstracta. Una ventaja de este modelo es la escalabilidad: para agregar un nuevo sitio al conjunto de los ya existentes, basta con agregar una clase que herede de "*MainScraper*" e implemente el método "*perform_search*"

Otra ventaja de implementar por separado cada scraper específico a cada página es quela recuperación de información puede realizarse de manera concurrente. Cuando un usuario quiere buscar un producto, se disparan todos los scrapers al mismo tiempo, y luego se unen los resultados que cada uno retorna, por lo que la búsqueda es más eficiente que si se llamaran uno por uno. Como todos implementan la misma interfaz propuesta por "*MainScraper*", no existen problemas de incompatibilidad de datos, y se asegura que en promedio el tiempo de ejecución de las búsquedas será igual al del scraper más lento.

Con el fin de optimizar la performance de las búsquedas también utilizamos una base de datos a modo de caché. Cada vez que se realiza una búsqueda, primero se corrobora que

los productos solicitados están guardados en la base de datos. Si están, y el tiempo transcurrido es menor a cierto umbral (predefinido desde la última vez que se realizó la consulta que arroja esos productos), se retorna al usuario los productos almacenados en base de datos. Luego de probar repetidas veces la búsqueda pudimos comprobar que la performance aumentó considerablemente al eliminar el tiempo total de scrapeo que de otra forma se requeriría. Por otro lado, si transcurrió un cierto tiempo mayor al del umbral predefinido, entonces se considera que los productos en la base de datos están desactualizados, y se procede a borrarlos y a realizar nuevas búsquedas contra los sitios originales.
La principal razón de definir dicho umbral de tiempo es que los precios de los productos pueden cambiar, ya sea por ofertas o promociones, o porque nuevos productos que salen al mercado, alterando así los precios de los productos ya existentes. Ésto implica que cualquier producto previamente persistido en base de datos puede tener cambiado , por lo cual una estrategia que mantenga inmutables los resultados en base de datos no nos pareció ser la adecuada.
En un principio también se pensó en actualizar la búsqueda y extracción de los datos de forma automática de forma de asegurar la consistencia de la información en base con la de los sitios web y de aminorar los tiempos de las consultas para los usuarios. Debido a cuestiones de tiempo, ésto no fue posible, por lo que decidimos que mantener la base de datos como caché era una solución aceptable.

4.4- Análisis y extracción de datos

Como se mencionó anteriormente los datos de los productos fueron extraídos de 3 páginas web. A continuación se describe el proceso realizado para cada una de ellas:

- Motociclo y Zona Laptop: Como ambas páginas no presentaban Javascript para cargar ninguno de los elementos en la página, se pudo realizar el web scraping utilizando sólo la librería Mechanize. El proceso para ambas fue muy similar. Todos los productos estaban organizados de acuerdo a su categoría, por lo que fue necesario posicionarse en la categoría de celulares. Una vez allí se obtuvo la página actual con todos los celulares y se fue iterando sobre cada uno. Al contener todos los tipos de celulares en la misma categoría en cada iteración hubo que ir filtrando los productos que no se correspondieran con la consulta realizada por el usuario. Para los filtros se examinó que la parte del nombre contuviera por lo menos alguna

de las palabras de la consulta. Por último se devuelven los datos obtenidos en un arreglo que contiene una estructura de diccionario por cada artículo. La página de Motociclo está paginada por lo que también hubo que iterar sobre las diferentes páginas de resultados. Por otro lado Zona Laptop tiene el problema de que en la misma página muestra artículos que no eran celulares pero que están relacionados con los mismos, como por ejemplo carcasas o vidrios templados. Por lo que también hubo que tener que filtrar este tipo de artículos, teniendo especial cuidado de no agregarlos ya que parte del nombre contenía palabras incluidas en la consulta inicial.

- Zona Tecno: en el caso de este sitio fue necesario utilizar una herramienta que pudiera ser capaz de procesar Javascript, debido a que las consultas sobre productos se realizan asincrónicamente contra un servidor de datos, y no hay pasos intermedios en los cuales se pueda obtener un documento html para su scrapeo. CasperJS fue la herramienta utilizada para solucionar el problema y scrapear, y éste se realizó siguiendo el siguiente esquema: primero se configuró al buscador del sitio para que limitara su búsqueda exclusivamente a celulares. A diferencia de los sitios mencionados anteriormente, Zona Tecno provee dicha funcionalidad.
  Luego se ingresó el texto que el usuario introdujo en el buscador del sitio, y se procedió a disparar el evento Javascript en el sitio que realiza la consulta en su base de datos. Una vez que se detectó que el sitio terminó de mostrar en pantalla los resultados se procedió a interpretarlos y a armar un arreglo de objetos con los datos especificados en el diagrama de arquitectura.
  Por una cuestión de complejidad y tiempos de entrega no se realizó una búsqueda exhaustiva por todas las páginas de resultados que el sitio retorna, sino que solo se retornan los resultados obtenidos en la primera. La navegación por las diversas páginas resulta una tarea bastante compleja en Casper debido a la estrategia asíncrona de búsqueda que el sitio implementa, y debido al hecho de que la cantidad de páginas puede ser variable según el producto buscado.

Los datos que se obtuvieron por cada producto fueron nombre del sitio fuente, url de la imagen en el sitio, nombre del producto (éste incluye modelo y marca), url a la página específica del producto, precio y moneda en la que se encuentra el mismo. En un principio se pensó en traer solamente los celulares cuyo nombre coincidiera exactamente con la consulta ingresada por el usuario pero al final se optó por traer y mostrar también todos aquellos que coincidieran parcialmente. Ésto fue debido a que todos los sitios fuentes tampoco "*matcheaban*" exactamente por consulta, por lo que iba a ser más complejo el

filtrado. Por ejemplo si el usuario ingresa en el buscador “*samsung S6*” como resultados va a obtener también todos los celulares cuyo nombre contenga la palabra “*samsung*”.

5- Conclusiones

Podemos decir que se logró implementar un sistema funcional, el cual logra centralizar y homogeneizar la información de los productos celulares en un solo sitio. Es posible buscar y filtrar información según las características que se especificaron, sobre todo de forma fácil y sencilla para los usuarios. Por lo tanto podemos afirmar que nuestra aplicación cumple los requerimientos básicos definidos por el equipo para el alcance del proyecto.

Como principales conclusiones surgidas de los problemas que se tuvieron que resolver a la hora de la implementación de la aplicación, destacamos los puntos que consideremos de mayor importancia:

- Normalización de la información: dado que los diversos sitios manejan y muestran de manera distinta la información, consideramos fundamental la definición de un esquema de datos propio que capture la mayor cantidad de información posible. En particular encontramos que todos los sitios mostraban información en común para sus productos, como por ejemplo modelo, precio, una imágen del mismo, etc. Tomando estos atributos en cuenta a la hora de diseñar el modelo de datos utilizado en la aplicación, logramos capturar una gran cantidad de información relevante de todos los sitios.
- Escalabilidad de la aplicación: al proponer un modelo de de datos y una interfaz común a todos los sitios desde los que se recupera información se consigue maximizar la escalabilidad del sitio. Para agregar nuevos módulos que scrapeen información de otros sitios, alcanza con inyectarlos en el sistema, hacer que sus métodos hereden de la interfaz propuesta, y retornen así su información siguiendo el esquema de datos detallado.
- Utilización de base de datos: dado que la recuperación de la información directamente desde los sitios a veces puede ocurrir muy lentamente (pueden llegar a ser varios segundos), encontramos que la utilización de una base de datos para almacenar resultados de búsquedas previas aporta una gran mejoría en el tiempo promedio de espera a la hora de realizar una búsqueda.

- Paralelización de las búsquedas: dado que todos los módulos de scrapeo están desacoplados entre sí, es posible ejecutar en paralelo las funciones que recuperan información para cada sitio. De esta manera se logra que en promedio el tiempo de ejecución total de una consulta sea igual al tiempo que le toma al scraper más lento recuperar la información solicitada. Es menester aclarar que el tiempo de ejecución de una consulta de un scraper no depende únicamente de la calidad del código, sino que también de factores tales como la tecnología de scrapeo utilizada y la disponibilidad de sitio. Por ejemplo, el scraper que se basa en la herramienta CasperJS presenta en promedio un tiempo de respuesta superior a los otros debido a que no está implementado en el lenguaje nativo de la aplicación (Ruby), y también porque a que tiene que ejecutar grandes cantidades de código Javascript provenientes del sitio. En general tomará más tiempo recuperar información de un sitio con una arquitectura más compleja (e.g: arquitectura Single Page Application) que de un sitio que siga un enfoque arquitectónico más tradicional (e.g: arquitectura Model View Controller).

6- Trabajos a Futuro

Durante el desarrollo de la aplicación web fueron surgiendo ideas para nuevas funcionalidades o simplemente para mejoras de la aplicación o soluciones a problemas, que nos parecen interesantes y posibles para aplicar en el futuro. Las mismas se describen a continuación:

- Ampliar la cantidad de sitios fuente para la recuperación de los datos.
- Ampliar las categorías de productos, centrándonos siempre dentro de productos tecnológicos, como por ejemplo computadoras portátiles.
- Poder scrapear el sitio de Zona Tecno de forma más eficiente e implementar la iteración sobre todas las páginas que fueron devueltas como resultado en lugar de sólo la primera.
- Poder extraer más información sobre los productos, haciendo hincapié en las características técnicas que son importantes e influyen la hora de buscar y comprar. Además agregar los índices y filtros correspondientes para que los

usuarios puedan buscar sobre ellos también. También sería útil contar con el dato de si los productos se encuentran en ese momento en stock o no.

- Automatizar la extracción de los datos de forma que sea periódicamente, por ejemplo cada dia o cada semana, permitiendo así tener al día la información y ser más eficientes a la hora de las consultas. Además contar con una limpieza de datos viejos automatizada.
- Implementar la funcionalidad de poder crear una cuenta de usuario para guardar preferencias de productos y poder recibir sugerencias de productos similares o notificaciones en caso de disponibilidad de stock u ofertas.

7- Referencias

Sitios web de donde se extraen los datos:

Zona Laptop: http://zonalaptop.com.uy/
Zona tecno: http://www.zonatecno.com.uy/
Motociclo: http://www.motociclo.com.uy/