UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

MANUAL DE USUARIO

SISTEMA DE RECUPERACIÓN DE INFORMACIÓN

Programado en Dr Racket de Scheme.

Profesor: Roberto González

Alumno: Roberto Orellana

Fecha de Entrega: 22/10/2017

Santiago de Chile

21 - 10 – 2017

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCION

El siguiente documento presentará de forma explicativa y sencilla cómo utilizar la siguiente aplicación, dentro de lo que se le procede a explicar, se encuentran las siguientes interrogantes: ¿Cómo compilar la aplicación?, ¿Cómo utilizarla de manera apropiada?, ¿Cuáles son los posibles errores que se pueden encontrar durante la ejecución del problema?, ¿Qué funcionalidades cumple la aplicación?, etc. Es recomendable poseer un nivel de leguaje técnico capaz de comprender los conceptos que se mencionen, para poder hacer buen uso de esta aplicación.

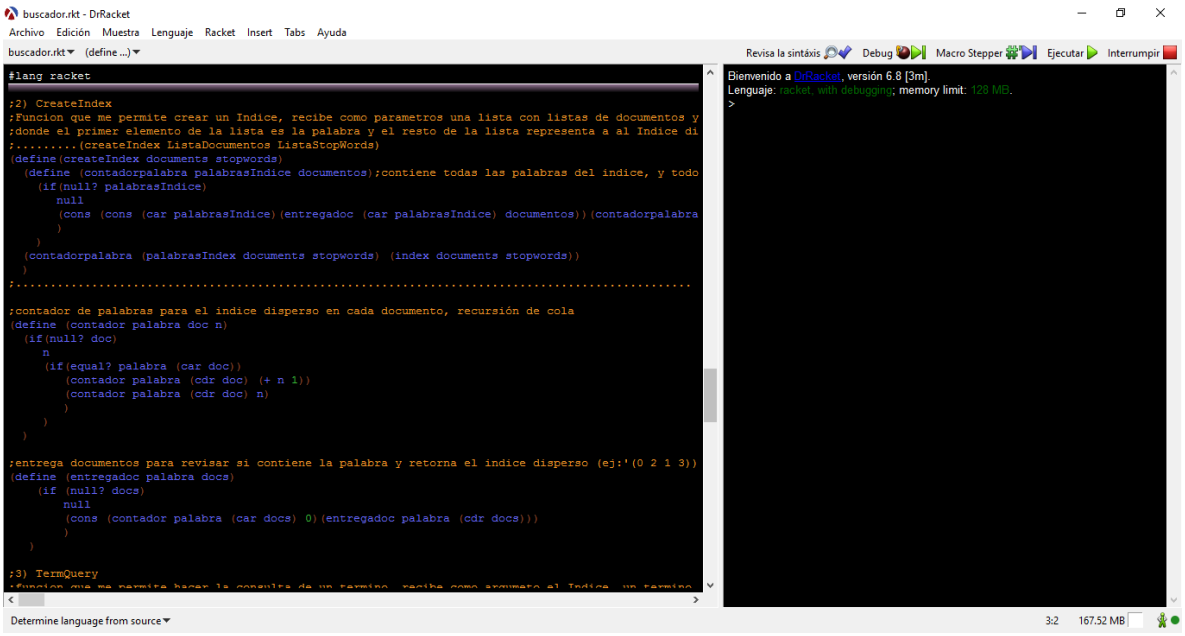
MANUAL DE USUARIO “BUSCADOR”

CAPÍTULO 2. INSTRUCCIONES DE USO

En el siguiente manual se explica la manera de operar de las reglas generadas en el programa, además de ejemplos de cómo implementarlas.

Para su ejecución es necesaria la instalación de DrRacket un intérprete de Scheme, además de conocimientos básico-intermedio sobre programación.

Luego de instalar la aplicación debe abrir el archivo (buscador_17411947_OrellanaTamayo.rkt), que lo enviara a la siguiente ventana donde podrás ingresar los comandos en la parte derecha de su pantalla (a veces la puede estar en la parte inferior de la pantalla) y llamar a las funciones que contiene el Buscador.



2.1 CreateIndex

```
(define(createIndex documents stopwords)
```

Esta función crea un índice, el Índice me permite encontrar información en documentos y colecciones de documentos, esta función solicita 2 argumentos como parámetros de entrada, documents y stopwords que son una lista de documentos y palabras que vienen precargados en el programa. Por ejemplo:

documents

```
(define doc '('(1 "titulo1" "autores1" "biblio1" "texto numero uno")
              '(2 "titulo2" "autores2" "biblio2" "texto numero dos")
              '(3 "titulo3" "autores3" "biblio3" "texto numero tres")
              '(4 "titulo4" "autores4" "biblio4" "texto numero cuatro")))
```

Stopwords

```
(define ListaStopWords '("a" "about" "above" "across" "after" "afterwards" "again" .....
```

La función se llamaría de la siguiente manera:

```
(createIndex ListaDocumentos ListaStopWords)
```

Esta función crea un Índice disperso representada por una lista de lista de elementos, donde el primer elemento de la lista corresponde a la palabra del índice y el resto de la lista cada elemento corresponde a la cantidad de veces que se repite la palabra en cada documento.

Palabra Doc1 Doc2 Doc3 Doc4 Doc5

```
'(("experimental" 3 0 0 0 0
```

Resultado:

```
> (createIndex ListaDocumentos ListaStopWords)
'(("experimental" 3 0 0 0 0)
  ("investigation" 2 0 0 0 0)
  ("aerodynamics" 2 0 0 0 0)
  ("wing" 4 0 0 0 0)
  ("slipstream" 6 0 0 0 0)
  ("brenckman" 1 0 0 0 0)
```

2.2 TermQuery

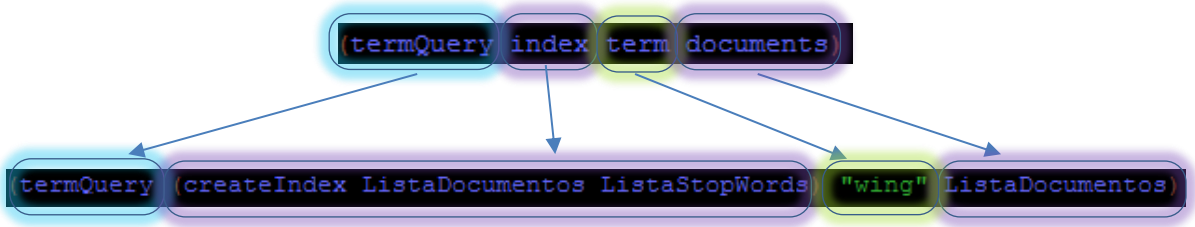
```
(define (termQuery index term documents)
```

Esta función solicita 3 argumentos como parámetros de entrada, index que corresponde al índice entregado por la función createIndex, term corresponde al string que se consulta, y documents que sirve para retornar la lista de documentos que contengas incidencias con la palabra consultada. Por ejemplo:

index: corresponde al índice disperso (createIndex ListaDocumentos ListaStopwords)

term: corresponde a la palabra a consultar. (“wing”)

documents: corresponde a toda la lista de documento precargados. (ListaDocumentos)



Esta función permite hacer una consulta al Índice y genera una lista de resultados desordenadas de solo los documentos que contengan la palabra,

Resultado:

Cantidad de veces que se repite la palabra en el documento 1

```
> (termQuery (createIndex ListaDocumentos ListaStopWords) "wing" ListaDocumentos)
'(4
  1
  "experimental investigation of the aerodynamics of a\nwing in a slipstream ."
  "brenckman,m"
  "j. ae. scs. 25, 1958, 324"
  "experimental investigation of the aerodynamics of a\nwing in a slipstream .\n  an
experimental study of a wing in a propeller slipstream was\nmade in order to determine
the spanwise distribution of the lift\nincrease due to slipstream at different angles
of attack of the wing\nand at different free stream to slipstream velocity ratios .
the\nresults were intended in part as an evaluation basis for different\ntheoretical
treatments of this problem .\n  the comparative span loading curves, together
with\nsupporting evidence, showed that a substantial part of the lift
increment\nproduced by the slipstream was due to a /destalling/
or\nboundary-layer-control effect .  the integrated remaining lift\nincrement, after
subtracting this destalling lift, was found to agree\nwell with a potential flow
theory .\n  an empirical evaluation of the destalling effects was made for\nthe
specific configuration of the experiment"))
```

2.3 PhraseQuery

```
(define (phraseQuery index stopwords phrase documents)
```

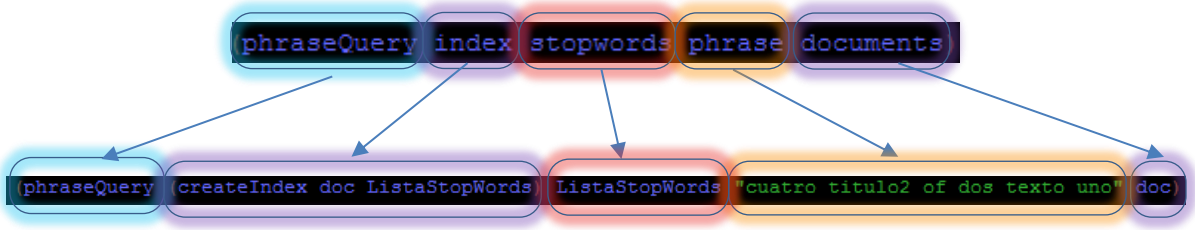
Esta función solicita 4 argumentos como parámetros de entrada, index que corresponde al índice entregado por la función createIndex, stopwords corresponde a la lista de palabras comunes que se eliminan en la consulta, phrase corresponde al string(frase) que se consulta, y documents que sirve para retornar la lista de documentos que contengas incidencias con la palabra consultada. En este ejemplo usaremos otra lista de documentos (doc) para que el ejemplo sea mejor visible Por ejemplo:

index: corresponde al índice disperso (createIndex doc ListaStopwords)

stopwords: corresponde a las palabras comunes (ListaStopwords)

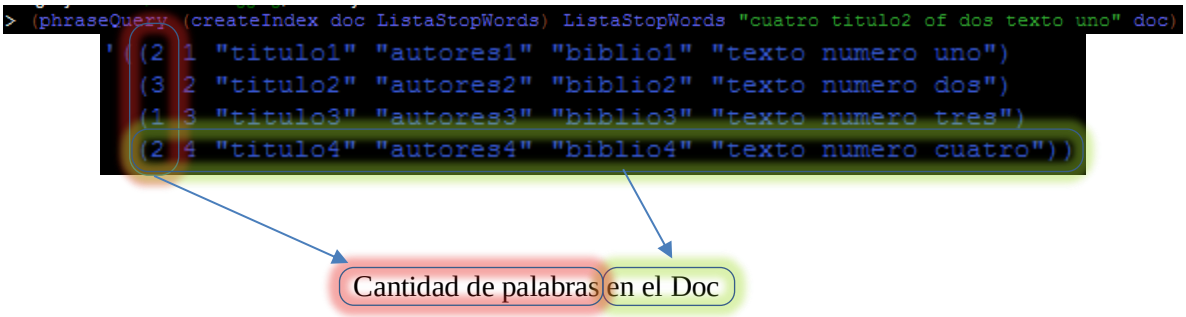
phrase: corresponde a las palabra o frase a consultar. (“cuatro titulo2 of dos texto uno”)

documents: corresponde a toda la lista de documento precargados. (doc)



Esta función permite hacer una consulta de palabras al Índice y genera una lista de resultados desordenadas de solo los documentos que contengan las palabras,

Resultado:



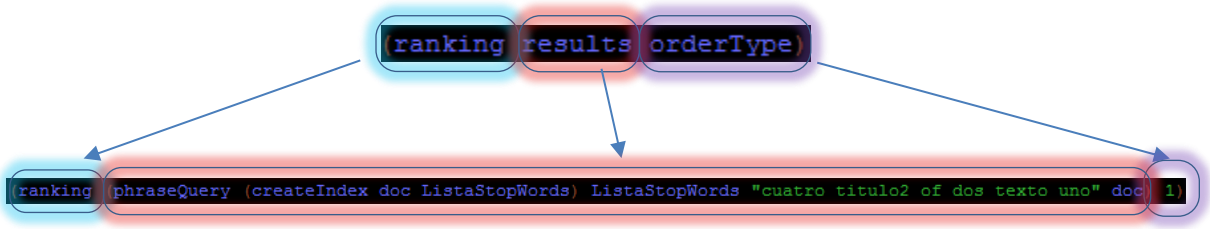
2.4 Ranking

```
(define (ranking results orderType)
```

Esta función solicita 2 argumentos como parámetros de entrada, results que corresponde al resultado entregado por la función TermQuery o PhraseQuery, orderType que corresponde al orden ascendente o descendente.
En este ejemplo usaremos otra lista de documentos (doc) para que el ejemplo sea mejor visible
Por ejemplo:

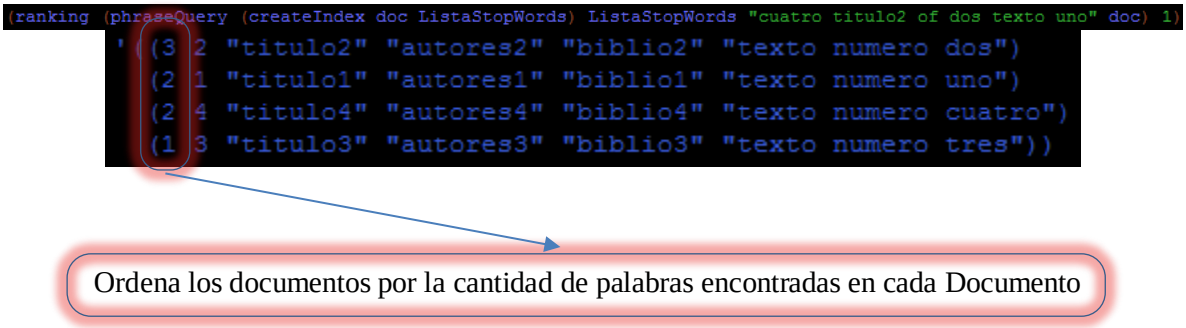
result: corresponde al resultado de la consulta (termQuery o phraseQuery)

orderType: corresponde al orden ascendente(2) o descendente (1)



Esta función nos permite ver el resultado de todos los documentos que contengan las palabras consultadas en orden ascendente o descendente

Resultado:

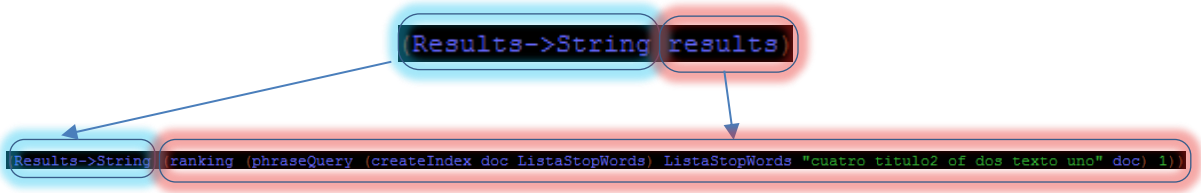


2.5 Results->String

```
(define (Results->String results)
```

Esta función solicita 1 argumento como parámetro de entrada, results que corresponde al resultado entregado por la función Ranking TermQuery o Ranking PhraseQuery.
En este ejemplo usaremos otra lista de documentos (doc) para que el ejemplo sea mejor visible
Por ejemplo:

result: corresponde al resultado de la consulta Ranking(termQuery) o Ranking(phraseQuery)



Esta función permite imprimir y mostrar el resultado lo más claro y legible posible para el usuario.

Resultado:

```
* Contiene 3 similitud con la(s) palabra(s) en el documento
* Documento: 2
* Titulo: titulo2
* Autores: autores2
* Bibliografia: biblio2
* Texto: texto numero dos

* Contiene 2 similitud con la(s) palabra(s) en el documento
* Documento: 1
* Titulo: titulo1
* Autores: autores1
* Bibliografia: biblio1
* Texto: texto numero uno

* Contiene 2 similitud con la(s) palabra(s) en el documento
* Documento: 4
* Titulo: titulo4
* Autores: autores4
* Bibliografia: biblio4
* Texto: texto numero cuatro

* Contiene 1 similitud con la(s) palabra(s) en el documento
* Documento: 3
* Titulo: titulo3
* Autores: autores3
* Bibliografia: biblio3
* Texto: texto numero tres
```

CAPÍTULO 3. POSIBLES ERRORES

Se solicita que el usuario que utilice la aplicación posea conocimientos básicos de programación funcional para utilizar cada una de las funciones, la única forma de que el programa buscador tenga errores, es ingresando datos que no correspondan a cada función.

La aplicación fue pensada y creada para que funcione con ciertos parámetros, por lo que, si estos no son recibidos, existe una gran posibilidad de que la aplicación falle.

Cualquier error por favor comunicarse al número 555-5555555 o al Mail

sin.soporte@SeBusca.cl