Proyecto Webir

Curso 2016

Grupo 14

QuePintoHoy

Autores:

Santiago Hitta 4519536-3

Eduardo García 4862736-3

Docente:

Libertad Tansini

Noviembre 21, 2016

Índice

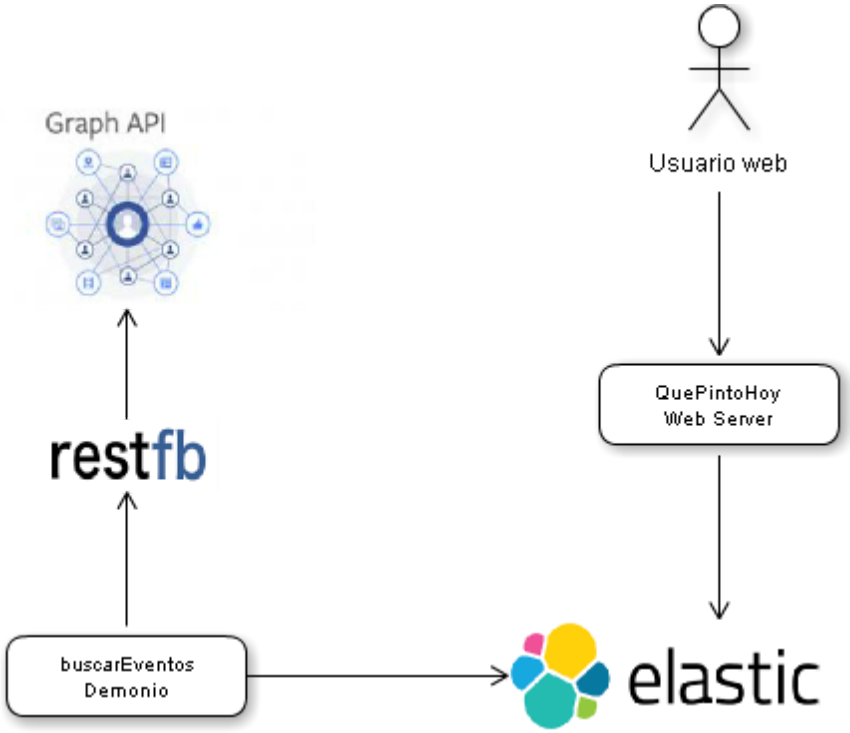
Introducción

QuePintoHoy es una plataforma que permite a los usuarios encontrar eventos de facebook de forma rápida y sencilla, de acuerdo a ciertos filtros aplicados por el usuario para obtener determinado tipo de eventos que sean de su interés.
Este proyecto surge frente a la necesidad de poder encontrar eventos de facebook de forma práctica, lo cual hoy por hoy no es posible.
Hoy en día Facebook es una de las plataformas sociales más utilizadas y en ella se publican grandes cantidades de eventos a diario. Sin embargo la plataforma aún no provee facilidades para la búsqueda de eventos, la única forma que existe para encontrarlos es a través de invitaciones de amigos a algún evento en particular o sino buscando en la sección ¨eventos relacionados¨ que aparece al acceder a un evento cualquiera y que muestra eventos al azar que incluso podrían ser de otro país, además de que para ver la información de los mismos es necesario acceder a ellos uno por uno. Como consecuencia la búsqueda de eventos mediante facebook se torna un proceso tedioso y al azar, en el que el usuario va seleccionando eventos recomendados con la esperanza de dar con un evento de su agrado por casualidad.
La plataforma QuePintoHoy ofrece una solución a dicho problema, permitiendo al usuario obtener los eventos de acuerdo a sus gustos de forma rápida y sencilla.

Enfoque de la solución

Componentes del sistema

La aplicación sigue una arquitectura cliente servidor, el sistema cuenta con un servidor web para que los usuarios accedan a la aplicación. También hay un servidor que ejecuta ElasticSearch el cual guarda los datos y los ofrece a partir de un servicio REST. El servidor WEB obtiene los eventos consumiendo los servicios de éste. Y finalmente hay un componente que se encarga de obtener los eventos de Facebook periódicamente (buscarEventosDemonio) y guardarlos en el servidor de ElasticSearch.



Implementación

Herramientas

API GRAPH de Facebook

Descripción:

Esta API posibilita el acceso a los datos de Facebook, como por ejemplo los eventos, lugares, usuarios, etc, así como también la posibilidad de publicar datos. Es una API basada en HTTP ofrecida por Facebook para que los desarrolladores de aplicaciones puedan interactuar con su plataforma.

Pros:

- ❏ Sencilla de utilizar una vez que se sabe cómo, con pocas líneas de código permite recuperar grandes volúmenes de información.
- ❏ Permite obtener información variada y en grandes cantidades.

- ❏ Está basada en HTTP, por lo que es compatible con cualquier lenguaje que tenga una biblioteca HTTP.
- ❏ No requiere instalar ningún software particular ni descargar librerías, con una petición HTTP de tipo GET ya se puede obtener información.

Cons:

- ❏ Muy poca documentación disponible en Internet.
- ❏ Gran cantidad de versiones distintas de la API que han ido cambiando las funcionalidades que se ofrecen drásticamente, lo cual produce incertidumbre a futuro, además de que gran parte de los ejemplos o problemas enfrentados por otros desarrolladores tratan sobre versiones viejas que tenían comportamientos distintos y por lo tanto dichas soluciones no son aplicables a día de hoy.
- ❏ Comportamientos impredecibles por parte de la API. Muchas veces al realizar búsquedas con los mismos criterios los resultados obtenidos terminan siendo distintos, obteniéndose distintas cantidades de información incluso en períodos de tiempo muy pequeños. Al parecer la API decide la cantidad de información que retorna, la cual es cambiante de un momento a otro, posiblemente basándose en la carga de solicitudes que esté recibiendo en ese momento.

Restfb

Descripción:

Restfb es un cliente open source escrito en Java para utilizar la Graph API de forma sencilla.

Pros:

- ❏ Abstrae el uso de HTTP, ofrece métodos mediante los cuales se puede obtener y publicar información a Facebook mediante la Graph API, además implementa los tipos de objeto de Facebook (eventos, usuarios, mensajes) como POJOs lo cual facilita el manejo de los datos obtenidos como objetos.
- ❏ Fácil de integrar al proyecto, simplemente se incluye el .jar y ya está listo para usarse.

Cons:

- ❏ Si bien cuenta con buena documentación por parte del equipo de desarrolladores, no se cuenta con tanta información en la web en lo que respecta a la resolución de problemas por parte de otros desarrolladores que hayan utilizado la API.

ElasticSearch

Descripción:

Elasticsearch es un servidor de búsqueda, provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON. Elasticsearch está desarrollado en Java y está publicado como código abierto.

Pros:

- ❏ Nos permitió hacer búsquedas de un conjunto de palabras en los documentos json fácilmente.
- ❏ La herramienta es muy potente y luego de pasar la curva de aprendizaje es fácil de usar.

Cons:

- ❏ Uno de los puntos en contra es la mala documentación, la cual es escasa en ejemplos fundamentalmente para Java.

JSF

Descripción:

JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

Pros:

- ❏ Una de las ventajas de JSF es que es un framework que sigue el patrón de diseño MVC, lo cual hace que las aplicaciones JSF sean más mantenibles y claras ya que el código de interfaz de usuario está limpiamente separado de la lógica (el modelo).

Cons:

- ❏ No encontramos dificultades o desventajas respecto al uso de este framework.

Problemas encontrados

El mayor problema que enfrentamos fue respecto al uso de la Graph API para poder recuperar la mayor cantidad posible de eventos.

El primer problema fue que la API no permite obtener eventos por localización, con lo cual solamente podíamos buscar eventos que tuviesen ciertas palabras en su descripción o

título, como por ejemplo ¨Montevideo¨ o ¨Uruguay¨, lo cual no resulta útil ya que existen eventos en todo el mundo con estas palabras, y además se obtienen pocos eventos que sean realmente uruguayos.
La solución que encontramos a este problema fue buscar lugares en vez de eventos, ya que la API permite la búsqueda de lugares dadas las coordenadas y un radio. Una vez obtenidos los lugares se puede consultar por los eventos de cada lugar.

Aquí es donde surge otro problema ya que como se mencionaba anteriormente ésta API limita la cantidad de resultados obtenidos, con lo cual no es suficiente realizar una búsqueda en Montevideo para obtener todos los lugares, ya que dicha búsqueda retornará un número limitado y variable de resultados. La solución que le encontramos a este problema fue achicar el dominio de búsqueda y eso lo logramos dividiendo Montevideo en varios círculos superpuestos de radio 500m y buscando lugares dentro de dichos círculos. Probamos con distintos radios y observamos que a medida que se achicaban los mismos obteníamos mayor cantidad de lugares. De todos modos tampoco se puede abusar demasiado en este aspecto ya que a círculos más pequeños se requieren más búsquedas para abarcar todo Montevideo, lo cual resulta en más demoras para la obtención de nuevos eventos, así como también mayores bloqueos por parte de la API que al realizar demasiadas consultas lanza una excepción por abuso y prohíbe el uso de la misma por unos minutos.

Funcionalidades y uso

La aplicación se accede a partir de un explorador web. El usuario ingresa a la página principal, la cual muestra un conjunto de botones para filtros y una lista de los eventos más cercanos a la fecha de hoy. De los eventos se muestra nombre, descripción, URL a la página de Facebook, fecha de inicio, imagen, dirección. Por medio de los botones se puede filtrar por fecha, eligiendo una en particular, y/o por tipos de eventos. Algunos de los tipos de eventos que se pueden mostrar son:

- Promociones - palabras clave = { promociones, promos, 2x1, 3x3 }
- Gastronomía = { gastronomía, catas, maridaje }
- Cine = { cine, movies, pelicula, película, cortometraje }
- Fiesta = { fiesta, party, rave, dj }
- Conciertos = { toque, concierto }

Evaluación y resultados

Aunque se complicó la utilización de las herramientas y el acceso a los datos, pudimos resolver el problema planteado satisfactoriamente. Se logró hacer una implementación funcional que obtiene los eventos de Facebook, muestra la información de estos cómodamente y los categoriza en diferentes filtros para un acceso más fácil y rápido.

A continuación se detallarán los resultados obtenidos en cuanto a tiempos de respuesta de la aplicación, tiempos de recuperación y categorización de los datos, cantidad de eventos obtenidos, etc.

Lugares obtenidos: 5882
Se utilizó un radio de 500m para los círculos. Círculos utilizados: 4350.
Duración: 50 minutos.

Lugares obtenidos: 5595
Se utilizó un radio de 2km para los círculos. Círculos utilizados: 286.
Duración: 3 minutos.

Utilizando radios menores se logra obtener mayor cantidad de lugares, pero el método tiene una duración mayor.
Dado que los lugares se obtienen usando el método de los círculos explicado más arriba se guardan muchos lugares en los cuales nunca se realizan eventos. Estos se podrían eliminar para mejorar la búsqueda de eventos y disminuir las llamadas a la Graph API.

Eventos obtenidos: 684
Duración: varias horas.

Los resultados mostrados fueron obtenidos en la fecha 19/11/2016. El demonio obtiene eventos a partir de la fecha en la que se ejecuta, obteniendo eventos de fecha 19/11/2016 o posteriores.
La larga duración se explica debido a la cantidad de lugares obtenidos, ya que, para cada uno de los lugares se hace un llamado a la Graph API.

Conclusiones

Concluimos que el área de trabajo en recuperación de datos en la web es muy amplia y que se pueden encontrar muchas aplicaciones útiles para realizar, siendo un gran reto el encontrar las fuentes de información en la web y adaptarse a ellas y a lo que ofrecen para poder dar a los usuarios un servicio acorde a sus necesidades.
Aprendimos de la importancia de la estructuración de los datos y los índices para un manejo más óptimo y rápido de la información. La utilización de herramientas que posibilitan lo anterior y además dan soporte a búsquedas avanzadas es una manera más fácil de implementar soluciones a este tipo de problemas.
En nuestra opinión haber podido tomar un problema de la realidad, el cual nos planteamos nosotros mismos y el cual nos afecta desde hace años, y darle una solución real utilizando las herramientas sugeridas nos pareció una muy buena experiencia para nuestra formación.

Trabajo Futuro

Nuevas estrategias de recuperación de eventos

Aumentar el rango de búsqueda hacia otras ciudades, departamentos, paises, etc para obtener eventos de otros lados, así como también ofrecer filtrado por zonas.

Utilizar un WEB Crawler para obtener más eventos. Ya que no estamos seguros de la completitud de los eventos obtenidos, es decir, no sabemos si realmente estamos encontrando todos los eventos. Un crawler ayudaría además a no depender de la Graph API y sus limitaciones. La idea es que el crawler recorra los eventos, iniciando a partir de varios eventos semillas y obteniendo nuevos a partir de la lista de recomendados, generando un loop que pudiera llegar a abarcar todos los eventos. Este era nuestro Plan B por si la API no resultaba.

ElasticSearch

Establecer más índices y tipos de documentos en ElasticSearch para ayudar a mejorar la performance de éste, haciendo que los datos se puedan acceder más rápidamente. Dado que se consiguieron alrededor de 600 eventos, las consultas de texto completo se realizan lo suficientemente rápido, pero si se trabaja con volúmenes más grandes sería necesario hacer un preprocesamiento para ya tener la información categorizada.

Implementar un demonio que limpie los eventos viejos y cancelados.

Más filtros

Se pensaron varios filtros diferentes de los cuales no hubo tiempo para implementar, por ejemplo: filtros con respecto a la localización (eventos cercanos al usuario), filtros con respecto a la popularidad (eventos que tienen más *likes*, cantidad de usuarios que pusieron asistir e interesa), un filtro general que busque por palabras ingresadas por el usuario.

Mejoras en la interfaz

Al implementar la interfaz nos enfocamos más en las funcionalidades y no tanto en el aspecto visual, el cual quedó como un punto flojo en la aplicación. Como trabajo a futuro se podría mejorar este aspecto y además dar soporte al ambiente mobile.

Referencias

ElasticSearch
https://www.elastic.co/guide/en/elasticsearch/client/java-api/current/index.html
RestFB
http://restfb.com/documentation/
Graph API
https://developers.facebook.com/docs/graph-api
JSF
http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html