Universidad de Valladolid

E.T.S. Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Comparación de las posibilidades de dos máquinas de búsqueda (Lucene y MG4J) sobre textos XML

Autor:

Mª Carmen Loriente Yáñez

Tutor:

Pablo de la Fuente Redondo

Agradecimientos

A mi madre y amigos que me han apoyado a lo largo de este duro año.
A mi novio que al encontrarse en la misma situación durante este año ha sido capaz de darme fuerza cuando esta se agotaba.
A mi tutor Pablo de la Fuente que me ha guiado y ayudado a la generación de este trabajo fin de grado.
A mi empresa por las facilidades que me ha aportado y la ayuda cuando ha sido necesaria.

Resumen

La idea principal del proyecto radica en buscar tanto los puntos fuertes como los puntos débiles de dos máquinas de búsqueda llamadas Lucene y MG4J (Managing Gigabytes for Java), para poder poner en relieve las fortalezas y debilidades de cada una. De este modo cuando vayamos a diseñar una nueva aplicación con búsquedas, será más fácil obtener las ventajas e inconvenientes de usar una u otra tecnología, teniendo siempre en cuenta las necesidades y requisitos de la aplicación.

A lo largo del trabajo buscaremos las diferencias sobre el papel, es decir, de un modo teórico ya que hay bastante escrito sobre ambas tecnologías. Buscaremos las ventajas e inconvenientes propuestas por cada tecnología y en varios estudios previos realizados comparando estas tecnologías.

Pero no pretendemos quedarnos ahí. Queremos obtener resultados también de un modo práctico, para ello buscaremos medidas para cuantificar distintas variables, como pueden ser el tiempo de aprendizaje de cada tecnología, el tiempo de respuesta ante una misma búsqueda, la cantidad de resultados obtenidos, la dificultad a la hora de generar las consultas, el tiempo de añadir la infraestructura necesaria de cada tecnología en un proyecto, etc. Para poder obtener los resultados prácticos deberemos implementar una pequeña aplicación completamente centrada en ambas tecnologías para poder obtener resultados válidos a las medidas con las que hemos decidido trabajar.

Una vez obtenidas las ventajas e inconvenientes, y las medidas cuantificables para cada tecnología intentaremos obtener las conclusiones más objetivas posibles sobre bajo qué circunstancias debería usarse una tecnología o la otra.

Abstract

The main idea of the project is to search the strengths and weaknesses of two search engines called Lucene and MG4J (Managing Gigabytes for Java), to highlight the strengths and weaknesses of each one. Thus when we design a new application with search, it will be easier to get the advantages and disadvantages of using one technology or another, always taking into account the needs and requirements of the application.

Along the work, we will search the differences on the paper, that is a theoretical way, as there is enough written about both technologies. Look for the advantages and disadvantages for each technology proposed in several previous studies comparing these technologies.

But we do not intend to stay there. We also want to achieve results in a practical way, to quantify it we will seek measures to different variables, such as learning time of each

technology, the response time to a single search, the number of results, the difficulty of generate queries, the time to add the infrastructure necessary for each technology in a project, etc.. To obtain practical results we will implement a small application completely focused on these two technologies to obtain valid measures with which we have chosen to work.

After obtaining the advantages, disadvantages, and quantifiable measures we will try to obtain the conclusions as objective as possible about the circumstances under which technology should be used or the other.

Tabla de contenidos

Lista de Figuras

Índice de ilustraciones

Índice de diagramas

Lista de tablas

Índice de tablas generales

Índice de tablas de requisitos

Índice de Tablas de Pruebas

Índice de Tablas Comparativas

Introducción

1.1 Contexto

Hoy en día las máquinas de búsqueda se han convertido en un elemento indispensable en nuestras vidas. Vivimos un momento en el que la cantidad de información de la que disponemos es muy superior a la que podemos absorber, de modo que muchas veces nos encontramos con la necesidad de filtrar y sintetizar lo que nos llega.

Según el **Banco Mundial** (organismo de las Naciones Unidas encargado de dar asistencia financiera y técnica a países en desarrollo) el número de usuarios de Internet como porcentaje de la población hasta 2009 es del 61,2%. Podemos ver esta evolución en la gráfica obtenida a través de la herramienta **Google Data Public Explorer** (herramienta para generar visualizaciones de datos o informaciones públicas. Fundamentalmente sirve para realizar gráficas o mapas para entender fácilmente grandes conjuntos de datos) que se muestra a continuación.

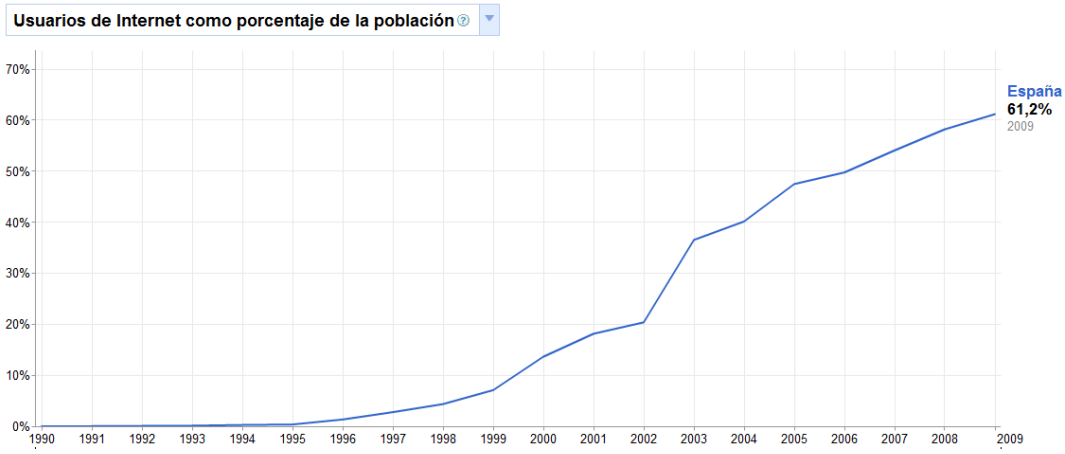

Ilustración 1- Usuarios de Internet como porcentaje de la población

Obtenido de www.undernews.com:

> *"El servicio de estadísticas online Alexa (uno de los más importantes junto con Compete) nos ofrece métricas bastante fiables sobre los principales aspectos de la analítica web (tráfico, usuarios únicos, páginas vistas, etc). Métricas que nos permiten analizar la evolución de las páginas web más importantes.*
>
> *Una de las informaciones que nos provee es el ranking –tanto a nivel nacional como a nivel mundial– de las webs más visitadas."*

ESPAÑA		
Ranking	**Página web**	**Categoría**
1	GOOGLE.ES	BUSQUEDAS
2	FACEBOOK.COM	REDES SOCIALES
3	YOUTUBE.COM	REDES SOCIALES, VIDEOS
4	GOOGLE.COM	BUSQUEDAS
5	LIVE.COM	BUSQUEDAS
6	BLOGGER.COM	REDES SOCIALES, BLOGS
7	YAHOO.COM	BUSQUEDAS, MAIL , CONTENIDO
8	MARCA.COM	MEDIA, DEPORTES
9	WIKIPEDIA.ORG	CONTENIDO
10	MSN.COM	REDES SOCIALES, CONTENIDO, MAIL

Ilustración 2 - Extracto de las 10 primeras páginas más visitadas en 2010 en España

A la vista de la última tabla podemos ver como entre los cinco primeros puestos 3 son buscadores estrictos, pero además podemos añadir youtube.com al conjunto de buscadores ya que per sé no es un buscador de información pero sí un buscador de contenido multimedia.

Como podemos observar cuando nos conectamos a internet para buscar información, contenidos, etc., dependemos absolutamente de que buenos motores de búsqueda nos devuelvan exactamente lo que buscamos.

Pero los buscadores no solo se encuentran en el ámbito de internet, existen otros muchos buscadores que empleamos muy a menudo casi sin darnos cuenta, los buscadores de archivos del sistema, los buscadores dentro de documentos para encontrar palabras, los buscadores de correos electrónicos, etc…

Es decir, que toda investigación y mejora en este campo sólo puede facilitar el día a día de muchos usuarios.

1.2 Importancia de los motores de búsqueda

Volviendo a basarnos en la tabla anterior podemos observar como existen preferencias entre los usuarios a la hora de elegir uno u otro.

Los motores de búsqueda nos devuelven listados de resultados lo más ajustado posible a lo que consultamos. Cuanto más arriba de la lista se encuentra el resultado que contiene la información que buscamos mejor percepción obtendremos del motor. No nos resultará tan importante un gran número de resultados como que podamos encontrar exactamente lo que buscamos entre los primeros resultados. Otra cualidad importante también será la cantidad de tiempo empleado para devolvernos dicho listado.

1.3 Objetivos

El objetivo principal de este estudio es lograr poner de relieve cuales son los puntos fuertes y débiles de cada una de las tecnologías empleadas (en este caso Lucene y MG4J) para la búsqueda sobre documentos XML.

La idea es lograr enfatizar bajo qué circunstancias será mejor emplear una tecnología u otra y de este modo a la hora de enfrentarnos a un nuevo proyecto elegir sin mucho esfuerzo cual encaja mejor ante nuestros requisitos y entorno.

1.4 Fases del trabajo

El trabajo se va a dividir en varias fases para intentar lograr alcanzar los objetivos de una forma organizada.

1.4.1 Investigación

Fase en la que nos dedicaremos a leeremos documentación, buscar información hasta llegar a la comprensión técnica de ambas tecnologías así como de las máquinas de búsqueda en general.

1.4.2 Comparación Técnica

Una vez tenemos toda la información sobre la que se sustentan ambas tecnologías vamos a intentar establecer una comparación sobre el papel de los elementos más relevantes de cada una de ellas.

1.4.3 Estudio Práctico a través de una aplicación

Esta fase la vamos a su vez a subdividir en varias fases:

1.4.3.1 Analizar y Diseñar la aplicación

A lo largo de esta fase nos dedicaremos fundamentalmente a la generación de un nuevo proyecto software para las búsquedas sobre documentos XML. Para ello tendremos que realizar un análisis y diseño de la aplicación antes de comenzar la implementación.

1.4.3.2 Investigación y generación de las infraestructuras

Para ello tendremos que generar la infraestructura necesaria para insertar tanto Lucene como MG4J. Tendremos que tener en cuenta todas las librerías que nos harán falta, la forma de desplegar y el mejor servidor de aplicaciones a utilizar, etc.

1.4.3.3 Implementación de la aplicación

A lo largo de esta fase implementaremos la aplicación diseñada según los requisitos de la fase de análisis y diseño.

1.4.3.4 Ejecución de los casos de prueba

Antes de comenzar a obtener resultados y medidas para la comparación práctica debemos asegurarnos de que la aplicación hace bien lo que tiene que hacer según las especificaciones generadas.

1.4.4 Comparación Práctica

En esta fase tomaremos ciertas medidas sobre la aplicación realizada para poder generar una nueva comparación entre ambas tecnologías.

1.4.5 Conclusiones

Obtendremos las conclusiones tanto de la comparación técnica como práctica para así lograr nuestro objetivo de determinar las circunstancias bajo las que funciona mejor cada tecnología.

1.5 Organización de la Memoria

La organización de la memoria para este trabajo de fin de grado comienza con un estudio básico y fundamental de lo que son en general los (ISR) o la búsqueda y recuperación de información. Será importante analizar bien esto para después introducirnos en dos casos más concretos como son MG4J y Lucene.

Una vez entendido el funcionamiento general el siguiente paso será comprender lo más profundamente posible cada una de las tecnologías a nivel conceptual al menos para poder ir realizando un estudio comparativo entre ambas.

La idea final es realizar una aplicación, con todo lo que ello conlleva, generar una arquitectura válida para las dos tecnologías, realizar un análisis y diseño de la misma, realizar unos casos de prueba básicos para comprobar que el resultado cumple las especificaciones.

Las comparaciones en general necesitan medidas, para poner de relieve las diferencias así como los puntos fuertes de los elementos a comparar. Esto nos proporcionará unos valores y por tanto será una comparación completamente objetiva. Pero las comparaciones también se pueden realizar con elementos no medibles o cuantificables, de modo que dicha comparación vendrá dada por la subjetividad del propio autor y su experiencia con las tecnologías.

Finalmente con todo el trabajo ya realizado solo deberemos obtener las conclusiones que hemos obtenido con el manejo y estudio de ambas tecnologías y cuales podrían resultar los trabajos futuros en ambos campos.

Desarrollo

Capítulo 1 Estudio Teórico

1.1 Términos previos

Para empezar a comprender adecuadamente en qué consiste cada tecnología tendremos que entender algunos términos previos para facilitarnos la comprensión y la comparación entre ambas máquinas. En este punto intentaremos acercar con un lenguaje llano todos estos términos.

1.1.1 Recuperación de información (IR)

Ciencia de la búsqueda de información sobre cualquier tipo de colección documental digital, que se encargada de buscar información sobre los documentos, los metadatos que describen los documentos, o sobre bases de datos relacionales, bien sea a través de internet o intranet con el fin de recuperar textos, imágenes, sonido o datos de otras características, de manera organizada y relevante.

1.1.2 Motor de búsqueda

Consiste en un sistema informático capaz de buscar archivos en servidores web. Este tipo de sistemas los usamos a diario en la red, ejemplos claros pueden ser Google, Yahoo! o Bing, por poner los más potentes hasta un buscador interno dentro de una página Web, como podría ser el buscador de la página de la Universidad de Valladolid.

En general los motores de búsqueda funcionan siguiendo estas operaciones:

- **Crawling o Araña Web** – Se encarga de recopilar información de los archivos y organizarla y priorizarla. El propio contenido de la página será quien determine las pautas para su indexación.
- **Indexación** - La indexación no es más que la generación de un listado ordenado de información. Para nuestro caso concreto será listar del modo que la araña lo solicite la información que se le pase. La existencia de índices facilita el trabajo a los motores de búsqueda otorgando una mayor rapidez de respuesta.
- **Consulta** - Suele ser la cadena que inserta el usuario en busca de resultados. A través de dicha cadena el motor de búsqueda a través de la tabla indexada de información devuelve los resultados más relevantes ante la consulta.

La importancia y utilidad de un motor de búsqueda radica en la relevancia del conjunto de resultados que devuelve tras una consulta.

1.1.3 Indexar

La pieza fundamental en todo motor de búsqueda es la indexación. Para buscar sobre grandes cantidades de texto, primero hay que indexar ese texto y convertirlo en un formato que permita realizar búsquedas rápidamente, eliminando el lento proceso de escaneo secuencial.

La indexación consiste en procesar la información original en una tabla de referencias cruzadas eficiente sobre la que realizar búsquedas más rápidamente.

1.1.3.1 Requisitos de un buen índice de búsqueda

- Reducir al mínimo los accesos a disco a costa del procesador.
- Indexación
  - Debe hacer hincapié en la indexación dinámica para ello la indexación tendrá que ser relativamente rápida sin comprometer el tiempo de búsqueda y deberá simultanear las búsquedas con la indexación.
  - Compresión del índice lo que hará que se acelere todo el proceso provocando un menor número de I/Os
- Búsqueda
  - Para lo que será necesario un índice inverso que logre que las búsquedas sean eficaces y además habrá que combinar los resultados de las búsquedas de consultas distintas.

1.1.3.2 Posibles estrategias de indexación

Estructuras de datos posibles:

Árbol B - O ($\log_b$ N)
- Requiere acceso aleatorio lo que implica accesos constantes a disco

Basado en ordenación por fusión – Θ(n log n)
- Ordenación en memoria, después fusiona los segmentos ordenados crea nuevos segmentos para nuevos documentos añadidos fusiona los segmentos existentes.

Basado en tries - O(longitud del término)
- Requiere de acceso aleatorio, lo que implica búsquedas frecuentes en disco

1.1.3.2.1 Arbol-B

Obtenido de la Wikipedia B-tree:

De acuerdo con la definición de Knuth's un árbol b de orden m es un árbol que satisface:

- *Cada nodo tiene al menos m hijos.*
- *Cada nodo no hoja (excepto el raíz) tiene al menos m/2 hijos*
- *La raíz tiene al menos dos hijos si no es un nodo hoja*
- *Un nodo no hoja con k hijos tiene que tener k-1 claves*
- *Todas las hojas deben estar en el mismo nivel y contener información.*

Cada clave interna de un nodo actúa como o un separador de valores que divide en hijos. Por ejemplo si un nodo interno tiene 3 nodos hijos (o subárboles) entonces debe tener 2 claves a_1 y a_2. Todos los valores del árbol izquierdo serán menores que a_1, todos los valores del nodo hijo central serán mayores que a_1 pero menores que a_2, y los valores del nodo de la derecha serán mayores que a_2.

Un **nodo interno** serán todos aquellos nodos excepto o los nodos hoja o el nodo raíz. Se suelen representar como un conjunto ordenado de elementos y punteros a hijos. Cada nodo interno contiene un máximo de U hijos y un mínimo de L hijos. Lo que implica que el número de elementos tiene que estar entre L-1 y U-1.

El número de hijos del **nodo raíz** tiene el mismo límite que los nodos hijos, pero no tiene límite inferior.

Los **nodos hoja** tiene la misma restricción de número de elementos per no tienen ni hijos ni punteros.

Un árbol B de profundidad n+1 puede contener unas U veces tantos elementos como un árbol B con profundidad n.

H va a ser el peso del árbol, n > 0 las entradas en el árbol y m el máximo número de hijos que puede tener un nodo, por lo que un nodo contendrá m-1 claves.

Por lo que un árbol-B con todas sus claves rellenas tendrá $n = m^h-1$ claves.

Complejidad mejor caso:
$\log_m (n+1)$

Complejidad en el peor caso:
$$\log_d \left(\frac{n+1}{2} \right) + 1$$

Búsqueda

La búsqueda es similar a la búsqueda en árboles binarios. Se comienza por la raíz y recursivamente se recorre desde arriba hasta abajo. Por cada nivel se busca el puntero entre cada valor del que se encuentra el valor buscado.

Inserción

Todas las inserciones comienzan en un nodo hoja. Para ello se busca entre todos los nodos hoja el nodo que deberá contener el valor.

- Si el nodo contiene menos que el máximo de elementos que puede contener el nodo añadimos el elemento en el nodo, manteniendo el orden de las claves
- Si el nodo está completo tendremos que dividir en dos nodos.
    - Se selecciona el valor intermedio como elemento padre
    - Los valores inferiores al padre se ponen en el nodo hoja de la izquierda y los mayores a la derecha.
    - El valor extraído se añade al nodo padre, lo que podría provocar una nueva división y así hasta el nodo raíz. Si al llegar al padre también desborda se crea un padre por encima manteniendo el orden de todas las claves.

Borrado

Se busca el valor a borrar
SI el valor se encuentra en un nodo hoja se elimina y se termina la operación.
Si es un nodo interno se re balancea el árbol.

- Elegimos una nueva clave para el nodo, entre el elemento más grande del árbol de la izquierda o el más pequeño en el de la derecha. Eliminamos la clave de la hoja nodo a la que pertenece y reemplazamos la clave a borrar con el nuevo valor
- Si al realizar el intercambio dejamos la hoja nodo con menos elementos de los que debe, realizamos un re balanceo de carga.
    - Si el nodo que no cumple las propiedades contiene una rama a la derecha con más del número de elementos mínimos lo rotamos.
        - Copiamos la clave separadora del padre al final del nodo padre
        - Reemplazamos el separador en el nodo padre con el primer elemento de la rama de la derecha
        - El árbol ahora está balanceado
    - Si el nodo problemático tiene nodos a la izquierda con más del número mínimo de elementos
        - Copiamos el separador a la primera posición del nodo padre
        - Reemplazamos el separador en el padre con el último elemento de la rama de la izquierda
        - El árbol ahora esta balanceado
    - Si tanto la rama de la izquierda como la de la derecha tienen el número mínimo de claves, entonces tendremos que eliminar el nodo y fusionar el resto de claves
        - Copiamos el separador al final del nodo del árbol izquierdo
        - Movemos todos los elementos del nodo derecho al izquierdo
        - Eliminamos el separador del padre y el nodo vacío hijo
            - Si el padre es raíz y no tiene elementos, vacíalo convierte el nodo fusión en nodo raíz
            - Si el padre tiene menos que el número de claves requeridas, re balanceamos de nuevo

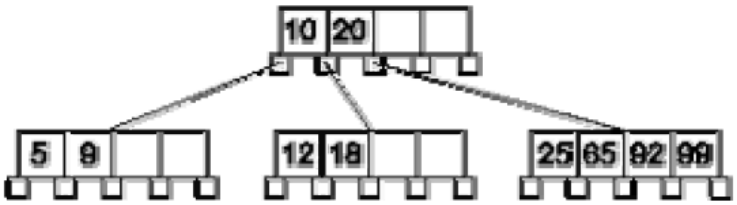

Ilustración 3 - Ejemplo de Árbol-B de orden 4 y profundidad 2

1.1.3.2.2 Ordenación por Fusión

Se trata de combinar dos estructuras de datos ordenadas logrando así una estructura de datos mayor.

Según Br. José Gregorio Justo Torres profesor del departamento de computación de la Universidad de los Ándes:

> *"Este método utiliza la técnica de Divide y Vencerás para realizar la ordenación del vector. Su estrategia consiste en dividir el vector en dos sub vectores, ordenarlos mediante llamadas recursivas, y finalmente combinar los dos sub vectores ya ordenados.*
>
> *Al ordenar un vector mediante las llamadas recursivas y después de fusionar las soluciones de cada parte, necesitamos un algoritmo eficiente para fusionar 2 matrices ordenadas en una única matriz cuya longitud sea la suma de las longitudes de las matrices ordenadas. Esto se puede lograr de una forma más eficiente y sencilla si se dispone de un espacio adicional al final en las matrices ordenadas para utilizarlo como centinela, este centinela es un valor previamente acordado y que se esté seguro que no va a ser mayor que cualquier elemento de las matrices ordenadas.*
>
> *Además otro punto para mejorar la eficiencia del ordenamiento se utiliza como sub algoritmo el ordenamiento por inserción."*

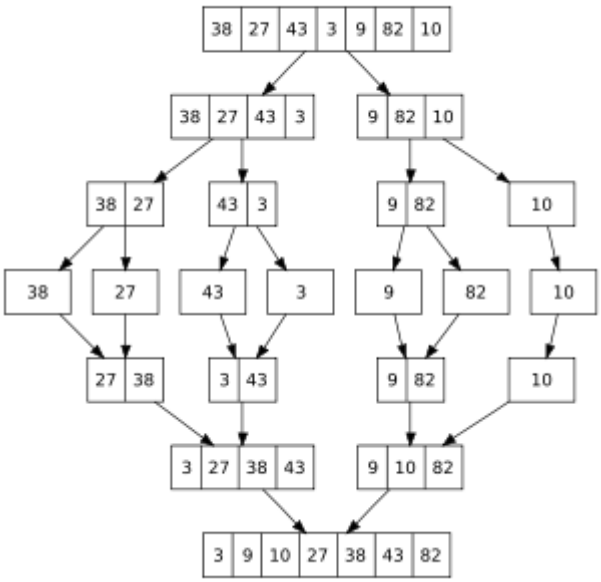

Ilustración 4 - Ordenación por Fusión

Complejidad en el mejor caso, en el peor y en el promedio es la misma:
Θ(n log n)

1.1.3.2.3 Trie

Un trie o árbol digital es una estructura de datos en forma de árbol ordenado que se utiliza para almacenar un conjunto dinámico donde las claves son normalmente cadenas. A diferencia de los árboles binarios de búsqueda, no hay ningún nodo en el árbol que almacene la clave asociada en el nodo, en su lugar, su posición en el árbol define la clave con la que está asociada.

Todos los descendientes de un nodo tiene un prefijo común a la cadena asociada con el nodo, y el nodo raíz está asociado con la cadena vacía.

Ventajas de los Tries sobre los árboles binarios:

1. Buscar en un trie es más rápido en el peor de los casos **O(m)**
2. No existen colisiones de distintas claves
3. No necesita una función de hash.
4. Un trie puede proporcionar una ordenación alfabética de las entradas por clave.

Desventajas:

1. Los tries pueden ser peores a la hora de buscar que las hash table. Sobre todo si para leer la información hay que tener acceso al disco o cualquier otro tipo de memoria secundaria.
2. Algunas claves como por ejemplo los numero en punto flotante, puedes ser muy largos y los prefijos no son particularmente descriptivos.
3. Algunos tries necesitaran más espacio que las hash tables.

Los tries suelen utilizarse bien para autocompletado, textos predictivos, corrección ortográfica o deletreo.

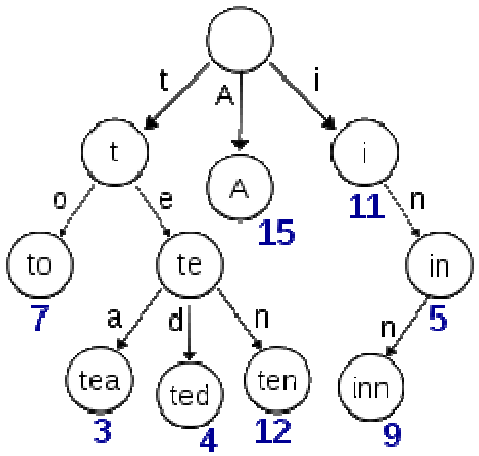

Ilustración 5 - Ejemplo de Trie

Obtenido de la Wikipedia española:

"*Un trie es un caso especial de autómata finito determinista (S, Σ,T,s,A), que sirve para almacenar un conjunto de cadenas E en el que:*

- *Σ es el alfabeto sobre el que están definidas las cadenas;*
- *S, el conjunto de estados, cada uno de los cuales representa un prefijo de E;*
- *La función de transición T: S x Σ → s; está definida como sigue: T(x, σ) = xσ si , x, xσ ∈ S, e indefinida en otro caso;*
- *El estado inicial s corresponde a la cadena vacía λ;*
- *El conjunto de estados de aceptación A ⊆ S es igual a E.*

*Su nombre procede del término inglés re**trie**val."*

1.1.3.2.4 Stemming

Esta técnica lo que pretende es eliminar las posibles confusiones semánticas de modo que aprovechando las estructuras de las palabras (sufijos, prefijos y raíz) trunca o reduce las palabras a su raíz. Una de las formas de implementación más típicas de esta técnica son los ya descritos Tries.

1.1.3.3 Compresión

La necesidad de comprimir el conjunto de índices se debe a que cuando se trata conjuntos muy grandes de información como podría ser buscadores web, estas tablas podrían llegar a tener tamaños realmente gigantescos.

No importa cómo va creciendo el espacio y la velocidad de transmisión con el tiempo. Siempre acabamos necesitando más espacio y más velocidad, por lo que la compresión de información acaba siendo esencial.

La compresión de textos en el ordenador implica el cambio de representación de un fichero de modo que ocupe menos espacio a la hora de almacenarlo o menos tiempo a la hora de transmitirlo que lo que tardaría el original. De esto extraemos las dos ideas principales de la compresión:

1. Ocupar menos espacio en disco
2. Si se transmite o envía menos tiempo de envío y por tanto menos coste también.

Por ello se buscó la idea de comprimir las tablas de índices para poder almacenar mucha información en menos espacio y aumentando la capacidad de almacenamiento de información. El problema ahora se traslada a la efectividad de la compresión, pero también a la rapidez en que se descomprime y re comprime dicha información

Como hemos visto esto implica un coste de compresión / descompresión para las búsquedas. Cuando los textos son muy grandes la búsqueda se complica ya que es secuencial, por lo que habrá que ir recorriendo todo el texto para resolver la consulta, si a esto le añadimos el tiempo de compresión y descompresión esto puede ser muy costoso. La forma de solucionar este es dividir el texto en bloques y comprimir cada uno de ellos, logrando así tener que descomprimir únicamente el bloque que nos interesa. Esto implica también un menor grado de compresión ya que cuantos más bloquees haya menos se podrá comprimirá ya que los patrones de compresión se minimizan. Por tanto hay que lograr encontrar el equilibrio entre ambas secciones.

.

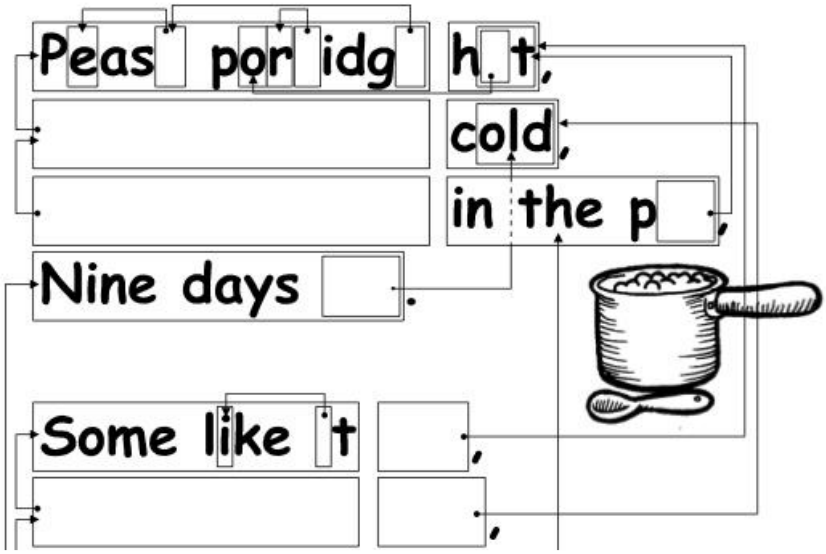

Ilustración 6 - Compresión de textos

1.1.3.3.1 Algoritmo de Huffman

Algoritmo utilizado para la compresión de datos. Se basa en la frecuencia de aparición de los caracteres. Para ello se realiza una codificación binaria. Es una codificación óptima cara compresión símbolo a símbolo.

La técnica consiste en crear un árbol binario donde cada nodo contendrá la información del símbolo a codificar y la frecuencia del mismo. La idea es ir uniendo los nodos de dos en dos (siempre escogiendo aquellos que tienen menor frecuencia) y generando un nuevo nodo con la suma de las frecuencias.

El paso final una vez generado el árbol será ir añadiendo unos y ceros por las aristas para obtener la codificación del símbolo.

Letra	Frecuencia
A	0.5
C	0.3
N	0.15
E	0.5

Tabla 1 Porcentajes codificación Huffman

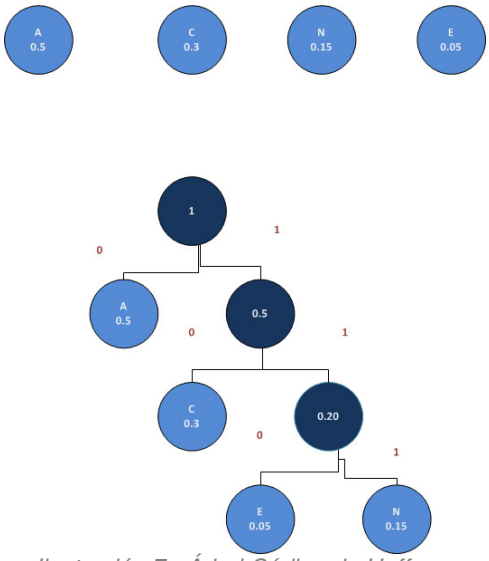

Ilustración 7 - Árbol Código de Huffman

La codificación de cada letra quedaría tal y como se muestra a continuación

Letra	Frecuencia	Codificación
A	0.5	0
C	0.3	10
N	0.15	111
E	0.05	110

Tabla 2 Codificación final

Suponiendo las palabras CENA, CAE o CE

Palabra	Tamaño en bits	Tamaño en bits con Huffman	Codificación
CENA	32	9	10 110 111 0
CAE	24	6	10 0 110
CE	16	5	10 110

Tabla 3 Codificación palabras cena, cae y ce

El tiempo de ejecución del método Huffman es muy eficiente ya que necesita **O(n log n)** operaciones para su construcción.

1.1.3.3.2 Índice Invertido

Son estructuras de datos utilizadas para recuperar palabras. Se trata de almacenar cada término que aparece en los documentos y la lista de documentos en los que

aparece. De este modo la búsqueda de documentos asociados a cada palabra ante una consulta es directa.

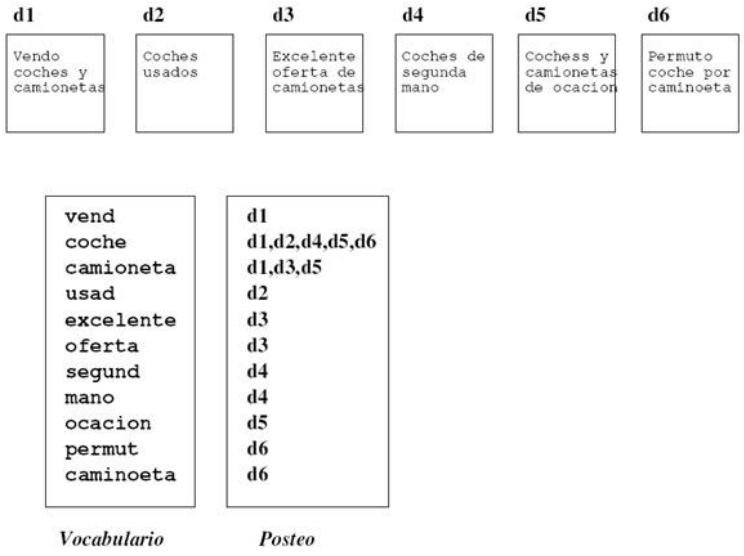

Ilustración 8 - Forma básica de un índice invertido

1.1.3.3.3 Codificación Aritmética

La codificación aritmética es una forma de codificación de entropía utilizada en la compresión de datos sin pérdidas. Normalmente una cadena de caracteres, como por ejemplo la palabra cama se representa mediante un número fijo de bits por carácter. Cuando una cadena se convierte a la codificación aritmética, los caracteres de uso más frecuente se almacenarán con el menor número de bits, mientras que los menos frecuentes se almacenaran con el mayor número de bits.

La gran diferencia entre el código de Huffman y la codificación aritmética es que en lugar de separar la entrada en símbolos y sustituir cada uno con un código, la codificación aritmética codifica el mensaje completo con un solo número.

En la codificación aritmética no se asigna una palabra de código a cada uno de los símbolos del alfabeto fuente. Se trata de asignar a cada símbolo un intervalo entre 0 y 1 de forma que la amplitud de cada intervalo sea igual a la probabilidad de cada símbolo. La suma de cada amplitud de los intervalos debe ser igual a la unidad. Para realizar la codificación de cada uno de los símbolos asociados a un mensaje entrante se siguen los siguientes pasos:

- Se selecciona el primer símbolo de la secuencia de entrada y se localiza el intervalo asociado a ese símbolo
- Ahora seleccionamos el siguiente símbolo y localizamos su intervalo. Una vez tenemos los dos se multiplican los extremos de este intervalo por la longitud del intervalo asociado al símbolo anterior (probabilidad del símbolo anterior) y

sumaremos estos resultados al extremo inferior del intervalo asociado al símbolo anterior.
- Realizamos el paso anterior hasta que hayamos recorrido todos los posibles símbolos contenidos en el mensaje.
- Finalmente seleccionamos un valor cualquiera contenido dentro del intervalo del último símbolo. Este valor será la secuencia que enviaremos.

1.1.4 Buscar

Buscar en principio es un concepto que todo el mundo conoce. A priori podríamos pensar que el proceso de buscar comienza desde que un usuario inserta una cadena (o consulta) sobre un cajetín de texto y selecciona el botón buscar, hasta que la aplicación devuelve el conjunto de documentos filtrados a través de la consulta. Pero vamos a simplificar el proceso al máximo y vamos a considerar una búsqueda como el proceso de encontrar sobre la tabla de índices el texto insertado devolviendo el conjunto de ficheros que contienen dicha consulta.

A la hora de buscar existen dos factores muy importantes como son la relevancia de los documentos encontrados y el tiempo de respuesta. El modelo ideal sería obtener justo el documento que se busca en la primera posición en un tiempo indetectable para el usuario, pero esto no resulta tan sencillo.

1.2 Búsquedas sobre ficheros XML

Habitualmente cuando hablamos de búsquedas e indexación de ficheros nos referimos a ficheros completos, sin ningún tipo de tratamiento, sin embargo los ficheros XML no se pueden indexar directamente, ya que al ser un lenguaje de marcas, tenemos que tratar el contenido previamente, para poder eliminar las marcas.

1.2.1 XML

Las siglas XML se traducen por e**X**tensible **M**arkup **L**enguage,, es decir, lenguaje extensible de marcas, lo creo el W3C para lograr almacenar información ordenada.

Es una tecnología muy extendida no solo utilizada en internet, ya que se ha convertido en un estándar para intercambiar información.

Según la Wikipedia:

"La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información. Ejemplos son un tema musical, que se compone de compases, que están formados a su vez por notas. Estas partes se llaman elementos, y se las señala mediante etiquetas."

La estructura de un fichero XML es la siguiente:

- Una declaración inicial donde se indica que es un fichero XML y el contentType
- Una definición de las reglas y la estructura que debe tener el XML a través de un DTD o XSD, servirá para validar que tiene un formato válido
- Un conjunto de etiquetas que siguen las reglas establecidas por el DTD o XSD
- Cada etiqueta podrá constar de atributos y/o contenido.

Supongamos un repositorio de emails almacenados como si fueran XML, el formato sería algo similar a:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
    <Mensaje>
        <Remitente>
            <Nombre>Nombre del remitente</Nombre>
            <Mail> Correo del remitente </Mail>
        </Remitente>
        <Destinatario>
            <Nombre>Nombre del destinatario</Nombre>
            <Mail>Correo del destinatario</Mail>
        </Destinatario>
        <Texto>
            <Asunto>
                Este es mi documento con una estructura muy sencilla
                no contiene atributos ni entidades...
            </Asunto>
            <Parrafo>
                Este es mi documento con una estructura muy sencilla
                no contiene atributos ni entidades...
            </Parrafo>
        </Texto>
    </Mensaje>
</Edit_Mensaje>
```

Tal y como podemos observar la información está organizada de un modo legible organizada entre etiquetas siguiendo las reglas establecidas por edit_mensaje.dtd (definición del tipo de documento).

1.2.2 Ventajas y desventajas

Las ventajas que nos proporcionan los XML son:

- Los XML tiene la capacidad de definir el contenido de los documentos de una forma jerárquica, lo que hace que los documentos contengan la información estructurada de un modo lógico.

- Los XML se auto describen, las propias etiquetas del XML dan la información necesaria para la comprensión del documento

- XML es una forma de almacenamiento estándar y muy extendida en muchos medios y elementos para el almacenamiento de información

- Los XML pueden contener estructuras muy complejas, son fácilmente extensibles y modificables.

- AL ir determinados estructuralmente por DTDs o XSDs se consigue un alto nivel de integridad en la información.

Las desventajas:

- Una de las desventajas de los XML es que pueden convertirse en documentos muy grandes y cuando las colecciones de estos documentos tambien son muy grandes puede resultar muy complejo.

- La complejidad que tienen los documentos XML aumenta tambien los costes, los usuarios han de conocer los XML y el software que procese los documentos debe ser más complejo que cuando los textos son en formato plano.

1.2.3 Dificultades de indexación sobre ficheros XML

El problema de los ficheros XML es que necesitan de un tratamiento previo para realizar la indexación ya que como vemos en el código anterior, al ser un lenguaje de marcado, contiene etiquetas organizando el contenido. Esto aunque a priori parece un inconveniente, también puede convertirse en una ventaja, ya que podremos indexar la información orientada a la propia organización del XML. A través de esta idea se pueden llegar a realizar búsquedas más refinadas y concretas teniendo en cuenta la información.

Siguiendo un poco con el ejemplo de los correos electrónicos podríamos llegar a indexar por cada uno de los campos que componen un email, generaríamos un campo remitente, otro campo destinatarios, otro sería el asunto y finalmente el párrafo. De este modo el usuario final podrá realizar búsquedas por campos concretos, pudiendo filtrar solo por un tipo de destinatario, o por conjuntos de campos, por ejemplo destinatario y asunto.

Si esto lo hiciéramos directamente sobre ficheros concretos los resultados no estarían tan refinados, ni el usuario final podría delimitar y filtrar tanto la información que desea encontrar.

1.2.3.1 Desafíos de la indexación XML

El primer desafío es que el usuario final va a querer una sección o parte del documento no todo el documento. Por lo tanto en la consulta deberemos encontrar la parte más específica del documento

Paralelamente tendremos el problema de que partes de los documentos por tanto indexar. En Documentos no estructurados el nivel de indexación queda claro, el fichero, pero en los documentos estructurados, pueden existir distintos niveles y aproximaciones.

Una posible aproximación sería subdividirlo en distintos tipos de documentos de modo que unos no afecten a otros.

Podemos también utilizar el elemento menos profundo (que se acercaría a la indexación de documentos no estructurados)

El enfoque menos restrictivo será indexar todos y cada uno de los nodos de todos y cada uno delos niveles, pero esta aproximación puede ser muy problemática ya que muchos elementos no son significativos, y probablemente en la recuperación de los documentos encontremos una gran reiteración de documentos.

Existen muchos casos en los que existirán varios esquemas XML para una misma colección porque serán varias fuentes los que generen estas colecciones. Por lo que podremos encontrar varios nombres que representen lo mismo.

1.3 Apache Lucene

Apache Lucene es la primera tecnología a estudiar para poder generar la comparación entre ambos buscadores. A continuación describiremos que es Lucene y a grandes rasgos como funciona.

1.3.1 ¿Qué es Lucene?

Según los autores *Otis Gospodnetic* y *Erik Hatcher* del libro *Lucene in Action:*

> Lucene es una librería de alto rendimiento y escalabilidad para la recuperación de información permitiendo agregar funcionalidades de indexación y búsqueda en aplicaciones informáticas.
>
> Se trata de un proyecto gratuito, maduro y de código abierto implementado en Java. Forma parte de la familia de proyectos de Apache Jakarta bajo la Apache Software License. Lucene es y ha sido durante muchos años la librería de Java más popular para la IR (recuperación de información).
>
> Lucene es una librería, un conjunto de herramientas si se quiere, pero no una aplicación de búsqueda directamente funcional, digamos que implementa una

nueva capa sobre nuestra aplicación encargada de indexar y buscar, sin importarle la lógica de negocio de la aplicación.

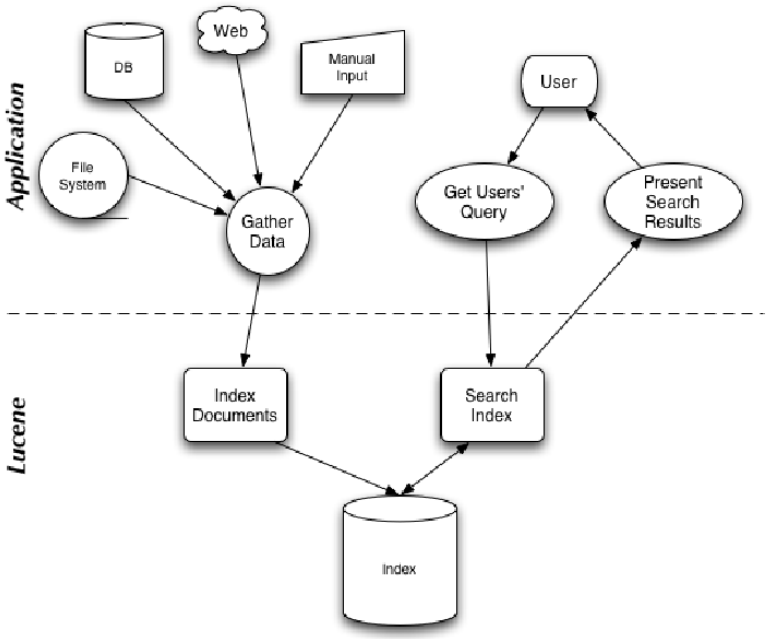

Ilustración 9 - Típica integración entre Lucene y una aplicación Web

1.3.2 ¿En qué consiste?

Según los autores *Otis Gospodnetic* y *Erik Hatcher* del libro *Lucene in Action:*

> Lucene proporciona una API del propio núcleo sencilla y poderosa que requiere conocimientos muy básicos de compresión e indexación de textos. Para poder integrar Lucene tan sólo será necesario incluir un conjunto reducido de clases en la aplicación.

1.3.3 Qué puede hacer Lucene para el desarrollador

Según los autores *Otis Gospodnetic* y *Erik Hatcher* del libro *Lucene in Action:*

> Lucene añade capacidades de búsqueda e indexación a aplicaciones, pudiendo indexar y buscar cualquier información que se puede convertir en texto.
>
> A Lucene no le importa el origen, formato o lenguaje de la información, siempre y cuando se pueda convertir en texto. Esto significa que se puede utilizar para indexar y buscar información almacenada en archivos del tipo: páginas web en

servidores remotos, documentos almacenados en sistemas de archivos locales, archivos de texto, documentos de Microsoft Word, archivos HTML o PDF, o cualquier otro formato desde el que se puede extraer la información textual.

1.3.4 Definición de Documento para Apache Lucene

Para Lucene un documento es la unidad fundamental tanto de indexación como de búsqueda. No es más que un contenedor donde almacenaremos uno o más campos sobre los que organizar la información de los ficheros.

1.3.5 ¿Cómo indexa Apache Lucene?

La indexación se organiza en tres fases:

1. Extraer la información de los documentos
2. Analizar la información extraída en la fase anterior
3. Guardar los índices

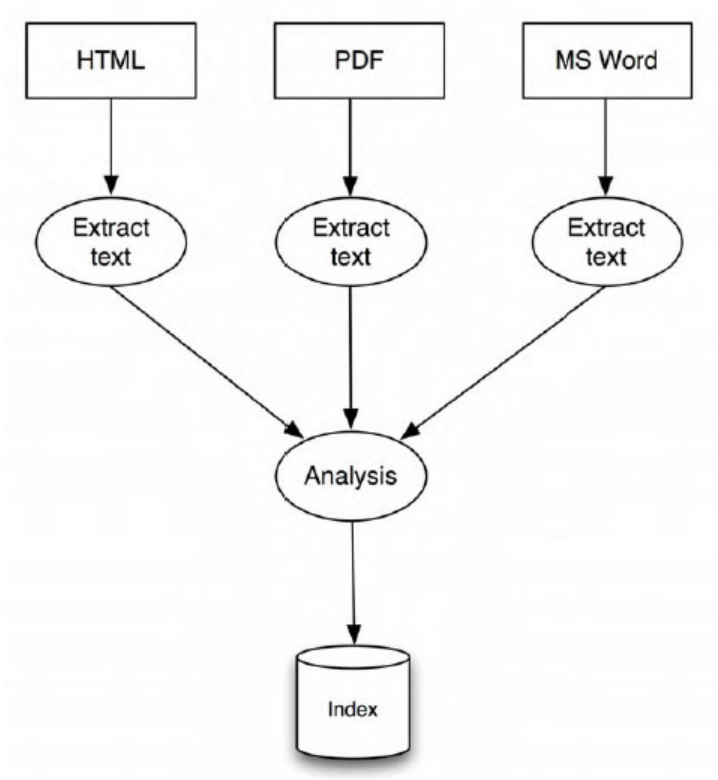

Ilustración 10 - Fases de la indexación Apache Lucene

Las bases para la indexación que aplica Apache Lucene son las siguientes:

- Un índice contiene una colección de documentos
- Un documento es una colección de campos
- Un campo es el nombre que se le da a una colección de términos
- Un término es un par de cadenas <nombre, valor>
- Índice invertido será adecuado para un búsqueda eficiente basada en términos

El primer paso para indexar es determinar qué contenido se tiene que indexar. Para cada contenido que se indexa Lucene crea un objeto del documento qué es la colección de campos. Cada campo está constituido por un par nombre/valor. Finalmente a cada campo le añadimos un campo objeto que determina si el valor del campo se debe indexar, almacenar o convertir en token.

Este campo lo que hace es determinar como Lucene va a utilizar el valor del campo:

- Indexado – Siempre que un campo esté indexado significa que se puede buscar. Luego a la hora de buscar si la búsqueda es global lo hará por todo tipo de campos o se podrá realizar búsquedas sólo por este campo.
- Almacenado – Cuando un campo está almacenado nos devolverá su valor como parte del listado de resultados de la búsqueda.
- Convertido en Tokens – Cuando un campo se convierte en tokens significa que el analizador convierte todo el contenido en tokens. Un token no es más que la unidad básica de indexación y representa a una palabra.

El analizador aparte de extraer el texto a indexar debe aplicar las lógicas de transformación pertinentes como eliminar palabras no relevantes como “el”, “la”, “a”, etc., ejecutan algoritmos de stemming para obtener las raíces de las palabras, convertir todo el texto a minúsculas para no hacer diferencias a la hora de buscar, etc.… De este modo el analizador logra reducir tamaño del índice ya que solo albergará los elementos principales.

Finalmente una vez analizado el texto de entrada este se almacena en índices. Para ello Lucene utiliza índices invertidos.

Las operaciones que se pueden realizar sobre un índice son las de añadir nuevos documentos, eliminar documentos del índice o actualizar documentos existentes en el índice.

1.3.5.1 Fields / Campos

A la hora de indexar los campos (Fields) son fundamentales ya que son los que contienen cada valor a almacenar. Por cada campo que forma un documento podemos establecer opciones para su análisis y para los términos que indexar. También se puede dar mayor o menor relevancia a los campos según las funciones de Boosting, ya que la información no

posee siempre la misma importancia. En el ejemplo del email podríamos establecer que por ejemplo los campos de asunto y párrafo son más relevantes que los de emisor, por ejemplo.

Los campos no tienen por qué ser siempre textuales, podemos tipar los campos para luego mejorar las búsquedas, pudiendo realizarlas por rangos, numéricas, etc. Tipos:

- Fechas y Horas
- Números
- Texto
- Truncado

1.3.6 ¿Cómo busca Apache Lucene?

Las fases de búsqueda de Lucene se las siguientes:

1. Inserción por parte del usuario los elementos a buscar
2. Generación de una consulta para buscar en los índices
3. Búsqueda a través de la consulta en los índices
4. Generación del listado de objetos resultado que devolver al usuario

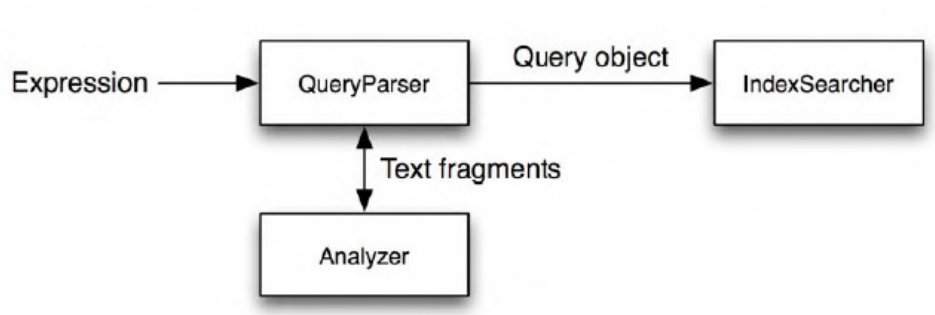

Ilustración 11- Fases de la búsqueda

La generación de consultas puede resultar en ocasiones muy sofisticada, para ello Lucene ha generado un objeto abstracto Query que admite multitud de formatos para generar la consulta. Parsear la expresión de consulta no es más que convertir el texto insertado por el usuario en una instancia correcta del objeto de consulta de Lucene. Las consultas podrán ser más o menos complejas dependiendo de las búsquedas que queramos realizar. Lo bueno es que como ya hemos visto antes los campos no solo son textuales sino también numéricos y con fechas lo que permitirá realizar búsquedas por rangos. También permite la posibilidad de crear consultas asignando valores a varios campos, refinando aún más la búsqueda.

Esta consulta y el directorio donde están almacenados los índices se le pasan a una nueva clase encargada de ejecutar la consulta sobre los índices y generar el objeto resultado de la búsqueda.

Este objeto resultado no es más que el conjunto de los Documentos que concuerdan con la búsqueda organizados según la importancia que tengan o no ciertos campos y el número total de resultados que se han encontrado.

Normalmente Lucene te devuelve un número reducido de resultados, por ejemplo, 10 no todos, de modo que estos se paginen. Al intentar ir a la siguiente página lo que hace Lucene es re-ejecutar los pasos de la búsqueda y generando el objeto de resultados empezando por el que corresponde según la paginación.

Lucene tiene implementado un mecanismo de puntuación para las búsquedas, de modo que cada vez que realizamos una búsqueda, si un documento está en esa búsqueda empezará a puntuar más y por tanto cada vez dicho documento aparecerá más arriba en la búsqueda.

1.4 MG4J

Managing Gigabytes 4 Java es la segunda tecnología a estudiar para poder generar la comparación entre ambos buscadores. A continuación describiremos que es MG4J y a grandes rasgos como funciona.

1.4.1 ¿Qué es MG4J?

MG4J es un motor de búsqueda gratuito de texto completo para grandes colecciones de documentos. Se ha implementado en Java y es un sistema personalizable y de alto rendimiento.

Según Sebastiano Vigna su creador:

> *"MG4J es un framework para la construcción de índices de grandes colecciones documentales basado en la clásica aproximación de índice invertido. El tipo de índices construidos son configurables. Para darle aun mayor potencia se generó también Archive4J que hace posible el acceso rápido y altamente comprimido a la información de los colecciones de documentos"*

1.4.2 ¿En qué consiste?

MG4J es un proyecto colaborativo para generar una implementación java gratuita de los técnicas de compresión de índices invertidos. Como producto ofrece clases optimizadas de propósito general, incluyendo cadenas mutables, entradas y salidas a nivel de bit, etc...

Las funciones de MG4J proporcionan un sistema de indexación de textos en toda regla. Es capaz de analizar, indexar y consultar grandes colecciones de texto.

1.4.3 Qué puede hacer por el desarrollador

- **Potente indexación**. MG4J hace posible el análisis la indexación y las consultas de un modo consistente sobre amplias colecciones de documentos otorgando facilidad de comprensión y poniendo de relieve los pasajes más importantes de los documentos recuperados.
- **Eficiencia**. Es capaz en pocos segundos de indexar sin esfuerzo cientos de millones de documentos.
- **Múltiples índices de intervalos semánticos**. Cuando se ejecuta una consulta MG4J devuelve para cada índice una lista de intervalos semánticos que satisfacen la consulta.
- **Operadores**. Cada operador es representado internamente como un objeto abstracto por lo que fácilmente se puede conectar con la sintaxis preferida.
- **Flexibilidad**. Se pueden construir índices grandes o pequeños según las necesidades.
- **Abierto**. Las factorías e interfaces otorgan un modo fácil de representar tus datos.
- **Procesos distribuidos**. Los índices se pueden construir de una colección y luego dividirlos en partes y combinarlos más tarde.
- **Multihilo**. Los índices pueden ser calculados y consultados concurrentemente.
- **Agrupamiento**. Los índices pueden ser agrupados tanto lexicalmente como documentalmente. El sistema de agrupamiento es completamente abierto y es el usuario quien decide las estrategias para combinar los documentos de distintas fuentes.

1.4.4 Definición de Documento para MG4J

Un documento proporciona el acceso a distintos campos que representan la unidad de información que debe ser indexada por separado. Cada campo (field) deberá ser accesible.

1.4.5 ¿Cómo Indexa MG4J?

MG4J para generar un índice necesitará una colección o secuencia de documentos organizados por los campos por los que se pretende indexar. También necesitará una factoría que se encarga de mantener la integridad entre los documentos que se van a indexar y los campos que debe tener. También necesitará un tipo de iterador que sea capaz de iterar por dicha colección de documentos.

MG4J generará un índice por cada campo que compone un tipo de documento, de modo que a la hora de filtrar por dicho campo accedamos a ese índice.

El codificador tiene dos fases, la primera se extrae el léxico de las palabras y se cuenta la frecuencia. En managing gigabytes una palabra se define como una secuencia continua de caracteres alfanuméricos de longitud máxima 15, con no más de 4 números.

En la segunda fase se comprime el texto utilizando los parámetros de léxico y frecuencia obtenidos en la primera fase.

Para la compresión del texto Managing gigabytes utiliza Huffman.

1.4.6 ¿Cómo Busca MG4J?

MG4J al ejecutar la consulta retorna el árbol de resultados que coinciden con la consulta, (no devuelve documentos) retorna un listado que ir recorriendo para poder localizar los documentos.

1.5 Ejemplo práctico de indexación

Vamos a suponer un repositorio de emails. Supongamos que queremos un buscador para ir encontrando los emails que nos hacen falta.

- La colección de documentos serán todos los emails.
- Cada tipo de documento tendrá una colección de campos, en este caso como todos los documentos son emails solo tendremos una colección de campos.
- La colección de campos estará formada por un conjunto de nombre y valor a continuación mostramos los nombres:
    - De
    - Para
    - CC
    - Adjunto
    - Asunto
    - Contenido
- Cada email contendrá esos campos con un valor asociado.
- Ahora a cada campo le vamos a añadir el campo objeto para indicar de que tipo queremos que sea:
    - De – Indexado y almacenado
    - Para – Indexado sin convertir a tokens
    - CC – Indexado sin convertir a tokens

- Adjunto – Indexado convertido a tokens
- Asunto - Indexado y almacenado
- Contenido – Indexado y convertido a tokens

- A través del analizador generará el índice.

En la siguiente figura se representa gráficamente todo el proceso que se ha analizado aquí.

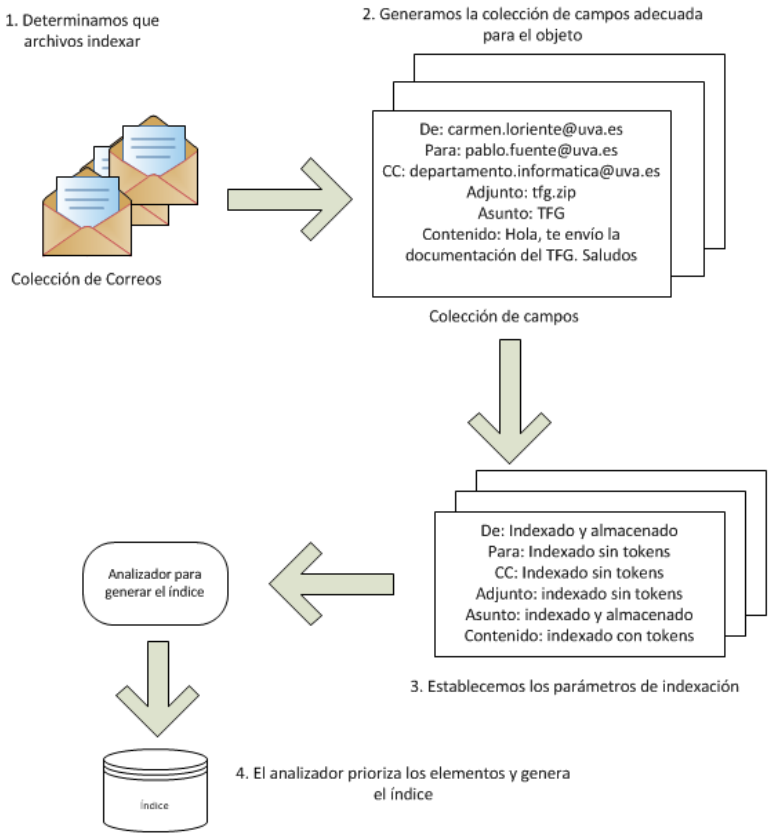

Ilustración 12 - Ejemplo práctico de Indexación

1.6 Comparación sobre el papel

A la vista de lo explicado hasta el momento, podemos sacar algunos datos y conclusiones al respecto.

1.6.1 Tipo de Indexación

MG4J utiliza un tipo de indexación por lotes lo que implica que ante nuevos documentos habría que realizar una re-indexación completa de la colección para poder añadir los nuevos documentos, sin embargo, Lucene utiliza una indexación incremental, de modo que en un momento dado se pueden añadir nuevos documentos, sin tener que re generar el índice completo. En este punto quizás Lucene ofrezca la mayor fortaleza.

1.6.2 Compresión

Lucene a la hora de analizar los textos para comprimirlos y generar los índices fundamentalmente utiliza las técnicas de eliminar stopWords, la codificación de byte variable para números y Codificación por delante de los términos, que en un ejemplo entenderemos fácilmente. Supongamos una serie de términos con la misma raíz ordenados por cantidad de caracteres: (8) autómata, (10) automático, (11) automatizar, (14) automatización, la forma que tendrá Lucene de almacenarlos será:

0	automata
7	ico
7	izar
7	izacion

Tabla 4 - Codificación por términos

Sin embargo MG4J utiliza en una primera fase stemming para obtener las raíces y las frecuencias y finalmente codifica todo el texto con el código de Huffman que como hemos visto anteriormente tiene un alto nivel de compresión y una de las ventajas que tenía es que es muy rápido.

1.6.3 Búsquedas

MG4J tiene implementadas dos tipos de búsquedas, las booleanas que se encargan de realizar búsquedas a través de la combinación de varias consultas y las consultas de tipo rango, que permiten buscar documentos por rangos de valores donde se define un valor mínimo y un valor máximo para determinar el conjunto de valores que retornar.

Lucene tiene mayor número de implementaciones para la generación de consultas. Además de las ya descritas para MG4J tiene también consultas por términos, consultas por prefijos y consultas con posibilidad de expresiones regulares, con uso de símbolos tales com (* y ?).

1.6.4 Puntuación

Lucene utiliza por defecto el modelo de espacio vectorial que básicamente lo que hace es representar los documentos en lenguaje natural mediante vectores, de modo que cada

vez que se realiza una consulta, también se genera un vector y se mide la diferencia de ángulos entre la consulta y los documentos. Las desventajas de este método son que los grandes documentos no quedan bien representados, algunas partes de palabras pueden dar falsos positivos y documentos con la misma temática pero distinto vocabulario no serán asociados e implicarán falsos negativos.

MG4J sin embargo utiliza BM25, que es una función de bolsa de palabras que puntúa un conjunto de documentos basados en los términos que aparecen en una consulta en cada documento independientemente de la relación entre los términos con el documento.

Actualmente la comunidad que soporta Apache Lucene ha desarrollado nuevos elementos de puntuación entre ellos un BM25 para Lucene, pero no deja de ser un añadido al sistema y no parte propia de Lucene o al menos aún, por lo que no se va a tener en consideración.

Capítulo 2 Estudio práctico

2.1 Introducción

Para poder realizar un estudio práctico entre ambas tecnologías tendremos que realizar una aplicación donde podamos obtener distintos parámetros más o menos objetivos y más o menos medibles para poner de relieve lo mejor de cada tecnología.

2.2 Qué y cómo

Se han buscado dos catálogos sobre los que realizar indexaciones y búsquedas. La idea es poder comparar las tecnologías teniendo en cuenta un pequeño catálogo XML como puede ser la colección de obras de Shakespeare, y un segundo catálogo como es el Boletín Oficial de Aragón que desde mediados del año 2010 publica sus boletines oficiales no solo en html y pdf sino también en XML.

La idea será generar una aplicación capaz de indexar ambas colecciones a distintos niveles de sus respectivos XML, en concreto vamos a crear cuatro escenarios por cada catálogo, y finalmente la aplicación tiene que ser capaz de buscar en dichas índices y retornar los documentos que cumplen la consulta. Para todos estos procesos vamos a tomar distintas medidas, no todas ellas cuantificables, ya que también queremos comparar la dificultad que puede entrañar una u otra tecnología.

2.3 Parámetros a medir

La idea es generar una aplicación desde cero para lo que vamos a intentar medir los siguientes parámetros.

2.3.1 Parámetros medibles y por tanto objetivos

- Tamaño final de la aplicación dependiendo de la tecnología en uso. Resolución de dependencias.
- Tamaño de los índices generados según el catálogo y el escenario a usar
- Tiempo de generación de los índices según el catálogo y el escenario a usar
- Tiempo de búsqueda según el catálogo y el escenario a usar
- Número de ficheros de cada tecnología en la indexación

2.3.2 Parámetros no cuantificables y por tanto subjetivos

- Dificultad de inserción de una tecnología en un proyecto desde cero
- Dificultad de aprendizaje entre una y otra
- Cantidad de documentación

Capítulo 3 Tecnologías a utilizar

3.1 Introducción

Necesitamos realizar una aplicación visual, que facilite la medición de los distintos parámetros que deseamos medir. Para ello se ha pensado en una aplicación Web de búsqueda. Para poder desarrollar esta aplicación se han tenido que utilizar distintas tecnologías además de MG4J y Lucene. A lo largo de este capítulo pasaremos a definirlas.

3.2 Tecnologías utilizadas

3.2.1 Tomcat 7

Servidor Web sobre el que va a correr nuestra aplicación. No es un servidor de aplicaciones como podría ser JBoss pero para la ejecución de servlets y JSP es más que suficiente para la ejecución de la aplicación.

3.2.2 Struts 2

Es un framework de código abierto para aplicaciones Web desarrolladas en Java. Está basado en una arquitectura MVC (modelo – vista – controlador). Este framework nos va a proporcionar la arquitectura básica para la aplicación web.

3.2.3 Castor

Es un framework open source para la vinculación de datos para mover información desde un XML a objetos Java o a Bases de datos. Para nuestra aplicación cumplirá la función de parsear los XML a objetos que internamente trataremos para ir generando los distintos índices.

3.2.4 Log4j

Log4j es un framework open source que nos permite formatear y establecer la salida del sistema. Sirve para manejar los logs de las apliaciones de un modo fácil, sencillo y transparente. Nos será de especial utilidad para la ejecución de los standalone que se encargan de lanzar la generación de índices.

3.2.5 Maven

Herramienta software para la gestión y construcción de proyectos Java. Es similar a a Apache Ant con la diferencia de que automáticamente resuelve todas las dependencias de las librerías automatizando su gestión y control. No es necesario tener las librerías en el proyecto sino que a través de Maven a la hora de generar la construcción este se encargará de descargar las versiones correspondientes de cada librería necesaria y generar el artefacto final, por ejemplo un .WAR.

Esta herramienta la utilizaremos para la construcción y despliegue del artefacto final, así como para facilitar futuros despliegues en otros entornos.

3.2.6 MG4J y Lucene

Obviamente para poder generar índices y realizar búsquedas a través de estas tecnologías deberemos añadirlas a nuestra aplicación.

Capítulo 4 Análisis de la aplicación

4.1 Introducción

Para la realización del estudio hemos decidido realizar una pequeña aplicación que integre ambas tecnologías, de modo que debemos realizar un análisis previo del sistema, para ello en las siguientes secciones vamos a ir describiendo las tareas que el sistema debe realizar, operaciones que no debe realizar y la organización en diversos grupos funcionales de tareas semejantes.

4.2 Catálogo de requisitos y casos de uso

En esta sección trataremos de escribir los distintos requisitos que nos hemos auto impuesto para la generación de la aplicación.

4.2.1 Catálogo de requisitos de usuario

La aplicación no posee unos requisitos específicos como tal ya que simplemente es una herramienta para la medición y comparación de dos tecnologías, por lo que la idea consiste en generar una aplicación sobre la que un usuario final pueda, indexar o buscar según el catalogo que desee y la profundidad establecida que prefiera.

El sistema deberá mostrar al usuario los tiempos que ha requerido la indexación o búsqueda, de modo que se pueda evaluar las diferencias entre unas tecnologías y otras.

4.2.2 Objetivos del sistema

4.2.2.1 Introducción

Los objetivos del sistema son los siguientes:

- **Indexación del catálogo de Shakespeare**
- **Búsqueda sobre el catálogo de Shakespeare**
- **Indexación del catálogo del BOA**
- **Búsqueda sobre el catálogo del BOA**

4.2.2.2 Catálogo de Objetivos del sistema

OBJ-01	Indexación catálogo Shakespeare
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Descripción	El sistema debe indexar tanto en MG4J como con Apache Lucene el catálogo de ficheros XML de las obras de Shakespeare en los cuatro niveles de profundidad establecidos: 1. Nivel Obra 2. Nivel Acto 3. Nivel Escena 4. Nivel Discurso
Comentarios	ninguno

Tabla Requisitos 4-1 - OBJ-01 – Indexación catálogo Shakespeare

OBJ-02	Indexación catálogo BOA
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Descripción	El sistema debe indexar tanto en MG4J como con Apache Lucene el catálogo de ficheros XML de las obras de BOA (Boletín Oficial de Aragón) en los cuatro niveles de profundidad establecidos: 1. Nivel Documento 2. Nivel Sección 3. Nivel Subsección 4. Nivel Detalle de texto
Comentarios	ninguno

Tabla Requisitos 4-2 - OBJ-02 – Indexación catálogo BOA

OBJ-03	Búsquedas booleanas sobre el catálogo de Shakespeare
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Descripción	El sistema debe realizar búsquedas sobre los índices generados en el OBJ-01 en ambas tecnologías y retornar el conjunto de valores que se corresponden con la consulta, indicando claramente el tiempo que ha tardado

	y el número de documentos obtenidos
Comentarios	ninguno

Tabla Requisitos 4-3 - OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakesperae

OBJ-04	**Búsquedas booleanas sobre el catálogo del BOA**
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Descripción	El sistema debe realizar búsquedas sobre los índices generados en el OBJ-03 en ambas tecnologías y retornar el conjunto de valores que se corresponden con la consulta, indicando claramente el tiempo que ha tardado y el número de documentos obtenidos
Comentarios	ninguno

Tabla Requisitos 4-4 - OBJ-04 – Búsquedas booleanas sobre el catálogo del BOA

Nuestro sistema queda por tanto delimitado tal y como se muestra en el diagrama inferior.

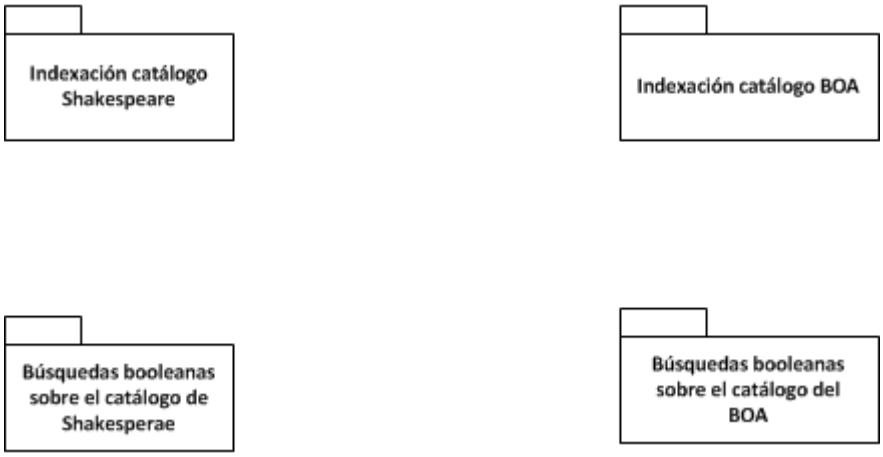

Diagrama 4-1 - Límites del Sistema

4.2.3 Identificación de los usuarios o Roles

4.2.3.1 Introducción

Primeramente debemos describir los perfiles de usuario necesarios en el sistema. Al ser un sistema tan sencillito tan sólo necesitamos un Rol para la aplicación:

- **Usuario -** Actor capaz de generar índices o realizar búsquedas

4.2.3.2 Actores del Sistema

ACT-01	Usuario
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Descripción	Este actor representa al usuario tipo de la aplicación que tiene que ser capaz de generar índices o realizar búsquedas sobre los catálogos
Comentarios	ninguno

Tabla Requisitos 4-5 - ACT-01 – Usuario

4.2.4 Catálogo de requisitos funcionales

El catálogo de requisitos funcionales viene delimitado por los subsistemas explicados en los objetivos del sistema. Por tanto vamos a ir subdividiendo este capítulo en los distintos subsistemas y grupos funcionales.

4.2.4.1 Indexación catálogo Shakespeare

Este grupo funcional será el encargado de generar los distintos índices del catálogo de Shakespeare para cada una de las tecnologías.

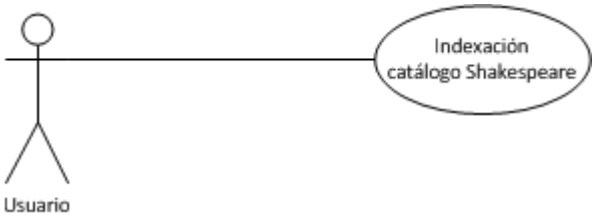

Diagrama 4-2 - Grupo Funcional Indexación Shakespeare

RF-000	Indexación Shakespeare
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Requisitos asociados	RF-001 – Indexación Nivel Obra Lucene RF-002 – Indexación Nivel Acto Lucene RF-003 – Indexación Nivel Escena Lucene RF-004 – Indexación Nivel Discurso Lucene RF-005 – Indexación Nivel Obra MG4J RF-006 – Indexación Nivel Acto MG4J RF-007 – Indexación Nivel Escena MG4J RF-008 – Indexación Nivel Discurso MG4J

Descripción	El sistema deberá generar índices a distintos niveles y con distintas tecnologías
Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Este Caso de uso es la Generalización de cualquiera de las indexaciones descritas en los casos de uso asociados.

Tabla Requisitos 4-6 - RF-000 – Indexación Shakespeare

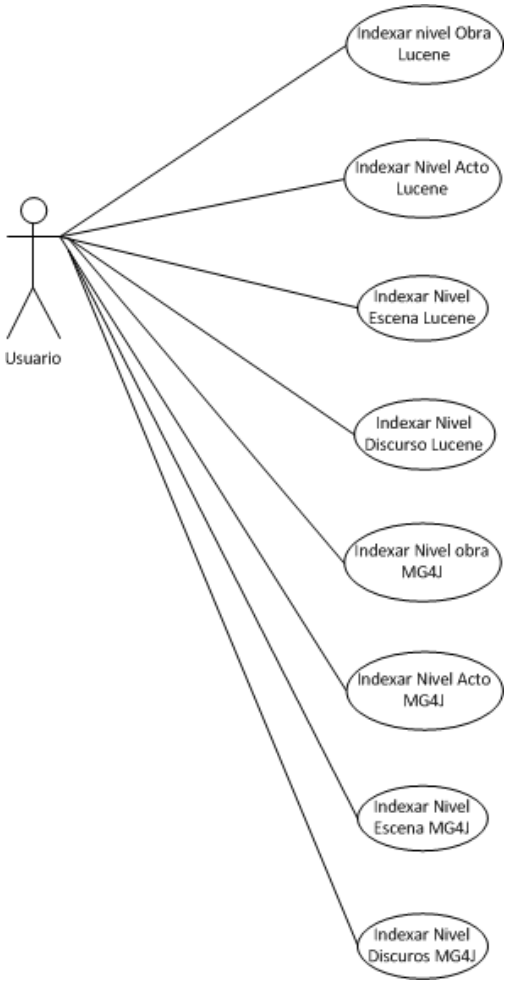

Diagrama 4-3 – Desglose Grupo Funcional Indexación Shakespeare

4.2.4.1.1 Indexación Shakespeare Lucene

Se trata de generar los índices necesarios en Lucene para lograr indexar los ficheros XML del catálogo de Shakespeare.

RF-001	**Indexación Nivel Obra Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Obra con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en tres campos. • RUTA – Fichero Xml al que pertenece • TITLE – Titulo de la obra • DOCUMENTO – Toda la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / Lucene / NIVELOBRA el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución

Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Ninguno

Tabla Requisitos 4-7 - RF-001 – Indexación nivel obra Lucene

RF-002	**Indexación Nivel Acto Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Acto con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • TITLE – Titulo de la Obra • SCNDESCR – Descripción de la obra • PLAYSUBT • ENTRADILLA – Entradilla de la obra • PERSONAE_TITLE – Titulo del conjunto de personajes • PERSONA – Personaje de la obra • INTRODUCCIÓN – Introducción de la obra • PROLOGO – Prologo de la obra • ACTO_TITULO – Titulo del acto • ACTO – Acto de la obra • EPILOGO – Epilogo de la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de

		un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / Lucene / NIVELACTO el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-8 - RF-002 – Indexación nivel Acto Lucene

RF-003	Indexación Nivel Escena Lucene
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare
Requisitos asociados	Ninguno
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Escena con Lucene.
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en quince campos. • RUTA – Fichero XML al que pertenece • TITLE – Titulo de la Obra • SCNDESCR – Descripción de la obra • PLAYSUBT • ENTRADILLA – Entradilla de la obra • PERSONAE_TITLE – Titulo del conjunto de personajes • PERSONA – Personaje de la obra • INTRODUCCIÓN – Introducción de la obra • PROLOGO – Prologo de la obra • ACTO_TITULO – Titulo del acto • ACTO_PROLOGO – Prologo del acto

	• ACTO_EPILOGO – Epilogo del acto • ESCENA_TITULO – Titulo de la escena • ESCENA – escena de la obra • EPILOGO – Epilogo de la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / Lucene / NIVELESCENA el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-9 - RF-003 – Indexación nivel Escena Lucene

RF-004	Indexación Nivel Discurso Lucene
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare
Requisitos asociados	Ninguno
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Discurso con Lucene.

<table>
<tr><td>Precondición</td><td colspan="2">El usuario no necesita ningún tipo de privilegio para realizar esta operación.
El sistema va a dividir los documentos XML en quince campos.
• RUTA – Fichero XML al que pertenece
• TITLE – Titulo de la Obra
• SCNDESCR – Descripción de la obra
• PLAYSUBT
• ENTRADILLA – Entradilla de la obra
• PERSONAE_TITLE – Titulo del conjunto de personajes
• PERSONA – Personaje de la obra
• INTRODUCCIÓN – Introducción de la obra
• PROLOGO – Prologo de la obra
• ACTO_TITULO – Titulo del acto
• ACTO_PROLOGO – Prologo del acto
• ACTO_EPILOGO – Epilogo del acto
• ESCENA_TITULO – Titulo de la escena
• DISCURSO – Discurso de la obra
• ORADOR – Orador que da el discurso
• EPILOGO – Epilogo de la obra</td></tr>
<tr><td rowspan="6">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario lanzará un standalone que se encarga de realizar la indexación</td></tr>
<tr><td>2</td><td>El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación</td></tr>
<tr><td>3</td><td>El sistema parsea los XML del catálogo convirtiéndolos en objetos</td></tr>
<tr><td>4</td><td>El sistema genera el índice de Lucene con los campos indicados en la precondición</td></tr>
<tr><td>5</td><td>Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación</td></tr>
<tr><td>Postcondición</td><td colspan="2">El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / Lucene / NIVELSPEECH el tamaño del índice generado.</td></tr>
<tr><td rowspan="3">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>5</td><td>El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución</td></tr>
<tr><td>5</td><td>Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución</td></tr>
<tr><td>Importancia</td><td colspan="2">Muy Alta</td></tr>
<tr><td>Urgencia</td><td colspan="2">Urgente</td></tr>
<tr><td>Estado</td><td colspan="2">Finalizado</td></tr>
</table>

Estabilidad	Alta
Comentarios	Ninguno

Tabla Requisitos 4-10 - RF-004 – Indexación Nivel Discurso Lucene

4.2.4.1.2 Indexación Shakespeare MG4J

Se trata de generar los índices necesarios en MG4J para lograr indexar los ficheros XML del catálogo de Shakespeare.

RF-005	**Indexación Nivel Obra MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Obra con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en tres campos. • RUTA – Fichero Xml al que pertenece • TITLE – Titulo de la obra • DOCUMENTO – Toda la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / MG4J / NIVELOBRA el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra

		la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-11 - RF-005 – Indexación nivel obra MG4J

RF-006	**Indexación Nivel Acto MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Acto con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • TITLE – Titulo de la Obra • SCNDESCR – Descripción de la obra • PLAYSUBT • ENTRADILLA – Entradilla de la obra • PERSONAE_TITLE – Titulo del conjunto de personajes • PERSONA – Personaje de la obra • INTRODUCCIÓN – Introducción de la obra • PROLOGO – Prologo de la obra • ACTO_TITULO – Titulo del acto • ACTO – Acto de la obra Epilogo – Epilogo de la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo

		convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / MG4J / NIVELACTO el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-12 - RF-006 – Indexación nivel Acto MG4J

RF-007	Indexación Nivel Escena MG4J
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare
Requisitos asociados	Ninguno
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Escena con MG4J.
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en quince campos. • RUTA – Fichero XML al que pertenece • TITLE – Titulo de la Obra • SCNDESCR – Descripción de la obra • PLAYSUBT • ENTRADILLA – Entradilla de la obra • PERSONAE_TITLE – Titulo del conjunto de personajes • PERSONA – Personaje de la obra

	• INTRODUCCIÓN – Introducción de la obra • PROLOGO – Prologo de la obra • ACTO_TITULO – Titulo del acto • ACTO_PROLOGO – Prologo del acto • ACTO_EPILOGO – Epilogo del acto • ESCENA_TITULO – Titulo de la escena • ESCENA – escena de la obra • EPILOGO – Epilogo de la obra	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / MG4J / NIVELESCENA el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-13 - RF-007 – Indexación nivel Escena MG4J

RF-008	Indexación Nivel Discurso MG4J
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare
Requisitos asociados	Ninguno

<table>
<tr><td>Descripción</td><td colspan="2">El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Discurso con MG4J.</td></tr>
<tr><td>Precondición</td><td colspan="2">El usuario no necesita ningún tipo de privilegio para realizar esta operación.
El sistema va a dividir los documentos XML en quince campos.
• RUTA – Fichero XML al que pertenece
• TITLE – Titulo de la Obra
• SCNDESCR – Descripción de la obra
• PLAYSUBT
• ENTRADILLA – Entradilla de la obra
• PERSONAE_TITLE – Titulo del conjunto de personajes
• PERSONA – Personaje de la obra
• INTRODUCCIÓN – Introducción de la obra
• PROLOGO – Prologo de la obra
• ACTO_TITULO – Titulo del acto
• ACTO_PROLOGO – Prologo del acto
• ACTO_EPILOGO – Epilogo del acto
• ESCENA_TITULO – Titulo de la escena
• DISCURSO – Discurso de la obra
• ORADOR – Orador que da el discurso
• EPILOGO – Epilogo de la obra</td></tr>
<tr><td rowspan="6">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario lanzará un standalone que se encarga de realizar la indexación</td></tr>
<tr><td>2</td><td>El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación</td></tr>
<tr><td>3</td><td>El sistema parsea los XML del catálogo convirtiéndolos en objetos</td></tr>
<tr><td>4</td><td>El sistema genera el índice de MG4J con los campos indicados en la precondición</td></tr>
<tr><td>5</td><td>Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación</td></tr>
<tr><td>Postcondición</td><td colspan="2">El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas Shakespeare / MG4J / NIVELSPEECH el tamaño del índice generado.</td></tr>
<tr><td rowspan="3">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>5</td><td>El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución</td></tr>
<tr><td>5</td><td>Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la</td></tr>
</table>

	ejecución
Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Ninguno

Tabla Requisitos 4-14 - RF-008 – Indexación Nivel Discurso MG4J

4.2.4.2 Indexación catálogo Boletín Oficial Aragón

Este grupo funcional será el encargado de generar los distintos índices del catálogo de BOA (Boletín Oficial de Aragón) para cada una de las tecnologías.

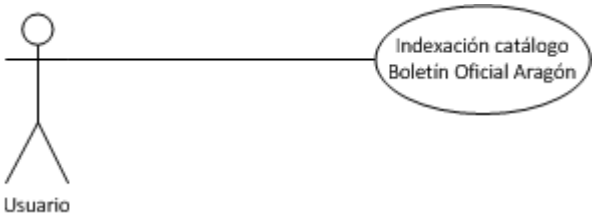

Diagrama 4-4 - Grupo Funcional Indexación Boletín Oficial Aragón

RF-009	Indexación BOA
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Requisitos asociados	RF-010 – Indexación Nivel Boletín Lucene RF-011 – Indexación Nivel Sección Lucene RF-012 – Indexación Nivel Subsección Lucene RF-013 – Indexación Nivel Detalle Lucene RF-014 – Indexación Nivel Boletín MG4J RF-015 – Indexación Nivel Sección MG4J RF-016 – Indexación Nivel Subsección MG4J RF-017 – Indexación Nivel Detalle MG4J
Descripción	El sistema deberá generar índices a distintos niveles y con distintas tecnologías
Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Este Caso de uso es la Generalización de cualquiera de las indexaciones descritas en los casos de uso asociados.

Tabla Requisitos 4-15 - RF-009 – Indexación BOA

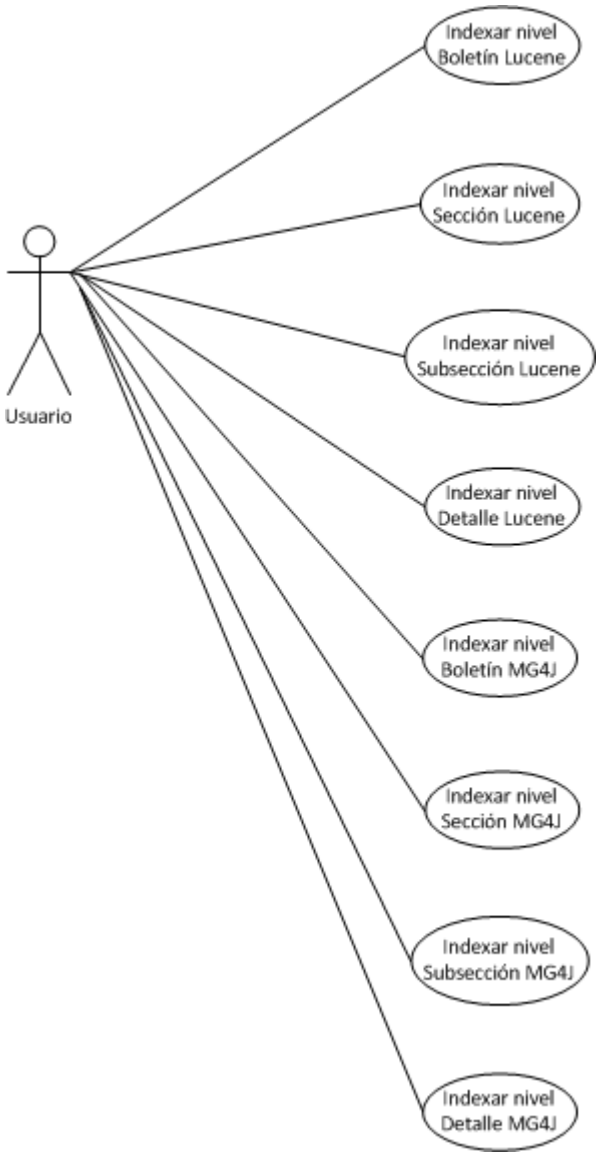

Diagrama 4-5 – Desglose Grupo Funcional Indexación BOA

4.2.4.2.1 Indexación Lucene

Se trata de generar los índices necesarios en Lucene para lograr indexar los ficheros XML del catálogo de BOA.

RF-010	Indexación Nivel Boletín Lucene
Versión	1.0 (18/08/2013)

Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-02 – Indexación catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Boletín con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en tres campos. • RUTA – Fichero Xml al que pertenece • FECHA – Fecha del boletín • DOCUMENTO – Toda el boletín	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / Lucene / NIVELBOLETIN el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-16 - RF-010 – Indexación nivel boletín Lucene

<table>
<tr><th>RF-011</th><th colspan="2">Indexación Nivel Sección Lucene</th></tr>
<tr><td>Versión</td><td colspan="2">1.0 (18/08/2013)</td></tr>
<tr><td>Autores</td><td colspan="2">Mª Carmen Loriente Yáñez</td></tr>
<tr><td>Objetivos asociados</td><td colspan="2">OBJ-02 – Indexación catálogo BOA</td></tr>
<tr><td>Requisitos asociados</td><td colspan="2">Ninguno</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Sección con Lucene.</td></tr>
<tr><td>Precondición</td><td colspan="2">El usuario no necesita ningún tipo de privilegio para realizar esta operación.
El sistema va a dividir los documentos XML en doce campos.
• RUTA – Fichero XML al que pertenece
• FECHA – Fecha del boletín
• ANYO – Año del boletín
• INDICE – Indice del boletín
• NUMERO – Número del boletín
• TITULO_SUMARIO – Titulo del sumario
• NEWPAGE
• SECCION – sección</td></tr>
<tr><td rowspan="6">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario lanzará un standalone que se encarga de realizar la indexación</td></tr>
<tr><td>2</td><td>El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación</td></tr>
<tr><td>3</td><td>El sistema parsea los XML del catálogo convirtiéndolos en objetos</td></tr>
<tr><td>4</td><td>El sistema genera el índice de Lucene con los campos indicados en la precondición</td></tr>
<tr><td>5</td><td>Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación</td></tr>
<tr><td>Postcondición</td><td colspan="2">El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / Lucene / NIVELSECCION el tamaño del índice generado.</td></tr>
<tr><td rowspan="3">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>5</td><td>El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución</td></tr>
<tr><td>5</td><td>Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución</td></tr>
</table>

Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Ninguno

Tabla Requisitos 4-17 - RF-011 – Indexación nivel sección Lucene

RF-012	**Indexación Nivel Subsección Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-02 – Indexación catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Subsección con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • FECHA – Fecha del boletín • ANYO – Año del boletín • INDICE – Indice del boletín • NUMERO – Número del boletín • TITULO_SUMARIO – Titulo del sumario • NEWPAGE • TITULO_SECCION – Titulo de la sección • CODIGO_SECCION – Código de la sección • CODIGOS_EMISOR – Conjutno de campos emisores • SUBSECCIÓN – Subsección del boletín	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el

		número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / Lucene / NIVELSUBSECCION el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-18 - RF-012 – Indexación nivel Subsección Lucene

RF-013	**Indexación Nivel Detalle Lucene**
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-02 – Indexación catálogo BOA
Requisitos asociados	Ninguno
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Detalle con Lucene.
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • FECHA – Fecha del boletín • ANYO – Año del boletín • INDICE – Indice del boletín • NUMERO – Número del boletín • TITULO_SUMARIO – Titulo del sumario • NEWPAGE • TITULO_SECCION – Titulo de la sección • CODIGO_SECCION – Código de la sección • TITULO_SUBSECCIÓN – Titulo de la subsección • CODIGO_SUBSECCION – Código de la subsección • EMISOR – Emisor de la disposición

	• TXT_EMISOR – Texto del emisor • FECHA_EMISOR – Fecha del emisor • CLAVE_DISPOSICION – Clave de disposición • TITULO_TEXTO – Título del texto • TEXTO – texto con la disposición	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de Lucene con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / Lucene / NIVELDETALLE el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-19 - RF-013 – Indexación Nivel Detalle Lucene

4.2.4.2.2 Indexación MG4J

Se trata de generar los índices necesarios en MG4J para lograr indexar los ficheros XML del catálogo de BOA.

RF-014	Indexación Nivel Boletín MG4J
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-02 – Indexación catálogo BOA

Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Boletín con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en tres campos. • RUTA – Fichero Xml al que pertenece • FECHA – Fecha del boletín • DOCUMENTO – Toda el boletín	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / MG4J / NIVELBOLETIN el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-20 - RF-014 – Indexación nivel boletín MG4J

RF-015	**Indexación Nivel Sección MG4J**
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez

<table>
<tr><td>Objetivos asociados</td><td colspan="2">OBJ-02 – Indexación catálogo BOA</td></tr>
<tr><td>Requisitos asociados</td><td colspan="2">Ninguno</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Sección con MG4J.</td></tr>
<tr><td>Precondición</td><td colspan="2">El usuario no necesita ningún tipo de privilegio para realizar esta operación.
El sistema va a dividir los documentos XML en doce campos.
• RUTA – Fichero XML al que pertenece
• FECHA – Fecha del boletín
• ANYO – Año del boletín
• INDICE – Indice del boletín
• NUMERO – Número del boletín
• TITULO_SUMARIO – Titulo del sumario
• NEWPAGE
• SECCION – sección</td></tr>
<tr><td rowspan="6">Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>El usuario lanzará un standalone que se encarga de realizar la indexación</td></tr>
<tr><td>2</td><td>El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación</td></tr>
<tr><td>3</td><td>El sistema parsea los XML del catálogo convirtiéndolos en objetos</td></tr>
<tr><td>4</td><td>El sistema genera el índice de MG4J con los campos indicados en la precondición</td></tr>
<tr><td>5</td><td>Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación</td></tr>
<tr><td>Postcondición</td><td colspan="2">El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / MG4J / NIVELSECCION el tamaño del índice generado.</td></tr>
<tr><td rowspan="3">Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td>5</td><td>El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución</td></tr>
<tr><td>5</td><td>Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución</td></tr>
<tr><td>Importancia</td><td colspan="2">Muy Alta</td></tr>
<tr><td>Urgencia</td><td colspan="2">Urgente</td></tr>
<tr><td>Estado</td><td colspan="2">Finalizado</td></tr>
<tr><td>Estabilidad</td><td colspan="2">Alta</td></tr>
<tr><td>Comentarios</td><td colspan="2">Ninguno</td></tr>
</table>

Tabla Requisitos 4-21 - RF-015 – Indexación nivel Sección MG4J

RF-016	**Indexación Nivel Subsección MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-02 – Indexación catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Subsección con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • FECHA – Fecha del boletín • ANYO – Año del boletín • INDICE – Indice del boletín • NUMERO – Número del boletín • TITULO_SUMARIO – Titulo del sumario • NEWPAGE • TITULO_SECCION – Titulo de la sección • CODIGO_SECCION – Código de la sección • CODIGOS_EMISOR – Conjutno de campos emisores • SUBSECCIÓN – Subsección del boletín	
Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / MG4J / NIVELSUBSECCION el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**

	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-22 - RF-016 – Indexación nivel Subsección MG4J

RF-017	Indexación Nivel Detalle MG4J
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-02 – Indexación catálogo BOA
Requisitos asociados	Ninguno
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la indexación a nivel Detalle con MG4J.
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación. El sistema va a dividir los documentos XML en doce campos. • RUTA – Fichero XML al que pertenece • FECHA – Fecha del boletín • ANYO – Año del boletín • INDICE – Indice del boletín • NUMERO – Número del boletín • TITULO_SUMARIO – Titulo del sumario • NEWPAGE • TITULO_SECCION – Titulo de la sección • CODIGO_SECCION – Código de la sección • TITULO_SUBSECCIÓN – Titulo de la subsección • CODIGO_SUBSECCION – Código de la subsección • EMISOR – Emisor de la disposición • TXT_EMISOR – Texto del emisor • FECHA_EMISOR – Fecha del emisor • CLAVE_DISPOSICION – Clave de disposición • TITULO_TEXTO – Título del texto • TEXTO – texto con la disposición

Secuencia normal	**Paso**	**Acción**
	1	El usuario lanzará un standalone que se encarga de realizar la indexación
	2	El sistema recupera los datos básicos de indexación del fichero de propiedades de la aplicación
	3	El sistema parsea los XML del catálogo convirtiéndolos en objetos
	4	El sistema genera el índice de MG4J con los campos indicados en la precondición
	5	Cuando el sistema finaliza nos indica a través de un log el tiempo transcurrido para la indexación, el número de ficheros indexados, y la ubicación
Postcondición	El usuario podrá visualizar en una ruta específica del sistema bajo las carpetas BOA / MG4J / NIVELDETALLE el tamaño del índice generado.	
Excepciones	**Paso**	**Acción**
	5	El sistema no encuentra alguno de los datos necesarios para realizar la indexación. Se muestra la excepción en el Log y se termina la ejecución
	5	Alguno de los ficheros a indexar no es correcto. Se muestra la excepción en el log y se termina la ejecución
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-23 - RF-017 – Indexación Nivel Detalle MG4J

4.2.4.3 Búsqueda catálogo Shakespeare

Este grupo funcional será el encargado de generar los distintos escenarios para poder realizar las búsquedas a través de los índices generados para el catálogo de Shakespeare con las dos tecnologías estudiadas.

Diagrama 4-6 - Grupo Funcional Búsqueda Shakespeare

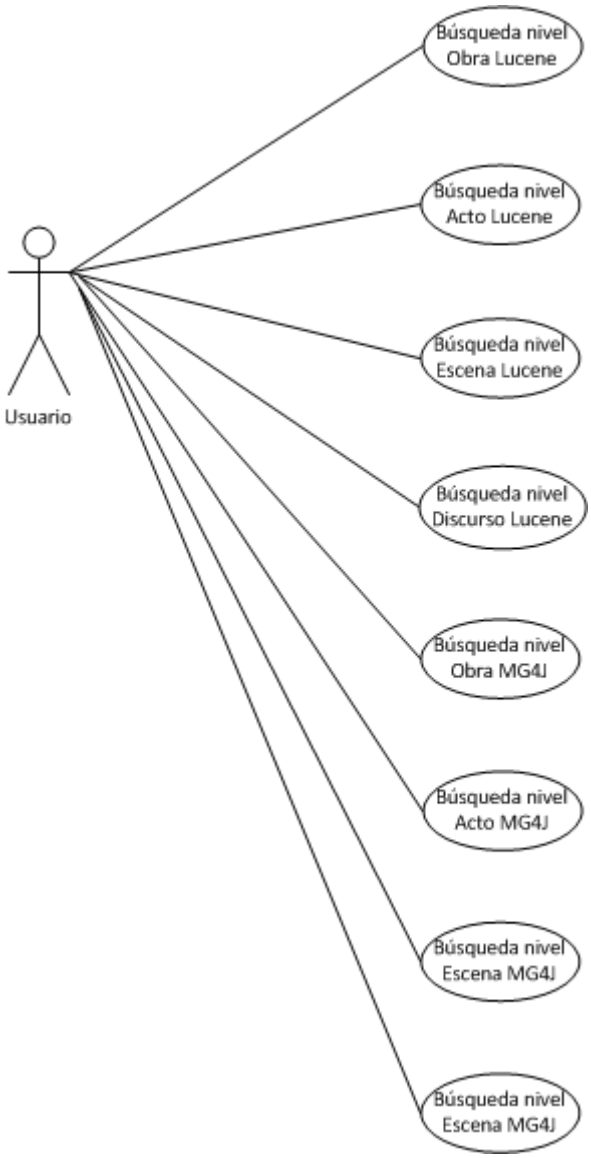

Diagrama 4-7 – Desglose Grupo Funcional Búsqueda Shakespeare

RF-018	Búsqueda Shakespeare
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Requisitos asociados	RF-019 – Búsqueda Nivel Obra Lucene RF-020 – Búsqueda Nivel Acto Lucene

	RF-021 – Búsqueda Nivel Escena Lucene RF-022 – Búsqueda Nivel Discurso Lucene RF-023 – Búsqueda Nivel Obra MG4J RF-024 – Búsqueda Nivel Acto MG4J RF-025 – Búsqueda Nivel Escena MG4J RF-026 – Búsqueda Nivel Discurso MG4J
Descripción	El sistema deberá retornar el conjunto de documentos que cumplen con la consulta y el tiempo que ha tardado en realizar la búsqueda
Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Este Caso de uso es la Generalización de cualquiera de las búsquedas descritas en los casos de uso asociados.

Tabla Requisitos 4-24 - RF-018 – Búsqueda Shakespeare

4.2.4.3.1 Búsqueda Shakespeare Lucene

Se trata de realizar las búsquedas para el catálogo de las obras de Shakespeare, utilizando Lucene.

RF-019	**Búsqueda Nivel Obra Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Obra con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Obra.
	2	El sistema mostrará los siguientes filtros: Titulo, Documento, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE
	4	El sistema construirá la consulta según los filtros

		rellenos y realizará una búsqueda sobre el índice nivelObra de Lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-25 - RF-019 – Búsqueda nivel obra Lucene

RF-020	**Búsqueda Nivel Acto Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Acto con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Acto.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Acto, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice

		nivelActo de Lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-26 - RF-020 – Búsqueda nivel Acto Lucene

RF-021	**Búsqueda Nivel Escena Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Escena con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Escena.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Prologo Acto, Epilogo Acto, Titulo Escena, Escena, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice

		nivelEscena de Lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-27 - RF-021 – Búsqueda nivel Escena Lucene

RF-022	**Búsqueda Nivel Discurso Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Discurso con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Discurso.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Prologo Acto, Epilogo Acto, Titulo Escena, Orador, Discurso, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice nivelSpeech de Lucene.

	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-28 - RF-022 – Búsqueda Nivel Discurso Lucene

4.2.4.3.2 Búsqueda Shakespeare MG4J

Se trata de realizar las búsquedas para el catálogo de las obras de Shakespeare, utilizando MG4J.

RF-023	**Búsqueda Nivel Obra MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Obra con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Obra.
	2	El sistema mostrará los siguientes filtros: Titulo, Documento, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros

		rellenos y realizará una búsqueda sobre los índices nivelObra de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-29 - RF-023 – Búsqueda nivel obra MG4J

RF-024	**Búsqueda Nivel Acto MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Acto con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Acto.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Acto, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivelActo de MG4J.

	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-30 - RF-024 – Búsqueda nivel Acto MG4J

RF-025	**Búsqueda Nivel Escena MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Escena con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Escena.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Prologo Acto, Epilogo Acto, Titulo Escena, Escena, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivelEscena de MG4J.
	5	Cuando el sistema finaliza nos devolverá el

		número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-31 - RF-025 – Búsqueda nivel Escena MG4J

RF-026	**Búsqueda Nivel Discurso MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Obra con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Discurso.
	2	El sistema mostrará los siguientes filtros: Titulo, Prologo, Epilogo, Introducción, Titulo Acto, Prologo Acto, Epilogo Acto, Titulo Escena, Orador, Discurso, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivelSpeech de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el

		tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-32 - RF-026 – Búsqueda Nivel Discurso MG4J

4.2.4.4 Búsqueda catálogo Boletín Oficial Aragón

Este grupo funcional será el encargado de generar los distintos índices del catálogo de BOA (Boletín Oficial de Aragón) para cada una de las tecnologías.

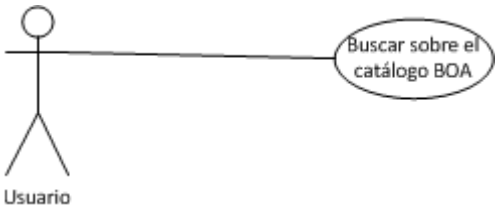

Diagrama 4-8 - Grupo Funcional Indexación Boletín Oficial Aragón

RF-027	Búsqueda Shakespeare
Versión	1.0 (18/08/2013)
Autores	Mª Carmen Loriente Yáñez
Requisitos asociados	RF-028 – Búsqueda Nivel Boletín Lucene RF-029 – Búsqueda Nivel Sección Lucene RF-030 – Búsqueda Nivel Subsección Lucene RF-031 – Búsqueda Nivel Detalle Lucene RF-032 – Búsqueda Nivel Boletín MG4J RF-033 – Búsqueda Nivel Sección MG4J RF-034 – Búsqueda Nivel Subsección MG4J RF-035 – Búsqueda Nivel Detalle MG4J
Descripción	El sistema deberá generar índices a distintos niveles y con distintas tecnologías

Importancia	Muy Alta
Urgencia	Urgente
Estado	Finalizado
Estabilidad	Alta
Comentarios	Este Caso de uso es la Generalización de cualquiera de las búsquedas descritas en los casos de uso asociados.

Tabla Requisitos 4-33 - RF-027 – Búsqueda BOA

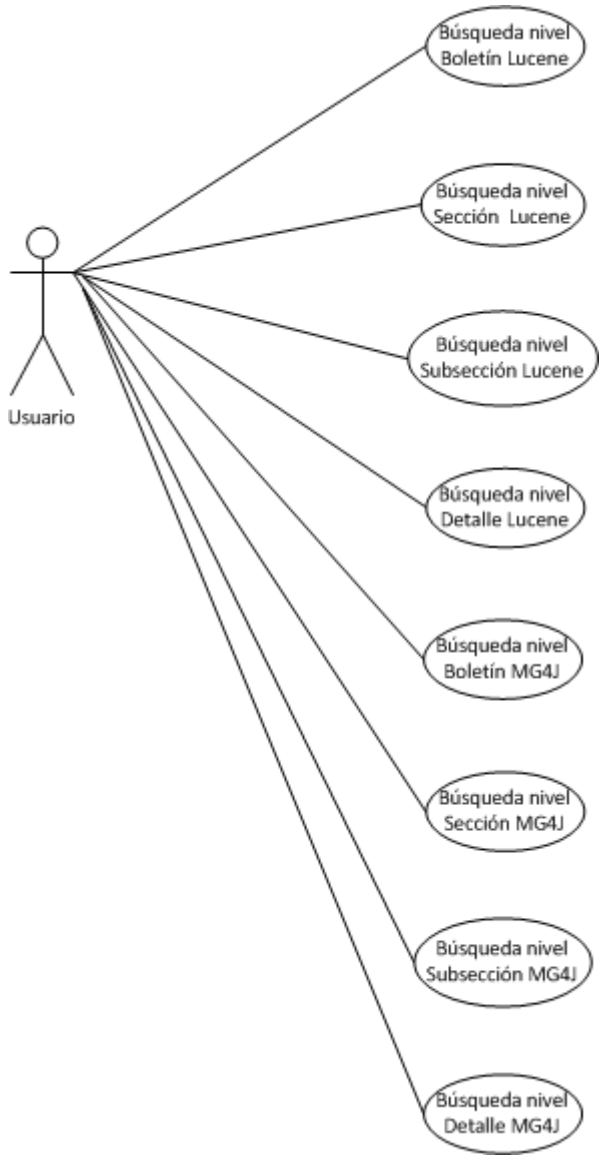

Diagrama 4-9 – Desglose Grupo Funcional Indexación BOA

4.2.4.4.1 Búsqueda BOA Lucene

Se trata de realizar las búsquedas para el catálogo del Boletín Oficial de Aragón, utilizando Lucene.

RF-028	**Búsqueda Nivel Boletín Lucene**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Boletín con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Boletín.
	2	El sistema mostrará los siguientes filtros: Fecha, Documento, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE.
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice nivelBoletin de lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-34 - RF-028 – Búsqueda nivel boletín Lucene

RF-029	Búsqueda Nivel Sección Lucene	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Sección con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Boletin.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Sección, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice nivelSección de lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-35 - RF-029 – Búsqueda nivel sección Lucene

RF-030	Búsqueda Nivel Subsección Lucene	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Subsección con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Subsección.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Titulo Sección, código sección, subsección, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsar el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice nivelSubsección de lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-36 - RF-030 – Búsqueda nivel Subsección Lucene

RF-031	Búsqueda Nivel Detalle Lucene	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Detalle con Lucene.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por detalle.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Titulo Sección, código sección, titulo subsección, código subsección, emisor, fecha emisor, clave disposición, titulo texto, texto, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsa el botón LUCENE
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre el índice Detalle de lucene.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-37 - RF-031 – Búsqueda Nivel Detalle Lucene

4.2.4.4.2 Búsqueda BOA MG4J

Se trata de realizar las búsquedas para el catálogo del Boletín Oficial de Aragón, utilizando Lucene.

RF-032	**Búsqueda Nivel Boletín MG4J**	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Obra con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Boletín.
	2	El sistema mostrará los siguientes filtros: Fecha, Documento, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivelBoletin de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-38 - RF-032 – Búsqueda nivel boletín MG4J

RF-033	Búsqueda Nivel Sección MG4J	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Sección con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por Sección.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Sección, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivelSección de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-39 - RF-033 – Búsqueda nivel Sección MG4J

RF-034	Búsqueda Nivel Subsección MG4J	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Subsección con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por subsección.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Titulo Sección, código sección, subsección, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivel Subsección de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-40 - RF-034 – Búsqueda nivel Subsección MG4J

RF-035	Búsqueda Nivel Detalle MG4J	
Versión	1.0 (18/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	Ninguno	
Descripción	El sistema deberá comportarse tal y como se describe en el siguiente caso de uso cuando se ejecute la búsqueda a nivel Detalle con MG4J.	
Precondición	El usuario no necesita ningún tipo de privilegio para realizar esta operación.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accederá a la pantalla búsqueda por subsección.
	2	El sistema mostrará los siguientes filtros: Año, Fecha, Indice, Numero, Titulo Sumario, Titulo Sección, código sección, titulo subsección, código subsección, emisor, fecha emisor, clave disposición, titulo texto, texto, Conjunción y los botones LUCENE y MG4J
	3	El usuario rellenará los filtros según los datos que desee buscar y pulsará el botón MG4J
	4	El sistema construirá la consulta según los filtros rellenos y realizará una búsqueda sobre los índices nivel Detalle de MG4J.
	5	Cuando el sistema finaliza nos devolverá el número de documentos totales que coinciden, el tiempo que ha tardado en encontrarles y un mapa con los documentos encontrados
Postcondición	El usuario podrá visualizar la información referente a cada documento encontrado.	
Excepciones	**Paso**	**Acción**
	4	El sistema no puede generar correctamente la consulta. Muestra una excepción en el Log y no retorna ningún resultado a la pantalla
	5	El sistema no ha encontrado coincidencias, nos muestra 0 resultados en el tiempo que le haya llevado
Importancia	Muy Alta	
Urgencia	Urgente	
Estado	Finalizado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla Requisitos 4-41 - RF-035 – Búsqueda Nivel Detalle MG4J

4.3 Diagramas de clases del sistema

4.3.1 Modelo de datos Shakespeare

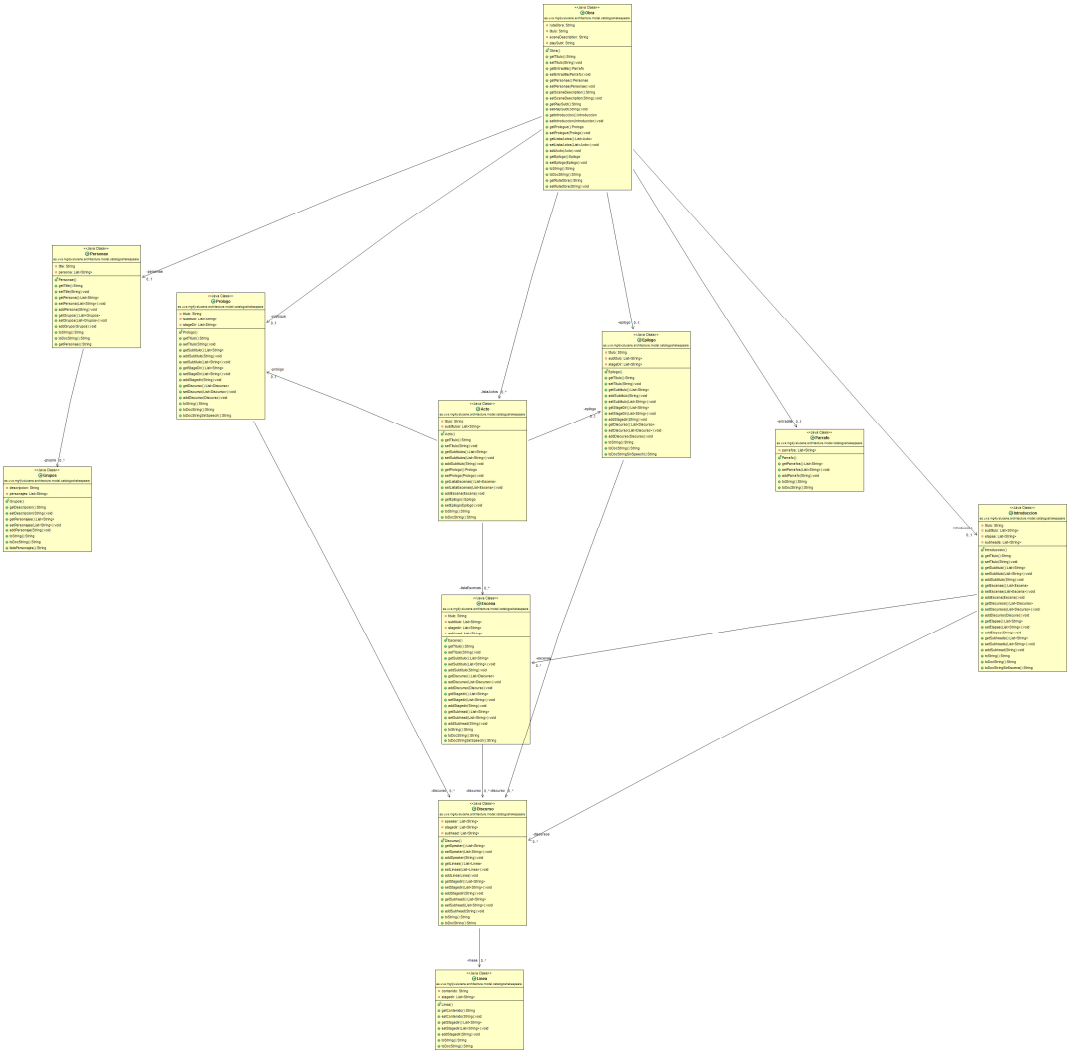

Ilustración 13 - Diagrama Modelo de datos catalogo Shakespeare

4.3.2 Modelo de datos BOA

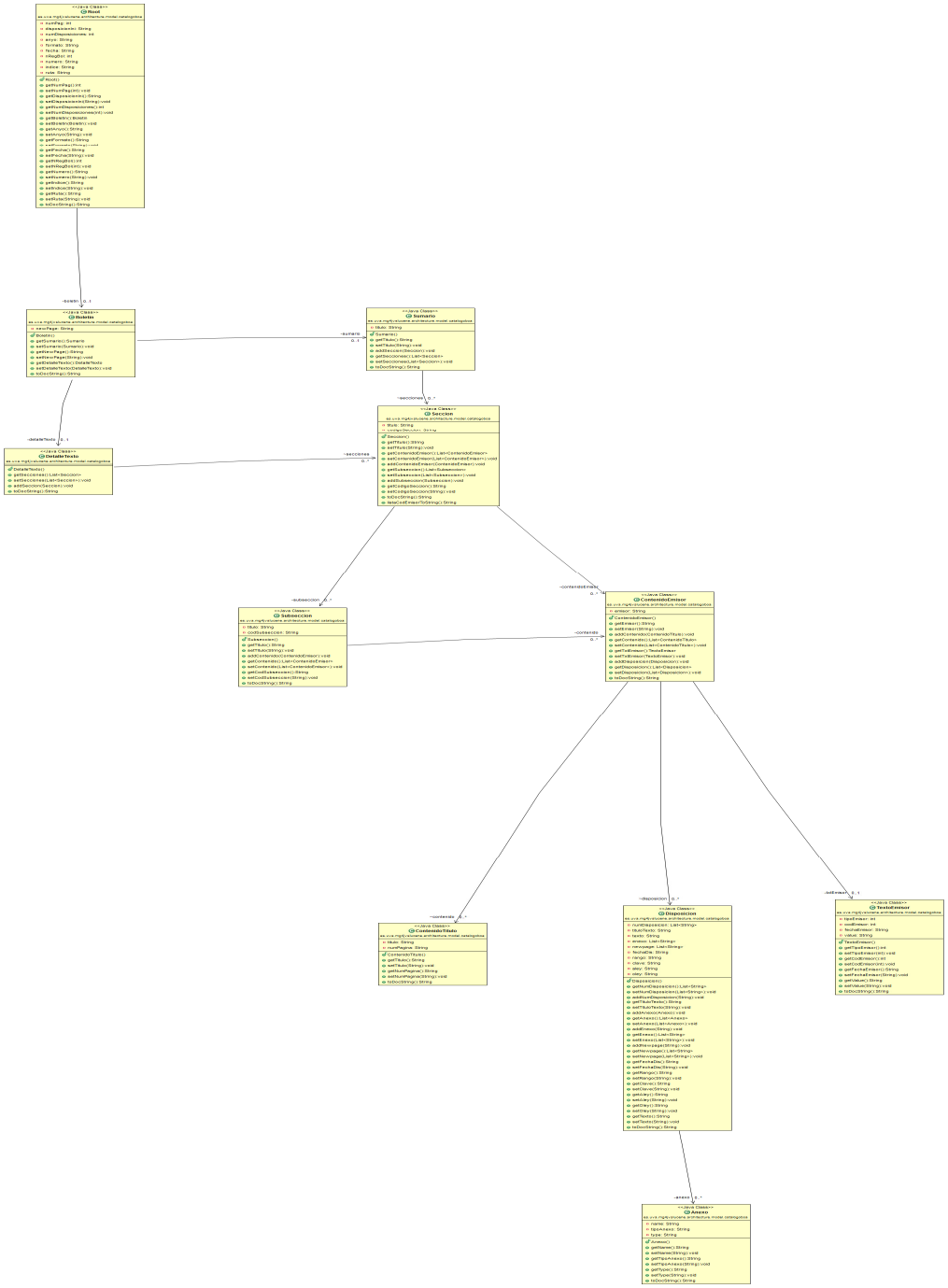

Ilustración 14 - Modelo de datos del catálogo BOA

Capítulo 5 Diseño e Implementación

5.1 Arquitectura del sistema

Aplicar Struts2 a la aplicación implica aplicar el patrón de modelo – vista - controlador a la aplicación.

El **modelo** serán los datos (o modelo de la aplicación) y todas las reglas de la lógica de negocio que operan sobre estos datos.

La **vista** es la encargada del interfaz con el que interacciona el usuarios. Este queda definido por los results que se definen en el xml de struts.

El **controlador** es el encargado de comunicar la vista y el modelo. Salta cada vez que un usuario interacciona con el interfaz y genera un evento con el que invocar cambios en el modelo.

El patrón modelo – vista - controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

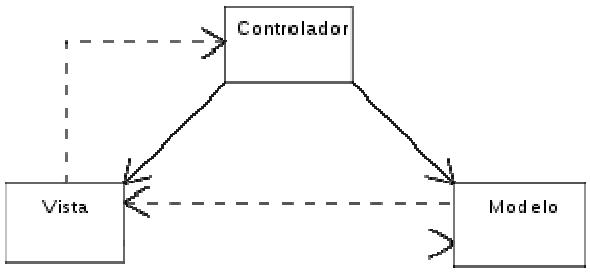

Ilustración 15- Modelo Vista Controlador

5.1.1 Arquitectura de la capa cliente

Siguiendo el archetype típico de Maven, la estructura para la capa cliente comienza bajo /src/main/webapp/ dentro podremos encontrar los siguientes elementos.

- **Css** – Contiene los archivos de estilos del portal
- **Images –** Contiene las imágenes del portal
- **Jsp –** Contiene una página de inicio para la aplicación y una subestructura por catálogo Boa y Shakespeare, que contiene cada subestructura los escenarios descritos anteriormente.
- **WEB-INF –** Aquí se encuentra el web.xml donde definimos los valores de configuración para la aplicación en el servidor de aplicaciones.

5.1.2 Arquitectura de la capa de servidor

La aplicación se organiza.

- **Arquitectura** - Aquí encontraremos todos los modelos y lógicas de negocio utilizadas a lo largo de la aplicación.
  - **Enums –** Clases enumeradas que contienen información sobre los catálogos, los niveles de indexación o los campos que tiene cada catálogo
  - **Ln -** Lógica de negocio, aquí es donde tenemos toda la lógica de negocio de la aplicación. Dónde podemos encontrar las estructuras comunes para realizar indexaciones o búsquedas según la tecnología e independientemente del modelo de datos.
  - **Modelo** – Aquí encontraremos organizados los distintos modelos de datos referentes a cada uno de los catálogos
  - **Script** – Abstracción de los scripts generales de la aplicación
  - **Clases varias** de lógica como las propiedades o las excepciones de la aplicación

- **Grupos Funcionales –** Conjunto de grupos funcionales implementados en la aplicación.
  - **GF01ShakespeareIndex –** Este grupo funcional tan solo contiene Scripts ya que no se interacciona de ningún modo con ninguna interfaz.
  - **GF02ShakespeareSearch -** Subdividido en la vista (view) dónde están todos los campos de pantalla referentes al catálogo Shakespeare y que lógicas implica cada uno. Y la lógica (script) que será independiente dependiendo de la tecnología.
  - **GF03BoaIndex –** Este grupo funcional tan solo contiene Scripts ya que no se interacciona de ningún modo con ninguna interfaz.
  - **GF04BoaSearch** – Para el catálogo de BOA, se ha subdividido en la vista (view) dónde están todos los campos de pantalla referentes al catálogo Shakespeare y que lógicas implica cada uno. Y la lógica (script) que será independiente dependiendo de la tecnología Abstracción de los scripts generales de la aplicación

5.2 Configuración de las distintas tecnologías

Para el desarrollo e implementación de la aplicación hay que configurar para su correcta convivencia las distintas tecnologías que lo conforman.

5.2.1 Log4j

Para insertar Log4J se ha añadido al pom de la aplicación la siguiente dependencia:

```
<dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
```

```
            <version>1.2.12</version>
    </dependency>
```

En la carpeta /src/main/resources/ del proyecto se genera el archivo log4j.xml donde configuramos los parámetros necesarios para la generación del log formateado.

5.2.1.1 Ejemplos

```
INFO (es.uva.mg4jvslucene.GF03BoaIndex.IndexBoaLuceneScript) [26 ago
2013 08:36:53] - [MG4JVSLUCENE] Tiempo de fin 1377499013501
   INFO (es.uva.mg4jvslucene.GF03BoaIndex.IndexBoaLuceneScript) [26 ago
2013 08:36:53] - [MG4JVSLUCENE] El tiempo indexando 32112 ficheros ha
llevado 14326 milisegundos
```

5.2.2 Struts 2

Para añadir la librería al proyecto al igual que con log4j se añade la siguiente dependencia:

```
<dependency>
      <groupId>org.apache.struts</groupId>
      <artifactId>struts2-core</artifactId>
      <version>2.3.1.2</version>
</dependency>
```

Ahora en el web.xml tenemos que añadir el filter Dispacher de Struts2 para que toda petición pase por struts.

```
<filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>
<filter-mapping>
```

En la carpeta /src/main/resources/ se genera el struts.xml que es el fichero que relaciona los accesos con las acciones a realizar y los resultados según la acción. Por ejemplo:

```
<!-- Busqueda por Obra -->
<action name="searchObra"
class="es.uva.mg4jvslucene.GF02ShakespeareSearch.view.ShakespeareSearchA
ction">
      <result name="input">jsp/Shakespeare/obra.jsp</result>
      <result name="error">jsp/Shakespeare/obra.jsp</result>
      <result>jsp/Shakespeare/obra.jsp</result>
</action>
```

5.2.1 Maven

Maven es la herramienta utilizada para la construcción del proyecto. Para su uso debe estar Maven instalado en la máquina con una jdk mínima 1.6. A mayores con el plugin m2eclipse que facilita la gestión de dependencias.

Para usar maven tan solo hay que generar un proyecto con su arquetipo base y luego ir añadiendo en el pom todas las dependencias necesarias.

5.2.2 Castor

Para utilizar Castor tan sólo será necesario añadir la dependencia al pom. No necesita ningún tipo de configuración.

```
<dependency>
	<groupId>org.codehaus.castor</groupId>
	<artifactId>castor-xml</artifactId>
	<version>1.3</version>
	<exclusions>
		<exclusion>
			<groupId>commons-logging</groupId>
			<artifactId>commons-logging</artifactId>
		</exclusion>
	</exclusions>
</dependency>
```

Los xml de mapeo se almacenan en /mg4jvslucene/src/main/resources/es/uva/mg4jvslucene/architecture/model

Estos serán necesarios a la hora de indexar.

Capítulo 6 Plan de Pruebas

6.1 Introducción

El plan de pruebas no es más que un conjunto de pruebas a realizar para comprobar la que la aplicación hace lo que tiene que hacer y no hace lo que no tiene que hacer.

Suelen ejecutarse tres veces, primero las ejecuta el propio desarrollador (pruebas unitarias), luego las debería ejecutar el responsable de proyecto (pruebas de integración) y finalmente (pruebas de aceptación) que son las que ejecuta el cliente final (si es que existe) para comprobar que el software funciona tal y como espera que funcione.

6.2 Grupo Funcional Indexación catálogo Shakespeare

CP-001	**Ejecución indexar todo Shakespeare**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-01 – Indexación catálogo Shakespeare	
Requisitos asociados	RF – 000 – Indexación Shakespeare	
Descripción	Se generan todos los índices para los cuatro niveles descritos para este catálogo en ambas tecnologías	
Entrada	Cadena Vacía	
Procedimiento	**Paso**	**Acción**
	1	Ejecutamos el fichero ubicado en /mg4jvslucene/src/test/java/es/uva/mg4jvslucene/indexar/shakespeare/IndexarTodoShakespeare.java
	2	Observamos la salida del log para comprobar que termina correctamente.
Salida Esperada	En la ruta establecida por configuración se ha tenido que crear la carpeta Shakespeare, dentro de esta una de mg4j y otra de lucene, y dentro de cada una de ellas las carpetas por cada nivel. Dentro deben estar los ficheros que componen los índices.	
Salida obtenida	Se ha generado la estructura de archivos necesaria y contiene los índices de cada nivel dentro de cada tecnología	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-1- CP - 001 – Ejecución indexar todo Shakespeare

6.3 Grupo Funcional Indexación catálogo BOA

CP-002	Ejecución indexar todo BOA	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-02 – Indexación catálogo BOA	
Requisitos asociados	RF – 009 – Indexación BOA	
Descripción	Se generan todos los índices para los cuatro niveles descritos para este catálogo en ambas tecnologías	
Entrada	Cadena Vacía	
Procedimiento	**Paso**	**Acción**
	1	Ejecutamos el fichero ubicado en /mg4jvslucene/src/test/java/es/uva/mg4jvslucene/indexar/boa/IndexarTodoBOA.java
	2	Observamos la salida del log para comprobar que termina correctamente.
Salida Esperada	En la ruta establecida por configuración se ha tenido que crear la carpeta BOA, dentro de esta una de mg4j y otra de Lucene, y dentro de cada una de ellas carpetas por cada nivel. Dentro deben estar los ficheros que componen los índices.	
Salida obtenida	Se ha generado la estructura de archivos necesaria y contiene los índices de cada nivel dentro de cada tecnología	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-2- CP - 002 – Ejecución indexar todo Boa

6.4 Grupo Funcional Búsqueda Shakespeare

CP-003	Búsqueda Obra
Versión	1.0 (25/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare
Requisitos asociados	RF-019 – Búsqueda Nivel Obra Lucene RF-023 – Búsqueda Nivel Obra MG4J
Descripción	Se debe realizar la búsqueda a nivel Obra en ambas tecnologías
Entrada	Rellenamos todos los campos

Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	1. Deben estar rellenos los campos una vez pulsemos buscar 2. Observar la consulta sobre el índice y observar que se utilizan todos los campos 3. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 4. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los cuatro casos de aceptación tanto para MG4J como para Lucene	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-3 - CP - 003 – Búsqueda obra

CP-004	**Búsqueda Acto**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	RF-020 – Búsqueda Nivel Acto Lucene RF-024 – Búsqueda Nivel Acto MG4J	
Descripción	Se debe realizar la búsqueda a nivel Acto en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	1. Deben estar rellenos los campos una vez pulsemos buscar 2. Observar la consulta sobre el índice y observar que se utilizan todos los campos 3. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 4. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene	

Resultado de la prueba	**Satisfactorio**

Tabla Pruebas 6-4 - CP - 004 – Búsqueda Acto

CP-005	**Búsqueda Escena**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare	
Requisitos asociados	RF-021 – Búsqueda Nivel Escena Lucene RF-025 – Búsqueda Nivel Escena MG4J	
Descripción	Se debe realizar la búsqueda a nivel Escena en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	1. Deben estar rellenos los campos una vez pulsemos buscar 2. Observar la consulta sobre el índice y observar que se utilizan todos los campos 3. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 4. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-5 - CP - 005 – Búsqueda Escena

CP-006	**Búsqueda Discurso**
Versión	1.0 (25/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-03 – Búsquedas booleanas sobre el catálogo de Shakespeare
Requisitos asociados	RF-022 – Búsqueda Nivel Discurso Lucene RF-026 – Búsqueda Nivel Discurso MG4J
Descripción	Se debe realizar la búsqueda a nivel Discurso en ambas tecnologías

Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	1. Deben estar rellenos los campos una vez pulsemos buscar 2. Observar la consulta sobre el índice y observar que se utilizan todos los campos 3. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 4. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-6 - CP - 006 – Búsqueda Discurso

6.5 Grupo Funcional Búsqueda BOA

CP-007	**Búsqueda Boletín**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	RF-028 – Búsqueda Nivel Boletín Lucene RF-032 – Búsqueda Nivel Boletín MG4J	
Descripción	Se debe realizar la búsqueda a nivel Boletín en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	5. Deben estar rellenos los campos una vez pulsemos buscar 6. Observar la consulta sobre el índice y observar que se utilizan todos los campos 7. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 8. Si X > 0 debe aparecer un listado con los resultados	

Salida obtenida	Se cumplen los cuatro casos de aceptación tanto para MG4J como para Lucene
Resultado de la prueba	**Satisfactorio**

Tabla Pruebas 6-7 - CP - 007 – Búsqueda boletín

CP-008	**Búsqueda Sección**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	RF-029 – Búsqueda Nivel Sección Lucene RF-033 – Búsqueda Nivel Sección MG4J	
Descripción	Se debe realizar la búsqueda a nivel Sección en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	5. Deben estar rellenos los campos una vez pulsemos buscar 6. Observar la consulta sobre el índice y observar que se utilizan todos los campos 7. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 8. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-8 - CP - 008 – Búsqueda Sección

CP-009	**Búsqueda Subsección**
Versión	1.0 (25/08/2013)
Autores	Mª Carmen Loriente Yáñez
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA
Requisitos asociados	RF-030 – Búsqueda Nivel Subsección Lucene RF-034 – Búsqueda Nivel Subsección MG4J

Descripción	Se debe realizar la búsqueda a nivel Subsección en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	5. Deben estar rellenos los campos una vez pulsemos buscar 6. Observar la consulta sobre el índice y observar que se utilizan todos los campos 7. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 8. Si X > 0 debe aparecer un listado con los resultados	
Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene	
Resultado de la prueba	**Satisfactorio**	

Tabla Pruebas 6-9 - CP - 009 – Búsqueda Subsección

CP-010	**Búsqueda Detalle**	
Versión	1.0 (25/08/2013)	
Autores	Mª Carmen Loriente Yáñez	
Objetivos asociados	OBJ-04 – Búsquedas booleanas sobre el catálogo BOA	
Requisitos asociados	RF-031 – Búsqueda Nivel Detalle Lucene RF-035 – Búsqueda Nivel Detalle MG4J	
Descripción	Se debe realizar la búsqueda a nivel Detalle en ambas tecnologías	
Entrada	Rellenamos todos los campos	
Procedimiento	**Paso**	**Acción**
	1	Pulsamos el botón Lucene
	2	Pulsamos el botón MG4J
Salida Esperada	5. Deben estar rellenos los campos una vez pulsemos buscar 6. Observar la consulta sobre el índice y observar que se utilizan todos los campos 7. Debe aparecer la frase *Se han encontrado X en Y milisegundos* una vez pulsemos alguno de los dos botones 8. Si X > 0 debe aparecer un listado con los resultados	

Salida obtenida	Se cumplen los tres casos de aceptación tanto para MG4J como para Lucene
Resultado de la prueba	**Satisfactorio**

Tabla Pruebas 6-10 - CP - 010 – Búsqueda Detalle

Capítulo 7 Comparación práctica

7.1 Comparaciones medibles

En este apartado vamos a intentar realizar una comparación objetiva entre ambas tecnologías teniendo en cuenta parámetros y variables que hemos podido medir y poner de relieve entre una tecnología y otra.

7.1.1 Aplicación

En este punto pretendemos medir el tamaño final de la aplicación en relación a las dependencias que tiene cada tecnología.

Al añadir lucene al proyecto, única y exclusivamente nos descarga una librería (lucene-core-3.6.2.jar) de 1,46 MB.

Al añadir MG4J sin embargo se baja 72Mb en dependencias. Si nos descargásemos el rar de dependencias que está en la página principal de MG4J podremos observar como hay 112 librerías de las que depende.

Por tanto podemos determinar que si añadimos MG4J a nuestra aplicación en vez de Lucene, está finalmente ocupará 49 veces más.

7.1.2 Índices

Al generar la aplicación hemos generado distintos índices según la tecnología y el nivel de profundidad especificado en los XML. Al generar estos índices hemos medido los tiempos de indexación y los tamaños finales en disco de dichos índices. Con esto podemos obtener datos de comparación.

7.1.2.1 Shakespeare

Para el catálogo de Shakespeare, hemos generado los siguientes índices con la siguiente información:

	NIVEL OBRA	NIVEL ACTO	NIVEL ESCENA	NIVEL DISCURSO
Nº Campos	3	12	15	16
Campos	1. Fichero 2. titulo 3. ruta	1. Ruta 2. título 3. scndescr 4. playsubt 5. entradilla 6. personae title 7. persona 8. introducción 9. prologo 10. acto título 11. acto 12. epilogo	1. Ruta 2. Título 3. scendescr 4. playsubt 5. entradilla 6. personae title 7. persona 8. introducción 9. prologo 10. epilogo 11. acto_titulo 12. acto_prologo 13. acto_epilogo 14. escena_titulo 15. escena	1. ruta 2. title 3. scndescr 4. playsubt 5. entradilla 6. personae tittle 7. persona 8. Induct 9. prologo 10. epilogo 11. acto_titulo 12. acto_prologo 13. acto_epilogo 14. escena_titulo 15. orador 16. discurso
Nº Ficheros	37			
Colección en Megas	7,6			

Tabla Comparativa 1 - Información Catálogo Shakespeare

7.1.2.1.1 Tiempo de indexación

A continuación una tabla con los tiempos de indexación que ha llevado a cada tecnología los distintos niveles de la colección:

	NIVEL OBRA (ms)	NIVEL ACTO (ms)	NIVEL ESCENA (ms)	NIVEL DISCURSO (ms)
LUCENE	374	327	359	1872
MG4J	1360	1455	1718	4665
Nº Documentos	37	185	750	31028

Tabla Comparativa 2 - Tiempos de indexación en milisegundos

Tal y como podemos observar Lucene es mucho más rápido que MG4J en todos los niveles.

NIVEL OBRA	NIVEL ACTO	NIVEL ESCENA	NIVEL DISCURSO
3,64	4,45	4,79	2,49

Tabla Comparativa 3 - Proporcionalidad de rapidez de Lucene vs MG4J

Aparentemente da la sensación de que cuantos más campos indexemos más rápido es Lucene, pero ante el caso de discurso, no se cumple, por lo que no podemos sacar una afirmación tajante ante esta posibilidad.

7.1.2.1.2 Tamaño Índice

Una vez generados los índices de todos los niveles podemos medir fácilmente lo que ocupa el índice por cada nivel de profundidad.

	NIVEL OBRA (MB)		NIVEL ACTO (MB)		NIVEL ESCENA (MB)		NIVEL DISCURSO (MB)	
	Real	**En disco**	**Real**	**En disco**	**Real**	**En disco**	**Real**	**En disco**
LUCENE	6,16	6,19	6,6	6,62	7,66	7,67	51	51,1
MG4J	1,79	1,94	1,94	2,51	2,17	2,84	8,57	9,25
Nº DOCUMENTOS	37		185		750		31028	

Tabla Comparativa 4 - Tamaño de los índices una vez generados

Ahora podemos observar como el tamaño de los índices de MG4J son muy inferiores a los de Lucene, también podemos observar como al medir el tamaño real y el tamaño en disco MG4J tiene un pequeño factor más de perdida que Lucene. Seguramente esto se deba al mayor número de ficheros que crea por índice tal y como veremos más adelante. Además los índices con poco texto como podrían ser los títulos o las rutas ocupan muy poco, una media de 1 a 4 ks, lo que implica un gran desperdicio de espacio.

NIVEL OBRA	NIVEL ACTO	NIVEL ESCENA	NIVEL DISCURSO
3,19	2,64	2,70	5,52

Tabla Comparativa 5 - Proporcionalidad del tamaño de MG4J vs Lucene

Como vemos está vez sí cuantos más campos hay en el índice más comprimida está la colección de MG4J con respecto a la de Lucene.

Finalmente vamos a mostrar una comparativa de cuanto más grande o más pequeño es el tamaño del índice respecto al nivel y al tamaño de la colección inicial que eran 7.6 MB.

	NIVEL OBRA	NIVEL ACTO	NIVEL ESCENA	NIVEL DISCURSO
LUCENE	-1,23	-1,15	-0,99	6,72
MG4J	-3,92	-3,03	-2,68	-0,82

Tabla Comparativa 6 - Proporcionalidad entre el tamaño inicial de la colección y una vez indexado

Como podemos observar en esta colección no muy grande en general el tamaño de los índices es inferior al de la colección inicial en XML. Tan sólo Lucene en el nivel discurso supera el tamaño de la colección inicial en 6.72 veces

7.1.2.2 Boletín Oficial Aragón

Para el catálogo de BOA, hemos generado los siguientes índices con la siguiente información:

	NIVEL BOLETIN	NIVEL SECCIÓN	NIVEL SUBSECCIÓN	NIVEL DETALLE
Nº Campos	3	8	11	17
Campos	1. Documento 2. Fecha 3. ruta	1. Ruta 2. Anyo 3. Fecha 4. Índice 5. Numero 6. Titulo sumario 7. Newpage 8. sección	1. Ruta 2. Anyo 3. Fecha 4. Índice 5. Numero 6. Titulo sumario 7. newPage 8. título sección 9. código sección 10. código emisor 11. subsección	1. Ruta 2. Anyo 3. Fecha 4. Índice 5. Numero 6. Titulo sumario 7. newPage 8. título sección 9. código sección 10. título subsección 11. código subsección 12. emisor 13. texto Emisor 14. fecha Emisor 15. clave Disposición 16. titulo Texto 17. texto
Nº Ficheros	902			
Colección en Megas	213			

Tabla Comparativa 7- Información Catálogo BOA

7.1.2.2.1 Tiempo de indexación

A continuación una tabla con los tiempos de indexación que ha llevado a cada tecnología los distintos niveles de la colección:

	NIVEL BOLETÍN (ms)	NIVEL SECCIÓN (ms)	NIVEL SUBSECCIÓN (ms)	NIVEL DETALLE (ms)
LUCENE	10154	12138	4518	9454
MG4J	11191	11237	6230	13972
Nº Documentos	902	6528	5682	32112

Tabla Comparativa 8 - Tiempos de indexación en milisegundos

Al igual que ocurría en la colección anterior podemos observar que Lucene tarda menos que MG4J, pero ahora si detectamos que la proporcionalidad de los tiempos es mucho más cercana, que cuando la colección era más pequeña.

NIVEL BOLETÍN	NIVEL SECCIÓN	NIVEL SUBSECCIÓN	NIVEL DETALLE
1,18	1,40	1,32	1,51

Tabla Comparativa 9 - Proporcionalidad de rapidez de Lucene vs MG4J

Esta vez sí que se da el caso en el que cuantos más campos hay más es la diferencia entre Lucene y MG4J.

Otra cosa que debemos resaltar es que en este caso se produce un caso curioso en el nivel subsección y es que el número de documentos indexados es muy inferior al de secciones. Este caso se produce porque en el DTD del XML podemos observar como las secciones pueden o no tener subsecciones, lo que implica que podría haber habido más o menos o las mismas pero al final resulta que en el cómputo global hay muchas menos subsecciones. Esto tiene un poco que ver con lo explicado en los desafíos explicados en el primer capítulo de esta misma memoria.

7.1.2.2.2 Tamaño Índice

Una vez generados los índices de todos los niveles podemos medir fácilmente lo que ocupa el índice por cada nivel de profundidad.

	NIVEL BOLETÍN (MB)		NIVEL SECCIÓN (MB)		NIVEL SUBSECCIÓN (MB)		NIVEL DETALLE (MB)	
	Real	**En disco**	**Real**	**En disco**	**Real**	**En disco**	**Real**	**En disco**
LUCENE	259	259	263	263	110	110	278	278
MG4J	48,9	49	50,6	50,9	21,8	22,2	54,5	55,2

TOTAL DOCUMENTOS	902	6528	5682	32112

Tabla Comparativa 10 - Tamaño de los índices una vez generados

En esta segunda colección podemos observar como los índices de MG4J tienen un altísimo nivel de compresión a diferencia de los de Lucene. Se está reduciendo el tamaño de la colección real en una proporción muy alta, prácticamente en ¼ de la misma.

NIVEL BOLETÍN	NIVEL SECCIÓN	NIVEL SUBSECCIÓN	NIVEL DETALLE
5,29	5,17	4,95	5,04

Tabla Comparativa 11 - Proporcionalidad del tamaño de MG4J vs Lucene

En contraposición al tiempo que ha disminuido considerablemente la proporción en el tamaño ha aumentado considerablemente, lo que nos hace deducir que cuanto mayor sea la colección más a la par estarán en los tiempos ambas tecnologías y más pequeños serán los índices de MG4J con respecto a Lucene.

Si ahora realizamos la proporcionalidad de cada índice de cada tecnología con la colección inicial obtenemos.

	NIVEL BOLETÍN	NIVEL SECCIÓN	NIVEL SUBSECCIÓN	NIVEL DETALLE
LUCENE	1,22	1,23	-1,94	1,31
MG4J	-4,35	-4,18	-9,59	-3,86

Tabla Comparativa 12 - Proporcionalidad entre el tamaño inicial de la colección y el indexado

Como podemos observar excepto en el caso excepcional del nivel de subsección Lucene ronda el mismo tamaño, un poco más que el de la colección inicial, mientras MG4J tiene unos tamaños muy inferiores.

7.1.2.3 Ficheros generados por los índices

Lucene genera para todo el conjunto de documentos uno o varios segmentos según la necesidad de espacio y el siguiente conjunto de campos para un único índice:

Extensión	Descripción
.fdt	Los campos guardados para los documentos
.fdx	Contiene los punteros al campo con la información
.fnm	Información acerca de los campos
.frq	Lista de documentos que contiene cada término con su frecuencia
.nrm	Almacena los factores de peso y puntuación para cada documento
.prx	Almacena información acerca de la posición de un término
.tii	Índice para los accesos al diccionario de términos
.tis	Diccionario de términos, almacena información de los términos

.gen Segments_b	Almacena información de los segmentos

Tabla Comparativa 13 - Archivos generados por los índices de Lucene

Tal y como ya hemos contado a lo largo de la memoria, MG4J genera un índice por cada campo, es decir, que genera los archivos que acontinuación se describen para cada campo que se indexe según el nivel.

Extensión	Descripción
.counts	Número de documentos
.countsoffsets	-
.frequencies	Frecuencia de cada término
.occurrencies	Ocurrencias de cada término
.pointers	Punteros a los documentos
.pointersoffsets	-
.positions	Posiciones de los términos
.positionsoffsets	-
.properties	Conjunto total de propiedades del índice
.sizes	Tamaños de los documentos
.stats	Estadísticas de los términos
.sumsmaxpos	-
.termmap	Mapa de términos e índices
.terms	Términos

7.1.2.4 Búsquedas

Para las búsquedas se han generado unas consultas para cada nivel y se ha medido los resultados, y el tiempo de recuperación con cada medida. Así podemos comparar las diferencias en tiempos y resultados. Además para que los tiempos sean promedio, se ha ejecutado cada consulta por cada tecnología 10 veces y se ha establecido la media de esas diez ejecuciones para cada consulta.

7.1.2.4.1 Shakespeare

Para el nivel obra se han generado 7 tipos de consulta con los siguientes resultados.

CONSULTA	TIPO BUSQUEDA	MG4J		LUCENE	
		T (ms)	Rdos	T (ms)	Rdos
TITLE:henry	ESTRICTA	9,84	7	0,26	7
TITLE:henry , DOCUMENTO:public	ESTRICTA	19,3	7	0,3	7
DOCUMENT:PUBLIC	ESTRICTA	16,2	37	0,2	37
TITLE:henry the fourth	ESTRICTA	14,8	2	0,5	2
TITLE:henry the fourth	CONTIENE	15,7	7	0,3	7

TITLE:henry the fourth, Document: public domain	ESTRICTA	16,7	2	0,6	2
TITLE:henry the fourth, Document: public domain	CONTIENE	15,7	7	0,7	7

Tabla Comparativa 14 - Comparativa búsquedas nivel Obra

Como podemos observar el número de resultados siempre es igual en MG4J que en Lucene con la diferencia que los tiempos de búsqueda de Lucene son muy inferiores a los de MG4J. Como veremos esto se va arepetir a lo largo de la comparativa.

CONSULTA	**TIPO BUSQUEDA**	**MG4J**		**LUCENE**	
		T (ms)	Rdos	T (ms)	Rdos
TITLE:henry the fourth, Acto: lord bardolph	ESTRICTA	49,4	2	0,3	9
TITLE:henry the fourth, Acto: lord bardolph	CONTIENE	72,8	135	0,4	35
TITLE:henry the fourth, Acto: lord bardolph, TITULO_ACTO:II, INDUCT: OPEN YOUR EARS	ESTRICTA	57,2	0	0,5	1
TITLE:henry the fourth, Acto: lord bardolph, TITULO_ACTO:II, INDUCT: OPEN YOUR EARS	CONTIENE	54,1	1	0,8	1

Tabla Comparativa 15 - Comparativa búsquedas nivel Acto

En este nivel existen discrepancias en los resultados encontrados, las búsquedas de tipo estricta para Lucene están reportando falsos positivos con la cadena lord bardolph, cuando debería ser estricta parece que está encontrando sólo lord y le está bastando como resultado válido.

CONSULTA	**TIPO BUSQUEDA**	**MG4J**		**LUCENE**	
		T (ms)	Rdos	T (ms)	Rdos
TITLE:henry	ESTRICTA	49,5	157	0,3	157
TITLE:henry , TITULO ESCENA: LONDON	ESTRICTA	49,9	35	0,3	35

TITLE:henry, INDUCT: Open your ears, TITULO ACTO: ii, TITULO ESCENA: london	ESTRICTA	64,9	3	1,6	3
TITLE:henry, INDUCT: Open your ears, TITULO ACTO: ii, TITULO ESCENA: london	CONTIENE	57,3	3	0,8	3

Tabla Comparativa 16 - Comparativa búsqueda nivel Escena

Aquí no hemos encontrado discrepancias con los resultdos por tanto volvemos a fijarnos en la diferencia que hay entre los tiempos de respuesta de una tecnología y otra.

CONSULTA	**TIPO BUSQUEDA**	**MG4J**		**LUCENE**	
		T (ms)	Rdos	T (ms)	Rdos
DISCURSO: TO BE OR NOT TO BE	ESTRICTA	52,2	1	0	0
TITLE:henry the sixth , TITULO ESCENA: LONDON, TITULO ACTO:act, ORADOR: Suffolk	ESTRICTA	51,6	13	1,4	13
TITLE:henry the sixth , TITULO ESCENA: LONDON, TITULO ACTO:act, ORADOR: Suffolk	CONTIENE	50,1	16	1,1	16
TITLE:henry, INDUCT: Open your ears, TITULO ACTO: ii, TITULO ESCENA: london	ESTRICTA	49,2	6	0,9	6

Tabla Comparativa 17 - Comparativa búsqueda Nivel Discurso

Para este nivel se está produciendo un caso curioso y es que en la primera búsqueda Lucene considera todas las palabras presentes como StopWords lo que está provocando una consulta vacía para Lucene, sin embargo MG4J encuentra perfectamente el resultado correcto.

7.1.2.4.2 Boletín Oficial de Aragón

CONSULTA	**TIPO**	**MG4J**	**LUCENE**

	BUSQUEDA	T (ms)	Rdos	T (ms)	Rdos
FECHA:2012	ESTRICTA	17,5	247	0,7	247
FECHA:2012, DOCUMENTO: HACIENDA	ESTRICTA	17,8	236	1,4	236
FECHA: ABRIL 2012	CONTIENE	17,2	307	1,2	307
FECHA: ABRIL 2012	ESTRICTA	17,1	0	1	17
FECHA: abril 2012, DOCUMENTO: calatayud	CONTIENE	22,4	175	3	175
FECHA: abril 2012, DOCUMENTO: calatayud	ESTRICTA	20,8	0	1,8	13

Tabla Comparativa 18 - Comparativa búsqueda nivel Boletín

Cuando en la consulta ponemos FECHA: abril 2012, y tipo de búsqueda estricta, MG4J no encuentra nada, el 'de', está haciendo que los resultados sean cero, mientras que Lucene al considerar 'de' una StopWord no está encontrando problemas en resolver la consulta. Otra cosa a destacar es que podemos observar cómo según va subiendo la cantidad de texto en un campo de búsqueda, los tiempos de búsqueda para lucene aumentan. También podemos destacar que contiene cuesta más a ambas tecnologías que la búsqueda estricta.

CONSULTA	TIPO BUSQUEDA	MG4J		LUCENE	
		T (ms)	Rdos	T (ms)	Rdos
AÑO:XXXI, SECCIÓN:obras	ESTRICTA	37,6	729	2	729
AÑO:XXXI, SECCIÓN:obras	CONTIENE	36,6	729	1,9	729
AÑO:XXXI, SECCIÓN:obras, Número:70, Fecha:abril	CONTIENE	36,3	4	1,6	4
AÑO:XXXI, SECCIÓN:obras, Número:70, Fecha:abril	ESTRICTA	35,3	4	1,1	4

Tabla Comparativa 19 - Comparativa nivel Sección

En este nivel hemos detectado que el campo índices compuesto de [0-9]{1,4}-[0-9]{1,4}[;] provoca errores en ambos índices. Por lo que en principio se ha excluido de las búsquedas. Por otro lado vemos como los resultados para este nivel coinciden, y los tiempos de Lucene siguen aumentando paulatinamente, aunque también los de MG4J que mantiene la distancia.

CONSULTA	TIPO BUSQUEDA	MG4J		LUCENE	
		T (ms)	Rdos	T (ms)	Rdos
AÑO:XXXII, FECHA: marzo, Titulo Seccion: V. Anuncios, CODIGO SECCIÓN:5	ESTRICTA	41,8	0	1,1	34
AÑO:XXXII, FECHA: marzo, Titulo Seccion: V. Anuncios, CODIGO SECCIÓN:5	CONTIENE	41,3	34	1,4	34
AÑO:XXXII, FECHA: marzo, CODIGO SECCIÓN:5, SUBSECCIÓN: AEROPUERTO	ESTRICTA	40	2	0,7	2
AÑO:XXXII, FECHA: marzo, CODIGO SECCIÓN:5, SUBSECCIÓN: AEROPUERTO	CONTIENE	39,8	2	0,9	2

Tabla Comparativa 20 - Comparativa búsqueda nivel Subsección

En la primera búsqueda estricta, se están produciendo 0 resultados para MG4J cuando V. Anuncios es una sección clara de varios boletines. Si eliminamos la V. de delante si funciona correctamente.

CONSULTA	TIPO BUSQUEDA	MG4J		LUCENE	
		T (ms)	Rdos	T (ms)	Rdos
AÑO:XXXI, FECHA:Marzo, FECHA EMISOR: 03/01/2012, TITULO_TEXTO: Universidad de Zaragoza	ESTRICTA	59,4	0	3,4	32
AÑO:XXXI, FECHA:Marzo, FECHA EMISOR: 03/01/2012, TITULO_TEXTO: Universidad de Zaragoza	CONTIENE	58,5	0	3,8	750
AÑO:XXXI, FECHA:Marzo, TITULO_TEXTO: Universidad de Zaragoza, TEXTO: De	CONTIENE	57,6	753	5,9	753

conformidad con la propuesta formulada por la Comisión constituida					
AÑO:XXXI, FECHA:Marzo, TITULO_TEXTO: Universidad de Zaragoza, TEXTO: De conformidad con la propuesta formulada por la Comisión constituida	ESTRICTA	57,9	10	5,1	10

Tabla Comparativa 21- Comparativa búsqueda nivel detalle

Volvemos a encontrarnos con problemas para MG4J, las fechas en formato texto parecen no funcionar correctamente. Una vez añadido el filtro Fecha emisor: 03/01/2012, no retorna ningún resultado, aun habiendo tratado dicha fecha desde el principio (incluida la indexación) como texto.

Por tanto podemos concluir que en líneas generales Lucene tarda menos en realizar la búsqueda, aunque ambos tiempos son casi desechables, proporcionalmente Lucene es mucho más rápido. Por otro lado MG4J suele ser más certero en el conjunto de resultados, provoca menos falsos positivos. Y el problema o no de las StopWords al final tiene que ser una decisión de la persona que decide generar los índices de la colección. Debe decidir si tenerlas en cuenta o no, ya que como hemos visto en algunos casos han jugado a nuestro favor y en otros, en contra.

7.2 Comparaciones no cuantificables

7.2.1 Dificultad de inserción

En este punto vamos a encontrar un claro empate técnico ya que la complejidad de añadir uno u otro, nos la ha solventado maven, obligándonos tan solo a añadir la dependencia directa de cada librería principal (lucene-core y mg4j-core).

Inicialmente en la propia página de MG4J encontré que MG4J necesitaba la JDK6 o superior para funcionar, y Lucene indicaba que para la versión que he incluido (las hay más modernas) JDK5 o superior. Fiándome de esto inicialmente establecí la JDK del proyecto a la 6 para asegurarme de que ambas tecnologías funcionarían correctamente. Mi sorpresa fue mayúscula cuando al ejecutar las clases de test de MG4J lanzaban excepciones. Para averiguar cuál era el problema tuve que depurar dentro de la propia librería para averiguar cuál era el problema. Resulta que en la última versión ya están utilizando las nuevas implementaciones de la JDK7, lo que hace no factible el uso de la 6, si queremos obtener todas las funcionalidades.

En este punto la información de instalación es mucho más fiable y certera la de Apache Lucene.

7.2.2 Dificultad de aprendizaje

En este punto sí que no hay guerra posible. Con Apache Lucene en un par de tardes era capaz de generar índices más o menos correctos y realizar búsquedas más o menos complejas.

Sin embargo conseguir indexar correctamente en MG4J fue excesivamente complicado. Me llevo bastante más de dos semanas conseguir hacerlo y hacerlo bien, y todo a base de entrar mucho en la propia librería a comprobar cómo iba siendo el funcionamiento. Sin querer extenderme demasiado en esto, que quedará mejor explicado en el siguiente punto, conseguir encontrar documentación válida de MG4J es imposible en la red, lo cual complica sobre manera entender los conceptos básicos de la librería.

Cómo digo la única forma de entender cómo funciona MG4J se redujo a ver cómo iba haciendo más o menos las cosas en las propias clases que conforman la librería, hasta conseguir recuperar la información de los documentos una vez ejecutada la búsqueda se ha convertido prácticamente en una misión imposible.

Además conceptualmente es más sencillo y conlleva mucha menos complejidad Lucene, si añadimos a esto la cantidad de ejemplos que existen en la propia página de Lucene, pues el tiempo de hacerte con la librería desciende considerablemente.

7.2.3 Documentación

En este punto podemos usar la famosa frase de las comparaciones son odiosas, ya que casi resulta hasta injusta esta comparación. Pero a la hora de desarrollar, este punto suele ser definitivo a la hora de decidir si añadir o no una nueva tecnología a nuestros proyectos.

Apache, en general, detrás de todos sus proyectos tiene una gran comunidad que los prueba, ayuda a desarrollar y da soporte. Además Lucene cuenta con la ventaja de ser una librería utilizada en gran cantidad de proyectos muy establecidos, sin ir más lejos el mismo eclipse con el que se ha generado esta aplicación tiene integrado Lucene. Es por esto que existe un gran número de desarrolladores que antes o después se han ido encontrando problemas similares a los que yo he ido encontrando. Esto favorece el rápido desarrollo ya que alguien antes que tu ha tenido el problema y ha publicado como solucionarlo.

Por otro lado MG4J es una tecnología minoritaria, especialmente utilizada como aplicación independiente, es decir para instalártela en el sistema y usarla tal cual y no como librería sobre la que desarrollar. Esto hace que exceptuando el API prácticamente no exista ningún tipo de información en la red. Lo más cercano es un pequeño grupo de goolge code, que no llega ni a 40 hilos, y la mayoría de estos son de gente que ha intentado ejecutarlo vía línea de comando, por lo que ha supuesto un claro reto, averiguar el funcionamiento y los conceptos básicos de la librería.

Conclusiones

Conclusiones

De algún modo u otro todos hemos utilizado herramientas de búsqueda alguna vez en la vida, pero quizás el simplismo de cierto motor de búsqueda web, nos dé la falsa idea de que es un proceso fácil y sencillo.

Tal y como hemos podido comprobar nada más allá de la realidad, retornar unos resultados correctos al usuario final implica un gran número de complicadas fases previas, en las que hay que tener claro que se indexa y cómo hacerlo del modo más correcto.

A nivel personal he de reconocer que ya conocía Lucene por ser una tecnología bastante utilizada en multitud de herramientas. Directamente nunca había utilizado la librería como tal, si había generado algún índice pero con herramientas de terceros, que facilitan la generación casi a tres clicks. Del mismo modo nunca antes había oído hablar de MG4J, y la poca documentación existente no me ha ayudado mucho en el camino.

En este punto me encantaría resaltar el reto que ha constituido aprender y programar con MG4J. Ha generado los momentos más duros y también los de mayor satisfacción personal a lo largo de la creación de la aplicación.

Como conclusiones finales respecto a la comparativa, me gustaría destacar que si queda claro que cuanto más grande es la colección MG4J logrará reducir considerablemente el tamaño de los índices, pero esto implica un mayor coste de tiempo en la indexación y aparentemente también en la búsqueda.

Las dificultades que conlleva añadir MG4J a nuestras aplicaciones de un modo funcional y la poca documentación existente, hacen que sin duda si algún día tengo que elegir entre añadir a mi aplicación una u otra tecnología, me decante por Lucene. Sólo me lo pensaría remotamente si los catálogos fueran tan gigantescos que mis servidores de verdad necesitaran el espacio que nos proporcionaría MG4J de ganancia.

Trabajo futuro

Este trabajo se ha realizado única y exclusivamente con campos textuales, lo que en principio puede ser suficiente, pero ambas tecnologías están capacitadas para indexar otro tipo de campos como pueden ser las fechas y los números. De este modo se podría estudiar los algoritmos de compresión e indexación para este tipo de campos e incluso probar las consultas de tipo Rango.

Una tarea pendiente que me ha faltado en el proyecto, es ser capaz de recuperar la información de los documentos resultantes de una búsqueda para MG4J, es decir, el

texto del título, el texto del acto, etc. Al generar la indexación desde stream y habiendo tenido que generar mis propios documentos e iteradores, parece que falta algún paso intermedio.

Quizás también una regeneración de los índices directamente desde la aplicación web.

Bibliografía, webgrafía y material

Bibliografía

Los libros consultados para la realización del proyecto son.

[1] Craig Larman, *UML y patrones: introducción al análisis y diseño orientado a objetos.* Prentice-Hall, 2006, 590 p. 21 cm.

[2] Deitel, Paul J., *Java: Cómo programar.* Pearson Educación, 2008, 1338 p. 26 cm.

[3] Donald Brown, Chad Michael Davis y Scott Stanlick, *Struts 2 in Action.*Manning Publications co. 424 p.

[4] Erich Gamma, *Patrones de diseño elementos de software orientado a objetos reutilizable.* Addison-Wesley, 2006, 364 p. 25 cm.

[5] Michael Mccandless, Erik Hatcher, Otis Gospodnetic, *Lucene In Action, Second Edition.* Manning Publications co. 380 p, 2010, 532 p.

[6] Ian H. Written (University of Waikato), Alistair Moffat (University of Melbourne), Timothy C. Bell (University of Canterbury), *Managing Gigabytes, Compressing and Indexing Documents and Images, Second Edition.* Morgan Kaufmann Publishers. 1999, 551 p.

[7] Srirangan, *Apache Maven 3 Cookbook, First Edition.* Packt Publishing Ltd. 2011, 224 p.

Webgrafía

[1] http://indiceri.netne.net/Indices%20Invertidos.html, Definición de índices invertidos, última visita el 10/06/2013.

[2] http://www.ganimides.ucm.cl/haraya/doc/Arboles_B_2.pdf, Árboles B, Profesor Hugo Araya Carrasco (Universidad Católica de Maule, Chile), última visita el 10/06/2013.

[3] http://slady.net/java/bt/view.php?w=600&h=450, Applet de árbol B, última visita el 10/06/2013.

[4] http://www.undernews.com/2010/05/06/las-100-paginas-webs-mas-visitadas-en-espana/450, Las páginas más visitadas en España, última visita 11/06/2013

[5] http://www.undernews.com/2010/05/06/las-100-paginas-webs-mas-visitadas-en-espana/450, Las páginas más visitadas en España, última visita 11/06/2013

[6] http://www.google.com/publicdata/explore?ds=d5bncppjof8f9_&met_y=it_net_user_p2&idim=country:ESP&dl=es&hl=es&q=uso+de+internet+en+espa%C3%B1a#ctype=l&strail=false&nselm=h&met_y=it_net_user_p2&scale_y=lin&ind_y=false&rdim=country&idim=country:ESP&ifdim=country&hl=es&dl=es, Información usuarios internet España, última visita el 11/06/2013.

[7] www.wikipedia.es, www.wikipedia.com, web de información general en la que consultamos entre otras:

- JSTL: http://es.wikipedia.org/wiki/JSTL, última visita 30/07/2013
- Motor de búsqueda: http://es.wikipedia.org/wiki/Motor_de_b%C3%BAsqueda, http://en.wikipedia.org/wiki/Web_search_engine, última visita 12/06/2013
- Consulta: http://en.wikipedia.org/wiki/Web_search_query, última visita 12/06/2013
- Araña Web: http://en.wikipedia.org/wiki/Web_crawling , http://es.wikipedia.org/wiki/Ara%C3%B1a_web , última visita 12/06/2013
- Recuperación de información: http://es.wikipedia.org/wiki/Recuperaci%C3%B3n_de_informaci%C3%B3n , última visita 12/06/2013
- XML: http://es.wikipedia.org/wiki/XML , última visita 11/08/2013
- Arbol- B: http://en.wikipedia.org/wiki/B-tree, último visita el 12/08/2013
- Algoritmo de Huffman: http://en.wikipedia.org/wiki/Huffman_coding, última visita el 13/08/2013
- Codificación aritmética: http://en.wikipedia.org/wiki/Arithmetic_coding, última visita el 14/08/2013

[8] http://mundogeek.net/archivos/2009/02/13/etiquetas-struts-2/, descripción del conjunto de etiquetas de Struts2 y su uso, última visita el 15/07/2013.

[9] http://java.dzone.com/articles/struts2-tutorial-part-27, última visita el 15/07/2013.

[10] http://stackoverflow.com/ Página de resolución de dudas en general, última visita el 31/08/2013.

[11] http://www.mkyong.com/struts2/struts-2-hello-world-example/, última visita 10/08/2013

[13] http://www.w3schools.com/, Página de tutoriales para diversos lenguajes, visitamos entre otras, última visita el 15/08/2013

- http://www.w3schools.com/jquery/default.asp
- http://www.w3schools.com/html/default.asp
- http://www.w3schools.com/css/default.asp

- http://www.w3schools.com/js/default.asp

[15] http://tomcat.apache.org/tomcat-7.0-doc/index.html , última visita el 24/07/2013.

[16] http://lucene.apache.org/core/3_6_0/api/all/, última visita 12/08/2013

[17] http://lucene.apache.org/, última visita 12/08/2013

[18] http://www.lucenetutorial.com/lucene-in-5-minutes.html, última visita 05/03/2013

[19] http://mg4j.di.unimi.it/, última visita 12/08/2013

[20] https://groups.google.com/forum/#!forum/mg4j, última visita 12/08/2013

[21] http://www.paxle.net/en/start , última visita 22/08/2013

[22] https://github.com/Paxle/Paxle/tree/v0.1.0/bundles/IndexMG4J, última visita 22/08/2013

Programas

A continuación listamos los programas que hemos usado, todos ellos se encuentran en el CD adjunto en la carpeta Software utilizado.

- **Eclipse Indigo**
  Es un famoso entorno de desarrollo integrado de código libre con el que hemos trabajado para diseñar la aplicación práctica para medir los resultados de ambas tecnologías.
  - Aptana Studio 2.0.5 PlugIn
    Es un plugin muy potente que facilita el manejo de eclipse en ciertas funciones como en el CSS o en la vista previa de la página.

- **Apache Tomcat 7.0**
  Servidor de aplicaciones sobre la que se ejecuta la aplicación práctica de la memoria.

  .

- **GIMP 2.8**
  Editor gráfico de código libre con el que hemos realizado algunos diagramas, editado de imágenes, iconos, pantallas etc...

- **Microsoft Office 2007**
  - Word
    Famosa suite ofimática de Microsoft con el que hemos escrito esta memoria y creado algunos diagramas.
  - PowerPoint

Programa con el crear presentaciones con el que preparamos la presentación para la lectura del proyecto.

- **ArgoUML 0.30.2**
  Programa de dibujo de diagramas UML con el que hemos dibujado los casos de uso, paquetes, etc….

- **Navegadores Web**
  Para realizar las pruebas de la práctica y para realizar las búsquedas necesarias en internet para completar esta memoria

- **Adobe Acrobat reader**
  Para la lectura de los libros en formato PDF y algunas informaciones encontradas en internet

- **Microsoft Visio**
  Para realizar diagramas que se han mostrado tanto en la memoria como en el Power Point de la presentación

- **Luke-All**
  Herramienta para medir, controlar, realizar búsquedas y optimizar los índices generados con Apache Lucene.

- **Notepad y Ultraedit**
  Herramienta para modificar xml, buscar en catálogos grandes, formatear y tratar ficheros grandes.

Librerías

Para este proyecto se ha utilizado Maven, lo que facilita muchísimo el trabajo con librerías. Es más en el proyecto no van las librerías, sino que será el propio pom.xml del proyecto el que gestione la descarga del repositorio central las librerías y sus dependencias, siendo un trabajo completamente transparente para el desarrollador, pero sobre todo para el montador del sistema final.

Las librerías que necesita el proyecto son:

- struts2-core-2.3.1.2.jar
- lucene-core-3.6.2.jar
- castor-core-1.3.jar
- xercesImpl-2.9.1.jar
- log4j-1.2.12.jar
- dsiutils-2.0.14.jar
- mg4j-5.2.jar

Para ello simplemente hemos añadido en el POM en la sección de dependencias los siguientes valores:

```
<dependency>
	<groupId>org.apache.struts</groupId>
	<artifactId>struts2-core</artifactId>
	<version>2.3.1.2</version>
</dependency>
<dependency>
	<groupId>org.apache.lucene</groupId>
	<artifactId>lucene-core</artifactId>
	<version>3.6.2</version>
</dependency>
<dependency>
	<groupId>org.codehaus.castor</groupId>
	<artifactId>castor-xml</artifactId>
	<version>1.3</version>
	<exclusions>
		<exclusion>
			<groupId>commons-logging</groupId>
			<artifactId>commons-logging</artifactId>
		</exclusion>
	</exclusions>
</dependency>
<dependency>
	<groupId>xerces</groupId>
	<artifactId>xercesImpl</artifactId>
	<version>2.9.1</version>
	<exclusions>
		<exclusion>
			<groupId>xml-apis</groupId>
			<artifactId>xml-apis</artifactId>
		</exclusion>
	</exclusions>
</dependency>
<dependency>
	<groupId>log4j</groupId>
	<artifactId>log4j</artifactId>
	<version>1.2.12</version>
</dependency>
<dependency>
	<groupId>junit</groupId>
	<artifactId>junit</artifactId>
	<version>4.10</version>
	<scope>test</scope>
</dependency>
<dependency>
	<groupId>it.unimi.dsi</groupId>
	<artifactId>dsiutils</artifactId>
```

```
            <version>2.0.14</version>
        </dependency>
        <dependency>
            <groupId>it.unimi.di</groupId>
            <artifactId>mg4j</artifactId>
            <version>5.2</version>
        </dependency>
```

Anexos

Apéndice A
Manual de instalación

A lo largo de este manual vamos a explicar cómo realizar una instalación limpia de la aplicación de Comparación entre Lucene vs MG4J

A.1 Instalación del Tomcat

Del servidor de aplicaciones tampoco necesitamos nada en especial. La versión de esta aplicación está desarrollada sobre Tomcat 7.0. Aunque se ha comprobado que si funciona bajo tomcat 6.0, aunque algunas consultas complejas podrían no funcionar.

Por tanto aconsejamos la instalación normal y corriente de Tomcat 7.0. Suele ser simplemente descomprimir el servidor donde queramos ubicarle. Si se desean configura puertos o características especiales del Tomcat (no será necesario para esta aplicación) se aconseja seguir el manual de Apache (Configuración).

A.2 Establecemos ruta catálogos

Debemos ubicar en un directorio visible para el servidor de aplicaciones los catálogos que vamos a indexar. Además ubicaremos los XML de mapeo llamados BOA-Castor.xml y Shakespeare-Castor.xml. Este paso es previo ya que después tendremos que decirle a la aplicación estas ubicaciones. Los ficheros xml de mapeo se encuentran en el código fuente en es.uva.mg4jvslucene.architecture.model cada uno bajo el paquete del catálogo que le corresponde (*catalogoboa*, *catalogoshakespeare*).

A.3 Despliegue de la aplicación

Para esta sección podemos encontrarnos ante dos situaciones. Por tanto vamos a describir las dos

A.3.1 La aplicación ya está generada en un artefacto .war

Si nos encontramos con que tenemos ya el .war generado tan solo tendremos que seguir dos pasos.

- El primero será abrir el .war y ubicarnos en la carpeta WEB-INF\classes\properties y editar el archivo aplication.properties modificando los parámetros de ubicación de los catalogos XML y la ruta bajo la que queramos que se almacenen nuestros índices. Este paso se puede realizar con el Win-Rar o similar.
- Una vez modificado y re-insertado en el .war el archivo aplication.properties movemos el archivo a la carpeta webapps del servidor.

A.3.2 Tenemos el código fuente y debemos compilar

Debemos tener instalado MAVEN en el servidor donde vayamos a generar la compilación.

En el Eclipse sobre la raíz del proyecto hacemos click en el botón derecho y buscamos Run as > Maven build …

Se nos abrirá un pop-Up. Buscaremos la línea donde pone Goals y añadiremos los comandos clean package install. Esto generará el artefacto WAR bajo la carpeta target del workspace.

Una vez termina la ejecución vamos a la carpeta target y copiamos el war bajo la carpeta webapp del servidor de apliaciones.

A.4 Arrancamos la aplicación

Ya tan sólo nos falta arrancar el Tomcat, y vía navegador acceder a la aplicación con el puerto configurado por el tomcat y al contexto tfg, por ejemplo:

http://localhost:8080/tfg/

Apéndice B
Manual de la aplicación

B.1 Indexar Shakespeare

Para indexar Shakespeare habría que ejecutar desde el eclipse cualquiera de los ficheros según lo que deseemos indexar.

Los ficheros de indexación se encuentran bajo la carpeta src/test/java/es.uva.mg4jvslucene.indexar.shakespeare

Podemos encontrar los siguientes ficheros:

1. IndexarObraLucene.java
2. IndexarActoLucene.java
3. IndexarEscenaLucene.java
4. IndexarDiscursoLucene.java
5. IndexarTodoLucene.java
6. IndexarObraMg4j.java
7. IndexarActoMg4j.java
8. IndexarEscenaMg4j.java
9. IndexarDiscursoMg4j.java
10. IndexarTodoMg4j.java
11. IndexarTodoShakespeare.java

Para ejecutar cualquiera de estos ficheros tan solo deberemos seleccionar botón derecho sobre el fichero Run as Java Aplication

Al ejecutar el sistema nos irá dejando la traza en el log tal y como se muestra a continuación:

```
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:33] - [MG4JVSLUCENE] El fichero [catalog] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [dsssl.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [fot.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [play.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [play.xsd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [playlist] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [shaksper.htm] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [style-sheet.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [vs] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [vx] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [xml.dcl] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [xml.soc] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Comenzamos la indexación del catalogo [NIVELOBRA] ubicado en D:/CONFIG/mg4jvslucene/indexdirs/shakespeare/lucene/NIVELOBRA
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Tiempo de inicio 1376851955136
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Tiempo de fin 1376851955589
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El tiempo indexando 37 ficheros ha llevado 453 milisegundos
```

Ilustración 16 - Log de salida de la indexación

Tal y como podemos observar existe una fichero por cada nivel de indexación y por cada tecnología, también podemos indexar todos los niveles de una tecnología o con IndexarTodoShakespeare.java realizaremos la indexación de todos los niveles de ambas tecnologías.

B.2 Indexar Boletín Oficial de Aragón

Para indexar el catalogo del boletín oficial de Aragón habría que ejecutar desde el eclipse cualquiera de los ficheros según lo que deseemos indexar.

Los ficheros de indexación se encuentran bajo la carpeta src/test/java/es.uva.mg4jvslucene.indexar.boa

Podemos encontrar los siguientes ficheros:

1. IndexarBoletinLucene.java
2. IndexarSeccionLucene.java
3. IndexarSubseccionLucene.java
4. IndexarDetalleLucene.java
5. IndexarTodoLucene.java
6. IndexarBoletinMg4j.java
7. IndexarSeccionMg4j.java
8. IndexarSubseccionMg4j.java
9. IndexarDetalleMg4j.java
10. IndexarTodoMg4j.java
11. IndexarTodoBOA.java

Para ejecutar cualquiera de estos ficheros tan solo deberemos seleccionar botón derecho sobre el fichero Run as Java Aplication

Al ejecutar el sistema nos irá dejando la traza en el log tal y como se muestra a continuación:

```
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:33] - [MG4JVSLUCENE] El fichero [catalog] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [dssi.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [fot.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [play.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [play.xsd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [playlist] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [shaksper.htm] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:34] - [MG4JVSLUCENE] El fichero [style-sheet.dtd] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [vs] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [vx] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [xml.dcl] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.architecture.ln.mappear.Mappear) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El fichero [xml.soc] no se parsea ya que no es de tipo [.xml].
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Comenzamos la indexación del catalogo [NIVELOBRA] ubicado en D:/CONFIG/mg4jvslucene/indexdirs/shakespeare/lucene/NIVELOBRA
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Tiempo de inicio 1376851955136
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] Tiempo de fin 1376851955589
INFO (es.uva.mg4jvslucene.GF01Shakespeare.IndexShakespeareLuceneScript) [18 ago 2013 20:52:35] - [MG4JVSLUCENE] El tiempo indexando 37 ficheros ha llevado 453 milisegundos
```

Ilustración 17 - Log de salida de la indexación

Tal y como podemos observar existe una fichero por cada nivel de indexación y por cada tecnología, también podemos indexar todos los niveles de una tecnología o con IndexarTodoBOA.java realizaremos la indexación de todos los niveles de ambas tecnologías.

B.3 Indexar todos los catálogos

Igual que existen ficheros para indexar por cada nivel de cada catalogo podemos lanzar la indexación total de ambos catálogos en sus cuatro niveles de profundidad en ambas tecnologías, para ello ejecutaremos del mismo modo que los anteriores el fichero IndexarTodo.java ubicado en src/test/java/es.uva.mg4jvslucene.indexar

B.4 Acceso a la aplicación

En esta zona encontraremos una página de inicio con todos los accesos a las distintos tipos de búsqueda dependiendo del catálogo y el nivel de búsqueda que queramos.

Ilustración 18- Página de Inicio

B.5 Búsquedas Shakespeare

La búsqueda de Shakespeare permite realizar búsquedas a distintos niveles de profundidad en el catálogo XML de Shakespeare. Esta sección de la aplicación se divide en cuatro pantallas (una por nivel) tal y como mostramos a continuación. Cada pantalla contiene un formulario con los campos que corresponden así como un botón para buscar por Lucene y otro para buscar por MG4J.

B.5.1 Búsqueda por Obra

En esta ventana podremos realizar búsquedas sobre el catálogo de Shakespeare a nivel Obra. Como este nivel es muy alto y apenas tiene campos, solo tendremos posibilidad de filtrar por título o por texto donde se encuentra indexado el conglomerado del fichero.

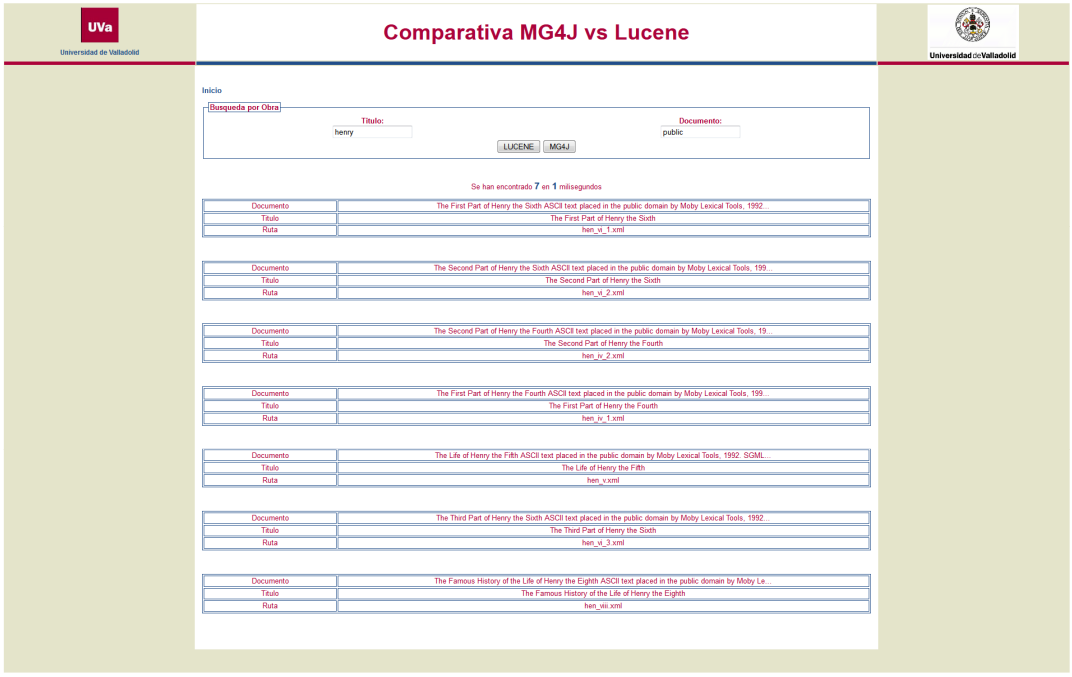

Ilustración 19 - Búsqueda Shakespeare a nivel Obra

B.5.2 Búsqueda por acto

En esta pantalla podremos realizar búsquedas de actos sobre el catálogo de Shakespeare. Al realizar la indexación a nivel acto poseemos muchos más campos por los que poder filtrar como título, prologo, epílogo, título del acto, texto del acto o introducción.

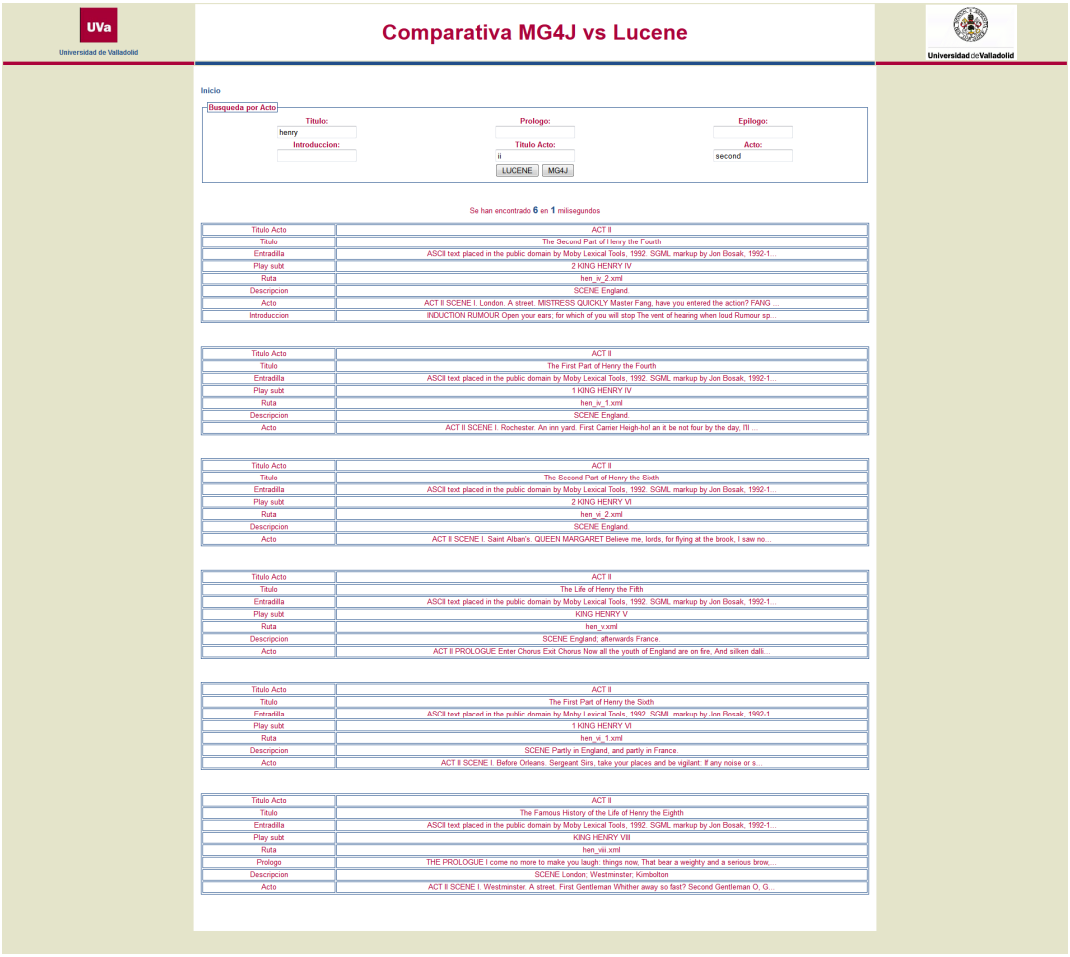

Ilustración 20 - Búsqueda Shakespeare a nivel acto

B.5.3 Búsqueda por Escena

En esta pantalla podremos realizar búsquedas de escenas sobre el catálogo de Shakespeare. Al realizar la indexación a nivel escena, ahora a tendremos también los campos de prólogo y epilogo del acto, título de la escena y texto de la escena.

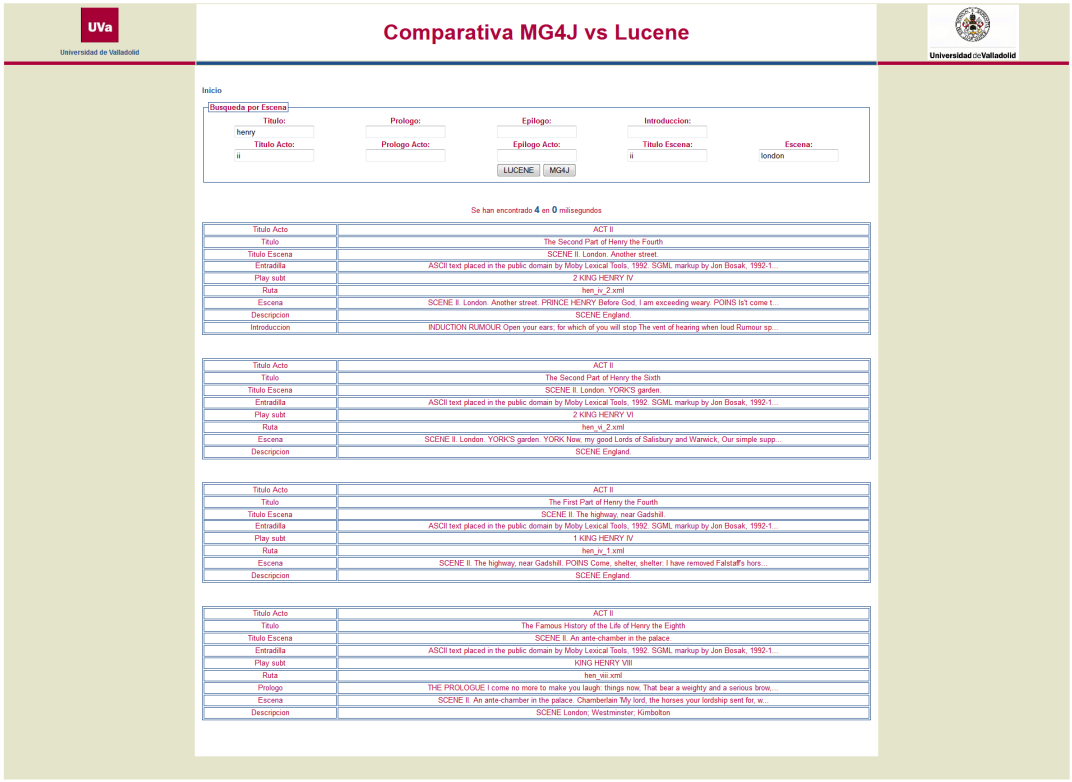

Ilustración 21 - Búsqueda Shakespeare nivel Escena

B.5.4 Búsqueda discurso

Finalmente en el catálogo de Shakespeare podremos filtrar por los discursos de los distintos personajes de cada obra. Tendremos por tanto campos como orador y discurso para poder filtrar a este nivel.

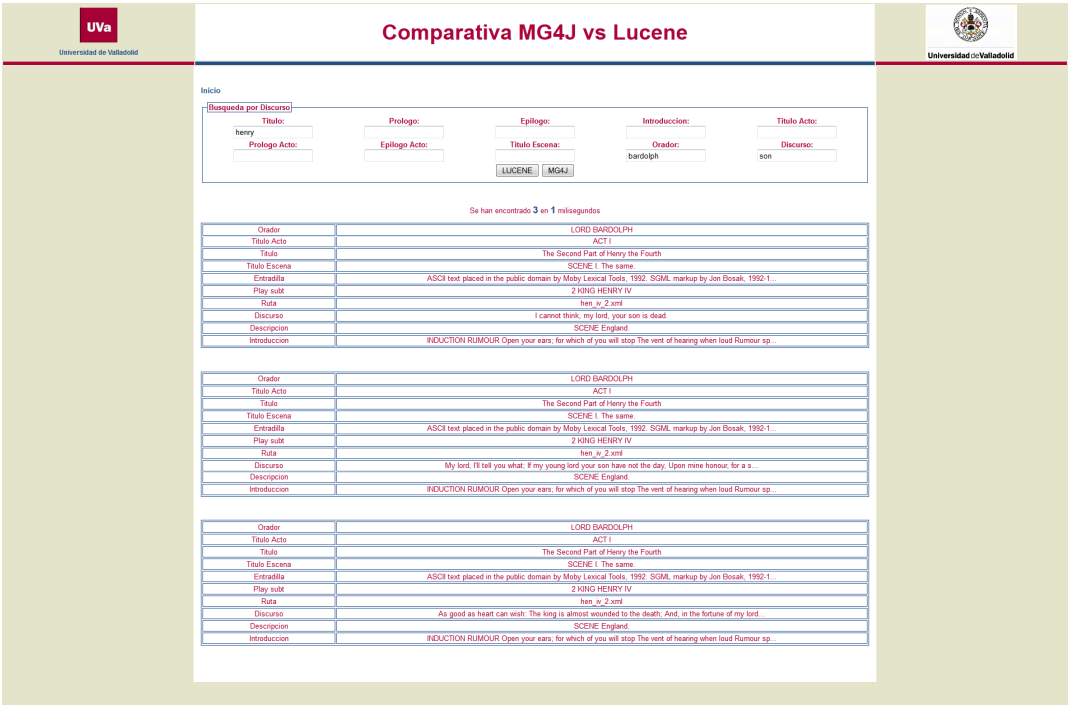

Ilustración 22 - Búsqueda Shakespeare nivel discurso

B.6 Búsqueda BOA

La búsqueda BOA permite realizar búsquedas a distintos niveles de profundidad sobre el catálogo de Boletines Oficiales de Aragón en formato XM. Esta sección de la aplicación se divide en cuatro pantallas (una por nivel) tal y como mostramos a continuación. Cada pantalla contiene un formulario con los campos que corresponden así como un botón para buscar por Lucene y otro para buscar por MG4J.

B.6.1 Búsqueda por boletines

Esta indexación es prácticamente a nivel fichero por lo que únicamente disponemos de dos filtros, el de la fecha del boletín y el documento completo.

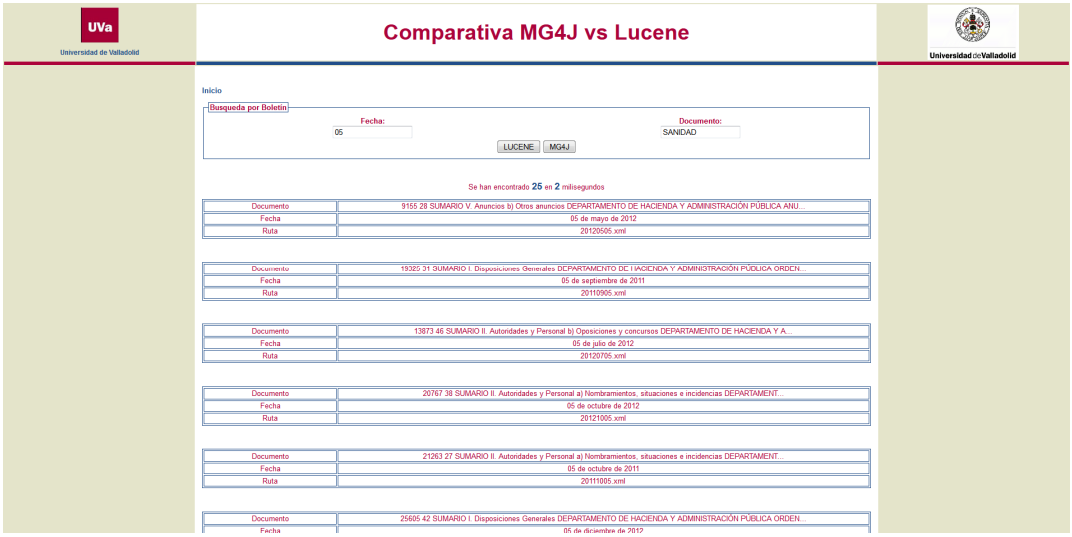

Ilustración 23 - Búsqueda BOA a nivel boletín

B.6.2 Búsqueda a nivel sección

En esta pantalla buscaremos secciones del Boletín Oficial de Aragón. Por tanto ahora dispondremos de más información para filtrar el catálogo, como año, el número, el índice o el texto de la sección.

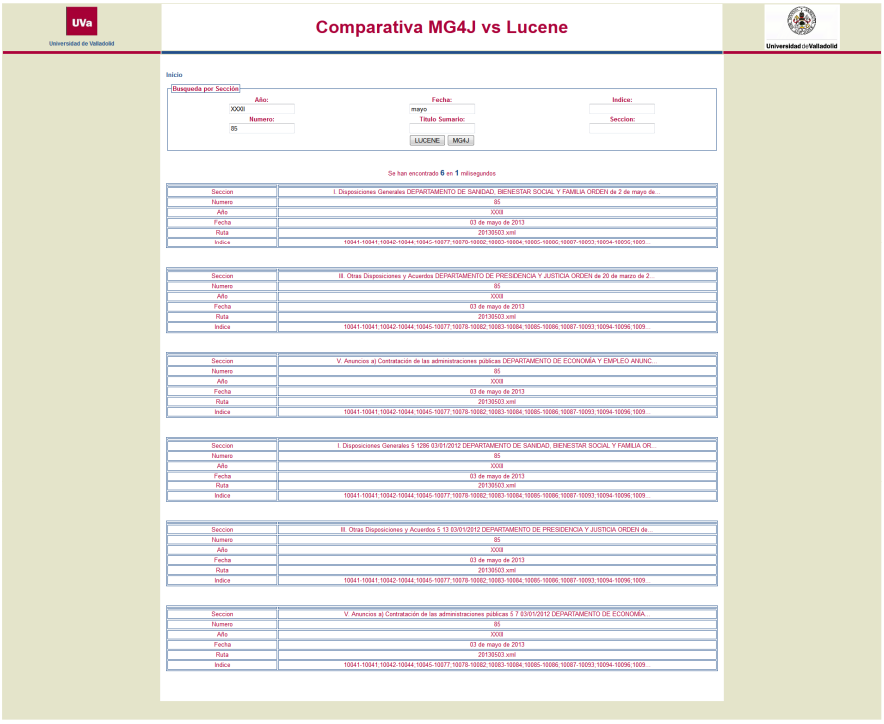

Ilustración 24 - Búsqueda BOA a nivel sección

B.6.3 Búsqueda por Subsección

Las secciones pueden estar compuestas por subsecciones, por tanto estamos un nivel más profundo que en la búsqueda por sección, por lo que ahora podremos filtrar por la información relativa a la sección a la que pertenece dicha subsección como puede ser el título o el código de sección.

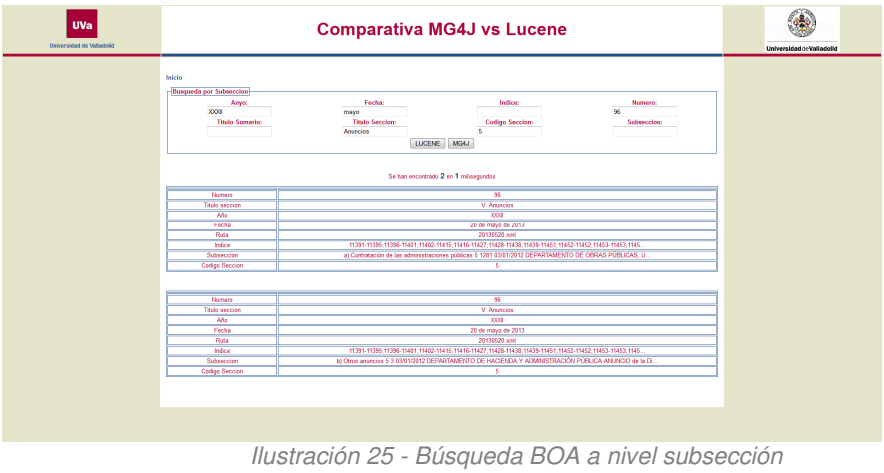

Ilustración 25 - Búsqueda BOA a nivel subsección

B.6.4 Búsqueda por detalle del boletín

En este caso vamos a realizar búsquedas en los textos de los boletines, en la sección de detalle de la información de cada disposición. Aquí tenemos catorce filtros por los que obtener más detallada la información.

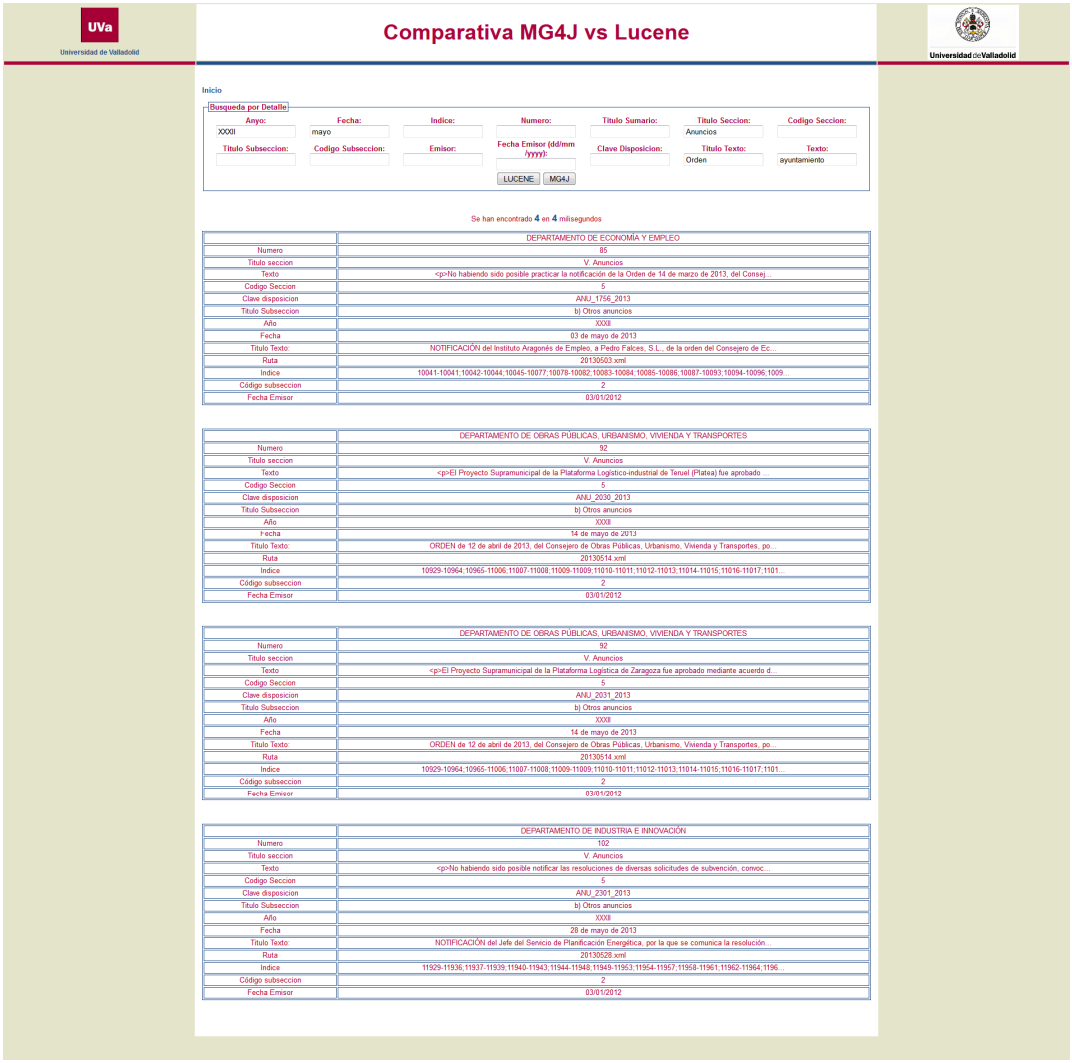

Ilustración 26 - Búsqueda BOA a nivel detalle

Apéndice C
Contenidos del CD

En el CD adjunto se encuentran los programas que se han utilizado así como el código fuente desarrollado.

La forma de organizar los contenidos del CD es la siguiente:

Código Fuente: Carpeta con el código fuente del proyecto con el formato típico de un proyecto Maven. A demás en la carpeta resources podemos encontrar los diagramas de modelo y secuencia de la aplicación.

Carpeta Imágenes: Esta carpeta contiene las siguientes carpetas:

- Imágenes de la memoria - Ilustraciones empleadas en la documentación.
- Imágenes del archivo de presentación - Todas las figuras y diagramas empleadas para la realización del archivo .ppt para la presentación.
- Diagramas - Los distintos diagramas que contiene la aplicación.

Carpeta programas: Son todos los archivos binarios y librerías necesarias para llevar a producción la aplicación.(Tomcat 7.0, LukeAll)

Carpeta configuración: Carpeta con los dos catálogos y el conjunto de índices generados.

Carpeta Proyecto: Generación binaria del proyecto mg4jvslucene (.WAR).

Carpeta Documentación: Contiene la memoria del proyecto, un fichero Excel con las medidas tomadas para la indexación, y el log de generación de todos los índices, otro fichero Excel con las medidas tomadas para las búsquedas y el archivo de presentación.