Recuperación de Información y Recomendaciones en la Web (2016)

Informe Grupo 9

Docente: Libertad Tansini

Integrantes:	CI:
Carmen Invernizzi	4.643.006-1
Diego Rey	4.064.156-1
Mauricio Morinelli	4.422.760-6
Gerardo Goñi	4.356.839-0
Martina Señoris	4.778.166-9

Introducción

En este documento se presenta información sobre el diseño e implementación del sistema desarrollado, se describe la arquitectura del sistema, así como cada uno de los componentes, se plantean conclusiones y se menciona el trabajo a futuro.

Problema

Dado que cada vez más personas se disponen a viajar buscando vuelos en la web, y hay una gran cantidad de sitios que ofrecen distintas ofertas de vuelos y de formas diferentes, surge la idea de implementar una aplicación web centralizada, que contenga ofertas de vuelos, de diferentes sitios, de una manera unificada, permitiendo aplicar filtros para que las búsquedas se ajusten a las necesidades del usuario. De esta forma un usuario podrá obtener el viaje de menor precio disponible, dado un presupuesto, un destino, etc, desde una misma aplicación.

Para realizar un viaje se deben tener en cuenta factores como: fechas de partida y de regreso, destinos, agencias de viaje, aerolíneas, las escalas de los vuelos, así como los aeropuertos. De todos modos el principal factor a ajustar es el presupuesto destinado al viaje y en particular al precio de los traslados.

Hoy en día, la información necesaria para realizar los viajes al menor costo, está disponible en distintas páginas publicadas en internet, y debido a la gran cantidad de páginas destinadas a la venta de vuelos y que cada una de estas brindan sus propias ofertas, resulta difícil al viajero poder acceder a determinada información como: Cuál es la agencia que tiene el precio más bajo dado una fecha y un destino, o cuando son las fechas de descuentos para determinada agencia o aerolínea.

Enfoque de la solución

Arquitectura

El sistema consta de una arquitectura cliente-servidor simple, donde la aplicación web *VuelosWirWeb* se comunica con el servidor *VuelosWir* mediante inyección JNDI. A su vez *VuelosWir* se comunica con un servidor de base de datos *MONGO,* donde se insertan los JSON que representan los vuelos, y con un servidor ElasticSearch, donde se realizan las consultas.

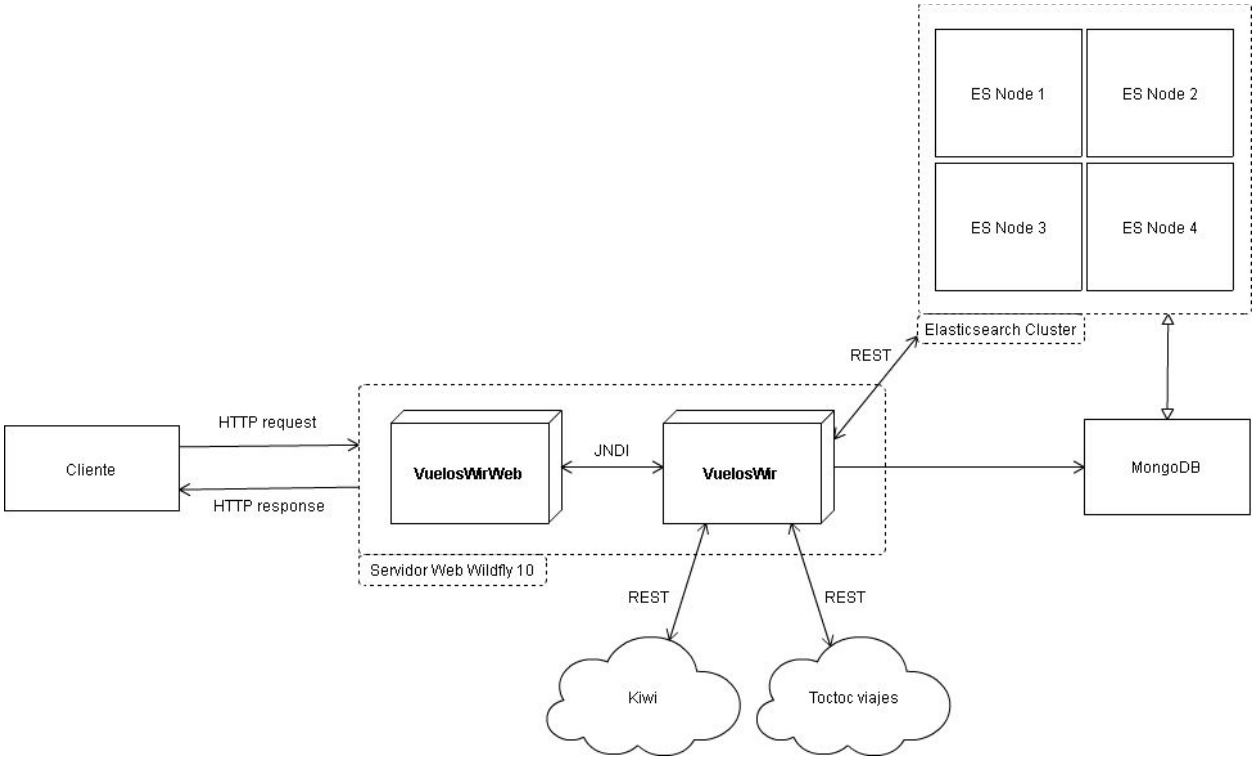


Figura 1. Arquitectura

Los sistemas *VuelosWir* y *VuelosWirWeb* se despliegan sobre un servidor WildFly 10. Además se dispone de un servidor de base de datos MONGO, y como search engine se utiliza elasticsearch, como se mencionó anteriormente.

Implementación

Elasticsearch

Elasticsearch es un motor de análisis y búsqueda en tiempo real distribuido. Se utiliza para búsquedas full-text, búsqueda estructurada, análisis y como combinación de los tres.

Se destaca por ser capaz de integrar en una única aplicación búsquedas full-text, un sistemas de análisis y la capacidad de escalar gracias a la distribución de las bases de datos.

Es de código abierto, construido sobre Apache Lucene que es considerada la librería de motor de búsqueda más avanzada y performante en el mercado, código abierto y propietaria.

Dado que Lucene es una librería, el desarrollador debe tener un conocimiento avanzado para integrarlo en su aplicación, es por esto que Elasticsearch lo utiliza internamente y apunta a esconder todas estas complejidades y proveer las mismas funcionalidades a través de una simple RESTful API.

Se decide utilizar Elasticsearch porque entre otros permitirá a la aplicación:

- Proveer un almacén distribuido de documentos donde cada atributo es indexado.
- Un motor de búsqueda distribuido con capacidades de análisis en tiempo real.
- Capacidad para escalar a cientos de servidores y manejar petabytes de datos estructurados y no estructurados (documentos).

Todo esto será accesible a través de un servidor autónomo el cual nuestra aplicación podrá comunicarse a través de una simple RESTful API.

Al recuperar información de los dos sitios seleccionados, obtenemos documentos representados como JSON, y cómo Elasticsearch se orienta a documentos la tarea resulta trivial y directa. La característica de mayor interés para este trabajo es que el contenido de cada documento será automáticamente indexado, permitiendo buscar, filtrar y ordenar la información.

En las bases de datos relacionales se agregan índices, como el B-tree, en nuestro caso se agregaron índices algunas de las columnas para mejorar el tiempo de respuesta en las búsquedas. Elasticsearch y Lucene utilizan una estructura denominada índice invertido para tal fin.

Por defecto cada atributo en los documentos será indexado automáticamente, y junto a la capacidad de escalabilidad y tolerancia a fallos que elasticsearch provee naturalmente, esto hace de la herramienta la mejor elección para nuestro objetivo.

Destacamos que configurar un cluster con un nodo elasticsearch, insertar documentos y consultar la base de datos es un proceso que se realiza en cuestión de pocos minutos, lo que permitió al equipo comprobar el potencial de la herramienta.

Mongodb

Mongodb es un sistema de base de datos NoSQL (not only SQL), orientado a documentos de código abierto. En lugar de guardar los datos en registros, los datos se guardan en documentos, almacenados como BSON, una representación binaria de JSON.

Nuestra aplicación deberá eventualmente manejar grandes cantidades de datos al integrar más proveedores de vuelos por lo que es necesario plantear un sistema que pueda escalar en forma

horizontal. Se busca que la aplicación crezca para adaptarse a las circunstancias de demanda cambiantes de forma rápida y sencilla, manteniendo tiempos de respuesta.

En el mundo de las bases de datos relacionales generalmente se escala en forma vertical, esto es, aumentando recursos de un equipo. En entornos donde se manejan muchos datos esto se puede volver impracticable pues la cantidad de recursos destinados a un único nodo son finitos y los costos aumentan a medida que se alcanza dicho límite.

Sumado a lo anterior, se tiene en cuenta que se están integrando distintas fuentes de datos, por lo que la estructura de los mismos es distinta. Es conveniente tener un sistema flexible a la hora de persistir la información, ya sea por las distintas estructuras que manejan las distintas fuentes de información o ante una eventual evolución y cambios de estructuras en cada uno de ellos.

Destacamos que la periodicidad con la que se genera o modifica información es baja, se estima que la carga de la base de datos se realizará una vez al mes, evitando el problema de llevar a cabo muchas operaciones de escritura en nuestra base mongo, pues en cada operación de escritura se bloquea el acceso a toda la base de datos.

Las operaciones de escritura se ralentizan cuantos más índices tenga una colección, pero dado que la periodicidad de inserciones será baja y que la búsqueda de información se realizará con elasticsearch (encargado de indexar la información), no será necesario la creación de estos índices, mejorando los tiempos a la hora de insertar las nuevas ofertas en el sistema.

Se decide entonces, por las razones mencionadas anteriormente y principalmente por el manejo de grandes cantidades de datos, la necesidad de escalar horizontalmente y la necesidad de un manejo no estructurado de la información, que un sistema de base de datos como Mongodb es una buena elección para la solución.

Mongodb y Elasticsearch

¿Cómo se relacionan estos dos servidores en nuestra solución? La idea que aquí se propone es que los datos que provienen de las distintas empresas oferentes se persistan en una base de datos mongo, obteniendo todos los beneficios mencionados anteriormente. Esta se utilizará como principal recurso para alojar la información.

No se utilizara Elasticsearch para tal fin pues se ha investigado acerca de pérdidas en las inserciones de datos, por lo que utilizar este motor como principal base de datos resulta poco confiable. Se desea persistir toda la información proveniente de las fuentes, sin pérdidas de datos.

La información que se persiste en mongo se replicará automáticamente a nuestro cluster elasticsearch. Creemos que si bien se están usando más recursos para poder replicar los datos

tanto en mongo como en elasticsearch, los beneficios son mayores a los costos, obteniendo por ejemplo un alto nivel de paralelismo.

Esta solución provee un alto nivel de confianza a la hora de insertar los datos en mongo y un gran nivel en los tiempos de respuesta al consultar los datos debido a todas las ventajas ya mencionadas del motor de búsqueda elasticsearch.

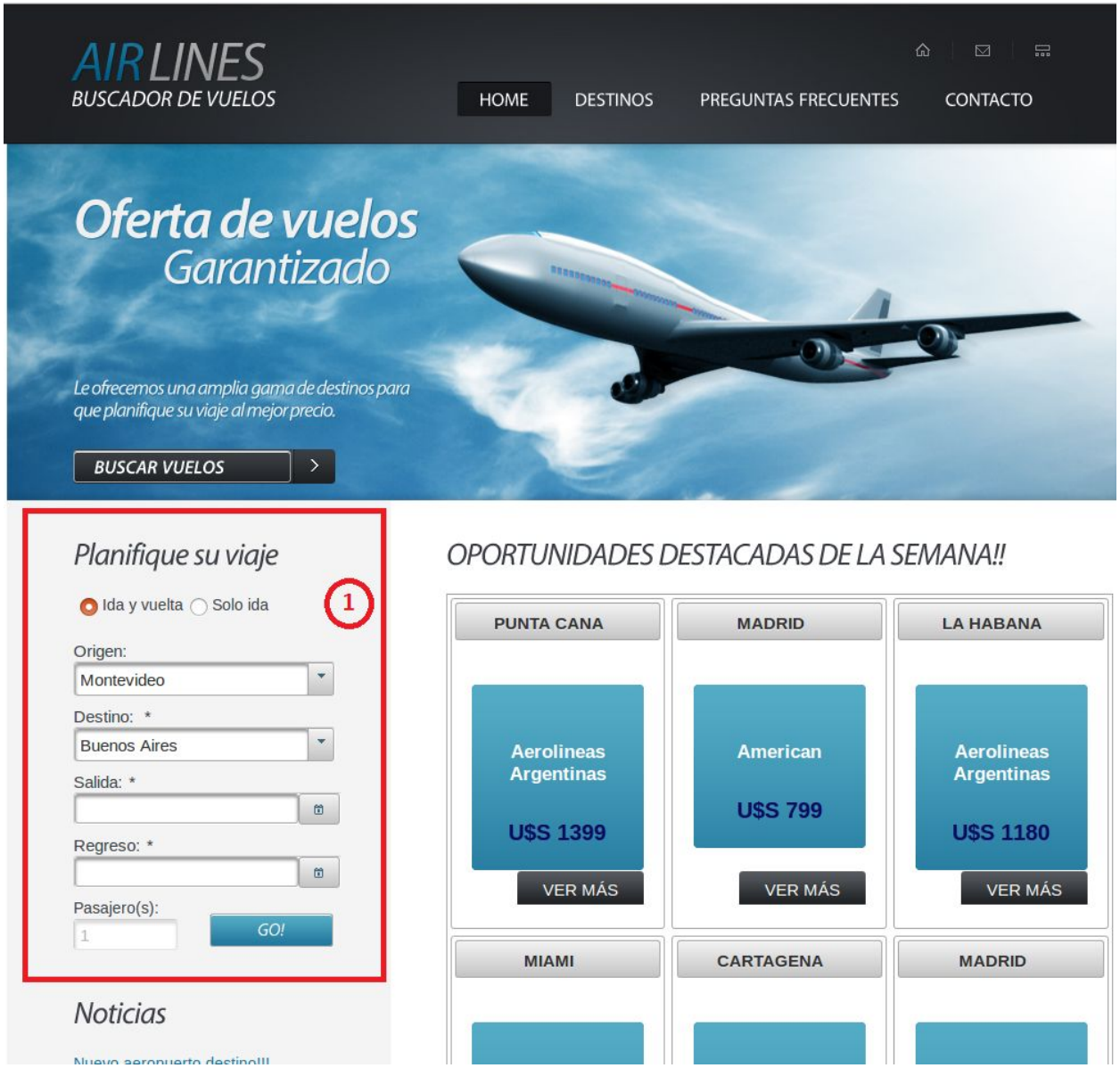
Se optó por obtener lo mejor de ambos mundos e integrarlo en un sistema para cumplir con nuestros objetivos.

VuelosWir

Este sistema implementa la lógica de negocio. Es un proyecto Java Empresarial que implementa una interfaz de EJB (Enterpirse JavaBeans), el cual expone métodos que son consumidos por el sistema *VuelosWirWeb.*
Este sistema consta de un Bean principal, el cual contiene las traducciones de código de aeropuerto- nombre del aeropuerto , código de aerolínea - nombre de aerolínea. Además de dos clases encargadas de consultar a las APIs de *TocTocViajes* y *SkyPicker* (*Kiwi*), parsear los JSON obtenidos como respuesta y guardalos en MONGO.

Funcionalidades y uso

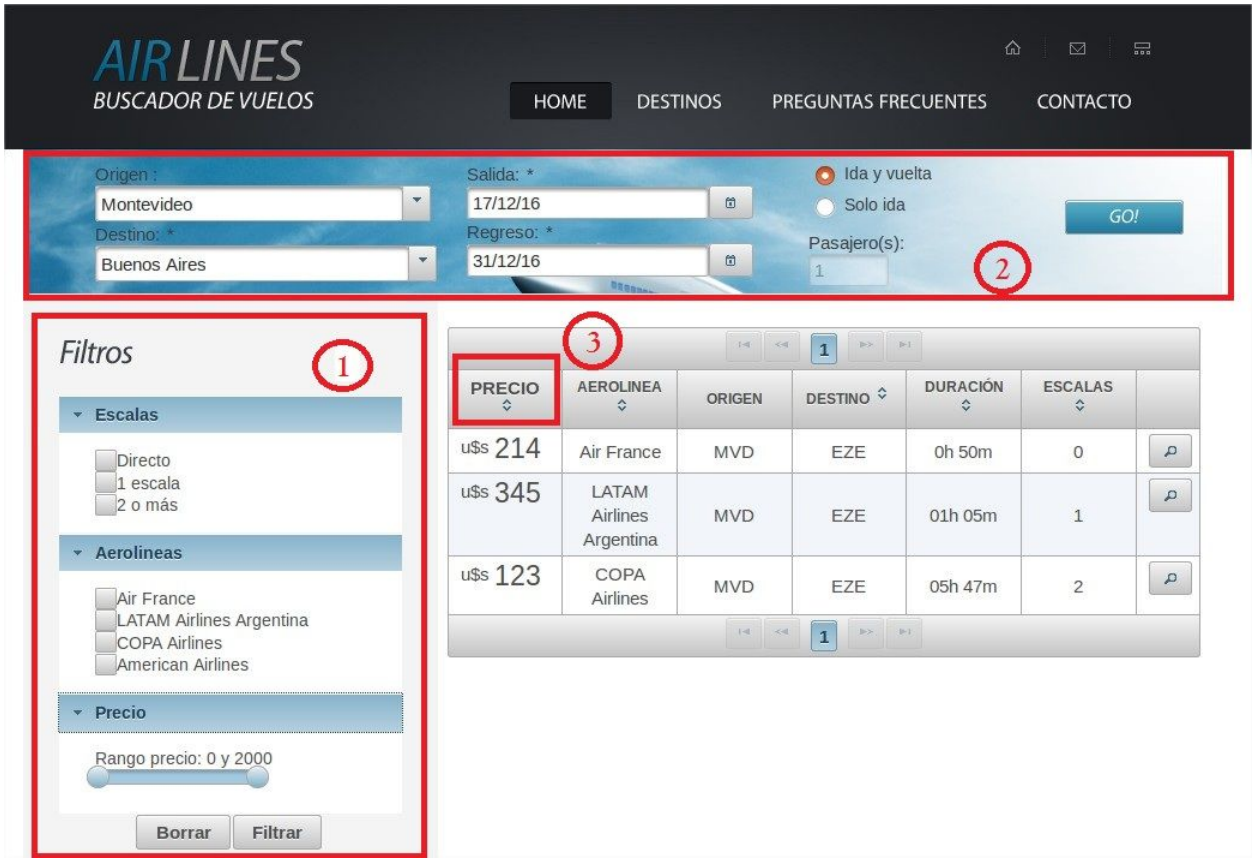
Con el objetivo de poder probar rápidamente las consultas a ElasticSearch y la carga de vuelos en MongoDB (casi 22.000 vuelos), se diseñó una interfaz simple que permita buscar vuelos con un origen, un destino y una fecha de salida como datos obligatorios, y una fecha de regreso como dato opcional, ver cuadro (1). También en esta página se muestra una lista de destinos filtrados mediante una búsqueda a ElasticSearch.



Al hacer click en “GO!”, se buscan los vuelos en ElasticSearch y se muestra la siguiente pantalla, donde se tendrá una lista paginada de vuelos que cumplieron los criterios de búsqueda, con datos de precio, aerolínea, origen, destino, duración y números de escalas.
En esta pantalla también se muestra un panel (1), donde se pueden aplicar distintos filtros a la búsqueda realizada. Estos filtros se aplican en el front-end sobre el json obtenido en la búsqueda inicial, para evitar así una sobrecarga de consultas innecesarias a ElasticSearch.

En el panel (2) se muestran los campos para poder realizar un nueva consulta de vuelos sin necesidad de tener que volver a la página inicial. Esta búsqueda al igual que la búsqueda de la página inicial, se realiza sobre ElasticSearch, ya que el criterio de búsqueda puede ser diferente. Además, por cada columna del listado es posible ordenar de mayor a menor los

vuelos resultantes, por ejemplo ordenando por precio (3), y por último permite seleccionar un vuelo y ver los detalles del mismo, con la información de cada una de las escalas.



Evaluación y resultados

Al momento de obtener los datos que se usaron para poblar de datos nuestro sistemas, se investigaron varias APIs que proveen información de vuelos y se evaluó hacer Web Scraping directamente sobre algunos sitios web. Posteriormente, se decidió usar 2 APIs que fueron las que reunieron características que hicieron viable su uso en este proyecto académico. Los factores a tener en cuenta para decidir usar una API fueron: costo económico, cantidad de pedidos soportados por tiempo, facilidad de uso y confiabilidad. Las páginas que se evaluaron para hacer Web Scraping no reunieron toda la información que se pretendía obtener o su obtención requería de un gran esfuerzo, por lo tanto se optó por usar APIs para obtener la información.

Las dos API seleccionadas fueron TocTocViajes y SkyPicker (Kiwi). Cada una de ellas exponen sus servicios como un servicio REST y como respuesta a un pedido de información con determinados filtros, retornan una colección de vuelos representada en formato JSON.

Se hizo un trabajo de integración de datos entre las dos APIs llevando los resultados de cada una a un formato común de JSON el cual es el que se guarda en la base de datos MONGO usada por nuestro sistema.
Considerando el tiempo que tarda en responder cada API se decidió recortar la cantidad de filtros por los que se hacen llamados a cada API, dado que cambiar combinación de valores de filtros implica hacer un nuevo llamado. Por lo tanto hay filtros que quedaron fijos, como la cantidad de pasajeros que es de uno, o limitados a un conjunto reducidos de valores, como es el caso de la cantidad de aeropuertos destino para los cuales se guarda información. Por la forma en la que se tiene que invocar cada API se decidió usar un periodo de tiempo limitado como fecha de partida y retorno de los distintos vuelos, ya que para consultar a cada API hay que definir de forma explícita la combinación de fechas de salida y retorno para cada llamado.

Conclusiones

Como conclusión se puede destacar el gran volumen de datos que genera cada sitio web, y cómo esto puede influir en el desarrollo de un sistema centralizado que contenga todos los vuelos de los diferentes sitios. Aquí se implementó un prototipo con ofertas de vuelos de dos sitios web, limitando los vuelos a un único mes y con parámetros fijos como son el tipo o número de pasajeros. Aún así, el volúmen de datos generado fue grande (en el orden de 22.000 vuelos diferentes).

Debido a esta gran cantidad de datos que es necesario manejar, pudiendo aumentar en gran cantidad al incluir nuevos sitios web, nos pareció indispensable contar con un sistema que realice consultas de forma eficiente. ElasticSearch nos pareció una buena opción ya que como se explicó anteriormente utiliza índices invertidos, los cuales mejoran los tiempos de búsqueda y además permite realizar búsqueda de texto por parecidos. Esta búsqueda por parecidos permite obtener un conjunto de resultados mayor ordenados por prioridad, que puede ser útil para recomendar vuelos al usuario y mejorar así su experiencia en el sitio.

Cabe señalar que se debería hacer un uso razonable de los índices creados por ElasticSearch, eliminando aquellos creados por atributos que no se necesiten consultar, ya que estos índices también ocupan espacio en la base de datos.

Luego de finalizado el proyecto, podemos concluir que logramos alcanzar el objetivo planteado al inicio del mismo, a pesar de las limitaciones que fuimos descubriendo en el desarrollo del prototipo. Destacamos un importante aprendizaje por el equipo sobre las herramientas utilizadas y la problemática de la recuperación de volúmenes de información tan grandes distribuidos por la web.

Trabajo Futuro

Debido al escaso tiempo que se contó para realizar este prototipo hay aspectos que se dejan para desarrollo futuro:

- Ampliar el rango de tiempo que se guarda en la base de datos.
- Ampliar la cantidad de parámetros por los que se puede filtrar y traer esa información (cantidad de pasajeros, aeropuertos destino, cant de niños, etc).
- Automatizar la carga de información y limpieza de datos viejos de la base de datos.

Referencias

[1] TocToc Viajes:
https://www.toctocviajes.com/
[2] Api skypicker:
http://docs.skypickerpublicapi.apiary.io/
[3] MONGODB:
https://www.mongodb.com
[4] Elasticsearch:
https://www.elastic.co/
[5] JDNI:
http://docs.oracle.com/javaee/6/tutorial/doc/gipjf.html
[6] JSON:
https://tools.ietf.org/html/rfc7159
[7] Wildfly 10:
http://wildfly.org/staging/
[8] JavaEE:
http://www.oracle.com/technetwork/java/javaee/ejb/index.html