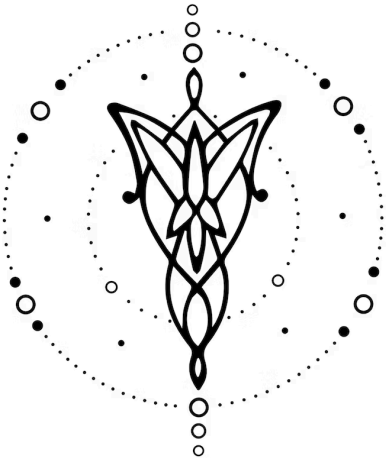

eSearch21

Diseño e Implementación de un Lenguaje

v1.0.0

1. Equipo

Nombre	Apellido	Legajo	E-mail
Tomas	Gonzalez Colasantti	63281	tgonzalezcolasantti@itba.edu.ar

2. Repositorio

La solución y su documentación serán versionadas en: TP-Automatas-y-Compiladores.

3. Dominio

Desarrollar un lenguaje de búsqueda sobre un cuerpo de archivos almacenados en una base de datos en base a tags asignados y metatags. Dicho lenguaje debe permitir fácilmente realizar operaciones complejas de lógica booleana sobre los tags y debe ser lo suficientemente potente para ser utilizado como una herramienta interna de desarrolladores, y lo suficientemente sencillo para ser usado como una herramienta externa a disposición de usuarios finales no expertos.

Las consultas realizadas en este lenguaje serán traducidas a SQL para operar sobre una base de datos relacional, pero tambien podria ser adaptado a una base de datos No-SQL o a alguna otra alternativa más eficiente a futuro, sin necesidad de hacer grandes cambios en la lógica interna de la aplicación que lo utilice. Esto permitiría, por ejemplo, que un emprendimiento comience con una base de datos de Firestore por su sencilla configuración, y que luego al necesitar manejar gran volumen de documentos, migren a una base de datos relacional en PostgreSQL normalizada para mayor performance, únicamente modificando el back-end del compilador (y migrando los contenidos de la base de datos).

4. Construcciones

El lenguaje desarrollado debería ofrecer las siguientes construcciones, prestaciones y funcionalidades:

(I). Realizar búsquedas básicas por tags asignados a archivos, separados por espacios o '\n'. Los tags podran contener únicamente letras del alfabeto inglés (no distingue de mayúsculas y minúsculas), números, guiones medio y bajo, y paréntesis. Los tags no podrán comenzar con paréntesis, pero si terminar.

(II). Agrupar tags en unidades lógicas mediante el uso de paréntesis para búsquedas de mayor complejidad. Se notará como " ' (' + *términos* + ') ' ". Nótese el espacio que separa los paréntesis lógicos de los demás tags.

(III). Realizar búsquedas por metatags. Dichos metatags se notarán colocando su nombre, seguidos de ':' y luego el parámetro a buscar (ej. "createdby:tgonzalezcolasantti"). Los mismos podrán recibir como parámetro, según el tag, un string que siga los lineamientos dados para los tags, un entero positivo o una fecha (DD/MM/AAAA). Cada metatag deberá ser único dentro de cada unidad lógica. Metatags definidos: *name:string*, *createdby:string*, *createdon:date*, *editedby:string*, *editedon:date*, *lasteditedby:string*, *lasteditedon:date*, *likedby:string*, *likes:integer*, *type:pdf|doc|img|vid|aud*, *size:integer*, *views:integer*, *pool:string*.

(IV). Permitir ordenar los resultados según fecha de creación, última modificación, cantidad de likes, tamaño, cantidad de accesos, o random. Solo podrá haber un tag de orden por consulta en el nivel lógico superior, dado que tener dos sería redundante, y poner tags en subniveles no producirá el resultado esperado por cómo funciona el DBMS.

(V). Realizar operaciones de lógica booleana AND, OR y NOT sobre los tags o grupos de tags. Estos también incluyen metatags, salvo *orden*, que no podrá verse afectado por

operadores lógicos OR y NOT. La operación predeterminada será AND, denotada por el separador '*espacio*' entre tags. El operador OR se indicará mediante el símbolo '~' separado de los términos, y el NOT, mediante un '!' pegado a un término. El orden de precedencia es NOT, OR, AND ("*tag1 tag2* ~ !*tag3*" -> "(tag 1 AND (tag2 OR (NOT tag3)))".

(VI). Permitir el uso de *wildcards* en búsqueda de tags. Los mismos serán notados con el caracter '*', que podrá colocarse antes o después de un término a buscar, pegado al mismo, para indicar si es prefijo, sufijo, o infijo. No se podrán colocar *wildcards* en medio de un tag (ej. "*calcu*dora*" no es una búsqueda válida).

(VII). Permitir definiciones de rangos para metatags que no reciban string como tipo de dato. Dichos rargos serán notados con '..' (ej. "*creationdate*:01/01/2024..01/07/2024" o "*score*:100..1000") .

(VIII). Permitir rangos indefinidos para metatags que no reciban string como tipo de dato mediante los operadores '>', '<', '>=', '<=' (ej. "*score*:>=100"). El operador no igual es reemplazado por aplicar NOT al metatag solicitado.

5. Casos de Prueba

Se proponen los siguientes casos iniciales de prueba de **`aceptación`**:

(I). Una búsqueda nula (sin ningún tag o metatag, que deberá retornar todos los docs).
(II). Una búsqueda de un único tag.
(III). Una búsqueda de múltiples tags separados por espacios.
(IV). Una búsqueda de múltiples tags mezclando AND, OR y NOT.
(V). Una búsqueda mezclando tags y metatags sin rangos.
(VI). Una búsqueda mezclando tags y metatags con y sin rango, definido e indefinido.
(VII). Una búsqueda utilizando unidades lógicas en combinación con operadores lógicos.
(VIII). Una búsqueda utilizando tags con y sin wildcards.
(IX). Una búsqueda aplicando el metatag *order*:.
(X). Una búsqueda combinando todas las funcionalidades descritas anteriormente.

Además, los siguientes casos de prueba de **`rechazo`**:

(I). Una búsqueda con un tag que comience con paréntesis.
(II). Una búsqueda con un tag que contenga un símbolo prohibido.
(III). Una búsqueda con paréntesis desbalanceados.
(IV). Una búsqueda con un metatag sin argumento.
(V). Una búsqueda con un rango determinado incompleto.
(VI). Una búsqueda con un operador lógico OR aislado (que termine o comience en '~').
(VII). Una búsqueda con un operador lógico NOT aislado ('... ! ...').
(VIII). Una búsqueda con metatags duplicados dentro de una unidad lógica.
(IX). Una búsqueda con un metatag con parámetro del tipo incorrecto (número en fecha, fecha en número, fecha inexistente, string con símbolos prohibidos).

6. Ejemplos

Una búsqueda de documentos clasificados como ‘urgente’, ‘contabilidad’, ‘proveedores’ y no ‘aprobado’, que haya sido editado por última vez por usuarioA o por usuarioB (pero en este caso creado por usuarioA), que sea del tipo ‘PDF’ y ordenado por cantidad de favoritos.

```
urgente contabilidad proveedores !aprobado
( lasteditby:usuarioA ~
    ( lasteditby:usuarioB createdby:usuarioA )
) filetype:pdf order:likes
```

Realizar una búsqueda de documentos pertenecientes al pool (carpeta) “1234_Expediente_Moderno” cuyo nombre contenga “Lavarden”, ya sea de tipo imagen marcados como ‘reclamos’, ‘seguro’, ‘robo’, creados entre el 01/01/2024 y el 01/02/2024; o documentos de tipo video marcados como ‘reclamos’ o ‘evidencia’ creados por usuarioC pero editados por usuarioD, o en su defecto likeados por usuarioE, que pesen al menos 100MB y que no hayan sido editados por usuarioF, ordenados por fecha de creación (default).

```
pool:1234_Expediente_Moderno name:*lavarden*
( type:img reclamo seguro robo createdon:01/01/2024..01/02/2024 ) ~
( type:vid reclamo ~ evidencia
    ( createdby:usuarioC editedby:usuarioD ) ~
    likedby:usuarioE size:>=100 !editedby:usuarioF
)
```