UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Implementación de software para extracción y manejo de datos de productos

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Oscar Hernández Mendoza

ASESOR DE INFORME

M.A. Luis Yair Bautista Blanco

Ciudad Universitaria, Cd. Mx., 2021

Índice

ByPrice

Esta empresa fue formada en septiembre del 2015 por Víctor Casanova y Ricardo Gamba, a quienes poco después se unió Julian Stastny con el fin de ofrecer al público un servicio que permitiera comparar todos los precios de medicamentos disponibles en el mercado mexicano. Esto debido a que llegaron a percibir diferencias de hasta 50 % en el precio de exactamente el mismo producto, lo cual afecta directamente al consumidor, en caso de no tener conocimiento de dicha diferencia. Fue entonces cuando surgió la idea de crear una página web, donde se difundieran todos los precios disponibles de cada medicamento ofrecido por las principales farmacias del país, permitiendo compararlos entre sí.

Durante el año 2016 se llevó a cabo la etapa de desarrollo; diseño, y pruebas tanto para el funcionamiento como el rendimiento de la infraestructura del sitio "byprice.com". Una vez con la primera versión funcional se lanzó el servicio al público. El crecimiento de la plataforma se dio de forma rápida, a tal punto que finalizando el año, se abrieron las puertas a negociaciones con empresas de la industria farmacéutica y alimentaria, con el fin de realizar comparaciones estadísticas directamente sobre los productos del interés que cada una de ellas tenía. Fue así como en 2017 surge ByPrice Intel, servicio dirigido a otras empresas.

En 2018 se sentaron las bases para la expansión internacional de la compañía; se consiguieron financiamientos, se comenzó a recopilar información de producto en países como EEUU y Puerto Rico. Además se lanzó la aplicación móvil, junto con la expansión del catálogo a todos los productos de supermercado; gracias a la inclusión de información de tiendas de autoservicio, lo cual aumentó el número de productos disponibles para comparar.

Figura 1. Logo actual de ByPrice

Contexto actual y visión a futuro

Actualmente se está trabajando en el mejor posicionamiento de la página en los motores de búsqueda más populares, en especial Google, para lo cual se llevan a cabo prácticas de SEO (*Search Engine Optimization*), que busca que la empresa aparezca en las primeras páginas cuando se realiza la búsqueda

de ciertas palabras clave, lo que permite aumentar el número de visitantes al sitio, que pueden convertirse en usuarios frecuentes de ésta, en caso de lograr que su experiencia sea agradable durante el uso de la plataforma.

También se está trabajando en aumentar la cantidad de productos relacionados entre distintas tiendas para permitir mayor variedad de comparaciones en cuanto a precio, además de que se planea llegar a relacionar productos similares; es decir, que compartan muchas características a pesar de no ser el mismo, de forma que se puedan comparar productos de distintas marcas y presentaciones; dando la opción al usuario de elegir un producto como el que busca, pero más barato o de mayor calidad.

En cuanto a la expansión internacional, se lanzó la versión beta de la página para EEUU, con productos de Walmart y Amazon, aumentando gradualmente el número de supermercados o plataformas de venta de productos para captar cada vez más usuarios y presentar el proyecto a inversionistas de aquél país.

Organigrama

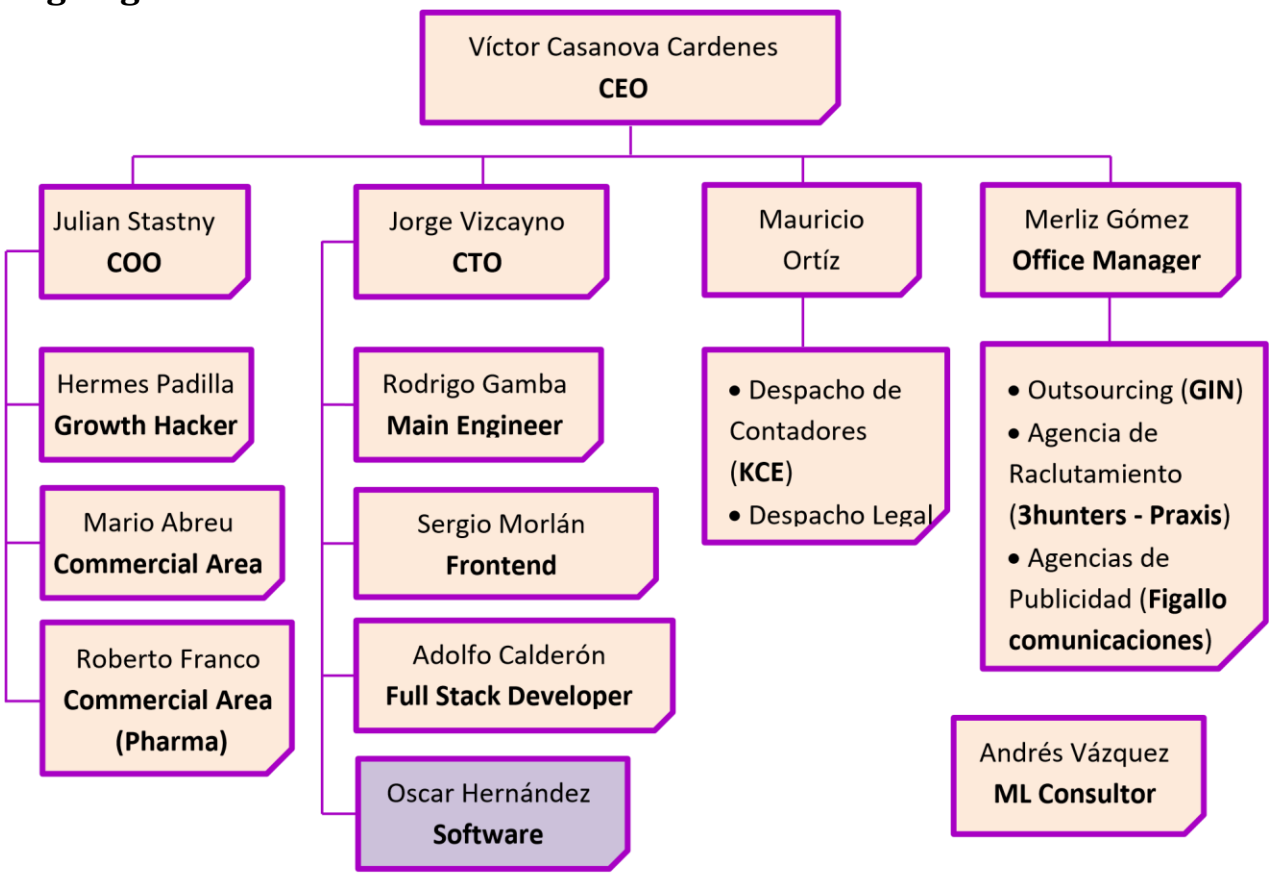

Figura 2. Organigrama de la empresa

Servicios que la empresa proporciona

Business to Consumer

Éste fue el primer modelo de negocio. Consiste en ofrecer al consumidor un servicio que les permite comparar precios del mismo producto disponibles en múltiples comercios de acuerdo con la región en la que se encuentran, permitiendo visualizarlos directamente en el sitio de la empresa que lo comercializa.

Existen además otras características de la página que pretenden mejorar la experiencia del usuario, como el servicio de alerta en caso de que un producto que haya dejado de estar disponible vuelva a estarlo, posibilidad de crear listas de productos que acostumbre comprar, una sección de preguntas y respuestas, etcétera. Igualmente se incluyen elementos que permiten la retroalimentación con el cliente como son la valoración de producto y la posibilidad de dar opiniones o comentarios respecto al sitio. Esto con la finalidad de hacer más agradable su recorrido por el mismo.

Recientemente la empresa extendió el alcance del servicio mediante la creación de una aplicación gratuita que cumple con los mismos fines que los de la página web.

Business to Business

Gracias a la información de precios por comercio que se ha logrado recopilar, es posible ofrecer un servicio que permite a las empresas realizar el seguimiento de precios de los productos de su interés; pueden conocer su variación durante un periodo específico de tiempo, en cada uno de los comercios disponibles en las regiones que se requiera. Dicho servicio se proporciona a través de una plataforma web de BI (*Business Intelligence*) dedicada a cada cliente.

En primera instancia, se le ofrece una prueba limitada de la plataforma a la empresa que tiene potencial de convertirse en cliente, con el fin de que conozcan qué se puede hacer con ella y si puede adaptarse a sus necesidades.

El precio base por el uso del sitio se define de acuerdo con la cantidad de productos y comercios a rastrear, acordados con el cliente. Debido a que cada empresa tiene requerimientos propios, hay una etapa de pruebas para sus usuarios, capacitaciones y pequeños ajustes para que la plataforma se adapte completamente a la necesidad del cliente, ajustando también el precio del servicio. Una vez adquirido el servicio, también se ofrece el soporte y mantenimiento del sitio.

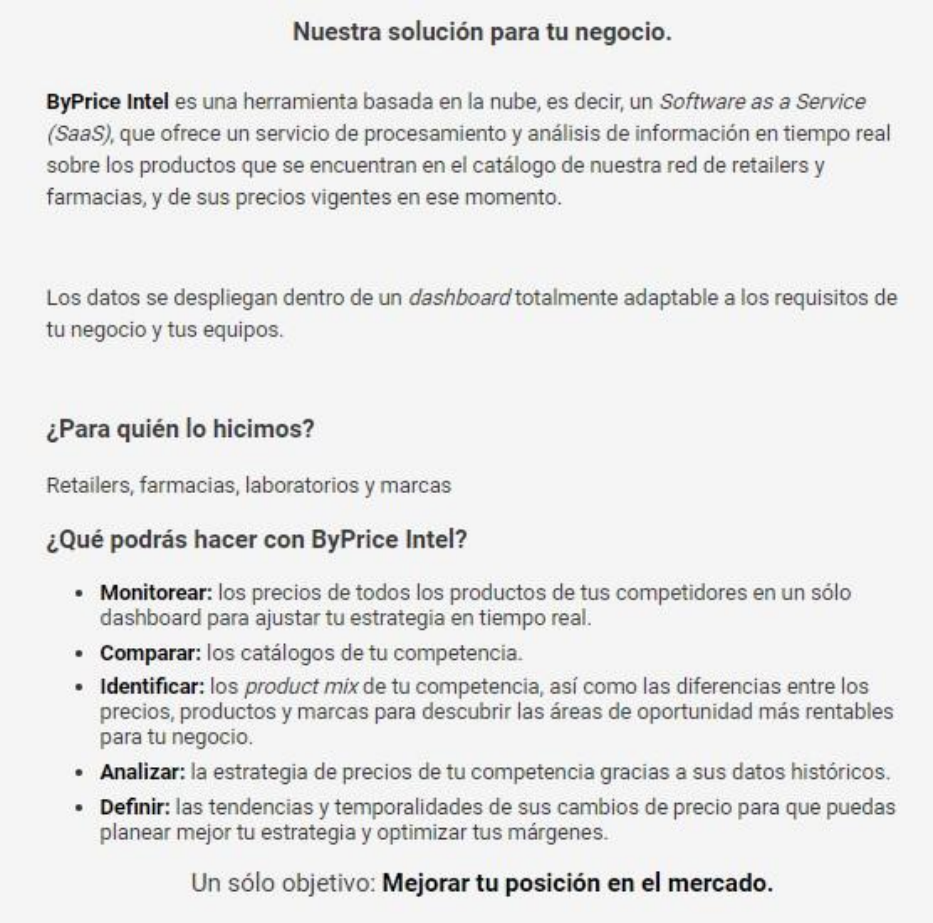

Figura 3. Captura de pantalla de infografía en Home Page de byprice.com (2019) [1]

Infraestructura de IT

El sitio web de la empresa es la única capa de toda la infraestructura con la que el usuario interactúa; detrás, existe un conjunto de aplicaciones que trabajan conjuntamente para poder brindar el servicio. En la base, se encuentran los *Web Scrapers* que se encargan de la obtención de la información de precios y productos en cada comercio. Después de esa recolección, los datos son enviados a una serie de "bloques" de procesamiento (microservicios) que validan, reformatean y guardan la información en bases de datos que posteriormente son consultadas mediante servicios web por el *frontend* de la página de internet, que representa la interfaz directa con el usuario.

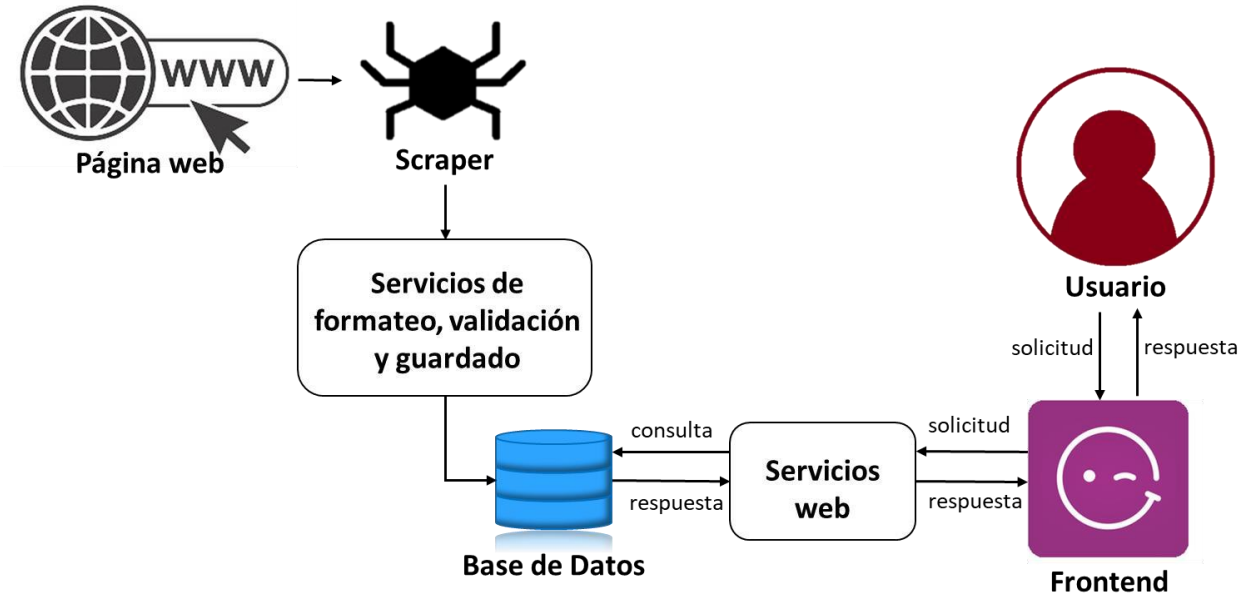

Figura 4. Diagrama de la infraestructura básica del flujo de información

Tanto los *scrapers* como los servicios que permiten guardar la información en las bases de datos se comunican mediante un "bróker de mensajería" o *message broker*, éste se define como un tipo de software que funge como mediador de comunicaciones entre aplicaciones, permitiendo intercambio de mensajes independientemente del lenguaje en el que estén programadas, haciendo posible la independencia entre ellas y su escalabilidad.

El bróker que se usa en la infraestructura se llama RabbitMQ, el cual funciona básicamente con cuatro elementos; el emisor del mensaje, un "intercambio" o *exchange*, una "cola" o *queue* y el receptor. El emisor publica el mensaje al intercambio que, dependiendo de su configuración, lo dirige a una o más colas. Éstas a su vez son "escuchadas" por receptores, que se encargan de manejar el mensaje (Ver Figura 5) [2].

Además de los servicios que manejan los datos existe otro de mucha importancia, ya que gracias a éste, es posible realizar la comparación de los productos entre comercios. Se trata del servicio denominado *matching* que se encarga de relacionar los artículos de una con otra tienda.

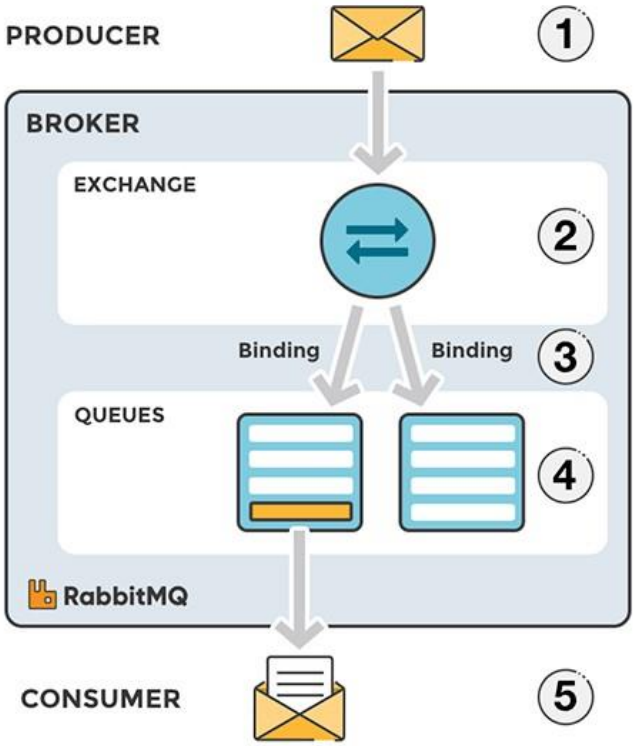

Figura 5. Diagrama de Funcionamiento de RabbitMQ ("RabbitMQ Exchanges, routing keys and bindings", 2015) [3]

Servicios en la nube

Debido a la cantidad de datos que necesitan procesarse y almacenarse, la empresa ha recurrido al uso de servicios web como Azure, Amazon Web Services y Google Cloud Platform, que ofrecen múltiples herramientas y recursos de cómputo.

A grandes rasgos, los servicios ponen a disposición de la empresa cierto número de máquinas virtuales según la demanda, con la configuración elegida por el usuario (versión de sistema operativo, memoria y CPU disponible, etcétera) para ejecutar los procesos necesarios. Adicional a ello, se maneja un software de código abierto llamado Docker, el cual permite encapsular distintas aplicaciones en un contenedor virtual, cada una con sus propias dependencias y recursos, los cuales se especifican en un archivo de configuración llamado por convención Dockerfile. Estos contenedores permiten flexibilidad al momento de escalar, desplegar o reiniciar algún microservicio, y además son de suma utilidad para mantener un buen funcionamiento y mantenimiento general de la infraestructura.

Tanto los servicios de administración de datos como los de almacenamiento y consulta se encuentran corriendo en plataformas web, que también proporcionan servicios de mantenimiento, autoescalamiento y soporte técnico. Que de otra forma tendría que ser cubierto por el equipo de IT de ByPrice.

Minado de datos

El Minado de Datos o *Data Mining* es un proceso de gran importancia en el "Descubrimiento de Conocimiento de Bases de Datos" o *KDD* por sus siglas en inglés y se define como el proceso de inferencia de algoritmos que se encargan de la exploración de grandes cantidades de datos, del descubrimiento de patrones y relaciones entre ellos, y del desarrollo de modelos con el fin de analizarlos y realizar predicciones de fenómenos de distinta índole [4].

Este campo integra herramientas de estadística e inteligencia artificial con manejo de bases de datos para analizar grandes colecciones de datos (conjuntos de datos o *data sets*). El Minado de Datos se aplica ampliamente en negocios (aseguradoras, bancos, supermercados), investigación científica (astronomía, medicina), y seguridad gubernamental (detección criminal o terrorista).

En el año 1996 un consorcio de empresas dedicadas al mercado de la minería de datos creó una metodología con el fin de formalizar y estandarizar dicho proceso. En ese entonces los nombres de las empresas eran Daimler-Benz, ISL, y NCR. La metodología fue denominada "Proceso Estándar de la Industria para la Minería de Datos" o CRISP-DM por sus siglas en inglés. Ésta consta de seis pasos, descritos a continuación [5].

- Entendimiento del negocio

Esta etapa se enfoca en conocer los objetivos y requerimientos del proceso, para permitir crear una definición del problema desde la perspectiva del minado de datos y diseñar un plan preliminar para lograr los objetivos planteados.

- Comprensión de los datos

Esta fase se inicia con un conjunto de datos, el cual se analiza para determinar su calidad y estructura, de forma que se puedan crear hipótesis sobre qué información se puede obtener.

- Preparación

Durante esta fase se construye el conjunto de datos definitivo; aquél que va a servir para alimentar la herramienta de modelado. Éste es un proceso iterativo, que requiere de tareas como la creación de tablas, selección de atributos, transformación y limpieza de los datos.

- Modelado

En esta etapa se eligen y aplican varias técnicas para el modelado, calibrando cada uno de sus parámetros. Usualmente existen múltiples técnicas para el mismo tipo de problema de minado de datos.

Algunos requieren que los datos estén en cierto formato, por lo que en ocasiones es necesario regresar al paso anterior.

- Evaluación

En esta parte del proyecto, se han construido uno o varios modelos, de cierta calidad. Es importante evaluar profundamente el modelo y revisar el proceso de su construcción para determinar si se cumplen con los objetivos establecidos para el proyecto. Un punto clave consiste en revisar si existe alguna cuestión que no se consideró lo suficiente. Al final de esta etapa se debe decidir cómo se van a utilizar los resultados del minado de datos.

- Despliegue

Una vez adquirido el conocimiento de los datos, éste se organiza y se presenta de modo que puede ser utilizado. La complejidad de esta fase depende de los requerimientos del proyecto. Es un proceso que debe adaptarse al usuario final de los datos de modo que sea sencillo implementar los modelos obtenidos.

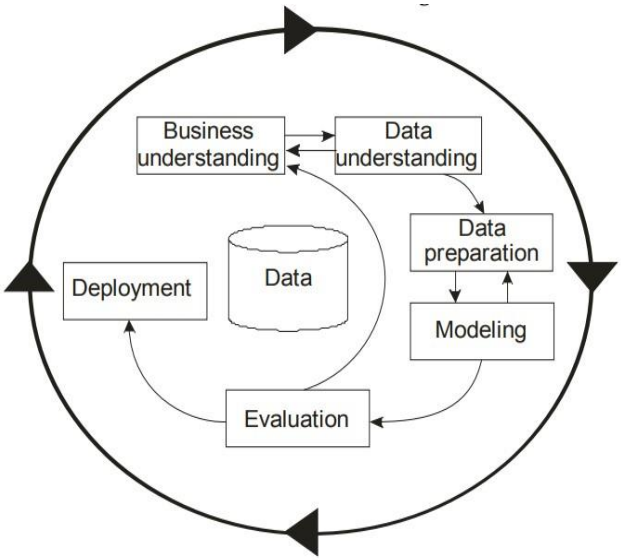

Figura 6. Modelo CRISP-DM (Chapman et al., 2000) [5]

Aplicación del Minado de Datos en la empresa

Actualmente ByPrice se involucra en el proceso de minería de datos con el fin de mejorar el servicio prestado a los usuarios tanto de B2C como de B2B. Como ya se sabe el principal objetivo es la comparación de precios de productos entre *retailers* ya que puede haber diferencias de incluso un 50% entre ellos; ese principio es en cual se basa el negocio. Para la fase de comprensión de los datos se realizó un análisis en cada una de las páginas web de los comercios para conocer qué datos se despliegan a los usuarios y cuáles de ellos son útiles para saber que se trata del mismo producto aunque se busque en otra página. La etapa de preparación consistió en definir las estructuras de las bases de datos para almacenar la información de una forma adecuada (formato y extensión de los datos), además de su extracción y limpieza. El modelado consiste en encontrar precisamente el modelo que permite localizar el mismo producto entre distintas plataformas de venta. Para la evaluación de los modelos obtenidos se realiza una observación de los resultados de productos que se etiquetaron como iguales para determinar qué tan precisos fueron. Cabe señalar que el modelado y evaluación son procesos que se ejecutan de forma cíclica en busca de mejora continua del servicio.

Personalmente, en la empresa he participado en los procesos que involucran preparación y modelado de los datos. A continuación se describen a fondo dichos procesos.

Extracción de datos

Para la extracción de información de las páginas públicas de los comercios, también llamados *retailers*, se programan *Web-Scrapers* cuyo objetivo es recorrer todo el sitio en búsqueda de los datos de interés, de forma organizada y tal que permita hacerlo de la manera más eficiente posible. El lenguaje de programación utilizado para ello es Python 3, apoyado en algunas bibliotecas que permiten hacer consultas a las páginas de internet o dar formato a las respuestas obtenidas, regularmente en fuentes HTML.

Desarrollo de Scraper

Cada *retailer* tiene distintas estructuras en sus páginas, así como diferentes flujos de información, por lo que es esencial comprender el funcionamiento del sitio para hacer el *scraper*.

El proceso para comprender cómo se da el flujo de información de cualquier sitio web empieza en su página principal o *home*, es importante observar los elementos que se cargan directamente con el HTML de la respuesta y las solicitudes asíncronas que se realizan tanto al propio internamente como a otras páginas, para cargar recursos como publicidad, métricas, contenido extra, etcétera.

Para ello, las herramientas de desarrollador en navegadores web son bastante útiles, ya que permiten visualizar el código fuente de la página, los archivos que se cargan (imágenes, código javascript, código HTML, etc.) y los tipos de solicitudes o *requests* que se realizan (Ver Figura 7).

Conocer cómo se carga la información en la página nos permite identificar de dónde proviene la información de interés que se observa en el sitio web, que en este caso son los detalles de los departamentos, las categorías, y de los productos pertenecientes a cada una de ellas. Algunos sitios cargan toda la página con código HTML, pero ésta práctica ha disminuido en los últimos años; también se recurre al uso de Javascript (cargado asíncronamente) para cargar el contenido en la página; otros utilizan las llamadas Interfaces de Programación de Aplicaciones (API por su acrónimo en inglés) para consultar la información tanto a sus propios servicios web como a proveedores externos (regularmente en formato json, XML o HTML).

Otra alternativa para obtener la información de la página es entrar al mapa del sitio o *sitemap* de la página web, ya que ahí se enlistan prácticamente todos los urls que la página contiene. La desventaja de éste método es que, en caso de que el sitio lo haya hecho público, el *sitemap* no siempre contiene toda la información o la más actualizada, por lo que es recomendable utilizarla como una técnica complementaria para extraer la información.

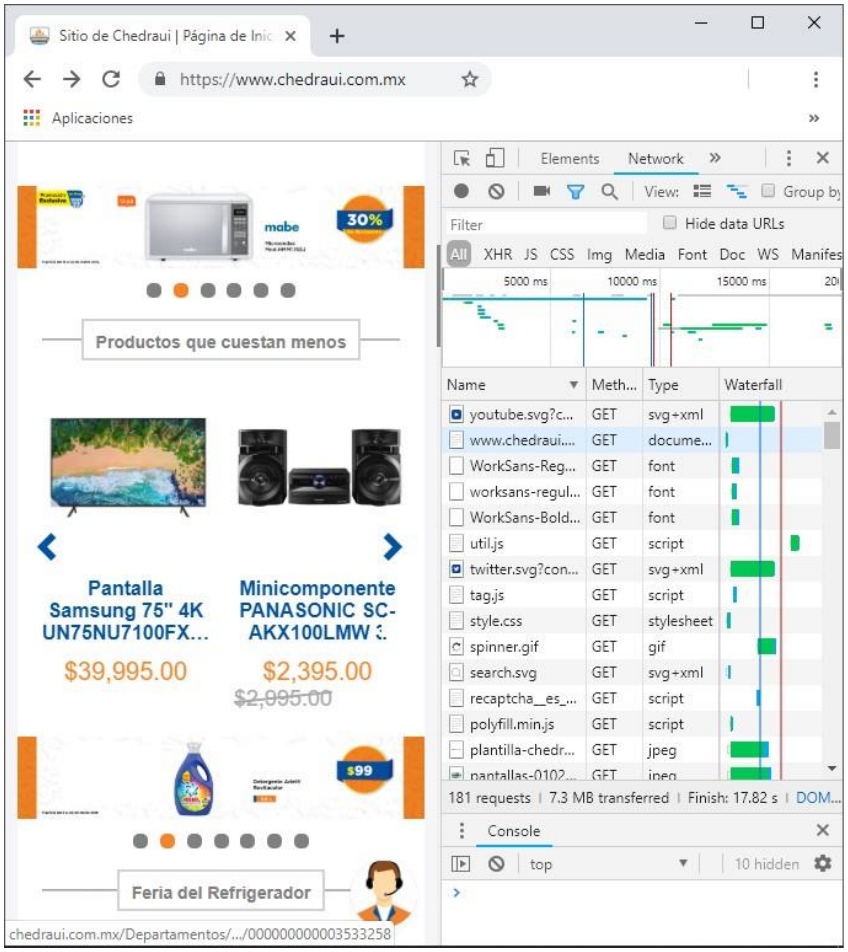

Figura 7. Del lado derecho se pueden observar las consultas que se realizaron al cargar la página, el método con el que se hicieron y el tipo de documento de la respuesta

Existe una tercera opción que permite obtener la información de interés. Ésta consiste en utilizar algún entorno de pruebas para aplicaciones web, como Selenium, y un driver de navegador web, como Chromium, con el fin de renderizar todo el contenido de la página sin necesidad de abrir alguna ventana de navegador, para extraer la información directamente, sin importar si ésta se carga de forma asíncrona. La desventaja de este método radica en que el uso de memoria aumenta bastante debido a que, a pesar de que no se muestra la página web a través de alguna interfaz, se realiza la carga completa de la misma en memoria, por lo que también se ralentiza el proceso.

Debido a que algunos *retailers* mantienen una estrategia de precios regional o geolocalizada, en la que los precios de algunos productos cambian de acuerdo con la zona donde se encuentre la tienda física, también es necesario cambiar la ubicación geográfica de la tienda desde la que se consultan los precios, simulando el proceso que sigue el usuario en el navegador. En ocasiones, esto ayuda también a conocer el inventario de cada una de las tiendas activas del *retailer*.

Ciertas páginas le piden al usuario que introduzca su dirección o código postal para realizar el cambio de ubicación, algunas permiten cambiar la tienda desde la que se desea buscar el precio, mientras que otras permiten localizar la tienda más cercana al usuario sólo hasta que se ha generado un carrito de compra. Una vez con la información de la ubicación del usuario, las páginas web recurren a diversos métodos para consultar internamente la información geolocalizada; algunos generan cookies que se envían con cada consulta para que éstas contengan la información de una tienda en particular y otros lo hacen añadiendo algún identificador de tienda en las consultas internas a sus API's, ya sean internas o reflejadas directamente en la url de consulta.

Para diseñar el *scraper* y dependiendo de cuál de estos métodos utiliza el sitio para mostrar la información, podemos utilizar herramientas de software como Postman, la cual permite conocer todas las consultas que realiza un sitio al momento de cargar una página, además de tener la capacidad de replicarla si así se desea; una función muy útil al momento de realizar pruebas de funcionamiento.

Paralelización

El número de urls que contiene un sitio puede llegar a sobrepasar fácilmente los 100 mil. Si por cada uno de ellos tenemos una respuesta en aproximadamente 500 ms (promedio del tiempo de respuesta para distintas páginas de comercios online), recorrer toda la página tomaría casi 14 horas. Es importante contemplar que diariamente se debe de extraer la información de aproximadamente otros veinte dominios mediante servicios externos de cómputo, donde el tiempo de uso para cada instancia del servicio tiene un representa cierto costo para la empresa. Por ello se desea que el tiempo de procesamiento sea el menor posible. Para lo cual los procesos del *scraper* se paralelizan mediante el uso de software que permite realizar tareas simultáneas de forma continua y asíncrona, llegando a reducir el tiempo de procesamiento hasta en un 70%.

Dicho software, llamado Celery, se denomina como “administrador de tareas” o *task manager*. Para realizar cualquier proceso en paralelo usualmente se requiere definir un “maestro” o *master* y uno o varios “esclavos” o *workers* a los que se les puede definir la concurrencia; es decir, el número de tareas simultáneas que puede ejecutar el esclavo (se recomienda definir un número de concurrencia igual a la cantidad de núcleos de procesamiento de la máquina donde se trabaja). Cabe señalar que el maestro y los esclavos generalmente se definen por separado; es decir, se utiliza una terminal para correr los *workers* y otra para inicializar el *master*. Ambas partes se comunican mediante un bróker de mensajería el cual administra los mensajes dependiendo de la carga de trabajo de los esclavos, además de informar el *status* de las tareas que están siendo o que ya fueron procesadas. La ventaja de esta estructura de procesamiento radica en que cada esclavo trabaja de forma independiente; procesan las tareas que les son asignadas sin importar el estado de aquellas que los demás realizan [6].

Ejecución automática

El modo de ejecución que la empresa acogió para el lanzamiento de los *scrapers* consiste en ejecutar el código del maestro en una instancia de servicio de cómputo en la nube, después se verifica de forma continua que se haya generado el mensaje de “inicio” en el bróker, para finalmente lanzar un número determinado de esclavos en otras instancias si se percibe la creación del mensaje o para cancelar el lanzamiento si se llega a cierto límite de tiempo sin detectarlo. Es posible correr N número de esclavos en las instancias del servicio web de cómputo, los *workers* que recién se ejecutan se sincronizan con los existentes y reciben nuevas tareas a realizar, de modo que el *scraper* no realiza las tareas de forma repetida sin importar cuántos nuevos *workers* se corran.

Esta forma de ejecución presenta algunos problemas debido a que el tiempo necesario para que se envíe el mensaje de inicio varía de acuerdo con la disponibilidad que tiene el servicio de cómputo para generar nuevas instancias; éste depende de factores internos, como el número de instancias o de contenedores activos, y de factores externos, ajenos a la empresa, que no se pueden detectar o medir. Debido a ello, hay ocasiones en las que el mensaje de inicio se pierde y el *scraper* no se inicia.

Con el fin de lograr que la ejecución de los *scrapers* fuera completamente efectiva y que éstos corran siempre que se requiera, diseñé una implementación con menos puntos de interconexión, reduciendo así los puntos de falla. Para ello recurrí a investigar algunas funciones avanzadas de Celery y el uso que se le da a clases específicas como lo son los esclavos o *workers*. El problema principal radicaba en que si éstos se corrían automáticamente, cada uno mandaría el mensaje de inicio y las tareas se realizarían repetidamente por distintos esclavos debido a la replicación en cadena de las tareas a realizar. Para solucionar este problema, programé los *workers* para que al iniciarse, supervisaran si había otras réplicas del mismo ya en funcionamiento para sincronizarse con ellos cuando así fuera o mandando el mensaje de inicio en caso contrario. El resultado fue una forma de ejecución tal que necesita únicamente las instancias del esclavo para iniciarse.

Resultados

Para esta parte del proyecto, participé en el desarrollo de los scrapers de empresas como Walmart, La Europea y Cornershop de México; y Walmart Grocery en EEUU, aumentando el catálogo de productos en aproximadamente 20 mil elementos, lo cual aumenta el número de productos que podemos monitorear para los clientes y permitiendo a los usuarios comparar más productos, ampliando la posibilidad encontrar precios menores.

Modelado de datos

El proceso que consistía en relacionar los productos, que en un principio sólo eran medicamentos, realizaba una comparación entre los números *Global Trade Item Number* o GTIN que representan mundialmente un artículo comercial y es único para cada producto, ya que prácticamente todas las páginas lo difundían. Actualmente son muy pocos comercios los que continúan publicando los GTIN en sus sitios web, por lo que se decidió cambiar el método para correlacionar productos.

La metodología actual consiste en tomar las descripciones de los productos y compararlas mediante herramientas de procesamiento de lenguaje y álgebra lineal, como las bibliotecas Scipy y NLTK (*Natural Language Toolkit*) para Python. Para tener mejores resultados con esta metodología es de gran utilidad contar con la mayor cantidad de datos posible, lo que lleva a un problema de procesamiento y almacenamiento.

Mi participación en el proyecto consistió en la implementación, paralelización, adaptación, pruebas, despliegue y mantenimiento de la nueva metodología, integrada con el servicio general de *matching*, el cual también se encarga de actualizar las bases de datos y guardar un historial para facilitar la corrección en caso de haber cometido un error al relacionar dos productos.

Funcionamiento General del algoritmo

A partir de la información existente en la base de datos se extraen los nombres de los productos con su identificador único, el texto se normaliza, removiendo preposiciones dando un formato adecuado y convirtiendo las palabras en sus raíces o lexemas. Posteriormente se genera un set con todos los lexemas presentes en los documentos para después generar una representación matricial de los mismos. Dicha matriz es dispersa, lo que significa que contiene muchos elementos cuyos valores son iguales a cero, por lo que se emplean funciones de la biblioteca de Scipy para manejar este tipo de matrices y optimizar el uso de la memoria en el procesamiento de los datos. Después de haber obtenido la matriz dispersa, se realiza una descomposición en valores singulares de la misma, con el fin de reducir la dimensión de los datos y generar estructuras más ligeras en memoria.

Una vez con los datos cargados, se utiliza la información nueva para crear más interrelaciones entre productos; uno a uno, se normalizan y se convierten en un vector con el mismo formato que el de los primeros datos. Posteriormente se calculan las distancias entre cada nueva entrada de información y la estructura ya cargada y se determina cuáles son los diez vectores más cercanos para después aplicarles

una métrica desarrollada en la empresa y así determinar la probabilidad de que el nuevo producto sea el mismo que alguno de sus vecinos. Si se llega a un umbral superior o igual a 0.9, se determina que se trata del mismo producto y se modifica la base de datos con dicha información.

Proyecto de implementación

Debido a la gran cantidad de datos de producto que se tienen almacenados, cargar toda la información para posteriormente procesarla en un solo hilo sería bastante tardado y computacionalmente costoso. Es por esto que se decidió paralelizar con las herramientas utilizadas para los *scrapers*.

Según los requerimientos del proyecto, el resultado final sería un servicio web que permitiera administrar el procesamiento de los datos y además, hacer consultas sobre nueva información de manera que se identifique si ya se tiene cierto producto en la base de datos.

Se definió que la arquitectura interna del proyecto consistiría en *workers* cargados con aproximadamente 20 mil datos de productos de un mismo *retailer*, lo que representa un aproximado de 200 [Mb] en memoria. Para procesar la información nueva, se consultarían los vecinos más cercanos a cada uno de ellos, comunicándose mediante un bróker de mensajería. Posteriormente, con el conjunto de resultados obtenido, se determinará si el nuevo producto coincide con alguno ya existente, para actualizar las bases de datos en caso de ser así.

Procesos internos

1. **Reconocimiento de los *workers* activos**

   Es necesario conocer el número de instancias activas para considerar en cuántos lotes se van a dividir los datos. Para ello, se asigna un identificador a cada una de ellas, con el nombre del *retailer* del cuál va a cargar información. Cabe señalar que puede haber más de una instancia para el mismo *retailer*. La biblioteca de Celery, la cual permite el manejo interno de los *workers*, tiene una función que permite enviar un “ping” a todos ellos y de esta forma conocer cuáles se encuentran activos según la respuesta recibida.

2. **Carga de los datos por *retailer***

   Para realizar dicha tarea se utiliza un servicio web que se conecta directamente a la base de datos del catálogo y descarga la información.

   a. **Modificación de *endpoint* de catálogo**

      Con base en el funcionamiento del algoritmo para relacionar los datos, la información que se debe cargar debe ser aquella que corresponde a productos ya identificados. Por ello se realizó una modificación al servicio web, agregando una ruta que hacía un *query* específico a la base de datos, permitiendo descargar sólo la información necesaria.

   b. **Reconocimiento del estado de *workers***

      Con el fin de su manejo, otro aspecto importante a conocer es el estado de los *workers*; si ya se cargó la información para empezar a realizar consultas y a qué *retailer* corresponde.

3. **Consulta de los datos**

   Con la información completamente cargada, se procede a consultar la información a cada instancia. La cual se agrupa por *retailer* y probabilidad de relación.

4. **Configuración del envío de mensajes a través del Bróker**

   Por defecto, la forma en la que el bróker trabaja consiste en mandar los mensajes de ejecución de alguna tarea a algún esclavo que se encuentre disponible. Para efectos del proyecto se requería enviar el mismo mensaje a todos ellos, para la consulta tanto del estado como de la información.

   La función de reconocimiento *workers* activos fue de bastante utilidad ya que permitió hacer un recuento y replicar el mensaje para cada uno de ellos.

5. **Configuración de Aplicación Web**

   Esta aplicación consiste en habilitar un servicio, accesible vía internet, que permita administrar ciertos elementos del proceso de modelado de los datos, como desplegar el número de *workers* activos por cada *retailer*, además de mostrar cuáles de ellos ya tienen cargada la información correspondiente y de hacer posible la consulta de algún producto.

   La aplicación fue programada con un marco de trabajo para programación de aplicaciones web en Python llamada Flask, la cual permite mapear las distintas rutas y métodos de consulta hacia la ejecución de distintas funciones. También permite manejar un contexto de aplicación para facilitar, por ejemplo, el uso de una variable u objeto por parte de varias funciones [7].

6. **Actualización de base de datos**

   Al manejar distintas bases de datos debido a las funcionalidades de cada una; se debe actualizar toda la in formación que éstas comparten. Debido a ello se realizó un integración con los servicios web de cada base, permitiendo realiza esta tarea. Además el registro de los cambios y actualizaciones que se realizan se agregan en otra base de datos con el fin de identificarlos y hacer una corrección en caso de haber existido un error.

Servicio de cómputo para despliegue

Gracias a la oficina de operaciones de la empresa, el equipo consiguió créditos de Google Cloud Platform para utilizar sus servicios. Por lo que se decidió dar de alta el nuevo servicio utilizando las herramientas de dicha plataforma. De esa forma se aprovechó para adquirir el conocimiento sobre el uso de la misma, ya que no se había utilizado con anterioridad. Así, el equipo podría adaptarse a este nuevo servicio de cómputo y recurrir a su utilización en caso de ser necesario en el futuro, aumentando sus recursos.

Google Cloud Platform utiliza Kubernetes como el servicio para manejar distintas unidades de cómputo que trabajan como clúster; es decir, que trabajan en conjunto, relacionadas a través de una red y actuando como unidad. Además, permite administrar el uso de aplicaciones alojadas en contenedores como los de Docker [8].

La unidad básica desplegable para uno o más contenedores que utiliza Kubernetes se denomina *Pod*, el cual permite que las aplicaciones utilicen un mismo espacio de almacenamiento, dirección IP y puertos internos. Existe una unidad más general para el manejo de *Pods* denominada *Deployment*. Éste contiene

una descripción del estado deseado de los contenedores, como la memoria y CPU a utilizar, y el número de réplicas a desplegar, además de los puertos, variables de entorno y comandos a ejecutar, que también se pueden especificar en los *Pods*.

En ocasiones varios grupos de contenedores necesitan comunicarse entre sí, esto se realiza a través de uno o varios puertos. Las direcciones IP creadas con cada *Pod* son aleatorias y efímeras, ya que permanecen vigentes sólo mientras existe uno activo, cambiando si éste se reinicia o escala. Para asegurar la comunicación entre distintas aplicaciones se utilizan servicios, los cuales son abstracciones que definen a un grupo de *Pods* y una política que permite acceder a ellos; funge como un enlace que permite el desacople y la modularidad del proyecto.

Existen cuatro tipos de servicio, ofreciendo distintos niveles de enlace y acceso:

1. **ClusterIP**
   Expone el servicio a una IP de clúster interna, a la que sólo se puede acceder desde dentro del mismo clúster.
2. **NodePort**
   Expone el servicio en cada uno de las IP de nodo (instancia particular del clúster). Crea una IP de clúster a la cual el servicio de nodo va a redirigir la comunicación. Se puede acceder mediante la dirección <IP de Nodo>:<Puerto del nodo>.
3. **LoadBalancer**
   Expone el servicio para acceso externo a través de un servicio de balanceo de cargas. Se crean servicios NodePort y ClusterIP para dirigir la comunicación.
4. **ExternalName**
   Mapea el servicio a un nombre externo, similar a un Sistema de Nombres de Dominio o DNS, por sus siglas en inglés, el cuál apuntará a la dirección IP del servidor.

Para comenzar a utilizar el servicio de Kubernetes es necesario realizar una configuración inicial que involucra tanto establecer parámetros de GCP y del propio servicio. Para ello es necesario instalar el "Equipo de Desarrollo de Software", o SDK por sus siglas en inglés, de Google. Al configurar el SDK, se debe configurar el proyecto y la "zona de cómputo", la cual se refiere a la zona geográfica, perteneciente a cierta región, donde se localizan físicamente las herramientas de cómputo que ofrece la plataforma y que ofrecen distinta disponibilidad de servicios.

Posteriormente y mediante la plataforma de internet de GCP, se creó un clúster de tres nodos, uno por cada zona de cómputo en la región (llega a variar dependiendo de cada región), con máquinas virtuales estándar (1 CPU virtual y 3.75 [GB] de memoria), adecuadas para las necesidades del proyecto.

Una vez con el clúster activo y Kubernetes instalado, se obtienen las credenciales de acceso para controlarlo de forma remota mediante un comando de autenticación del SDK de Google, el cual conecta internamente el clúster configurado con Kubernetes en el equipo local. Una vez realizado este

procedimiento, es posible realizar el manejo de *Pods*, *Deployments*, *Services* y otras herramientas para hacer funcionar nuestra aplicación.

Cada objeto de Kubernetes puede describirse a través de un documento de extensión YAML, que define parámetros como el tipo de herramienta de la que se trata; los recursos computacionales, variables, y comandos a ejecutar en caso de ser necesario; los puertos a exponer y algunos selectores . La funcionalidad de estos últimos es etiquetar cierto tipo de instancias, de tal forma que si se aplica un cambio sobre dicha etiqueta, éste se aplique a todas las instancias. Por ejemplo, si a un *Deployment* se le define un selector y posteriormente se define un servicio de tipo NodePort con ese selector, será posible acceder a los puertos especificados en el *Deployment* a través de este servicio.

Google Cloud Platform ofrece un servicio llamado Container Registry que permite alojar de forma privada las imágenes de Docker que se van a utilizar para crear los contenedores en el clúster. Todos los clústers tienen acceso a aquellas alojadas mediante este servicio, si ambos se encuentran en el mismo proyecto de GCP. El SDK de Google permite subir imágenes creadas en el ordenador propio al servidor del servicio mediante las autenticaciones pertinentes.

Una vez creado el archivo YAML, basta con un comando de Kubernetes para crear el objeto especificado en él. También existe una biblioteca de este software para Python3 que permite hacerlo desde algún *script* de programación.

Kubernetes ofrece una interfaz de usuario, también llamada UI (*User Interface*) en la que se puede visualizar el estado general del clúster; *Pods*, *Deployments*, *Services* activos y sus configuraciones. Además permite aplicarles modificaciones si así se requiere. Esta característica facilita realizar cambios que de otra forma sólo podrían realizarse a través de la línea de comandos del ordenador (Ver Figura 8).

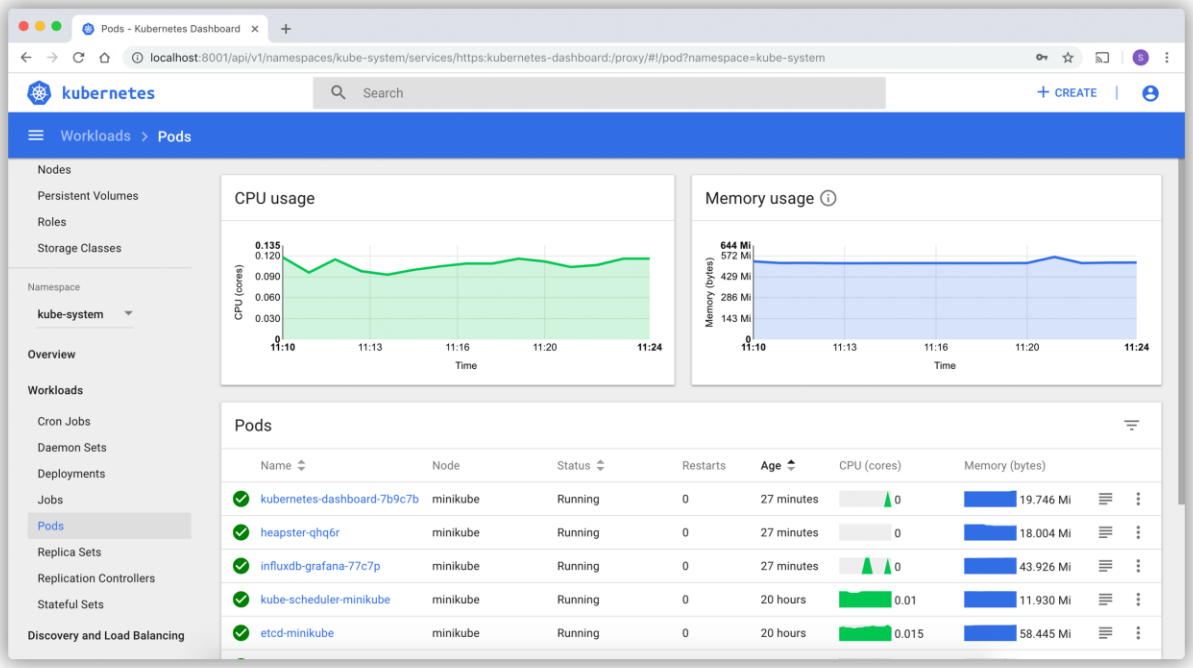

Figura 8. UI de Kubernetes para administración del Clúster (Web UI Dashboard, 2019) [9]

Se utilizó la biblioteca de Kubernetes para Python con el fin de facilitar el proceso ya que se transformaban los datos de configuración de la aplicación para lanzarla en el clúster y además guardar un archivo YAML local para visualización de los parámetros definidos.

El proyecto requirió de la generación de dos tipos de *Deployment*; aquel que corren los *workers* que cargan la información y el otro, que lanza el servicio web para consultar la información. Para los primeros es necesario especificar el *retailer* para el cuál van a cargar los datos y el número de réplicas (*Pods*) que se van a lanzar, por lo que se definen los comandos pertinentes en la plantilla del archivo. Esto se hace para cada *retailer* del cual se requiere consultar la información.

Para la configuración del servicio web se requiere una Interfaz de Puerta de Enlace de Servidor Web (WSGI por sus siglas en inglés) el cual sirve como una interfaz entre el servidor web y el marco de trabajo que ejecuta el código de Python. En este caso se utilizó la WSGI llamada *Green Unicorn* o *Gunicorn* disponible para sistemas operativos Linux, entre otros, el cual permite utilizar varios *workers* internos, cada uno con acceso a la aplicación de Flask, permitiendo manejar múltiples peticiones de forma simultánea [10].

Además del WSGI, se utiliza un servidor de proxy y HTTP, el cual se encarga de resolver la petición web realizada por el cliente y reenviándola a Gunicorn para ejecutar el procesamiento correspondiente.

Los procesos de instalación tanto de la WSGI como del servidor proxy se agregan en la construcción de la imagen de Docker para que su uso esté disponible cuando se ejecute el contenedor de esta aplicación. Finalmente en el archivo YAML del *Deployment* del servicio web se agregan los puertos a escuchar, las variables de entorno necesarias, el comando a ejecutar para el arranque del mismo, además de las réplicas a lanzar.

Además del lanzamiento de las aplicaciones tanto de los módulos de carga de información como del servicio web, es necesario correr una instancia en la que se ejecute el bróker de mensajes para permitir el intercambio de información entre ellas. Así que también se creó un *Deployment* para ello, especificando la versión de la imagen que se desea utilizar y los puertos correspondientes.

Se necesitan dos tipos de servicios para el proyecto; uno para comunicar el bróker con las aplicaciones, y otro para permitir la comunicación externa (a través de internet) con el servicio web. Ambos se establecen como servicios de tipo LoadBalancer para permitir la comunicación a cualquier nodo del clúster.

En primera instancia el servicio que comunica el bróker podría ser de tipo ClusterIP ya que el envío de mensajes es interno. Pero debido a que éste ofrece el uso de una interfaz de usuario a la que se puede ñconectar de forma externa, que permite la administración de los mensajes, colas e intercambios que utiliza; se configuró de forma que también se pueda acceder a ella desde un ordenador externo, a través de internet.

Una vez configurados y lanzados todos los elementos del proyecto, se procedió a la fase de pruebas con el fin de observar el rendimiento de la plataforma y el manejo de posibles errores. Se hicieron algunos cambios con el fin de mejorar el tiempo de respuesta del servicio web, como recurrir a la API del bróker con el fin de inferir qué *workers* se encontraban activos, ya que éste era un método más rápido que el uso de la función “ping”. Además sucedía que algunas colas permanecían registradas en el bróker cuando el *worker* encargado de recibir mensajes de ésta se había dado de baja a pesar de la adición de protocolos de eliminación de las mismas en dichos casos. La causa de ello es que en ocasiones, cuando se reescala el número de instancias de cierta aplicación, algunos contenedores terminan de correr de forma súbita, por lo que no hay tiempo suficiente para que se elimine la cola generada aleatoriamente según los protocolos. Es por ello que se añadió una ruta en el servicio web que permitiera borrar las colas que ya no están siendo escuchadas por ningún receptor.

Una adición al proyecto fue el empaquetamiento del modelo que realiza las comparaciones del producto, lo que ha permitido usarlo en otros proyectos, ya que puede ser instalado como una biblioteca más de Python.

Resultados

Al final se obtuvo un servicio que permite procesar un gran volumen de información con el fin de relacionar los nuevos productos que llegan a nuestras bases de datos con los que ya se encuentran presentes en ellas. Este servicio también permite controlar la cantidad de información que se utiliza para

hacer las comparaciones, además de que permite realizar consultas de la similitud de productos específicos bajo libre demanda.

Conclusiones

A lo largo del trabajo en los proyectos mencionados, he aplicado y desarrollado múltiples habilidades ingenieriles; las cuales contemplan desde aquellas muy generales, como el diseño de un proyecto basado en requerimientos de un cliente o de la propia empresa, hasta las particulares como la comprensión del flujo de información en una página web, la creación de APIs o el uso de herramientas de cómputo provistas por un servicio en la nube para crear un proyecto desde cero.

Al formar parte de una empresa relativamente joven y con poco personal, tanto las responsabilidades como oportunidades de aprendizaje son grandes, puesto que se nos presentan nuevos retos de forma frecuente, además de que buena parte del resultado de un proyecto recae en cada miembro del equipo.

En un proyecto de tal magnitud, en este caso relacionado con el minado de datos, es de suma importancia establecer estándares sobre las entradas y las salidas que se tienen, con el fin de facilitar el flujo de la información; además de dejar un registro de los procesos que se realizan, con el fin de permitir su replicación y evitar caer en problemas que fueron resueltos con anterioridad. Es por ello que cada proyecto debe documentarse de forma adecuada. En el campo del desarrollo de software esta tarea se realiza por medio de la adición de comentarios en el código y de la creación de documentos que expliquen el funcionamiento general de la aplicación que se programó.

El proyecto en el que participé me ayudó también a mejorar el nivel de abstracción empleado en el proceso de programación, ya sea para la paralelización adecuada de tareas o el uso eficiente de los recursos de cómputo. También pude relacionarme con el uso y almacenamiento de datos en bases tanto relacionales como no relacionales, conociendo ventajas y desventajas propias de cada tipo.

Así como mi participación en el proyecto me ha permitido aprender y aumentar mis habilidades como ingeniero, también he aportado al mejoramiento de procesos y al desarrollo de la empresa, traducido en cumplimiento de objetivos y satisfacción de los clientes.

Bibliografía

[1] ByPrice Intel. Consultado en Abril de 2019, de https://byprice.com/byprice-intel

[2] Rabbit Bróker https://www.rabbitmq.com/

[3] L. J. (2015, September 03). RabbitMQ Exchanges, routing keys and bindings. Consultado Mayo 1, 2019, de https://www.cloudamqp.com/blog/2015-09-03-part4-rabbitmq-for-beginners-exchanges-routing-keysbindings.html[4] O. M., & L. R. (Eds.). (2010). Data Mining and Knowledge Discovery Handbook. Springer (2da edición).

[5] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. CRISP-DM 1.0: Step-by-step data mining guide. The CRISP-DM consortium, 2000.

[6] Celery. Consultado en Abril de 2019, de http://docs.celeryproject.org/en/latest/index.html

[7] Flask, Web sevice app. Consultado en Mayo de 2019, de http://flask.pocoo.org/

[8] Kubernetes. Consultado en Mayo de 2019, de https://kubernetes.io/

[9] Kubernetes Web UI (Dashboard). Consultado en Junio de 2019, de https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

[10] Gunicorn. Consultado en Junio de 2019, de https://gunicorn.org/