Resumen de lo aprendido

¿Qué es el SEO?

Sus siglas significan Search Engine Optimization.

Surge con la necesidad de atraer más gente (tráfico) a las páginas, de forma que las páginas con posiciones más altas (primeras que aparecen) reciben más tráfico que otras que están más abajo.

Definición: El posicionamiento en buscadores u optimización de motores de búsqueda **es el proceso para mejorar la visibilidad** de un sitio web en los resultados orgánicos de los diferentes buscadores.

Existen **2 factores** básicos en los que se **basa un motor de búsqueda** a la hora de posicionar una página web: la autoridad y la relevancia.

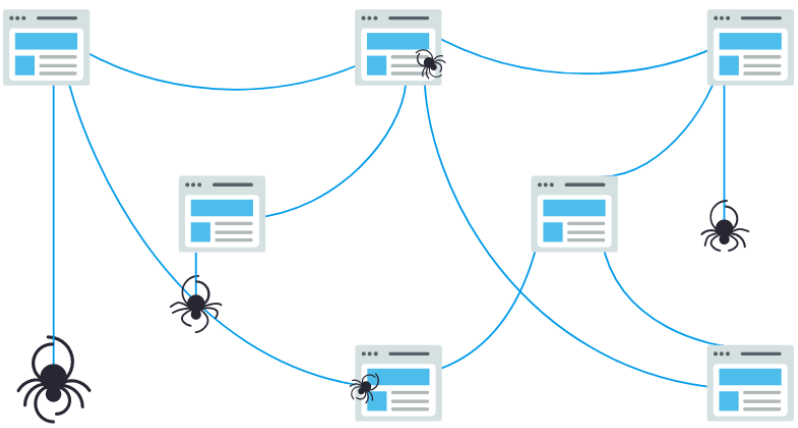
- **La autoridad:** o popularidad de la web. Cuanto más se comparte una web, es porque ha sido útil para un usuario y por lo tanto la información que contiene es más valiosa que otra que se ha compartido menos.
- **La relevancia:** Es la correspondencia que debe existir entre las palabras clave que ha introducido el usuario en su consulta en los buscadores y el contenido y código de la página web. (hay cientos de factores on-site)

¿Cómo funciona?

Podemos resumir el funcionamiento de un motor de búsqueda en 3 pasos: rastreo, indexación y clasificación.

- **Rastreo:** Un motor de búsqueda recorre la web rastreando con unas entidades llamadas **bots o arañas** (las cuales recorren las páginas a través de enlaces al igual que haría un usuario) de forma que recopilan datos sobre esas páginas webs. Los bots se sienten atraídos por los sitios nuevos y los cambios en las webs existentes.

  Son los propios bots los que deciden que páginas visitar, con qué frecuencia y cuánto tiempo van a rastrear esa web, por eso es **importante** tener un **tiempo de carga óptimo** y contenido actualizado.



– **Indexación:** Una vez que un bot ha rastreado una web y ha recopilado la información necesaria, estas páginas se incluyen en un índice y se ordenan de acuerdo a algún criterio (por ejemplo contenido, autoridad y relevancia). De forma que cuando hagamos una consulta al motor de búsqueda, le resultará mucho más fácil mostrarnos resultados relacionados con nuestra consulta.

  Al principio se basaban en el número de veces que se repetía una palabra, ahora dan más prioridad a la calidad del contenido.

– **Clasificación (rank):** Una vez son rastreadas e indexadas, llega el momento en el que actúa el algoritmo: éste decide que páginas aparecen antes o después en los resultados de búsqueda.

¿Qué tipos de información entra en juego en el SEO?

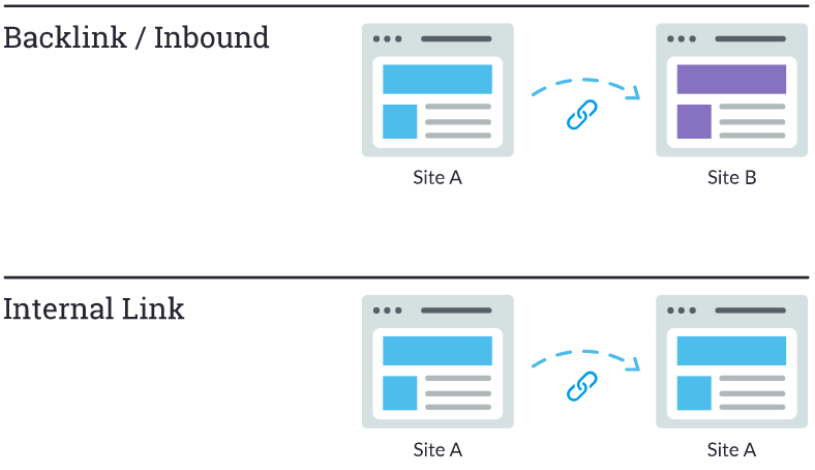
El contenido es más que solo palabras, es cualquier cosa destinada a ser consumida por los buscadores: por ejemplo contenido de **video, de imágenes** y por supuesto **texto**.

Una gran parte de determinar donde se clasificará la página para una consulta determinada se basa en cuánto coincide el contenido de la página con la intención de la búsqueda. Nos podríamos preguntar lo siguiente: ¿esta página coincide con las palabras que se buscaron y ayuda a completar la tarea que el buscador estaba tratando de lograr?

¿Por qué son **importantes los enlaces** en SEO?

Hemos comentado antes que los motores de búsqueda necesitan ayuda para determinar que página web es más confiable que otra para clasificarlas posteriormente. Una forma de hacer ésto es calcular la cantidad de enlaces que apuntan a dicho sitio web.

Distinguir que hay **2 tipos de enlaces**: Los "entrantes" o de retroceso, que son de un sitio web a otro diferente y los "internos" que son de un propio sitio web a otras páginas del mismo.



Se creó **PageRank** *(parte del algoritmo central del motor de búsqueda de Google)*. **Estima la importancia de una página web** midiendo la **calidad y cantidad de enlaces**

que apuntan a ella. La suposición es que cuanto más relevante, importante y confiable es una página web, más enlaces habrá obtenido.

Además, mientras más enlaces entrantes tenga de sitios webs de alta autoridad (confiables), mayores serán sus probabilidades de obtener una clasificación más alta en los resultados de la búsqueda.

Documentos .JSON

(Siglas de **J**ava**S**cript **O**bject **N**otation) es un **formato de texto** sencillo para el **intercambio de datos**.

Se trata de un subconjunto de la notación literal de objetos de Javascript, aunque debido a su fuerte adopción como alternativa a XML, se considera un formato independiente del lenguaje desde 2019.

Una ventaja respecto a XML es que resulta más sencillo escribir un analizador sintáctico o parse para éste. Además JSON se emplea en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia.

Los **tipos de datos** disponibles con JSON son: Números, cadenas, booleanos, arrays y objetos (colecciones de pares de la forma $< nombre >:< valor >$

Figura 1: Ejemplo de codificación en JSON

Podemos ver que tenemos un vector con 3 localizaciones, cada una de ellas con los atributos: latitud, longitud, ciudad y descripción.

API de Java para trabajar con JSON

La API de Java para procesamiento JSON proporciona rutinas API portátiles que permiten analizar, generar, transformar y consultar JSON usando rutinas API de **modelos de objetos** y de **streaming**.

La API de modelos de objetos crea una estructura de árbol, de acceso aleatorio, que representa los datos JSON almacenados en la memoria. Es posible recorrer el árbol y formular consultas. **Este modelo** de programación es el **más flexible** y posibilita el **procesamiento** en casos en que se **requiera acceso aleatorio** a la totalidad del contenido de la memoria. Sin embargo, a menudo no es tan eficiente como el modelo de streaming y requiere más memoria.

Las clases principales de la API de modelos de objetos son:

- **Json**: Contiene métodos estáticos para crear lectores, escritores, constructores de JSON y sus objetos de fábrica.
- **JsonGenerator**: Escribe datos JSON en forma de stream, con un valor por vez.
- **JsonReader**: Lee datos JSON de un stream y crea un modelo de objeto en la memoria.
- **JsonObjectBuilder** y **JsonArrayBuilder**: Crean un modelo de objeto o un modelo de matriz en la memoria agregando valores del código de aplicación.
- **JsonWriter**: Escribe un modelo de objeto de la memoria en un stream.
- **JsonValue, JsonObject, JsonArray, JsonString** y **JsonNumber**: Representan tipos de datos para valores en datos JSON.

Por ejemplo, el código para efectuar una búsqueda en los posts públicos de Facebook usando la API de modelos de objetos sería de la siguiente forma:

```
1
2  URL url = new  URL("https://graph.facebook.com/search?q=java&type=post");
3     try (InputStream is = url.openStream();
4          JsonReader rdr = Json.createReader(is)) {
5          JsonObject obj = rdr.readObject();
6          JsonArray results = obj.getJsonArray("data");
7          for (JsonObject result : results.getValuesAs(JsonObject.class)) {
8              System.out.print(result.getJsonObject("from").getString("name"));
9              System.out.print(": ");
10             System.out.println(result.getString("message", ""));
11             System.out.println("-----------");
12         }
13    }
```

Las líneas 1 a 3 crean JsonReader; la línea 5 crea JsonObject para los resultados; la línea 7 itera respecto de cada resultado; y las líneas 8 a 11 obtienen el nombre de la persona que publicó el post y el post en sí, y los imprimen.

Considerando que la API de Facebook devuelve los resultados de la búsqueda en el formato JSON que mostramos a continuación (para que así los `getString` por cada uno de los valores del JSON coincidan con el código anterior):

```
{
        "data" : [
           { "from" : {  "name" : "xxx", ... }, "message" :  "yyy", ... },
           { "from" : {  "name" : "ppp", ... }, "message" :  "qqq", ... },
           ...
        ],
        ...
}
```

La API de streaming ofrece un modo de analizar y generar JSON en streams. Le **otorga al programador el control sobre el análisis y la generación**. La API de streaming ofrece un analizador basado en eventos y brinda al desarrollador de aplicaciones la posibilidad de pedir el evento siguiente en lugar de tener que ocuparse del evento en una devolución de llamada.

De este modo, el desarrollador cuenta con mayor control procedimental del procesamiento JSON. El código de aplicación puede procesar o descartar el evento del analizador y pedir el siguiente evento (extraer el evento). **El modelo de streaming es adecuado** para el procesamiento local cuando **no se requiere acceso aleatorio** a otras porciones de la información. De manera similar, la API de streaming permite generar JSON bien formado en stream escribiendo un evento por vez.

Las clases principales de la API de streaming son las siguientes:

- **Json**: Contiene métodos estáticos para crear analizadores y generadores JSON, y sus objetos de fábrica.
- **JsonParser**: Representa un analizador basado en eventos que puede leer datos JSON en un stream. Proporciona acceso directo de solo lectura a datos JSON usando el modelo de programación con análisis pull. En este modelo, el código de aplicación controla el subproceso y llama métodos en la interfaz del analizador para hacer avanzar el analizador u obtener datos JSON del estado actual del analizador.
- **JsonGenerator**: Escribe datos JSON en forma de stream, con un valor por vez. Proporciona métodos para **escribir** datos JSON en un **stream**. El generador puede usarse para escribir pares de nombres/valores en objetos JSON y valores en matrices JSON.

A continuación, utilizaremos la API de streaming con el mismo fin que usamos la API de modelos de objetos, es decir, llevar a cabo una búsqueda de posts públicos de Facebook sobre java

```
1  URL url = new  URL("https://graph.facebook.com/search?q=java&type=post");
2     try (InputStream is = url.openStream();
3           JsonParser parser = Json.createParser(is)) {
4          while (parser.hasNext()) {
5              Event e = parser.next();
6              if (e == Event.KEY_NAME) {
7                  switch (parser.getString()) {
8                     case "name":
9                          parser.next();
10                        System.out.print(parser.getString());
11                    System.out.print(":  ");
12                    break;
13                 case "message":
14                    parser.next();
15                    System.out.println(parser.getString());
16                    System.out.println("---------");
17                    break;
18             }
19          }
20       }
21    }
```

Las líneas 1 a 3 crean un analizador de streaming; las líneas 4 a 5 obtienen el evento siguiente; la línea 6 busca el evento KEY_NAME; las líneas 8 a 11 leen los nombres y los imprimen; y las líneas 14 a 16 leen los posts públicos y los imprimen.

El uso de la API de streaming proporciona un modo eficiente de obtener acceso a nombres y sus posts públicos si se compara con el uso de la API de modelos de objetos para llevar a cabo la misma tarea.