CREACIÓN DE UN CRAWLER SEMÁNTICO Y DISTRIBUIBLE PARA SU APLIACIÓN EN UN BUSCADOR WEB

DAVID DEL PINO TOLEDANO
INGIENERIA INFORMÁTICA - UNIVERSIDAD CARLOS III DE MADRID

Tutor: ÁNGEL LAGARES LEMOS **Director: JUAN MIGUEL GÓMEZ BERBIS**

1. ÍNDICE DE CONTENIDOS

1.1. Índice de figuras

1.2. Índice de tablas

2. INTRODUCCIÓN

"Scientia potestas est." - "El conocimiento es poder." (Francis Bacon, *Meditationes Sacrae*, 1957).

A Sir Francis Bacon le fue atribuida esta frase desde que la acuñase en sus meditaciones filosóficas en el siglo XVI. Muchos otros autores han relacionado los conceptos de poder y conocimiento en sus ensayos, dejando entrever la relevancia que tienen en una sociedad cada vez más modernizada. Aunque evidentemente la frase puede ser interpretada en muchos sentidos, la base de su significado radica en que con el conocimiento las habilidades y potenciales de un individuo se incrementan. A su vez, la implicación directa de estas palabras es que quien sea poseedor del conocimiento tendrá ventaja sobre los demás.

En los tiempos actuales, la necesidad de tener a disposición de la sociedad diversas fuentes de información es cada vez más creciente. En un ámbito económico, debido a la velocidad a la que se mueven las finanzas en estos tiempos computarizados, la posesión de información implica estar por delante de la competencia y la diferencia temporal de la capacidad de obtención del conocimiento afecta al equilibrio de la situación de las empresas. Debido a que la cantidad de información es cada vez más amplia, la necesidad de mantener una estructura que permita un acceso rápido y de calidad a la información es cada vez más relevante. Este último aspecto, tiene una implicación directa con el uso de la información para posibilitar el análisis de la estructura de almacenaje y su futura recuperación.

Aplicando lo anteriormente mencionado a un sistema de información podemos establecer tres características importantes que debe ofrecer para garantizar el acceso a la información: cantidad, velocidad y estructura. Estas características están ligadas unas con otras y los desequilibrios en las mismas afectarán a los resultados. Este proyecto intenta realizar una búsqueda de soluciones que permitan elaborar un sistema de información equilibrando estas características mediante el uso de las tecnologías actuales.

3. ESTADO DEL ARTE

3.1. Principios

A lo largo de la historia, las civilizaciones que han dado lugar a la nuestra en la era moderna han ido evolucionando en sus técnicas de salvaguardar sus conocimientos, generalmente a través de la escritura. Desde la era del nacimiento del hombre como ser inteligente capaz de realizar trazos sobre la pared de una caverna para plasmar una cacería de bisontes. Más adelante, las épocas más antiguas como en el tiempo de las dinastías faraónicas egipcias, la utilización de escritura pictórica y jeroglíficos fueron la base para acuñar el conocimiento de la civilización y los murales de los templos que perduran hasta nuestros días. Las técnicas de plasmar el conocimiento en papel y otros soportes han ido evolucionando durante los siglos mediante la investigación de nuevos métodos como la imprenta hasta el almacenamiento digital.

El uso de la escritura y el consecuente aumento de la cantidad de obras literarias generadas llevó a la necesidad de la catalogación de las mismas y a la creación de colecciones en bibliotecas públicas o privadas. Los primeros lugares donde se podía acceder a la información contenida en estas colecciones fueron concebidos en el imperio romano donde pergaminos y manuscritos se almacenaban en lugares secos para su posterior uso a través de ciertas ciudades del imperio. La primera biblioteca pública conocida fue el London Guildhall establecida en la capital del imperio británico hacia el 1425 por Lord Mayor Dick Whittington [1]. A lo largo de los siglos venideros otras ciudades se unieron a la causa estableciendo la legislación que actualmente hoy conocemos en cuanto al acceso de la información.

3.2. La información en la tecnología y su organización

El mundo moderno está cada día más orientado al tratamiento y el uso de la información, no solo de los conocimientos científicos sino también los relacionados con el ser humano, la sociedad y sus interrelaciones. Las tecnologías de la información, en muchas ocasiones están orientadas al almacenaje de información para salvaguardar el conocimiento obtenido y previsiblemente utilizarlo en un futuro para su beneficio.

Si en la antigüedad la mejor manera de almacenar el contenido de la información era catalogar los manuscritos para su futura consulta de una manera ordenada y sencilla, actualmente existen sistemas de información capaces de estructurarla y ofrecernos una respuesta rápida de los documentos que se deseen consultar. Esto se consigue mediante la utilización de una catalogación similar de los contenidos para su posterior consulta a través de lo que se denomina comúnmente Buscador.

Un buscador es un sistema que permite la obtención de resultados de documentos aunque por lo general conocemos este término de una manera más aproximada como un buscador de documentos web que basa su arquitectura en la aplicación de la misma filosofía para documentos contenidos en la red. Sin embargo, existen otras muchas aplicaciones a las búsquedas de documentos.

Un claro ejemplo es la utilización de almacenes automáticos en la industria general para grandes y pequeñas piezas que deben ser categorizadas según su forma, color o material. Estas piezas tienen que ser almacenadas en grandes superficies, pero por lo general se busca que el espacio sea el más pequeño para ganar en eficiencia. Por otro lado, es muy importante poder encontrar una pieza en concreto entre todas las miles de piezas que haya almacenadas. Para ello, se utilizan sistemas de información sobre almacenes automáticos que devuelven las piezas necesarias a los operarios cuando éstos las necesitan. Los beneficios obtenidos por las empresas que implantan este tipo de sistemas informatizados de búsqueda de piezas almacenadas es bastante amplio y permite además ahorrar en tiempo de producción y espacio necesario de almacenaje. La empresa americana Kardex Remstar ofrece sistemas de almacenaje a terceros. Entre sus clientes, la empresa Frydenbø Bilsenter localizada en Bergen, una distribuidora de piezas de automóviles europeos ha obtenido un aumento en los ingresos de 1 millón de euros netos, unido a un ahorro de en la producción y el transporte anual de 200.000 euros. Además, la mayor eficiencia en la distribución de las piezas debido a que un solo

trabajador rinde más en realizar su actividad laboral arroja unos ahorros anuales de 100.000 euros. Por otra parte, la empresa ha pasado de tener un espacio de almacenaje de cerca de 130 m² a tan solo 6 m². [2]

Un ámbito más conocido por todos y que seguramente usamos diariamente es el de los buscadores web que permiten al usuario consultar una serie de documentos relevantes para una búsqueda introducida. Los documentos relevantes son extraídos a través de ciertos algoritmos y cálculos realizados previamente sobre ellos, es decir, una catalogación de los documentos bajo ciertos criterios. De la misma manera que los sistemas de almacenaje para ser precisos necesitan que las piezas estén previamente catalogadas y la información sea introducida en el sistema, los algoritmos que ofrecen documentos relevantes en un sistema de búsqueda web necesita realizar un análisis de éstos documentos. Este proceso se basa en la generación de un índice de relevancia entre los documentos para los términos incluidos en los mismos. Generalmente el análisis realizado para generar este índice es realizado por el motor de búsqueda pero la recuperación de la información es realizada por un web crawler o araña web.

3.3. Crawlers

La World Wide Web ha crecido desde unos pocos miles de páginas en 1993 hasta más de veinte billones de páginas en el último año según los documentos indexados por los buscadores Google y Bing.

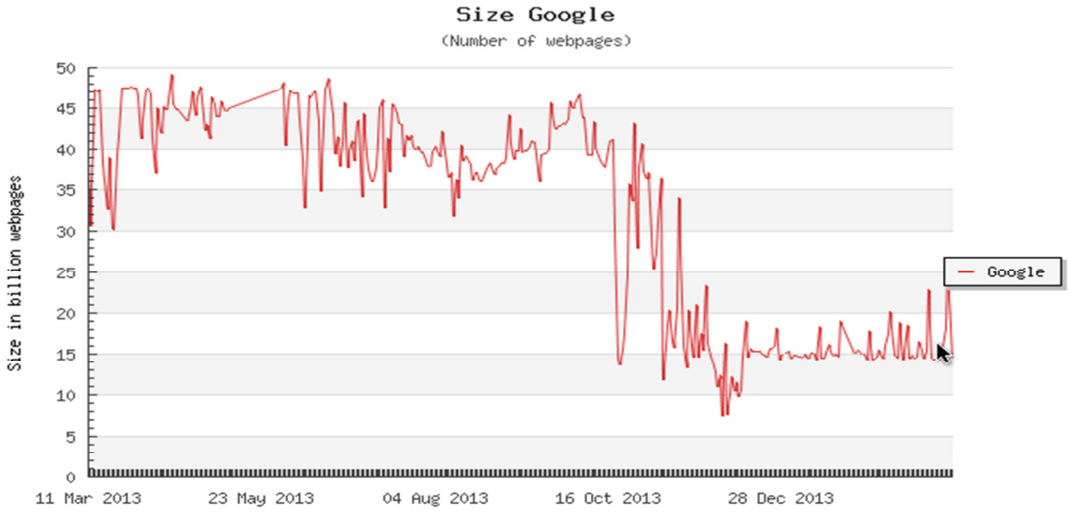

Figura 1 - Documentos indexados por Google en billones durante el año 2013

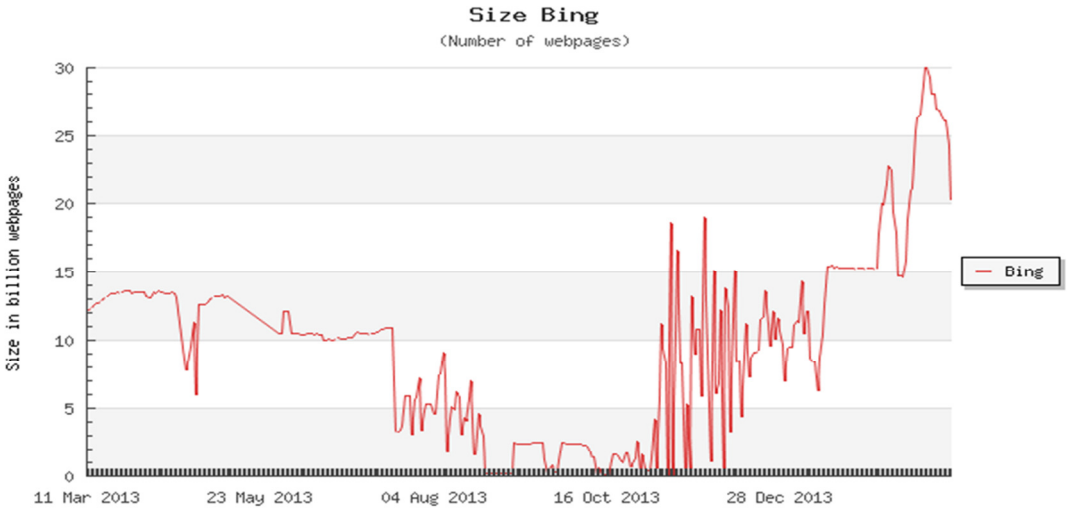

Figura 2 - Documentos indexados por Bing en billones durante el año 2013

Debido al aumento considerable en el número de documentos web que pueden ser consultados desde todo el mundo es indispensable poder acceder a sistemas que permitan la búsqueda de cierto contenido que para el usuario sea relevante. Estos sistemas son los llamados motores de búsqueda que se basan en coleccionar documentos y clasificarlos según lo que contengan. Para obtener la información de los contenidos alojados a lo largo de la red es necesario el uso de sistemas que permitan dicho propósito, estos sistemas se denominan web crawler o araña web. [3]

Un crawler es un sistema que realiza un análisis de un documento web y extrae la información en cuanto a contenido se refiere pero también extrae información del meta-lenguaje, en este caso HTML. Dicha información contiene hipervínculos que enlazan a otros documentos web. El crawler organizará estos hipervínculos e irá visitándolos de manera secuencial, según ciertos criterios, para continuar obteniendo contenido que nutra el índice para las futuras búsquedas.

Existen multitud de técnicas y estrategias con las que dotar a un crawler aunque la base sea la misma. Estas distintas estrategias permiten abordar el problema desde distintos puntos de vista:

Búsqueda en anchura: Si se busca construir un índice de documentos muy grande, los crawlers comienzan su ejecución sobre un conjunto pequeño de páginas siguiendo en cada una de ellas los enlaces mediante ciertas heurísticas. Algunas estrategias implican la visita estricta de cada uno de los hipervínculos contenidos en el documento y otras utilizan estrategias más complejas como por ejemplo la consulta de las páginas más importantes primero.

Actualización de los contenidos: Tras visitar cada una de las páginas se establecen criterios para consultar de nuevo el documento en busca de posibles actualizaciones en el contenido. En caso de que el contenido cambie en desde la anterior consulta, la información indizada, el contenido almacenado y en definitiva la relación de los elementos incluidos en el documento habrá cambiado al igual que su relevancia. Por estas razones, es necesario establecer políticas para volver a extraer la información de las páginas visitadas anteriormente. Existen multitud de estrategias para afrontar este problema y mantener un índice de búsqueda actualizado.

Recuperación de información selectiva: Ciertos crawlers se concentran en la recuperación de documentos con contenido de cierto tipo con el fin de extraer la información que se adecue mejor al objetivo final de los documentos que se van a buscar. Otros crawlers realizan una recuperación selectiva en función del lenguaje en el que esté escrito el contenido del documento. Por lo general, la recuperación de estos contenidos se hace consultando la información meta del documento a recuperar y se introducirá en el índice si satisface los criterios previamente decididos.

Recuperación de información oculta: Multitud de páginas en la red están ocultas o enmascaradas por sistemas introducidos en los propios documentos. Esto ocurre generalmente en páginas web donde es necesaria una sesión válida de usuario, certificaciones para poder acceder al contenido o la página contiene formularios que ofrecerán cierto contenido en función de los datos introducidos. En estos casos, la estrategia utilizada por los crawlers que buscan introducir en su sistema de relevancia estos contenidos ocultos es la interacción directa de estos formularios.

Como vemos, independientemente de las estrategias a seguir dentro del funcionamiento del crawler, éstos se caracterizan por contener dos componentes fundamentales:

- Uno o varios servicios que mantengan una cola de documentos a los que se irá accediendo, lo que se denomina araña y se encargará de ofrecer uno o varios URLs de documentos que otro servicio se encargará de procesar.
- Un segundo servicio que procesará las URLs de los documentos ofrecidos por el anterior servicio, descargando su contenido y lo ofrecerá al motor de búsqueda para su procesamiento.

El primer crawler público fue RBSE [4] y estaba basado en esta arquitectura. Por lo general la arquitectura de estos servicios debe ser un sistema adaptable y escalable en todos los sentidos, es

decir, que pueda ser estructurado en varios procesos que funcionen paralelamente e incluso de manera distribuida.

Las estrategias de funcionamiento del crawler anteriormente enumeradas, están integradas en el funcionamiento de los servicios especificados en los puntos anteriores o bien están implementadas en servicios alternativos que gestionan su funcionamiento. Por lo general, la arquitectura de un crawler intenta aislar de manera sistemática los procesos de tal manera que no sea necesaria una comunicación expresa ni la utilización de una gran cantidad de información por cada uno de ellos.

Además de los distintos criterios que puedan ser utilizados a la hora de recuperar la información, es necesario tener en cuenta que los documentos no tienen por qué estar donde ha apuntado un hipervínculo. En estos casos, se establece una relación entre el documento en cuestión con una serie de parámetros que en ocasiones sirve para penalizar a quien ofrece dicha información. De esta manera, se establecen posibles servicios alternativos que permitan categorizar las fuentes de documentos y valorarlas frente al resto antes de orientar los servicios para recuperar sus contenidos. Esto puede ser igualmente aplicado a la hora de recuperar información actualizada y los motores de búsqueda existentes en el mercado actualmente tienen en cuenta estos factores a la hora de ofrecer la relevancia de los documentos ante una búsqueda.

Un aspecto de fondo e importante es tener en cuenta las posibles contingencias que deban tener lugar ante el nivel de servicio ya no solo de nuestro servicio sino de las fuentes del contenido. Es necesario seguir los estándares establecidos y adoptados a lo largo de la red en cuanto a la exclusión de robots se refiere. Esto suele establecerse en un fichero alojado en el servidor del cual se vaya a recuperar información (robots.txt) que introduce políticas de acceso y denegación de acceso a ciertos directorios y contenidos donde un robot no debería acceder.

Del mismo modo que en lo establecido anteriormente, es necesario tener en cuenta cierto tiempo de espera entre las ordenes de recuperación de la información contra los servidores para no afectar al funcionamiento de los mismos. En ciertos casos, algunos servidores consultados que no sean de la propiedad del desarrollador del crawler, puede decantar en una denegación de acceso a todo el servidor durante un periodo de tiempo o de manera indefinida.

Por lo general, los servidores saben cuándo un crawler ha pasado por ellos debido a que introducen en las cabeceras del protocolo de conexión ciertos valores que lo identifican, este parámetro se denomina User Agent. En este valor se incluye además información de dónde procede el Bot y en ocasiones que estándar o versión utiliza e incluso los lugares donde poder consultar información sobre qué es y sus funcionalidades. Algunos de los crawlers existentes utilizan los siguientes User Agents para sus robots [5] :

- **Bingbot**: Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)
- **Googlebot**: Googlebot/2.1 (+http://www.google.com/bot.html)
- **FAST-WebCrawler**: FAST-WebCrawler/3.8 (atw-crawler at fast dot no; http://fast.no/support/crawler.asp)
- **Altavista robot**: AltaVista V2.0B crawler@evreka.com

3.4. Motores de búsqueda

Una vez recuperada la información, el contenido de los documentos dispersos por el conjunto recuperable debe ser analizado, almacenado y organizado.

3.4.1. Análisis de la información

A la hora de hacer el análisis de la información recuperada hay que tener en cuenta ciertos aspectos que van a tener relación directa con el resultado final que se obtengan en el proceso.

Es necesario entender que el análisis a realizar se realizará sobre millones de documentos, ya sean páginas web, imágenes o otro tipo de información. En cualquiera de los casos, el análisis a realizar no

debe confiar en que el formato de la fuente es el correcto, por ejemplo, nos podemos encontrar sitios web que no cumplan los estándares o tengan una estructura incorrecta y esto afectará directamente al análisis a realizar. Una de las posibles soluciones es ignorar estos documentos o incluso los fragmentos que no sean aceptables dentro de ciertos criterios.

Los motores de búsqueda existentes en el mercado, por lo general, intentan solventar los posibles errores en los fragmentos del contenido con el fin de continuar capturando información, pero no siempre es posible.

En otros casos, lo más lógico es utilizar estándares aplicados a la información que se intenta recuperar y si se da la situación en la que los estándares no se cumplen se ignoran los posibles resultados.

3.4.2. Relevancia del contenido

Una vez que la información es analizada se puede proponer distintas estrategias para analizar el contenido como tal y comenzar a aplicar diversas funciones computacionales para aprovecharlo. En ocasiones, ciertos fragmentos del contenido pueden tener más relevancia que otros o ciertos términos pueden tener una implicación directa con la información que se muestra.

Supongamos el caso de un documento en el que se muestre la información biográfica de un personaje histórico. En este caso, los contenidos más relevantes para encontrar dicho documento seguramente tengan relación directa tanto con el nombre del personaje histórico en sí como otros términos que lo caractericen.

Para realizar el análisis de la relevancia del contenido es necesario aplicar distintos algoritmos que permitan conocer la importancia de los términos en el documento a analizar. Estos algoritmos pueden tener un funcionamiento muy básico hasta complementarse con técnicas de análisis semántico de los términos y obtener relaciones entre los mismos.

El funcionamiento más básico de un la captura de relevancia del contenido se basa en la realización de un análisis de frecuencia de los términos diseminados por el texto, es decir, el número de repeticiones de cada término en el documento.

Estas técnicas suelen aplicarse teniendo en cuenta que algunos términos son muy usados en ciertos idiomas y atendiendo al léxico de los textos, pueden no ofrecer gran cantidad de información relevante sobre los documentos. En estos casos, dichos términos suelen ignorarse en los documentos o no ofrecer valores de relevancia para los mismos aunque puedan analizarse mediante otras técnicas.

Por lo general se aplican ciertos algoritmos matemáticos que se resumen en algunas ecuaciones que son aplicadas sobre el documento para obtener la relevancia de los términos. Sin embargo, estos algoritmos, su complejidad y las heurísticas que se utilizan en cada motor de búsqueda no tienen por qué ser iguales en los distintos productos existentes en el mercado. Cada motor puede nutrirse de las funcionalidades de otros productos ya desarrollados ya sean de código propietario o abierto y en otros casos desarrollar sus propias estrategias para abordar el problema. Por esta razón no tiene sentido de hablar de ecuaciones que permitan obtener la relevancia de un término en concreto dentro de un documento, aunque sí podemos tener una visión periférica de las posibles soluciones atendiendo a ejemplos generales. Para ello, veamos los siguientes ejemplos de motores de búsqueda:

- **Apache Lucence** [6]: Se trata del motor de búsqueda que sirve de núcleo para Apache Sorl, una suite que sirve de capa de abstracción para el propio motor de búsqueda. El motor, Lucene, es un sistema que permite generar un índice de relevancia de términos con una gran eficacia y eficiencia debido a que permite una escalabilidad y alto rendimiento sin utilizar demasiados recursos. Utiliza algoritmos muy eficaces que permiten generar índices múltiples de manera simultánea. Además, ofrece funcionalidades para la búsqueda de documentos en función de los términos incluidos en los mismos, ordenando los resultados según la relevancia de dichos términos en los documentos del resultado.

- **Okapi BM25** [7]: Es una implementación de funciones de cálculo de relevancia usadas por diversos motores de búsqueda para obtener la relevancia de los términos en un documento. Existen distintas versiones de la función en la que se ven afectados los parámetros que intervienen para conseguir distintos resultados de relevancia. La más importante de todas estas versiones es BM25F, una modificación del algoritmo original que permite diferenciar partes de la estructura del documento con el fin de asociar distintos grados de importancia. Además, permite también la ejecución de búsquedas sobre la colección de documentos indexada, ofreciendo igualmente un orden según la relevancia de dichos documentos para los términos que se deseen buscar.

Estos motores, utilizan distintos algoritmos para manejar la relevancia de los términos en los documentos para ofrecerlos de manera ordenada ante una búsqueda. A continuación se muestran los índices más representativos de manera teórica [8] para la recuperación de información relevante de una colección de documentos y su versión en los motores de búsqueda referenciados anteriormente:

- **Frecuencia booleana**: Si el término t tiene una o más ocurrencias en el documento d, la frecuencia booleana es 1.

$$tf(t,d) = 1$$

- **Frecuencia escalada**: una orientación logarítmica del anterior:

$$tf(t,d) = 1 + \log\, f(t,d)$$

  $\text{donde } f(t,d)$ es la frecuencia de un término en un documento. Siendo así, la frecuencia escalada será 0 si la frecuencia del término es nula.

- **Frecuencia normalizada de un término (tf o en ocasiones también denominada score)**: indica con qué frecuencia aparece un término en un documento. Cuanto más veces aparezca un término en un documento mayor es su relevancia. De esta manera, los documentos que contengan los términos de la búsqueda con mayor frecuencia tendrán más relevancia.
  - La aproximación teórica:

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d) \;:\; w \in d\}}$$

  - La versión de Lucene: implementa el cálculo de este parámetro como:

$$tf(t,d) = \sqrt{f(t,d)}$$

  - Por su parte la versión clásica de BM25:

$$R(q,d) = \sum_{t \in q} \frac{f(t,d)}{k\,((1-b) + b\,\frac{l(d)}{l_{AVG}} + f(t,d)}$$

    Donde $l(d)$ es la longitud del documento, l_{AVG} es la longitud media de los documentos de la colección, k es una constante arbitraria y b es un valor entre [0,1], usualmente 0,75.

- En el caso de la versión BM25F: Se obtiene primeramente el valor del peso del término de entre todos los campos siguiendo:

$$peso(t,d) = \sum_{c\ en\ d} \frac{f(t,d,c) \cdot boost_c}{((1-b_c) + b_c \cdot \frac{l_c}{avg(l_c)}}$$

Donde $boost_c$ es un factor de potenciación para el campo c. Se aplica después una saturación no lineal sobre el peso calculado:

$$R(q,d) = \sum_{t \in q} \frac{peso(t,d)}{k + peso(t,d)} \cdot idf(t)$$

Donde $idf(t)$ es la frecuencia inversa que veremos a continuación.

- **Frecuencia inversa del documento (idf)**: indica si un término es común o no en el conjunto de la colección de documentos:
  - La aproximación teórica:

$$idf(t,d) = \log \frac{|D|}{|\{d\ \in D : t\ \in d\}|}$$

Donde $|D|$ es el número de documentos de la colección y $|\{d\ \in D : t\ \in d\}|$ es el número de documentos de la colección donde aparece el término.

  - La versión de Lucence:

$$idf(t,d) = \log \frac{|D|}{|\{d\ \in D : t\ \in d\}|} + 1$$

  - La versión de BM25 y BM25F:

$$idf(t,d) = \log \frac{|D| + 0{,}5 - |\{d\ \in D : t\ \in d\}|}{|\{d\ \in D : t\ \in d\}| + 0{,}5}$$

3.4.3. Algoritmos de análisis de enlaces

En el ámbito de internet y las páginas web, muchos motores de búsqueda aplican otras técnicas para conocer la relevancia de los contenidos de un documento. A diferencia de los algoritmos de búsqueda de relevancia de los contenidos en sí, se basan en la calidad de la información que los documentos ofrecen y cómo otros documentos los citan o enlazan su información.

Durante años, las estrategias de estos algoritmos han ido evolucionando para evitar que el conocimiento de los mismos pueda ser usado de manera fraudulenta por los creadores de los contenidos. Teniendo en cuenta las distintas aproximaciones de estos algoritmos se pueden realizar ajustes y modificaciones sobre los contenidos con el fin de que sean referenciados desde otros documentos externos y así conseguir beneficios en cuanto a la relevancia de los mismos.

A la hora de realizar una búsqueda en un conjunto de documentos, una vez obtenido el listado de documentos ordenados por su relevancia frente a los términos indicados en la búsqueda, este orden puede ser modificado según algunas pautas que se quieran tener en cuenta.

El planteamiento inicial de la idea de estos algoritmos es que los documentos en la red más referenciados seguramente tendrían información con más calidad y su contenido sería más relevante. Esto significa que las páginas web a las cuales apunten más hipervínculos desde otras páginas web tendrán mayor relevancia que el resto, teniendo en cuenta el resultado de la relevancia inicial extraído desde un motor de búsqueda.

Esta estrategia se ha aplicado a ciertos motores de búsqueda y aunque en la teoría la idea parezca tener muchos sentido, en la práctica puede verse afectada por un uso inadecuado del algoritmo. La razón fundamental de que estas estrategias no son siempre adecuadas es que cierta empresa puede estar interesada en beneficiar a si misma o a terceros por medio de la creación de enlaces de manera discriminada contra documentos alojados en sus servidores. Esto podemos verlo más adelante con el ejemplo de PageRank y cómo el algoritmo se ha ido evolucionando con el tiempo.

Algunos de los algoritmos más conocidos son:

- **HITS** [9]: Es uno de los algoritmos precursores de estos métodos y sobre todo de PageRank, que se comentará a continuación. HITS también es el acrónimo en inglés de Hypertext Induced Topic Selection y es un algoritmo creado por Jon Kleinberg para valorar y clasificar la importancia de una página web. El algoritmo también es conocido como "hubs & authorities", núcleos y autoridades. Se puede describir de la siguiente manera:
    - La autoridad (authority) ofrece el valor de importancia de una página mediante la suma de los valores de los núcleos (hubs) que apuntan a dicha página.
    - El núcleo indica (hub) indica la calidad de la información que se consigue siguiendo los enlaces a otras páginas y se calcula mediante la suma de los valores de autoridad (authority) de las páginas a las que ésta apunta.
    - El algoritmo utiliza un método repetitivo ya que el cálculo de un valor afecta al opuesto y su resultado puede ir variando durante el tiempo en el conjunto de documentos.
    - HITS aplica el algoritmo de análisis de enlaces en el momento de ejecución de una búsqueda debido a que se ve afectado por la relevancia de los términos introducidos en la misma. Esto tiene un impacto directo contra la eficiencia del motor de búsqueda.
    - Se establece el algoritmo para ser usado en un subconjunto de documentos y no sobre todo el conjunto completo.
- **PageRank** [10]: Es posiblemente el algoritmo de análisis de enlaces más conocido debido a que el motor de búsqueda que lo llevó a su utilización fue el creado por la empresa Google Inc. Su funcionamiento es prácticamente igual que HITS con algunos cambios fundamentales:
    - El valor final de una página no corresponde a dos valores como en HITS, corresponde a un índice de 0 a 10 en el que se indica, cuanto mayor sea el valor, mayor relevancia tendrá dicha página.
    - El algoritmo se ejecuta en el momento de la indexación por el motor de búsqueda, de esta manera la experiencia del usuario es mejor ya que la ejecución de las búsquedas es más eficiente. Además, se asegura que cuando una página se analizada de nuevo por el motor de búsqueda, su PageRank se volverá a calcular.

3.4.4. Arquitectura de los motores de búsqueda

Es importante que el motor de búsqueda sea flexible y permita una escalabilidad adecuada. Generalmente, estos sistemas están funcionando en máquinas que permitan un acceso rápido al almacenamiento ya que es crucial tener una alta velocidad para mantener la información actualizada. Además, las estructuras de información almacenadas por los motores de búsqueda son muy grandes y el crecimiento es exponencial.

3.5. Sistemas de recuperación de información en la actualidad

A continuación se enumeran algunos de los sistemas de recuperación existentes actualmente en el mercado. Por lo general, la mayoría integran crawlers y motores de indexación de las búsquedas desarrollados por las empresas que gestionan su funcionamiento y arquitectura. En otros casos, algunos de los servicios son realizados por software de terceros o de código abierto. Por otro lado, muchas empresas no tienen una dedicación fundamental en cuanto a ofrecer servicios de búsquedas a sus clientes. Sin embargo, debido a que su dedicación es ofrecer contenidos o documentos sí que precisan ofrecer servicios de búsqueda sobre los mismos a sus clientes o usuarios. Estos casos suelen darse en medios de información, para los cuales se aplican servicios a medida o productos ya implementados que aplicar sobre los contenidos ya existentes. La lista a continuación incluye sistemas de recuperación de información que incluyen ambos tipos de productos:

- **Comerciales**:
    - **Googlebot** [11]: Posiblemente uno de los crawlers más conocidos debido a que el uso del buscador que lo gestiona es el más utilizado a través de todo el mundo. Aunque no se conoce la manera en la que está implementado actualmente, sí se conoce cómo fue su origen. Se basa en una arquitectura donde el crawler está integrado con el sistema de indexación de los contenidos porque el análisis de los mismos implica también la extracción de los hipervínculos. Durante el análisis, los listados de URLs son enviados a un servicio de planificación del propio crawler, que gestiona si una URL debe ser visitada y cuando debe hacerse.
    - **WebCrawler** [12]: Crawler orientado a generar el primer índice de texto en un subconjunto de páginas de la red. Basa el funcionamiento del crawler en una librería de funciones externa al desarrollo central del motor de búsqueda, así como el sistema de analisis de los documentos recuperados.
    - **IDOL (Autonomy)** [13]: Permite la extracción de documentos dentro de un conjunto seleccionado con la particularidad de cubrir más de 1.000 tipos distintos de información. Es independiente del lenguaje y es usado por grandes empresas, generalmente orientada a los contenidos como los medios de información. Sus servicios, aunque no exclusivamente, están orientados a ofrecer una gestión de los documentos generados por el CMS Autonomy, distribuido por la misma empresa que IDOL, Hewlett Packard.
- **De código abierto**:
    - **Scrapy** [14]: Algo distinto a lo descrito anteriormente, Scrapy es un framework que permite la extracción de documentos y sus contenidos desde un sitio web concreto. Mediante un crawler implementado en Python, se puede obtener la información de un conjunto de páginas web para las cuales no está implementado un API donde recoger los contenidos. Es un framework de código abierto y los desarrolladores pueden participar activamente en la implementación del software.
    - **Sphinx** [15]: Es un motor de búsqueda de código abierto utilizado para la indexación de información almacenada en una base de datos o ficheros XML. Permite la integración con APIs de información en distintos tipos de codificaciones

de caracteres y basa la ordenación de los resultados en la relevancia de los contenidos bajo factores adicionales incluidos en estándares internacionales.

3.6. Un paso más, la búsqueda semántica

Como hemos visto, las estrategias de recuperación de información relevante se ajustan por lo general al cálculo de la relevancia de los términos contenidos en los documentos y textos. Sin embargo, la tendencia actual en el mercado de los motores de búsqueda y sobre todo los orientados a grandes conjuntos de información como los buscadores de páginas web, incluyen heurísticas para el análisis semántico de estos términos tanto en la indexación de documentos como a la hora de hacer una búsqueda. Estas nuevas estrategias establecen por lo general relaciones entre los contenidos de un documento para compararlo o averiguar las implicaciones que tienen con otras terminologías o documentos de información similar.

La base de la búsqueda semántica es atribuir valores de relevancia a la información que está detrás del propio término de búsqueda pero que está implícita en su significado o en su relación con otros términos con el fin de ofrecer resultados más óptimos.

Para que un motor de búsqueda sea capaz de ofrecer un análisis semántico de los documentos, es necesario que implemente una serie de estrategias:

- **Análisis morfológico**: Mediante el análisis morfológico de los textos de un documentos se intenta afrontar el problema mediante una heurística propia del lenguaje. Para ello, el motor de búsqueda debe ser capaz de comprender los lenguajes de los documentos que recupera y posteriormente analizar la sintaxis y la morfología de sus textos. De esta manera, el motor de búsqueda sería capaz de establecer relaciones entre las acciones (verbos), su posición en el tiempo (tiempos verbales), los elementos que realizan una acción (sujeto y predicado en la sintaxis de una frase) y aquellas implicaciones o elementos que intervienen en la acción (complementos directos, indirectos y circunstanciales).

  Como se comentaba, para que esto pueda tener sentido y el motor de búsqueda sea capaz de realizar este tipo de análisis debe ser capaz de situarse el mismo nivel que los documentos que analiza. Los motores de búsqueda deben especializar sus implementaciones para conseguir trabajar sobre textos de diferentes idiomas si su misión es analizar un grupo de documentos multilingüe, si por el contrario, se centra sobre un subconjunto de documentos que está centrado en un idioma en concreto, bastaría con que contase con la capacidad de analizar la morfología de ese lenguaje.

- **Descripciones y generalizaciones**: Dentro de todos los lenguajes se establecen relaciones entre los elementos de una frase, por lo general nombres, y algunos atributos que lo describen de una manera más concreta. De esta manera, un motor de búsqueda, mediante un análisis de las frases contenidas en un documento, sería capaz de establecer relaciones entre ciertos términos del documento y otros que les ofrezcan más significado. En cualquiera de los casos, sería necesario un análisis morfosintáctico para obtener este capacidad y estaría centrada dentro del idioma que morfológicamente entienda el motor de búsqueda.

- **Sinónimos y significados similares**: Un gran reto para este tipo de motores es tener la capacidad de establecer relaciones entre los documentos y términos que aunque no aparezcan en los mismos si que pueden tener relación debido a su similitud en el significado.

  Las técnicas más sencillas de este tipo de análisis de sinónimos se basan en la capitulación de términos y agruparlos mediante su significado en bancos de información que puedan ser usados tanto en el análisis de los documentos como en el momento de establecer una búsqueda contra la información recopilada. Además de esto, mediante técnicas más sofisticadas se puede atender al análisis de los documentos para que el propio motor de

búsqueda establezca significados coherentes entre los términos. Mediante el funcionamiento rutinario del motor de búsqueda estos sinónimos irán apareciendo en los análisis haciendo que el sistema aprenda por si mismo.

Como hemos visto, los análisis semánticos que pueda hacer un motor de búsqueda sobre un conjunto de documentos van a depender del lenguaje en el que estén escritos. Sin tener en cuenta que la ortografía sea la correcta o no, un motor de búsqueda debería ser capaz de catalogar primeramente la morfología de las frases de los textos de un documento para poder analizarlo semánticamente.

Algunos motores de búsqueda se centran en ciertos ámbitos más cerrados, ya no solo en un lenguaje en concreto, sino en contextos relacionados con la ciencia, el ámbito de los documentos legales, etc.

4. ALCANCE Y OBJETIVOS

4.1. Alcance

El sistema a diseñar y desarrollar es un crawler que permita la recuperación de documentos alojados en servidores de internet con el fin de generar un índice de documentos, para los cuales:

- Se generará un índice de relevancia de los términos contenidos en los documentos,
- Se realizará un análisis semántico de los textos encontrados en los documentos
- Se obtendrán resultados de los datos almacenados en función de la relevancia y relaciones semánticas de los contenidos según la consulta realizada.

A su vez, el sistema deberá permitir su escalabilidad futura con el fin de mejorar el servicio de consulta de la colección de documentos así como la posibilidad de aumentar las funcionalidades existentes. Para ello el sistema dispondrá de un control de acceso a recursos compartidos a los cuales accederán ciertos procesos que realizarán el análisis de los documentos recuperados de manera ordenada.

Por otro lado, el sistema deberá estar disponible para su funcionamiento en una plataforma que permita el acceso concurrente de un número alto de usuarios de manera simultánea. De esta manera, se establecerá una política de ahorro de recursos así como la búsqueda de la eficiencia a la hora de realizar el diseño del sistema y de la plataforma.

4.2. Objetivos del proyecto

Los objetivos del presente proyecto son los siguientes:

- Análisis e implementación de un crawler que permita obtener la información que reside en documentos en páginas web.
- Implementación de un motor de búsqueda que genere un índice de documentos con el fin de poder ser consultados según su relevancia en una búsqueda.
- Generación de un buscador que permita realizar consultas contra el motor de búsqueda.
- Integración de funcionalidades semánticas dentro de las partes del sistema con el fin de interpretar búsquedas contra documentos elaborados en castellano.

Para realizar los objetivos expuestos, será necesario hacer uso de tecnologías orientadas a la web, y además enfocar el desarrollo del sistema de tal manera que permita flexibilizar el uso de los recursos. Además, se hace hincapié en la necesidad de que el sistema deba ser escalable, por lo que la arquitectura del sistema se basará en procesos que puedan interactuar entre ellos a través de recursos compartidos y puedan ser distribuidos en distintas máquinas.

5. ANÁLISIS DE LA ARQUITECTURA DEL SISTEMA

En el siguiente apartado se realiza un análisis de la arquitectura que tendrá el sistema. Dicho análisis será efectuado a partir de los requisitos que el sistema debe cumplir tanto a nivel interno y de funcionalidades como la abstracción de los mismos para ofrecérselos al usuario final.

De esta manera, para concluir en un análisis final de la arquitectura global del sistema pasaremos analizando los requisitos que

5.1. Requisitos de usuario

A continuación se muestra una tabla que indica los requisitos de usuario que han sido extraidos durante el análisis del proyecto. Sobre estos requisitos se elaborará un análisis que dará como resultado un listado similar de requisitos de sistema para los cuales ciertos componentes de sistema deberán cumplir las funcionalidades apropiadas.

CÓDIGO	NOMBRE	DESCIPCIÓN	FUENTE	PRIORIDAD
UR01	Consultas de búsqueda	El usuario debe ser capaz de realizar una consulta contra una base de conocimiento de contenidos web a través de una serie de términos introducidos en una cuadro de dialogo ofrecido por la interfaz del sistema.	Análisis	Alta
UR02	Resultados de la consulta	El usuario obtendrá a la hora de hacer una consulta una serie de resultados que puede contener tanto documentos web como imágenes.	Análisis	Alta
UR03	Entidades en los resultados	Los resultados mostrarán cuando proceda, información de las entidades relevantes que tengan relación con la consulta realizada.	Análisis	Alta
UR04	Relaciones semánticas	El usuario obtendrá asociaciones semánticas de los resultados obtenidos.	Análisis	Alta
UR05	Lenguaje de las búsquedas	El usuario enfocará consultas contra el sistema en el idioma español, esperando obtener resultados del mismo lenguaje sin descartar otros relevantes en idiomas alternativos	Análisis	Alta
UR06	Eficiencia de las búsquedas	Los resultados se obtendrán de una manera rápida al realizar una búsqueda, con un tiempo de espera menor de cinco segundos	Análisis	Alta

Tabla 1 - Requisitos de usuario

5.2. Requisitos de sistema

A continuación se describen los requisitos de sistema provenientes de los requisitos de usuario analizados. Para todo requisito de sistema se establece la relación con el requisito de usuario establecido y su tipología.

CÓDIGO	NOMBRE	DESCIPCIÓN	FUENTE	TIPO
SR01	Recuperación de contenidos	El sistema debe ser capaz de recuperar contenidos desde servidores de sitios web	UR01, UR02, UR03, UR04	Funcional
SR02	Procesamiento de los contenidos	Los contenidos recuperados serán analizados a nivel de términos, entidades y semánticamente	UR01, UR02, UR03, UR04	Funcional
SR03	Análisis de términos	El sistema recuperará los términos contenidos en un documento y ofrecerá una valoración para los mismos según su relevancia	UR01, UR02	Funcional
SR04	Análisis de entidades	El sistema recuperará las entidades contenidas en un documento y ofrecerá una valoración para las mismas según su relevancia	UR01, UR03, UR04	Funcional
SR05	Indexación de resultados	El sistema debe ser capaz de ordenar por relevancia los términos contenidos en un documento y relacionarlo con el mismo	UR01, UR02, UR03	Funcional
SR06	Indexación de entidades	El sistema debe ser capaz de ordenar por relevancia la relación de las entidades contenidas en un documento	UR01, UR03, UR04	Funcional
SR07	Generación de nuevas entidades	El sistema debe ser capaz de introducir nuevas entidades en la estructura de información existente	UR03	Funcional
SR08	Análisis semántico de los contenidos	El sistema debe ser capaz de encontrar relaciones semánticas entre los contenidos analizados	UR03	Funcional
SR09	Salida del formulario buscador	El buscador tendrá una salida web en la que mostrará un formulario simple con un cuadro donde introducir los términos de búsqueda y un botón para realizar la ejecución de la misma	UR01	Funcional
SR10	Salida de resultados de la búsqueda	El resultado de una búsqueda mostrará un listado de los documentos relevantes que tengan relación con los términos de la búsqueda. El orden del listado será de más relevante a menos	UR02	Funcional

SR11	Paginación de resultados	Los listados de resultados serán paginados para no introducir numerosos resultados en una sola página y que el usuario pueda navegar por las distintas páginas de documentos relevantes	UR02	Funcional
SR12	Salida de resultados de imágenes	Las imágenes relevantes para los términos de búsqueda se mostrarán en la zona superior de la página de resultados	UR02	Funcional
SR13	Imágenes asociadas a un resultado	Las imágenes relacionadas con un documento relevante encontrado y que se muestra en el listado se mostrará junto a la información del mismo	UR02	Funcional
SR14	Lenguaje	El sistema se enfocará en documentos del lenguaje español ignorando, si es posible, documentos o contenidos en otros idiomas	UR05	No Funcional
SR15	Eficiencia	El sistema debe ser eficiente tanto en la recuperación de la información, su procesamiento y sobretodo en la recuperación de los resultados que espera el usuario	UR06	No Funcional
SR16	Escalabilidad	El sistema debe ser escalable y permitir la interacción de los distintos procesos en sistemas que permitan su ejecución en paralelo	UR06	No Funcional
SR17	Robustez de la estructura de la información	La estructura de la información debe ser robusta para que la información almacenada en el sistema sea fiable permitiendo introducción de nuevos datos, modificación y eliminación de los mismos	UR06	No Funcional
SR18	Flexibilidad de la estructura de la información	La estructura de la información debe ser flexible y permitir un crecimiento sostenido de los elementos introducidos en la misma	UR06	No Funcional

Tabla 2 - Requisitos de sistema

5.3. Matriz de trazabilidad

A continuación se muestra la relación entre requisitos de sistema y requisitos de usuario.

	UR01	UR02	UR03	UR04	UR05	UR06
SR01	X	X	X	X		
SR02	X	X	X	X		
SR03	X	X				
SR04	X		X	X		
SR05	X	X	X			
SR06	X		X	X		
SR07			X			
SR08			X			
SR09	X					
SR10		X				
SR11		X				
SR12		X				
SR13		X				
SR14					X	
SR15						X
SR16						X
SR17						X
SR18						X

Tabla 3 - Matriz de trazabilidad UR/SR

6. DISEÑO DETALLADO DEL SISTEMA

En el siguiente apartado, se realizará un análisis exhaustivo de todos los elementos que intervienen en el sistema. En este análisis se incluyen los elementos software que deberán ser desarrollados, las estructuras de datos y los medios físicos donde se almacenarán así como la arquitectura general del sistema para una futura implantación.

6.1. Diseño de software

El diseño de software elegido para el desarrollo del sistema se basa en la metodología MVC (Model View Controller) que es ampliamente utilizado en proyectos de software orientados a la web. Este modelo se basa en el desarrollo modular de los elementos que componen el sistema, pudiendo ampliar sus necesidades en un futuro de una manera cómoda y sencilla sin afectar a las funcionalidades del resto de elementos. Además, se intentará realizará una aproximación del sistema para que las labores de mantenimiento o las extensiones agregadas al sistema en momentos futuros del desarrollo puedan ser realizadas por un equipo de personas de una manera precisa.

Con el fin de que el sistema no dependa de manera trascendental en el sistema físico donde está contenida la información; que este pueda ser sustituido por otro u otras razones relativas al mantenimiento, se dotará al sistema de una capa de persistencia de la información entre el modelo de datos y los datos en sí.

A continuación, veremos los elementos que intervendrán en el sistema y qué funciones deberán realizar, sus dependencias y finalmente un diagrama que nos permitirá comprobar lo ya establecido en los siguientes puntos.

6.1.1. Elementos que intervienen en el sistema

- **Paquete Model**: Un paquete de clases donde está contenida la capa de estructura de la información en el sistema y que permitirá dotar al mismo de una capa de abstracción adicional. Este paquete de software contiene elementos relacionados entre sí, que utilizan funcionalidades unos de otros y que además son utilizados por otros paquetes externos.
- **Paquete View**: Se trata de un paquete de clases que contiene las funcionalidades necesarias para generar las vistas para el usuario final. En este paquete aunque no intervienen directamente elementos del Paquete Model, sí intervienen elementos dentro del Paquete Cache, que veremos a continuación.
- **Paquete Controller**: Contiene la esencia del funcionamiento del sistema. En este paquete se incluye toda la lógica que permite gestionar la creación de información para ofrecer al usuario final a través del Paquete View. El conjunto de elementos contenidos en el paquete interactúa con los elementos del Paquete Model y el Paquete Database.
- **Paquete Database**: A modo de capa de abstracción entre la estructura de información contenida en el Paquete Model y el componente físico que contiene dicha información se construye esta capa de abstracción. En esencia, esta capa permite obtener un modelo de persistencia que no nos haga depender de la estructura física donde se almacena la información.
- **Paquete Cache**: Por razones de rendimiento, el usuario final no puede estar realizando consultas directamente contra la base de datos y la información final que se consultará estará cacheada. Este paquete permite controlar el sistema de cachés que será implantado en conjunción con el Paquete Database (donde se incluye la lógica de persistencia de la información en ambos recursos) y el Paquete View (que utilizará el usuario final para consultar la información).
- **Paquete RXFramework**: Se trata de un framework que permite dotar al sistema y al desarrollo final de una capa de abstracción sobre las funcionalidades básicas del sistema. Esta propuesta se toma debido a que el Framework elegido (RXFramework), cumple las expectativas necesarias para desarrollar el sistema.

6.2. Diagrama de paquetes y clases

A continuación se muestra el diagrama de paquetes y clases que se ha explicado en el punto anterior. En él se pueden observar las relaciones entre los paquetes y elementos descritos.

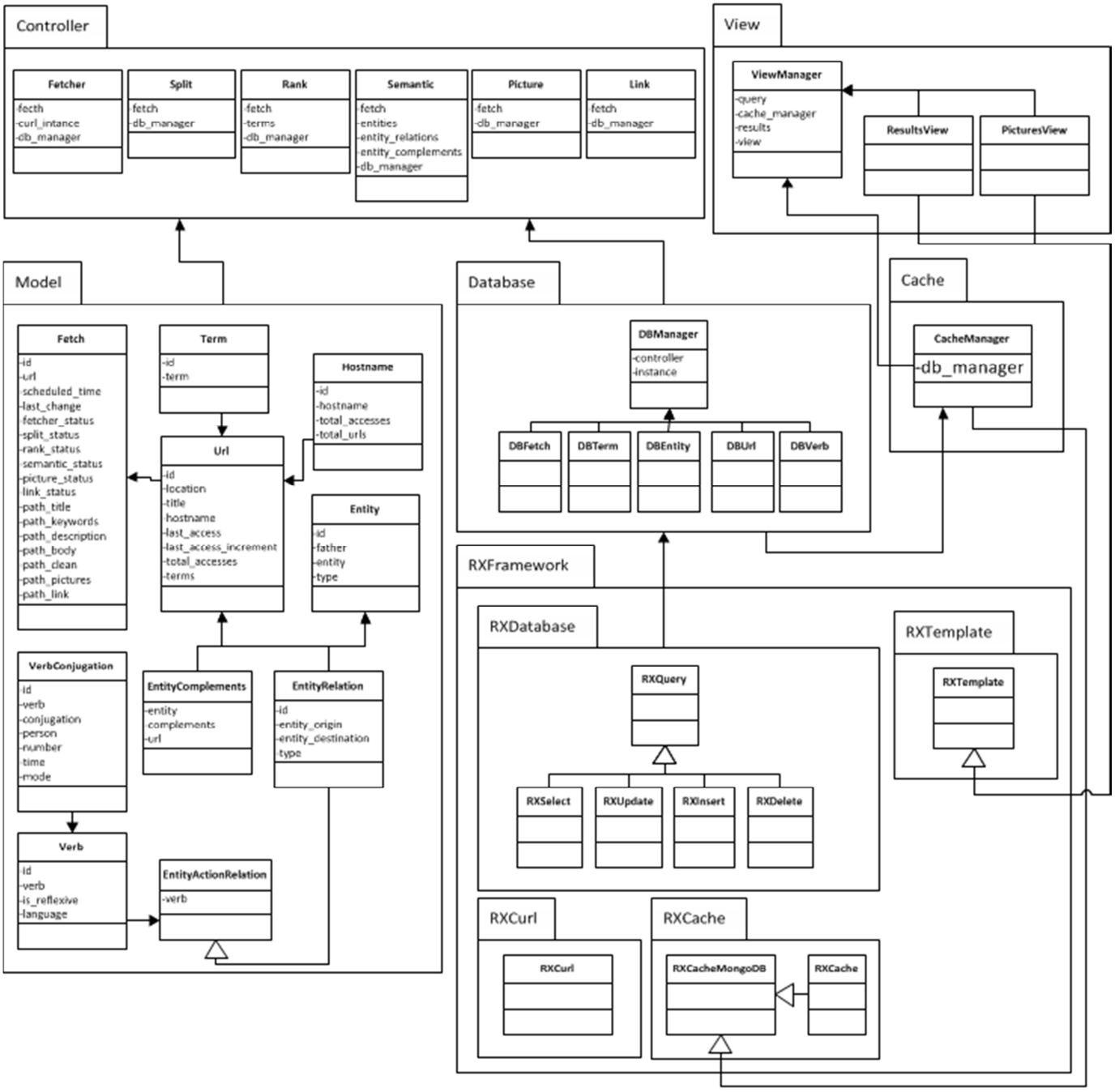

Figura 3 - Diagrama de paquetes y clases

6.3. Estructuras de datos

A continuación se detalla la información que el sistema almacenará y los soportes físicos donde se realizará dicho almacenaje.

Además, en el primero de los apartados de este punto se analizarán las entidades que intervienen en el sistema con el fin de arrojar algo de luz al sentido de la información.

6.3.1. Entidades que intervienen en el sistema

Las entidades que componen la estructura de información del sistema son las siguientes:

- **Url**: Identificará una localización de un documento en internet. Este elemento es el que está relacionado con el resto de entidades del sistema.
- **Hostname**: Identifica un nombre de dominio al que pertenece una o varias URLs.
- **Fetch**: Es el elemento de sistema sobre el que se trabajará. Esta entidad contiene información de cuándo debe ser procesada una URL y por qué punto del proceso se está analizando. Es la entidad sobre la que trabajarán los procesos del sistema para consultar información externa y analizarla posteriormente.
- **Término**: Es el elemento básico del lenguaje para ser analizado de manera básica. Tiene una relación elemental con las URLs ya que éstas identifican los documentos y al fin y al cabo la relación entre términos y documentos es la que nos permite considerar la relevancia entre ambos ante una búsqueda concreta.
    - **Relación de Términos y URLs:** Como ya se ha dicho, es la estructura básica del análisis simple de los documentos que se expondrán al sistema.
- **Entidades**: Estas entidades son las que intervienen a la hora de dotar de información semántica a un documento, en concreto a una URL. Una entidad además, puede tener un tipo concreto, es decir, puede contener información relativa a una localización a una personalidad o a una entidad no personal. Las entidades, a su vez, están relacionadas con las URLs de manera intrínseca. Las relaciones con las mismas vienen dadas por las siguientes entidades:
    - **Relación de Entidades**: Una relación uno a uno entre dos entidades donde se identifica a su vez el tipo de relación que tienen. Además, deberá incluir información sobre el tipo de relación que tienen ambas entidades.
    - **Relación de Complementos y Entidades:** En un texto concreto pueden aparecer relaciones semánticas, lo que puede dotar a un documento de información adicional a la hora de ser analizado.
    - **Relación de Entidades y Acciones:** Se tratan de relaciones semánticas entre entidades y las acciones realizadas entre ellas.
- **Verbos:** Son elementos que nos permitirán extraer información semántica de los documentos a analizar y permiten realizar relaciones semánticas con entidades por medio de la relación que ejecutan.
- **Conjugaciones Verbales:** Una base de datos de conjugaciones verbales que permite analizar los textos de una manera más detallada, ofreciendo información temporal, del calibre de la acción y del tipo de entidad que lo realiza. Las razones de la utilización de esta estructura de información se analizarán más adelante y están estrechamente relacionadas con la búsqueda de un análisis semántico exhaustivo sobre los documentos.

6.3.2. Bases de datos

Aunque en este punto no se realizará una especificación exhaustiva del sistema de base de datos a utilizar, sí se detalla la estructura que tendrán los datos en dicho soporte.

Sí es importante entender que la base de datos a utilizar debe ser relacional y se intentará aprovechar las características ofrecidas por el Framework a utilizar así como la búsqueda del rendimiento a la hora de realizar consultas a través del gestor de base de datos. Para ello, se precisa que la base de datos esté organizada bajo los estándares PLSQL.

La arquitectura de las bases de datos del sistema se analizará también más adelante.

A continuación se detalla un diagrama que permite comprobar las relaciones entre las entidades descritas con anterioridad y la estructura de información de las mismas:

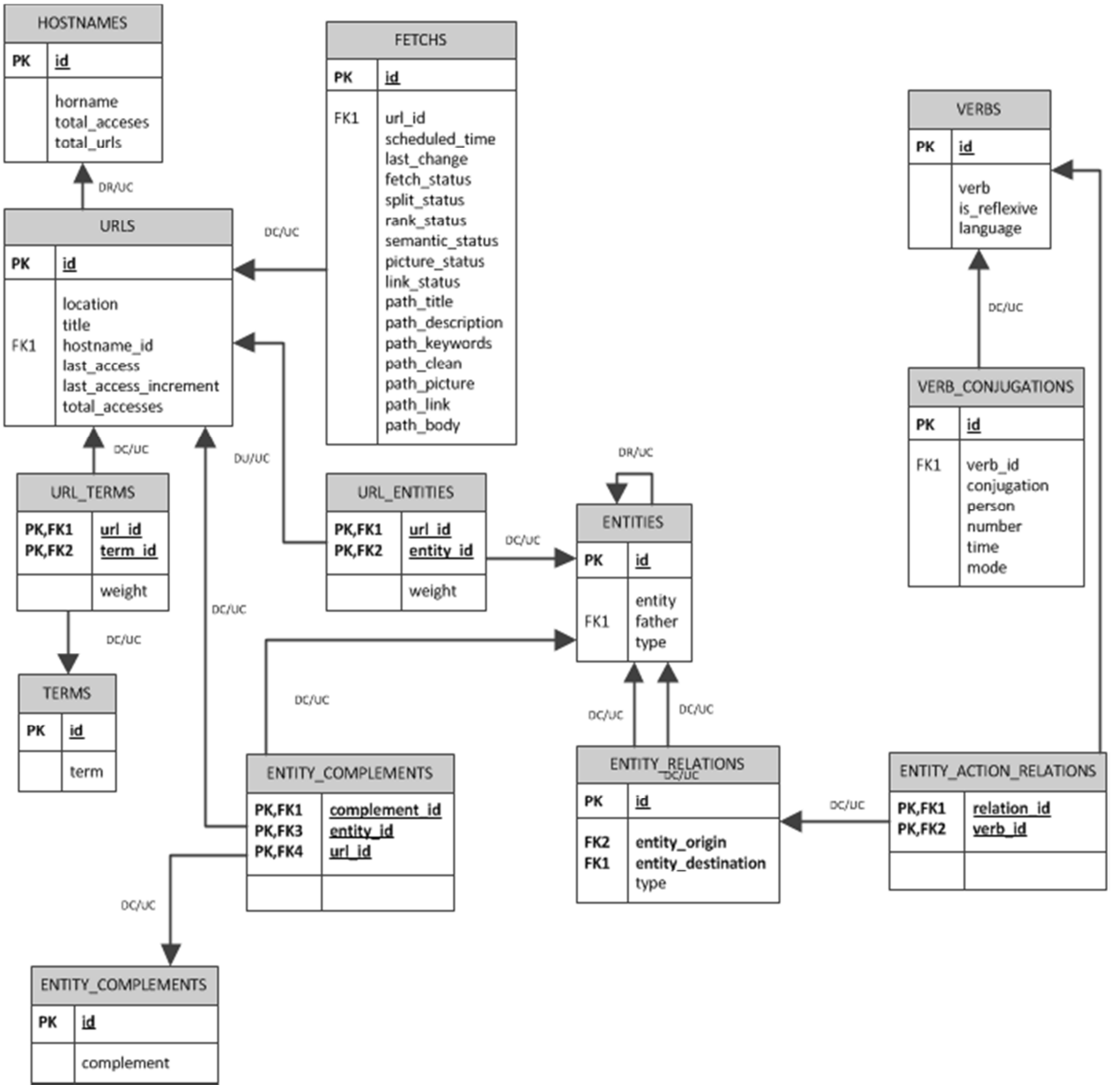

Figura 4 - Diagrama de base de datos

6.4. Arquitectura del sistema

La arquitectura del sistema tendrá los siguientes objetivos:

- Ofrecer una consistencia del sistema y un rendimiento óptimos.
- Ofrecer posibilidades de recuperación adecuadas ante los posibles riesgos que puedan encontrarse tanto de malfuncionamiento de la infraestructura como a posibles ataques desde fuera de la red.
- Ofrecer la posibilidad de aumentar las funcionalidades del sistema por parte de un equipo de desarrollo.

De esta manera, tenemos el siguiente diagrama:

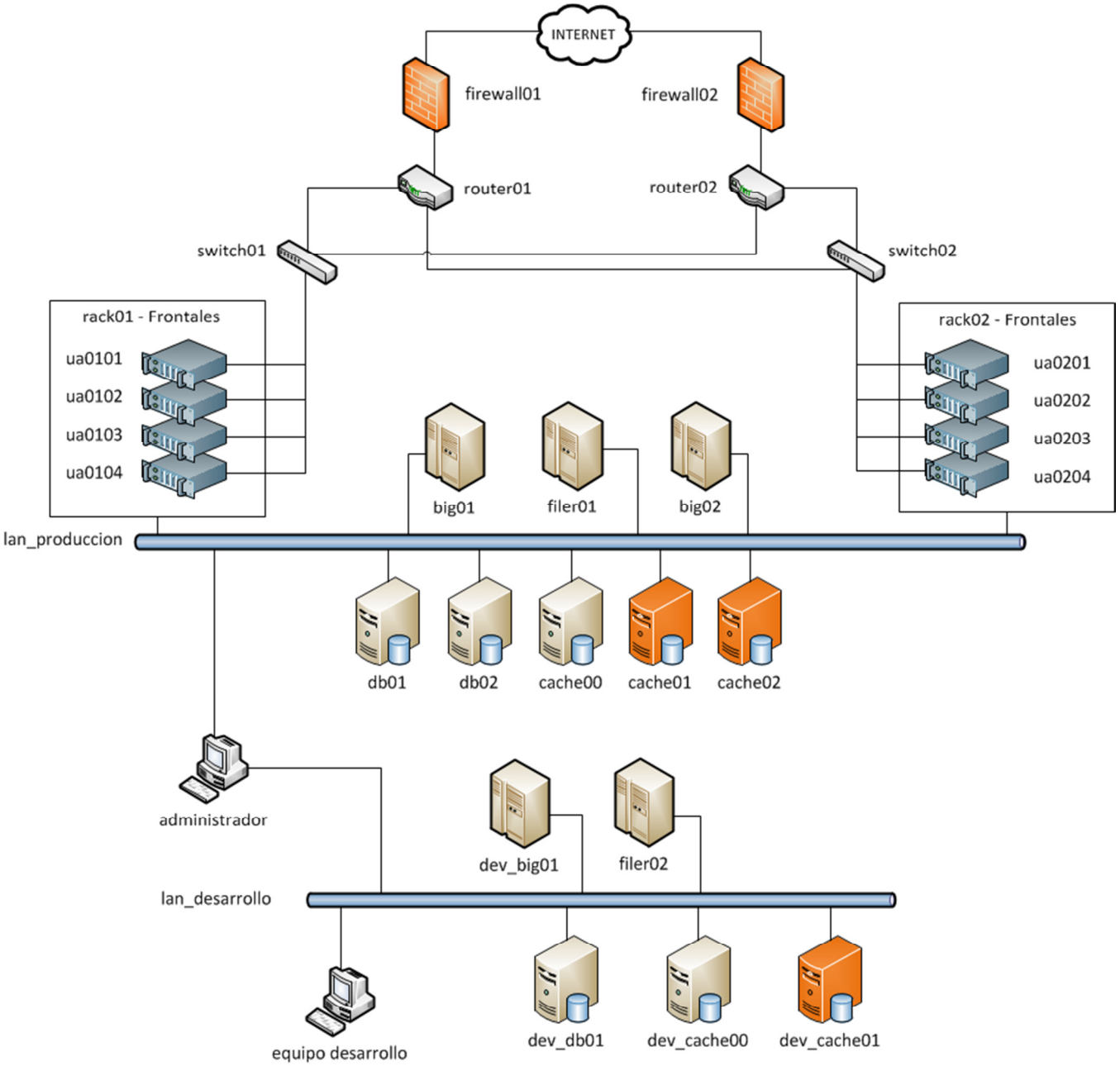

Figura 5 - Diagrama de arquitectura del sistema

Los elementos que se muestran en el diagrama son los siguientes:

- **Firewalls**: Realizan un rechazo selectivo de los paquetes que no nos interesa recibir desde fuera de nuestra red. En concreto, la intención de estos firewalls es salvaguardar el sistema que está funcionando detrás y que éste no se vea alterado frente a posibles ataques. En concreto la arquitectura del sistema contará con dos firewalls:
    - firewall01
    - firewall02
- **Routers**: Las peticiones que se reciban desde fuera de la red del sistema, es decir, desde internet, y que han sido filtradas por los firewalls acabarán directamente en los servidores frontales. Para que la carga de estos servidores sea equilibrada, los routers efectuarán labores de balanceo.

  Se obvian de momento de la arquitectura todos los routers, switches y conectores de red que están involucrados en la red interna para simplificar el esquema.

  En concreto habrá dos elementos presentes en la arquitectura de sistema:
    - router01
    - router02
- **Switches**: Los servidores frontales recibirán un número de peticiones elevado, por lo tanto, para dirigir el tráfico a cada una de las máquinas se utilizará un switch conectado a cada rack, que permitirá efectuar un direccionamiento de las peticiones adecuado. Existirán dos switches en la arquitectura del sistema:
    - Switch01: contra rack01
    - Switch02: contra rack02
- **Racks**: Son los armarios físicos donde residirán los servidores frontales, en concreto hay dos para diferenciar la localización de cada servidor por razones de organización y de mantenimiento:
    - rack01: contiene los servidores frontales ua0101, ua0102, ua0103, ua0104
    - rack02: contiene los servidores frontales ua0201, ua0202, ua0203, ua0204
- **Servidores frontales**: Agrupados en dos estructuras de red en grupos de cuatro, por un lado el rack01: ua0101, ua0102, ua0103, ua0104 y por otro el rack02: ua0201, ua0202, ua0203, ua0204.

  Estas máquinas se encargarán de realizar las comprobaciones y procesamiento de las búsquedas efectuadas por los usuarios. Hay que tener en cuenta que este procesamiento de la información debe ser muy ágil, eficiente y no utilizar recursos que puedan frenar el dinamismo de las respuestas. Es por esto que los desarrollos que se realicen hacia fuera no efectuarán ninguna petición contra una base de datos ni contra recursos de bajo rendimiento. Estas máquinas estarán conectadas con las máquinas de cachés que contengan la información ya procesada de los recursos que se quieran servir.

  Es necesario tener en cuenta que para el entorno de desarrollo no se va a precisar máquinas frontales ya que quien hará uso de dicho entorno será un equipo especializado de un número limitado de personas. De momento y debido a que las necesidades de dicho entorno están muy ajustadas, se utilizará la máquina de procesamiento del entorno de desarrollo (dev_big01) como servidor frontal.

Esta estructura dividida permite realizar un balance de las peticiones recibidas desde internet y además salvaguardar el sistema ante posibles fallos del mismo, caídas de red o similares:

  - En el caso de que uno de los frontales de un rack deje de funcionar, el tráfico puede derivarse al resto de los frontales del mismo rack.
  - En el caso de que uno de los racks deje de funcionar, siempre quedará uno activo para manejar las respuestas a las peticiones recibidas desde internet.
  - En caso de que cualquier máquina deje de funcionar, no estará de más contar con máquinas de respaldo para cualquier suceso imprevisto que pueda tener lugar.

- **Máquinas de procesamiento avanzado**: La arquitectura contará con dos máquinas para realizar la ejecución de los procesos que intervienen en el sistema. Estos procesos no tienen nada que ver con el tratamiento de la información que se ofrecerá al usuario pero sí con los procesos internos del sistema. Estos procesos involucrarán esencialmente al análisis de los documentos y su indexación a través de los registros organizados en la base de datos. Los recursos finales serán preparados en cachés para la consulta de los frontales.

  Las máquinas que realicen la ejecución de los procesos deben ser potentes y tener disponibles grandes cantidades de memoria. Un número bastante alto de procesos ejecutarán en esas máquinas continuamente y es necesario que sus recursos sean adecuados.

  Además se contará con una máquina exclusiva para el entorno de desarrollo.

  En concreto habrá tres máquinas que permitan realizar estos procesamientos:

  - En el entorno de producción: big01 y big02
  - En el entorno de desarrollo: dev_big01

- **Servidores de bases de datos SQL**: Con el fin de que el sistema sea rápido y escalable se contará con dos máquinas que serán servidores de bases de datos SQL. Las bases de datos que contendrán serán las mismas debido a que estarán replicadas. La razón de esto es salvaguardar el sistema ante un posible fallo de una de las máquinas. En el caso de una máquina falle, siempre se tendrá la otra de respaldo para su utilización.

  Además se contará con una máquina exclusiva para el entorno de desarrollo.

  Se tendrá entonces:

  - En el entorno de producción:
    - db01: Servidor de bases de datos en modo MASTER
    - db02: Servidor de bases de datos en modo SLAVE (réplica de db01)
  - En el entorno de desarrollo:
    - dev_db1: Servidor de bases de datos en modo MASTER

- **Servidores de cachés**: El sistema contará con máquinas que se dediquen exclusivamente a almacenar información ya procesada por las máquinas big01, big02 (ambas en el entorno de producción) y dev_big01 (en el entorno de desarrollo).

  Los servidores de caché tendrán una arquitectura especial. En concreto, siempre habrá un servidor en modo MASTER para cada entorno y por otro lado uno o varios servidores en modo SLAVE a los cuales se accederá desde las máquinas frontales en modo solo lectura.

La razón de esta elección de arquitectura es por motivos de rendimiento. Al acceder los servidores frontales a máquinas que tienen restringida la escritura no tendrán lugar operaciones de bloqueo ni restricciones por escrituras de manera sistemática.

Por otro lado, es necesario entender, que la máquina en modo MASTER realizará una copia cada cierto tiempo de la información que contenga a las máquinas en modo SLAVE.

En concreto se tendrá la siguiente arquitectura:

- En el entorno de producción:
  - cache00: en modo MASTER
  - cache01: en modo SLAVE y al cual accede solo los frontales de rack01
  - cache02: en modo SLAVE y al cual accede solo los frontales de rack02
- En el entorno de desarrollo:
  - dev_cache00: en modo MASTER
  - dev_cache01: en modo SLAVE

- **Redes Ethernet**: Las redes internas del sistema tendrán dos claras divisiones. La razón de esto es para dotar al sistema de una independencia en los entornos de producción y desarrollo. De esta manera, un equipo de desarrollo podrá trabajar en mejoras o realizar un mantenimiento de software sobre el sistema sin afectar a lo que ya esté funcionando en producción:
  - lan_produccion: Correspondiente a los elementos que intervienen tanto en los procesos de producción como los involucrados en servir el contenido adecuado al usuario.
  - lan_desarrollo: Una red restringida de la anterior a la cual solo tiene acceso el equipo de desarrollo y los sistemas configurados para dicho entorno.
- **Equipos conectados a las redes:** Se tendrá una serie de equipos conectados a las redes:
  - lan_produccion: Uno o varios equipos accesibles solo por administradores.
  - lan desarrollo: Uno o varios equipos accesibles solo por el equipo de desarrollo, esto incluye a los administradores del sistema.
- **Sistema de ficheros NFS**: Todas las máquinas estarán conectadas a dos sistemas de ficheros NFS:
  - filer01: sistema de ficheros de producción (solo en la red de producción)
  - filer02: sistema de ficheros de desarrollo (solo en la red de desarrollo)

6.5. Planificación de procesos

Con el fin de realizar un uso eficiente de los recursos tanto de la máquina donde se realice la ejecución de los mismos como de las estructuras de la información, el procesamiento completo de los resultados se realizará una partición de las tareas en distintos procesos.

Debido a que los procesos funcionarán en paralelo, el recurso sobre el que mantendrán la ejecución corresponderá a una secuencia de procesamiento sobre un documento en concreto. El proceso comenzaría con la recuperación del documento desde el sistema donde resida, su análisis, procesamiento y planificación para la siguiente recuperación.

Para que los distintos procesos no realicen las tareas que sean precisas de una manera desordenada o se pueda dar la situación en que un mismo recurso sea utilizado por dos procesos distintos, se integrará un sistema de semáforos sobre estos recursos. Este sistema de semáforos marcará cada uno de los recursos en distintos estados.

Este mecanismo ejecutará los distintos procesos para que cada uno de los recursos pase por ellos en el siguiente orden:

- **FETCH**: Realiza una recuperación de la información relacionada con el recurso.
- **SPLIT**: Con la información recuperada por el proceso anterior, se realiza una división del contenido. Cada uno de los fragmentos se almacenan para su posterior análisis.
- **RANK**: Sobre los fragmentos almacenados se realiza un análisis de frecuencia de términos.
- **SEMANTIC**: Se realiza un análisis semántico sobre el fragmento de texto limpio.
- **PICTURE**: Cada una de las imágenes contenidas en el recurso es almacenada recuperándose previamente desde el servidor donde esté alojada con el fin de ofrecerla en los resultados de búsqueda.
- **LINK**: Se realiza una planificación de futuras ejecuciones del proceso FETCH sobre cada uno de los hipervínculos contenidos en el recurso.

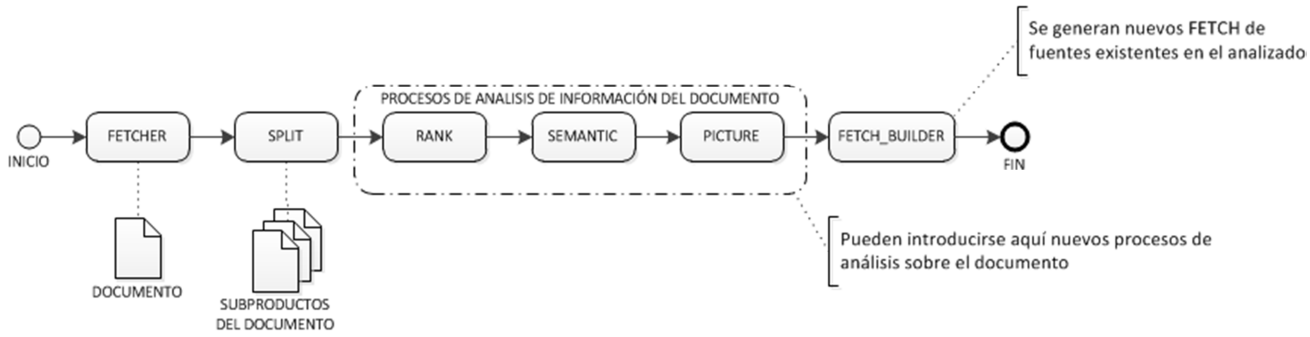

Figura 6 - Diagrama de flujo de procesos

Además, cada uno de los recursos, tendrá un tiempo de planificación. La lógica de planificación producirá que los recursos se introduzcan en una cola por lo que el primero que entre será el primero que sea procesado. De esta manera, cada uno de los procesos, recuperará el primer recurso de la cola en el estado adecuado para dicho proceso.

Los procesos que interactuarán con los recursos, y la evolución que tendrán los mismos durante el procesamiento completo de un recurso será la siguiente:

6.5.1. Proceso FETCH

Condiciones de inicio del proceso:

- Se recupera un recurso en cuyo estado del proceso FETCH sea 0 (en espera de ser recuperado) y se cambia su estado a 1 (recuperando contenido).

Funcionamiento del proceso:

- El recurso contendrá una URL, se realiza una llamada HTTP o HTTPS en función de la URL para recuperar su contenido.
- Se incrementa el número de recuperaciones realizadas sobre el mismo recurso.

- Se comprueba el estado de la recuperación, comprobando que existe el documento al que apunta el recurso, que no ha habido error o no se ha producido un timeout.
- El contenido del documento se copia en almacenamiento en disco.
- La ruta al contenido del documento se referencia en el recurso.
- Modificaciones de finalización del proceso sobre el recurso:
- El estado del proceso FETCH del recurso se cambia a 2 (recurso recuperado).

6.5.2. Proceso SPLIT

Condiciones de inicio del proceso:

- Se recupera un recurso cuyo estado del proceso FETCH sea 2 (recurso recuperado) y su estado del proceso SPLIT sea 0 (recurso recuperado y en espera de ser procesado).
- El estado del proceso SPLIT para el proceso se cambia a 1 (procesando SPLIT).

Funcionamiento del proceso:

- Se recupera el contenido del recurso almacenado.
- El contenido es particionado en una serie de fragmentos relevantes para el motor de búsqueda.
- Se copian los fragmentos en distintas rutas de almacenamiento.
- Uno de los fragmentos que se almacenará será un fichero de texto limpio del documento.
- Se recuperan también todas las URLs contenidas en hipervínculos en la URL
- Las rutas de los fragmentos se introducen en la estructura de información del recurso.
- Modificaciones de finalización del proceso sobre el recurso:
- El estado del proceso SPLIT del recurso se cambia a 2 (recurso procesado).

6.5.3. Proceso RANK

Condiciones de inicio del proceso:

- Se recupera un recurso con estado de proceso FETCH y SPLIT cuyo valor sea 2, comprobando además que el estado del proceso RANK sea 0 (recurso en espera de ser analizado).
- El estado del proceso RANK pasa a tomar el valor 1 (analizando).

Funcionamiento del proceso:

- Para cada uno de los fragmentos a los que hace referencia el recurso:
    - Se recupera el fragmento y se analizan la frecuencia de los términos.
    - Al número total de repeticiones de un término se le multiplica el peso del fragmento.
- Para cada término en el documento se realiza el cálculo de la relevancia total en función de las apariciones y pesos de los fragmentos donde aparezca.
    - Cada término se introduce en la base de datos (si no existía antes) y se relaciona con el documento al que referencia el recurso con el valor de relevancia calculado.
- El recurso cambia de estado del proceso RANK a 2 (recurso analizado y en espera del proceso de análisis semántico).

6.5.4. Proceso SEMANTIC

Condiciones de inicio del proceso:

- Se recupera un recurso en cuyo estado en los procesos FETCH y SPLIT sea 2 además de tener el estado del proceso SEMANTIC a 0 (recurso en espera de ser analizado semánticamente).
- El estado del proceso SEMANTIC se cambia a 1 (analizando semánticamente).
- Nótese que en este proceso no se tiene en cuenta el estado del proceso RANK. Esto es debido a que la información que recupera el proceso RANK no es relevante para el proceso SEMANTIC y ambos pueden actuar en paralelo.

Funcionamiento del proceso:

- Se recupera el fragmento de texto limpio del recurso.
- Se realiza un análisis de las entidades existentes en el texto.
- Por cada entidad, se analiza su relevancia a través de su frecuencia en el documento.
- Se filtran todas las entidades recuperadas y el proceso se queda con las más relevantes.
- Cada entidad se introduce en la base de datos (si no existía antes) y se relaciona con el documento al que referencia el recurso con el valor de la relevancia calculado.
- Se realiza un análisis de los verbos involucrados en el fragmento de texto limpio.
- Para cada verbo se buscan relaciones semánticas con entidades que aparezcan en sus cercanías o otras relaciones semánticas.
- Cada relación se introduce referenciando al documento en la base de datos.
- El recurso cambia su estado del proceso semantic a 2 (recurso analizado semánticamente).

6.5.5. Proceso PICTURE

Condiciones de inicio del proceso:

- Se recupera un recurso en cuyo estado en los procesos FETCH y SPLIT sea 2 además de tener el estado del proceso PICTURE a 0 (recurso en espera de analizar las imágenes contenidas).
- El estado del proceso PICTURE toma el valor 1 (analizando imágenes del recurso).
- Nótese que al igual que en el proceso SEMANTIC, en el proceso PICTURE no se tiene en cuenta el valor de los procesos anteriores (rank y semantic). Esto es debido a que la información tanto del proceso RANK y del proceso SEMANTIC no es relevante para el análisis de imágenes contenidas en el recurso.

Funcionamiento del proceso:

- Se recupera el listado de imágenes relacionadas con el recurso del fichero generado por el proceso SPLIT.
- Por cada imagen se realiza una petición contra el servidor en la URL a donde apunta y se recupera el contenido. Después de realizar esta petición y siendo esta satisfactoria se guarda la imagen en un formato estándar con unas dimensiones adecuadas para el almacenaje. Es necesario tener en cuenta que las imágenes a visualizar desde una búsqueda no tendrán un tamaño muy grande, por lo que la imagen, si sus dimensiones exceden de unas indicadas por constantes en el código, será recortada. De la misma manera, para mostrar las imágenes en miniatura dentro de los resultados ofrecidos por el buscador, se tomará un recorte y se almacenará posteriormente.

- Tras haber almacenado los recursos de imagen recogidos del servidor, se introduce la información en la base de datos y se relaciona con el recurso que se ha recuperado.
- El recurso cambia su estado del proceso PICTURE a 2 (imágenes del recurso analizadas).

6.5.6. Proceso LINK

Condiciones de inicio del proceso:

- A diferencia de los anteriores, este proceso precisa de la finalización de todos los procesos anteriores. De esta manera, todos los estados de los procesos anteriores han de indicar que han finalizado (con valor de 2) y el proceso está marcado con estado 0 en el proceso LINK.

Funcionamiento del proceso:

- Se recupera el listado de hipervínculos del recurso desde el fichero generado por el proceso SPLIT.
- Por cada uno de las URLs de los hipervínculos, se comprueba si existe en base de datos y se introduce en ella si no existe con anterioridad.
- Posteriormente (si la URL existía), se comprueba si existe algún recurso que esté siendo procesado o esté planificado para procesar de la misma URL. En caso de que no exista se planifica. Esta planificación se estudiará más adelante en el apartado dedicado a la planificación de la recuperación de los recursos.

6.6. Recuperación de contenidos

La recuperación del contenido de los documentos a analizar por el motor de búsqueda se realiza dentro de varios de los procesos que han sido descritos a lo largo del apartado anterior. En concreto, el contenido general del documento es recuperado desde el proceso FETCH. Por otro lado, las imágenes que son utilizadas para dar más información de los resultados de una búsqueda, son recuperadas desde el proceso PICTURE.

6.6.1. Petición contra el servidor y cabeceras

Las llamadas contra servidores serán realizadas a través del protocolo HTTP. Los procesos que realizan estas llamadas HTTP utilizan procedimientos habituales en la implementación de llamadas a servidores mediante la librería Curl. Esta librería está implementada en numerosos lenguajes de programación y es utilizada ampliamente en el mercado.

En el ámbito general, el uso de Curl precisa establecer ciertos parámetros en la librería que permitan realizar una petición contra un servidor de manera adecuada. En concreto, se establece la URL del destino donde se quiere realizar la petición, el protocolo a usar (si es seguro se utilizará HTTPS), el tiempo de espera antes de ofrecer un error en la respuesta, etc. Estos parámetros se envían al servidor en la petición. En particular, para el proceso FETCH, se establecen los siguientes parámetros:

- **URL del destino**: Es extraída del recurso sobre el que se realizará la petición y es recuperado por el proceso en cuestión.
- **Método de la petición**: Se establece por defecto a GET. Por lo general las URLs que reciben peticiones desde un navegador, por ejemplo, no realizan peticiones a través de otros métodos cuando se sigue un hipervínculo. Generalmente esto no ocurre así cuando por ejemplo se envía información a un servidor a través de un formulario. En este caso, por lo general se utiliza el método POST de HTTP para realizar las peticiones debido a que los límites en los tamaños de las llamadas no están tan limitados como ocurre con el método GET y permiten la introducción de conjuntos de información más compleja en las peticiones.

- **Tiempo de espera**: Se establece por defecto a 45 segundos. La ejecución de la librería es bloqueante para el proceso debido a que se espera la respuesta a la llamada realizada para continuar procesándola. Por esta razón, el tiempo de espera ante la respuesta del servidor se establece a menos de un minuto. La razón de esto es que no queremos que el proceso quede bloqueado durante mucho tiempo en caso de que un servidor no ofrezca una respuesta lo suficientemente rápido, pueda estar caído o existan otros problemas de comunicación.
- **Lenguajes aceptados**: Cualquiera que sea español. Aunque este parámetro los servidores no suelen tratarlo, se establece por defecto para el lenguaje español debido a que en ocasiones, algunos servidores si realizan redirecciones en función del lenguaje. Para estos casos, los servidores actúan de manera inteligente ofreciendo a la petición el lenguaje que están buscando y facilita el entendimiento del contenido al usuario.
- **Nombre del UserAgent**: Como se ha explicado con anterioridad, los servidores devuelven una respuesta ante una petición que haya realizado contra ellos. Estas peticiones quedan registradas en el servidor con una traza que indica la información del sistema que el usuario que ha realizado la petición tiene configurada. Por lo general, la información que reciben los servidores en este parámetro tiene que ver con el navegador web utilizado y la tecnología que implementa.

  En un navegador Mozilla Firefox, un ejemplo de User Agent puede ser el siguiente:

  ```
  "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:15.0) Gecko/20100101
  Firefox/15.0.1".
  ```

  Para este caso concreto y con el fin de que los servidores que recibirán las peticiones tengan claro que éstas han sido realizadas por un crawler se establecerá este parámetro como:

  ```
  "Semantic Search UC3M-RX Bot 1.0 (http://www.uc3m.es)"
  ```
- **Cookies**: En muchas ocasiones, la implementación de los servicios de los servidores integran variables en el cliente con el fin de almacenar configuraciones de sesión. Aunque esto es propio de los navegadores web, se dotará al sistema de una gestión de cookies que permita almacenar configuración ofrecida por el servidor. Es probable que esto no tenga demasiado interés técnico, pero sí facilitará el reconocimiento de los servidores del sistema que realiza las peticiones cuando se prolonguen en el tiempo. De esta manera, se reserva un fichero de cookies que se irá completando con las respuestas recibidas desde los servidores.

6.6.2. Tratamiento de la respuesta del servidor

Una vez se ha recibido una respuesta del servidor ante la llamada realizada contra la URL del recurso, se realizará un procedimiento estándar para todas las peticiones siguiendo los siguientes pasos:

- Lo primero que se debe comprobar cuando se recibe la respuesta del servidor es que ésta ha sido satisfactoria. Para ello, se comprueba el código de respuesta HTTP recibido y siempre debe ser 200 (OK). En caso contrario se retira el recurso de la cola de procesos para no realizar más comprobaciones. En caso de que la URL haya ofrecido un valor 404 (No disponible), se registrará como tal para tratar en futuras peticiones estos recursos de una manera distinta.
- Posteriormente, se comprueba que el contenido del documento tiene una tipología apropiada. En este caso, se analizarán documentos en formato textual o HTML por lo que el valor de la cabecera de la respuesta ContentType tiene que tener un valor "text/html/. En caso contrario se desechará el recurso.

- Se recupera el contenido de la respuesta y se almacena en una variable temporal que será analizada en los siguientes pasos.
- Es necesario comprobar la codificación de caracteres de la respuesta recibida. Es indispensable la comprobación de este apartado debido a que si no se comprueba y se unifican las codificaciones de caracteres en los procesos del sistema se obtendrán resultados adversos.
    - La recomendación de W3C desde la versión 5.2.2 de HTML4 es la de introducir esta información en las cabeceras de HTTP de la respuesta para que pueda ser tratada con naturalidad por los navegadores web.
    - Si este parámetro no estuviese definido en las cabeceras de la respuesta es probable que esté definido en el contenido. Por lo general, en la estructura de un documento HTML se establece un apartado "head" donde se establecen "metas" o parámetros de definición del documento en si. En este apartado se suele introduce la codificación de caracteres del documento.
    - Si el parámetro anterior tampoco estuviese definido se realizará un análisis de la codificación para tratar de detectarla. En caso de ser imposible detectarla se desechará el recurso de la cola de procesos.
    - La codificación de caracteres que se usará a lo largo de los procesos y de las estructuras de información será siempre UTF-8. Si se ha detectado una codificación distinta en el documento recuperado se realizará una conversión a través de la librería de sistema iconv.
- El contenido de la respuesta se almacenará en un fichero que denominaremos RAW Content. Este fichero se tratará por los sucesivos procesos del sistema con el fin de analizarlo y procesarlo.
- El producto final del proceso es un fichero denominado RAW Content que se utilizará en los procesos sucesivos para su procesamiento. La ruta completa en el sistema de almacenamiento de este fichero es introducida en la información del recurso para su utilización más adelante.

6.6.3. Fragmentación del contenido

El contenido recuperado y que el proceso FETCH ha almacenado en un fichero que hemos denominado RAW Content es separado en distintos fragmentos o subproductos por el proceso SPLIT.

Estos subproductos serán procesados por los siguientes procesos más adelante según se describía en los apartados anteriores.

La fragmentación se realiza atendiendo a la estructura del documento en sí. Debido a que los documentos que serán analizados son HTML se atenderá a esta estructura en particular. Debido a que HTML es un lenguaje de marcado basado en XML su estructura por lo general está definida de la siguiente manera:

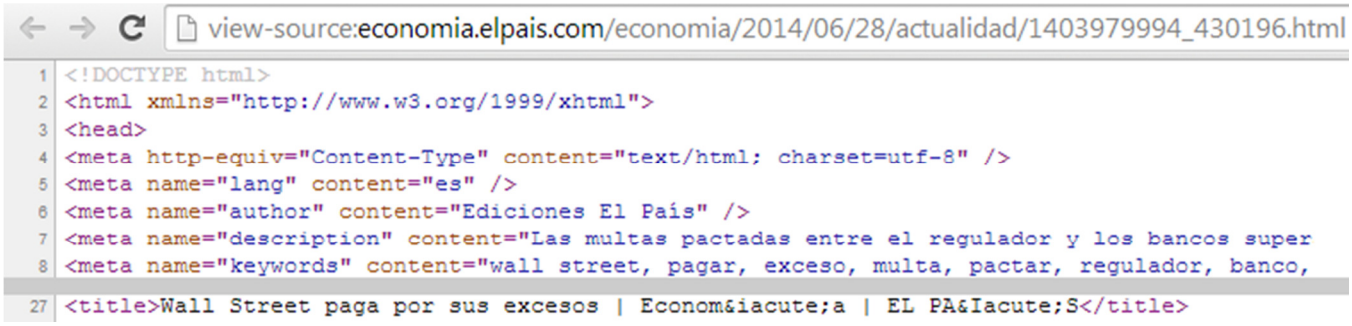

Figura 7 - Estrucutura de metainformación en HTML

Los subproductos son extraídos del contenido de estas estructuras del documento de la siguiente manera:

- **Title**: Fichero que incluye el título del documento recuperado. Es extraido de la información "meta" del documento si existiese.
- **Keywords**: Fichero que incluye la información de keywords del documento. Es extraido de la información "meta" del documento si existiese.
- **Description**: Fichero que incluye la información del campo "description" del documento. Es extraido de la información "meta" del documento si existiese.
- **Body**: Cuerpo del documento, el conjunto completo de la etiqueta Body.
- **Clean Content**: Contenido en claro del documento. Es la transformación del subproducto Body a un subproducto limpio de código y conjuntos de caracteres que no puedan usarse. En HTML se suele utilizar ciertas codificaciones de caracteres que permiten la visualización de los mismos en un navegador. En este caso estos caracteres son traducidos a un tipo de carácter corriente que puede ser analizado por el sistema. El subproducto final es un fichero limpio que será leído como un texto.
- **Pictures**: Listado de imágenes contenidas en el documento. En concreto es la extracción de todos los elementos <img> del cuerpo de HTML. Estos elementos pueden tener asociados textos alternativos o títulos que también son almacenados en el subproducto para su posterior análisis.
- **Links**: Hipervínculos a otros documentos. En concreto es la extracción de todos los elementos <a> del cuerpo de HTML. Estos elementos pueden tener asociados títulos alternativos e información propia del hipervínculo que son almacenados en el subproducto para su posterior análisis.

Cada uno de estos subproductos se almacena en el sistema como ficheros. La ruta de almacenamiento de cada uno de estos ficheros se introduce en la información del recurso para su posterior procesamiento.

6.7. Análisis y creación del índice básico de documentos

6.7.1. Análisis de frecuencia

El análisis básico comprende esencialmente un análisis de frecuencia sobre los términos existentes en los documentos. Este análisis, además, se realiza atendiendo a la estructura del documento HTML, ofreciendo distinta relevancia a ciertas partes del documento que por estándares se consideran más importantes que otras.

El sistema interno del motor de búsqueda permite otorgar a estas partes del documento con distintos grados de relevancia valores diferenciados que son configurables en tiempo de ejecución.

Esta característica permite elaborar un análisis exhaustivo del comportamiento del propio motor de búsqueda ante los resultados obtenidos a lo largo del tiempo.

El análisis de frecuencia de los términos contenidos en los documentos es realizado por el proceso RANK. Este proceso recoge para un recurso en concreto los fragmentos del contenido que se generaron en el proceso SPLIT. En concreto, se analizan los fragmentos Title, Keywords, Description y Body. El proceso general de análisis de frecuencia de términos para cualquiera de estos elementos es el siguiente:

- Para los fragmentos Title, Keywords y Description (donde no es apropiado según los estándares la aplicación de código HTML) se limpian de cualquier tipo de elemento que no sea un término incluyendo: fragmentos de código HTML, signos de puntuación, otros símbolos que no se tendrán en cuenta como símbolos. Este proceso de limpieza se realiza mediante expresiones regulares.
- Para el fragmento Body, se realiza el mismo procedimiento de limpieza, pero se almacenan en variables temporales aquellas partes del código que son consideradas más relevantes que otras, tales como:
    - Fragmentos <h1>, <h2> o <h3>: Cabeceras de HTML, integran por lo general títulos o elementos muy destacados del documento.
    - Fragmentos o <b>: Texto destacado en negrita.

Se realiza el análisis de términos, buscando repeticiones de cada uno de ellos a lo largo de los fragmentos. El resultado de este análisis ofrecerá para cada fragmento un listado de términos con el número de repeticiones similar a la siguiente tabla:

TÉRMINO	REPETICIONES
y	9
de	7
gobierno	5
coches	4
el	4
licencia	3
conducir	2
reglamento	2
coche	2
con	2
personas	1
para	1
obtener	1

Tabla 4 - Análisis de frecuencia básico I

En la tabla anterior, para un caso hipotético vemos dos casos que hay que tener en cuenta a la hora de hacer el análisis. En primer lugar, se obtienen muchas repeticiones para términos que no ofrecen ninguna relevancia al documento como conjunciones o preposiciones. Teniendo una lista de estos elementos de propio lenguaje a analizar se pueden extraer aquellos términos de la lista del análisis y obtener algo similar a lo siguiente:

TÉRMINO	REPETICIONES
gobierno	5
coches	4
licencia	3
conducir	2
reglamento	2
coche	2
personas	1
obtener	1

Tabla 5 - Análisis de frecuencia básico II

Además, nos damos cuenta que algunos términos ofrecen una distinción en el caso del plural. Esto sucede con el término "coche" y su plural "coches". Aparece varias veces en el texto analizado y es necesario tener en cuenta la posibilidad de la existencia de ciertas variaciones de los términos en el lenguaje para no ofrecer resultados inapropiados. En este caso, cada uno de los términos "coches" debería pasar a contar como repeticiones del término "coche", quedando la tabla de la siguiente manera:

TÉRMINO	REPETICIONES
coche	6
gobierno	5
licencia	3
conducir	2
reglamento	2
personas	1
obtener	1

Tabla 6 - Análisis de frecuencia básico III

6.7.2. Relevancia del contenido

Para cada uno de los fragmentos ofreceremos distintas variaciones en la relevancia de los términos. Siendo esta valoración configurable, obtendríamos una lista de términos que variaría de la

original y posiblemente algunos términos que apareciesen menos veces en el documento tendrían mayor relevancia que otros.

El sumatorio global de las repeticiones de cada uno de los términos ofrecería una visión primaria de la importancia de un término en un documento, pero aplicándole el valor de relevancia al fragmento del documento que tenemos entre manos, obtendremos una variación bastante más aproximada de la realidad.

Para una aproximación de la configuración de los pesos de los términos en cada uno de los fragmentos podríamos configurar estas variaciones de la siguiente manera:

FRAGMENTO	ETIQUETA HTML	PESO
Title	Cualquiera	10
Keywords	Cualquiera	5
Description	Cualquiera	3
Body	H1	5
Body	H2	3
Body	H3 o sucesivas	2
Body	STRONG o B	2
Body	Resto	1

Tabla 7 - Relación de fragmentos HTML y relevancia

El resultado de este análisis ofrecerá una lista de términos basada en las repeticiones de los mismos en el documento, pero también atenderá a la localización de los mismos en el texto. Términos contenidos dentro del título del documento tendrán mucha más relevancia que otros contenidos unicamente en la descripción del mismo. Además, muchos de los términos que deberían aparecer en el título aparecerán seguramente en el resto de fragmentos del documento. Por esta razón, aunque se de más importancia a términos que aparecerán en ciertas áreas del documento, si éste está bien formado y tiene una organización coherente, ofrecerá a los buscadores un esquema sencillo apuntando al posicionamiento del mismo.

6.8. Análisis y creación del índice semántico de documentos

Para conseguir realizar este tipo de análisis sobre un conjunto de documento, por lo general, se tiende a utilizar bases de datos que contengan información relativa al significado de las palabras, los verbos, los nombres y en definitiva, que permitan extraer cierta información morfológica de los textos. La razón fundamental de esto es que los sistemas de información, de manera natural, no son capaces de entender el significado de un texto ni de realizar análisis exhaustivos de la estructura gramatical a no ser que hayan sido programados para tal efecto. Algunas de estas bases de datos están disponibles de manera gratuita, otras para motivos educacionales o de investigación y algunas, generalmente las más completas, pertenecen a grandes empresas de sectores tecnológicos. Estos conjuntos de información normalmente están enfocados hacia un idioma específico aunque en ocasiones pueden encontrarse conjuntos de información para diversos idiomas. Una de las bases de datos de información más conocida y utilizada para fines generalmente educacionales y de experimentación es Wordnet [16], enfocada al inglés. En el caso del castellano, una gran cantidad de información puede encontrarse en recursos gestionados por la Real Academia Española de la Lengua [17]. En este proyecto, se utilizarán los recursos de la Real Academia de la Lengua debido a que está orientado al análisis en documentos en castellano. En el caso de que fuese una necesidad ampliar este conocimiento sería necesario enriquecer la base de datos del sistema con información proveniente de otras fuentes.

En adelante se describen los detalles de las diferentes vertientes semánticas que pueden trabajarse a la hora de introducir estos mecanismos en el análisis de los documentos.

6.8.1. Entidades

Conocer qué entidades tienen un cierto peso dentro de un documento es crucial si nuestra intención es ofrecer la posibilidad de realizar búsquedas sobre estos elementos semánticos. Para ello es necesario extraer la información adecuada del documento y si es posible relaciones con las entidades que intervienen en el mismo. Para entender el concepto de entidad es necesario acotar el conjunto de entidades y tener claro qué información se quiere destacar. Una primera aproximación de los subconjuntos de entidades que pueden encontrarse en un documento puede ser el siguiente:

- **Personas**.
- **Lugares**: países, provincias, comunidades autónomas, ciudades.
- **Nombres comerciales**: productos, modelos, marcas registradas, empresas, siglas.

Para extraer las entidades contenidas en un texto concreto, como se comentaba anteriormente, se pueden utilizar diferentes aproximaciones. Una posible aproximación sería el propio análisis del texto, utilizando reglas que permitan la extracción de palabras que estén escritas con mayúsculas y no sean el inicio de una frase por ejemplo, pero eso no nos posibilitaría ofrecer información de qué tipo de entidad estamos tratando. La segunda opción es hacer uso de algún conjunto de información del que ya hemos hablado con anterioridad o generar manualmente esta estructura de conocimiento. En este caso, es necesario valorar qué tipo de entidades deseamos catalogar y cuáles no. Esto va ligado intrínsecamente a la capacidad de construcción u obtención de la información que nos permita realizar este análisis y esto es al fin y al cabo la estructura de conocimiento con la que el sistema trabajará. En el caso de los nombres comerciales, puede hacerse uso de cierta información también, pero abarcarla toda es muy complicado y por lo general mucha información referente a productos comerciales es privada o está protegida por lo que hacer un uso de una manera gratuita de esta información puede ser complicado. En el caso de las entidades correspondientes a lugares, esta información puede recuperarse de una manera sencilla si la información está acotada:

- Listado de países,
- Listado de ciudades con más de un habitantes número de habitantes significativo.
- Listado de municipios de un país: Recuperar la información referente a los municipios de España es bastante sencillo, basta con consultar los datos ofrecidos por algún organismo oficial de estadística o censos de población activa.
- Listado de gentilicios relativos a países y provincias de un país: Puede asociarse la información descriptiva de un nombre concreto a la entidad de localización que tiene asociada.

Además de estas implicaciones algunas otras evoluciones de las mismas pueden tomarse en cuenta para tener más información semántica de lo que ocurre en el documento. En el caso de estas entidades, que podemos denominar simples o que definen un elemento concreto, se podría llegar a analizar un texto con el fin de recuperar enumeraciones de entidades o yuxtaposiciones en el texto que indiquen una asociación o agrupación de las mismas.

Por otro lado, una aproximación morfosintáctica nos ayudaría a comprender información sobre la entidad que actúa en una frase concreta del texto y localizarla de una manera más sencilla. Si en una frase podemos detectar qué acción tiene lugar y quién la realiza, se puede obtener información sobre ambos extremos, teniendo así la capacidad de relacionar la entidad con su acción y viceversa.

Una vez se tenga decidido sobre que espectro de entidades se quiere analizar los documentos, el siguiente paso sería la realización del análisis y la incorporación del mismo en al conjunto de procesos que cotejan los documentos. En el caso del sistema que tenemos ahora mismo presente, el análisis

semántico de entidades se localizaría en un proceso SEMANTIC, que abarcaría todas las posibles combinaciones.

6.8.2. Análisis morfológico

Este tipo de análisis de los textos solo puede realizarse sobre frases completas y además puede llevar a una complejidad bastante notable si una frase está comprendida por varias como en el caso de oraciones gramaticales complejas. Por otro lado, este tipo de análisis se centra sobre un idioma específico por lo que el hecho de tener la posibilidad de conseguir contenido documental de distintos idiomas conllevaría tener que analizar un análisis previo de los textos con el objetivo de conocer el idioma sobre el que trabajaremos.

En cuanto al análisis en sí, la información que se quiera recuperar dependerá de la complejidad con la que queramos dotar al sistema. Un primer acercamiento es la realización de un análisis morfosintáctico de oraciones simples que contengan un solo verbo. La localización de los verbos es de vital importancia debido a que ofrecen información de la acción que ocurre en la frase y de cuál es el sujeto de la frase, lo que implica que podemos tener información tanto de la entidad que realiza cierta acción como de las posibles implicaciones de la acción. Esto último va ligado a la extracción de complementos sintácticos de la frase como objetos directos, indirectos o complementos circunstanciales de una ocurrencia.

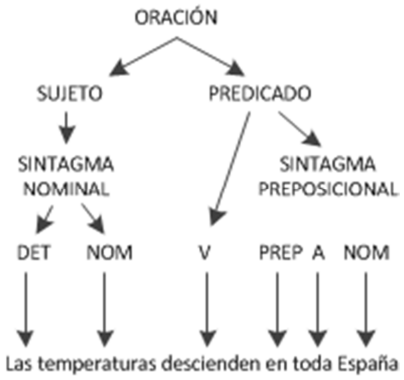

Figura 8 - Estructura de una oración simple

Para este proyecto en particular, se ha generado una base de datos de cerca de 800.000 conjugaciones verbales y sus relaciones tanto con la raíz verbal como con el tiempo verbal en el que se conjugan. Esta base de datos ha sido elaborada a partir de la información recuperada del Corpus de la Real Academia de la Lengua Española [18].

Conociendo esta información podemos obtener información de qué o quién realizó qué acción en un momento temporal (conocemos cuando el documento fue creado si tiene la información meta adecuada) y si es posible otra información relacionada. Esta información, generalmente contenida a modo de complementos en las frases de los textos, puede contener elementos que nos identifiquen algún tipo de relación entre las entidades e información secundaria. Los elementos que pueden darnos más información en este aspecto son las preposiciones. Un listado de preposiciones se puede generar rápidamente ya que éstas están muy acotadas a nivel gramatical en todos los idiomas y además ofrecen una gran información semántica contenida en los textos.

Por otro lado, es necesario tener en cuenta que el sistema, aunque sea capaz de conocer entidades, necesitará algún procedimiento añadido para recuperar información de entidades que son reconocidas pero están incompletas en los textos. Este es el caso de las entidades de persona que en muchas ocasiones son nombradas en textos mediante su apellido, apodo o similares. Para que estos casos puedan ser involucrados en los resultados apropiados ante la búsqueda de dichos términos en

documentos relevantes, es necesario elaborar un sistema satélite que permita evaluarlos. Una posible solución a este problema puede ofrecerse mediante la evaluación de entidades mediante candidatos, es decir, en el caso de que el proceso encuentre información relacionada con una entidad, no dar ese resultado como bueno hasta que haya sido superado un número de encuentros apropiado a lo largo del tiempo.

A continuación se enumeran algunos ejemplos de frases y las distintas acciones que deberían tenerse en cuenta a la hora de elaborar el procedimiento semántico en el motor de búsqueda:

- "La fiscalía pide el archivo de la causa contra Messi por delito fiscal"
    - El sistema deberá reconocer "Messi" como entidad.
    - Además, sería necesario que la entidad Messi esté relacionada uno a uno con la entidad "Lionel Messi" y en caso de encontrar entidades similares como por ejemplo "Leo Messi" también deberían reconocerse como la misma entidad.
    - La entidad "Lionel Messi" aparecería relacionada con el término "delito fiscal", que podría evaluarse como una entidad aparte.
- "Fernando Alonso obtuvo el primer puesto en el Gran Premio de Japón."
    - La entidad "Fernando Alonso" estaría relacionada con el "Gran Premio de Japón".
    - Del mismo modo, dicho documento podría ser evaluado como contenido geográfico orientado a "Japón" por el hecho de contener dicha entidad de localización.
- "El general Al Sisi apuesta por un gobierno continuista en Egipto"
    - La entidad "Al Sisi" estaría relacionada con "Egipto".
    - Del mismo modo que en el ejemplo anterior, el documento estaría relacionado a nivel de localización con "Egipto".
    - Además, se podría extrapolar la acción "apostar" generando una relación entre la entidad "Al Sisi" y un complemento de significado de la acción que en este caso sería "un gobierno continuista".

6.8.3. Relaciones de entidades con complementos

Como hemos visto en algunos casos, puede ser necesario obtener una relación entre una entidad y un conjunto de información. Este conjunto puede ser una o más entidades, una localización o información en contextos diferentes. Para detectar estos casos y generar relaciones entre entidades y estos complementos se tomarán algunas medidas que utilizarán información que hemos recuperado previamente u otra que puede ser de gran valor:

- **Gentilicios**: Mediante un análisis de las frases y su extracción de gentilicios se pueden crear relaciones con entidades que aparezcan en la misma frase o en contiguas. Esto puede ofrecer información de gran valor ya que se conocería la procedencia de una entidad de persona o de un elemento que haya sido creado o engendrado en una localización en particular. Mediante el uso del método de candidatos, el proceso deberá elaborar relaciones de procedencia o descriptivas entre las entidades encontradas y las localizaciones relacionadas con un gentilicio. Por ejemplo: en un contexto particular de una noticia donde se hable de la entidad "Antonio Banderas", por lo general, para no nombrar continuamente a la misma persona en una noticia, los periodistas acuden a los gentilicios para nombrar o describir acontecimientos. En estos casos, es muy utilizado el gentilicio de "malagueño". Esta relación para nosotros es evidente, pero el sistema deberá detectarla y posicionarla como una relación candidata a ser usada y ofrecida en los resultados. A lo largo del tiempo, el sistema irá reconociendo la misma relación numerosas veces hasta que sea adecuada para su utilización para ofrecer contenido relevante.

- **Sintagma nominal**: En muchas ocasiones, el proceso de análisis semántico encontrará frases del tipo "El presidente Barack Obama se reunió con su homónimo español". En este caso la entidad "Barack Obama", para nosotros estaría claramente relacionada con la presidencia de un país y el sistema debería ser capaz de detectarla. Este tipo de estructuras ARTICULO + NOMBRE + ENTIDAD + VERBO es fácilmente detectable y relacionar el término "presidente" con "Barack Obama" mediante una relación de descripción es bastante sencillo. Del mismo modo que en el caso anterior, estas relaciones deberían pasar un proceso por el cual sean candidatas durante un tiempo hasta que se obtengan los resultados apropiados como para obtener relevancia de documentos a través de ellas.
- **Unión de entidades en el sujeto**: En algunos casos se puede encontrar frases que contengan varias entidades que realicen cierta acción. Un ejemplo podría ser la siguiente frase: "Barack Obama y Mariano Rajoy firmarán un pacto para mejorar..." En este caso, la acción "firmar" tendrá una relación con las entidades "Barack Obama" y "Mariano Rajoy". El sistema deberá detectar dicha relación de yuxtaposición o enumeración en el caso de haber más entidades y obtener la información relativa a la acción para relacionarlas.

6.8.4. Proceso de análisis semántico

Aunque se ha hablado de ello pero sin entrar en detalle, vamos a entender cómo debería producirse el proceso de análisis semántico de los documentos.

Para la realización del análisis semántico de los textos se utilizará el texto en claro que será recuperado del recurso que almacenó el proceso SPLIT. Dicho texto en claro deberá ser dividido en frases, el sistema debe ser capaz de detectar el inicio y el final de una frase. Una vez tenemos la colección de frases que conforman el texto del documento habrá que procesar con cada una de ellas una serie de módulos que podrán activarse o desactivarse para realizar las acciones pertinentes. Para una mayor sencillez en la búsqueda del análisis semántico del documento vamos a reducir el número de frases a aquellas que contengan un único verbo, es decir aquellas que son oraciones morfológicamente simples.

Para cada frase se realizará el siguiente proceso:

- Buscar el verbo en la frase contrastando con la base de datos de verbos conjugados y extraer el infinitivo.
- Búsqueda de entidades de localización en el sujeto de la frase.
- Búsqueda de entidades en el sujeto de la frase.
- Relaciones de identificación de entidades en el sujeto de la frase.
- Relación de entidades del sujeto con la acción y el momento temporal en el que sucede. Esta información se extrae del tiempo verbal de la conjugación del verbo y de la información meta del documento si existe.
- Búsqueda de entidades de localización en el predicado de la frase y asociación a las entidades del sujeto a través de la acción.
- Búsqueda de entidades del predicado de la frase y asociación a las entidades del sujeto a través de la acción.
- Relación de la URL del documento con las entidades aparecidas en el resultado del análisis.

A través de este proceso de análisis, se pude recuperar información muy relevante a nivel semántico de los textos y además conocer la estructura de las frases. Estas operaciones son llevadas a cabo de manera particionada por parte del proceso SEMANTIC.

6.9. Planificación de la recuperación de documentos

El objetivo de realizar una planificación de la recuperación de los contenidos no se reduce solo a la obtención de información sobre documentos nuevos sino de mantener la información almacenada actualizada a lo largo del tiempo. Debido a que el alcance del procesamiento de cada uno de los documentos debe ser extrapolado hacia delante en el tiempo con el fin de comprobar futuras actualizaciones de contenido y salvaguardar la información almacenada en el sistema. Aunque muchos documentos cambiarán mucho a lo largo del tiempo, no todos lo harán y una gran mayoría llegarán a ser estáticos en concepto de contenido para siempre. El objetivo final de este proceso es ir introduciendo de nuevo en la cola de procesamiento tanto nuevos documentos como aquellos para los cuales ya conocemos su contenido.

La planificación de la recuperación de contenidos permitirá no solo obtener información de nuevos documentos, sino de recuperar información actualizada de los que ya se conocen, pero además, comprobar si esos documentos han cambiado lo suficiente a lo largo del tiempo como para adelantar su recuperación y reprocesamiento. Esta planificación se realiza en los siguientes pasos:

6.9.1. Extracción de los documentos

Se basa en la recuperación de los documentos a los que apunta un elemento en la cola de procesamiento. Para ello se recuperará el listado de hipervínculos extraído durante la ejecución del proceso SPLIT. En ese momento, se extrajo la información contenida en el documento que se estaba analizando para usarse más adelante. En este caso, uno de los fragmentos del contenido son los hipervínculos dispersos por el texto del documento. Cada uno de ellos será introducido de nuevo en la cola de procesamiento si fuese necesario.

6.9.2. Recuperación del contenido significativo

Cada entrada del fragmento de contenido del que habla el punto anterior contiene información referente al documento que referencia que nos servirá para tener más información sobre su contenido. Un hipervínculo en formato HTML tendrá el siguiente aspecto dentro de un documento:

```
<a href="/tag/siria/a/" title="Noticias sobre Siria">El conflicto sirio</a>
```

De este hipervínculo se puede extraer información incluida en el propio contenido del nodo XML o en los atributos de la misma. En este caso la información contenida en el mismo es la siguiente:

- **URL del documento al que apunta**: "/tag/siria/a/"
- **Contenido del hipervínculo**: "El conflicto sirio"
- **Título del hipervínculo**: "Noticias sobre Siria"

Con esa información podemos tener un alcance mayor del contenido que nos encontraremos en el documento. La URL nos servirá para poder introducir el documento de nuevo en la cola y realizar la ejecución del proceso FETCH que lo recuperará. El contenido y el título nos servirá para tener una información primaria del contenido incluido en el documento y puede servirnos para tener cierta información que quizá no encontremos en el documento objetivo.

Como nota final a este paso, la URL en el ejemplo resulta ser relativa al dominio del documento desde el que se ha extraído el hipervínculo. Será necesario concatenar la información de dominio del documento analizado junto a la URL objetivo para obtener la ruta final del documento a analizar a continuación.

6.9.3. Algoritmo de planificación de la recuperación

Una vez conocemos los documentos que queremos recuperar en las siguientes ejecuciones debemos saber cuándo hacerlo. Para saber cuándo realizar una recuperación de un documento es necesario saber si anteriormente ya se ha recuperado y si ha sido así y conocemos su contenido en varias ejecuciones, conocer cuánto ha cambiado. Esto es debido a que si el documento no ha cambiado lo suficiente durante varias recuperaciones, deja de tener sentido recuperarlo en intervalos de tiempo

muy reducidos porque el resultado del análisis será el mismo. Sin embargo, nunca debemos dejar de recuperar el contenido de un documento por el hecho de que no haya cambiado porque no podemos saber a ciencia cierta si esto puede ser distinto en un futuro cercano. Este hecho viene referenciado por lo que se ha denominado índice de diferencia temporal de un documento.

Durante el proceso de análisis básico de los términos contenidos en el documento (el proceso RANK) se ha generado un fichero que almacena las frecuencias de los términos del documento. Este fichero es contrastado con el que se hubiese generado durante un proceso de análisis anterior del mismo documento. Mediante una comprobación de cada uno de los términos existentes y los no existente se extrae el número de diferencias y términos totales del nuevo documento frente al recuperado con anterioridad. Una vez obtenido el número de diferencias y términos se genera la proporción de diferencia por cada término en el documento. Este valor obtenido de la proporción lo denominaremos índice de diferencia.

$$indice\ de\ diferencia = d\ = \frac{términos\ con\ frecuencia\ modificada + total\ términos\ nuevo\ documento}{total\ términos\ antiguo\ documento}$$

- El índice de diferencia tomará valores entre 0 y 1, pudiendo ser mayor que 1 siempre y cuando un gran número de términos nuevos sean incluidos en la nueva versión del documento.
- Tomará valores cercanos a 0 cuando la diferencia sea mínima.
- Tomará valores cercanos a 1 cuando la diferencia sea máxima. Incluso tomará valores mayores que 1 para los casos en que existan muchos nuevos términos.

Para normalizar el valor del índice de diferencia se calculará el valor del índice de cambio de un documento a través de la siguiente ecuación:

$$indice\ de\ cambio = c\ = \begin{cases} 1 & si\ d \leq 0{,}5 \\ sen(\pi d) & si\ 0{,}5 > d \leq\ 1 \\ -d + 1 & si\ d > 1 \end{cases}$$

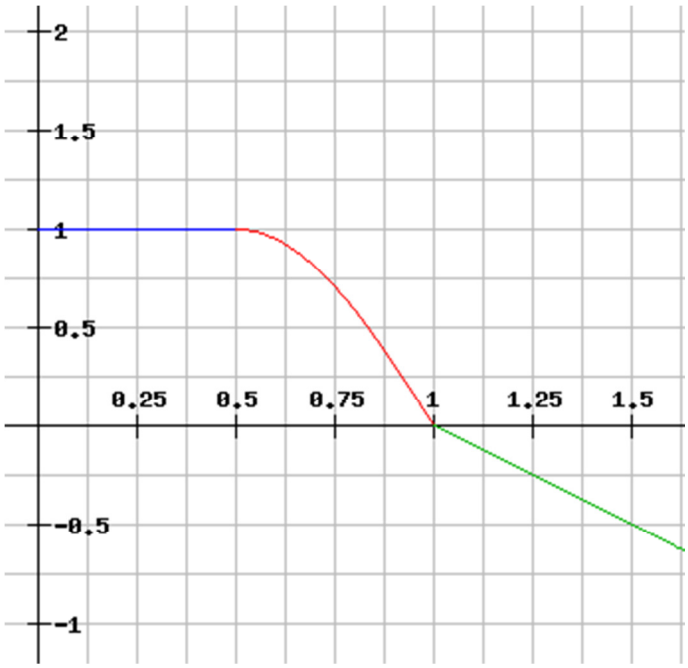

Figura 9 - Representación gráfica del índice de cambio

El algoritmo de planificación permitirá obtener un valor temporal para el cual el documento será recuperado y procesado. Aunque en realidad el valor temporal resultante no será el momento en que el documento se volverá a recuperar, sí permitirá ordenar los documentos para ser procesados en orden y así que aquellos que deben procesarse antes lo hagan de esta manera.

Se utilizará el algoritmo expuesto a continuación para calcular la planificación de la recuperación y procesamiento del documento. Este algoritmo hace uso de un valor almacenado para el documento que ya fue recuperado con anterioridad el incremento utilizado en ese momento. El incremento resultante será aplicado sobre la hora actual del sistema para calcular cuando se realizará la próxima recuperación del documento. Para la primera recuperación y procesamiento del documento no se tendrá un valor de incremento anterior, pero este sí se calculará al finalizar todo el proceso. Este factor, que al fin y al cabo repercute en el tiempo de cuando un documento es recuperado está intrínsecamente relacionado con el índice de cambio del documento a lo largo del tiempo. El índice de cambio es el que determina el incremento temporal sobre el momento actual.

Además, se utilizarán tiempos de planificación base e incrementos base que permitirán posponer la recuperación de los documentos en una horquilla de tiempo suficiente para que la cola de documentos a procesar no se vea alterada por la recuperación inmediata de documentos que ya se han procesado. Tomando los siguientes supuestos:

- Para un documento que termina su procesamiento en un momento t_0,
- Y se quiere planificar para que su planificación genere un momento de procesamiento futuro t_1,
- Cuyo índice de cambio es c,
- Y que i_0 es el incremento efectuado en el último procesamiento,
- T_{BASE} es una constante que define el tiempo de planificación base por defecto,
- I_{BASE} es una constante que define el incremento base por defecto,
- Teniendo en cuenta que i_1 es el incremento de t_1 sobre t_0.

$$i_1 = I_{BASE} \cdot c$$

$$t_1 = t_0 + T_{BASE} + i_0 + i$$

Caso de uso:

- Primer procesamiento del documento d. Cálculo de frecuencia de aparición de los términos:
  - $k_o = 3$
  - $k_1 = 2$
  - $k_2 = 1$
  - $k_3 = 4$
  - $t_1 = t_0 + T_{BASE} + 0 + 0 = t_0 + T_{BASE}$
  - La planificación genera un tiempo de planificación estándar para el siguiente procesamiento.

- Segundo procesamiento del documento d. Cálculo de frecuencia de aparición de los términos:
    - $k_o = 3$
    - $k_1 = 2$
    - $k_2 = 1$
    - $k_3 = 3$
    - $d = \frac{2}{4} = 0{,}5$
    - $c = 1$
    - $i = I_{BASE} \cdot 1$
    - La planificación genera un tiempo de planificación estándar más un incremento estándar para el siguiente procesamiento.
- Tercer procesamiento del documento D. Frecuencia de aparición de términos:
    - $k_o = 3$
    - $k_1 = 2$
    - $k_2 = 1$
    - $k_3 = 3$
    - $d = \frac{3}{4} = 0{,}5$
    - $c = 1$
    - $i = I_{BASE} \cdot 1$
    - $t_1 = t_0 + T_{BASE} + (2 \cdot I_{BASE})$
    - La planificación genera un tiempo de planificación estándar más el doble del incremento estándar debido a que el documento no ha cambiado prácticamente nada.
- Cuarto procesamiento del documento D. Frecuencia de aparición de términos:
    - $k_o = 6$
    - $k_1 = 4$
    - $k_2 = 2$
    - $k_3 = 5$
    - $d = 4$
    - $c = -1$
    - $i = -I_{BASE}$
    - $t_1 = t_0 + T_{BASE} + I_{BASE} - I_{BASE} = t0 = T_{BASE}$
    - La planificación genera un tiempo mucho más bajo que los anteriores porque el documento ha sufrido un gran número de cambios. Se genera un tiempo para el procesamiento futuro estándar.

6.10. Recursos externos

6.10.1. Lenguaje de programación e implementación

Para la implementación del sistema, se utilizará un lenguaje basado en el paradigma de la orientación a objetos. Debido a que será un sistema web y queremos que conviva fácilmente en servidores orientados a tal efecto se utilizará PHP. Este lenguaje es gratuito y libre de uso, está plenamente extendido sobre todo en plataformas orientadas a la web y su documentación es muy completa.

Además, los servicios que ofrece el lenguaje y su funcionamiento atiende a las necesidades que se tienen en el desarrollo del sistema, tanto de recursos necesarios para su acceso, bases de datos, sistemas de ficheros y cachés.

Se utilizará versión PHP 5.5.14 que la actualmente estable. A partir de las versiones 5, la orientación a objetos en el lenguaje tiene funcionalidades completas al contrario que con versiones anteriores. Por tanto, esta versión del lenguaje nos sirve suficientemente para desarrollar el sistema.

6.10.1.1. RX Framework

Al igual que otros muchos frameworks, RedXenon PHP Framework es un conjunto de librerías y clases que permiten realizar un desarrollo en el lenguaje de programación PHP mucho más sencillo, limpio y fácil de mantener. Este framework ha sido realizado por el autor de este proyecto con el fin de ser utilizado en diferentes proyectos. El conjunto de librerías ha ido evolucionando desde su nacimiento en el desarrollo de trabajos dirigidos en la Universidad Carlos III de Madrid y actualmente es usado en algunos sitios web profesionales.

La utilización del framework es bastante simple, solo es necesario hacer uso de las librerías cuando sea necesario mediante la importación de las mismas cuando sea necesario. Algunas de las librerías permiten ser extendidas desde otras clases de PHP, con el fin de que el desarrollador amplíe su funcionamiento y utilice funciones previamente implementadas.

Aunque existen muchos frameworks libres y gratuitos en el mercado, RedXenon PHP Framework tiene la flexibilidad suficiente como para elaborar tareas necesarias en sitios web generalistas y en proyectos afines. Las posibilidades del framework permiten la utilización y extensión de clases que facilitan el desarrollo de productos software a medida, en concreto consta de los siguientes paquetes de clases:

- **RXDate**: Un conjunto de clases que permite un manejo mucho más sencillo de elementos de fecha y temporales. Se abstraen mediante funcionalidades nativas las dificultades encontradas en el desarrollo de sitios multilenguaje mediante traducciones en tiempo de ejecución a través de diccionarios añadidos al paquete. Del mismo modo, permite centralizar la gestión de la información relativa a fechas y horas en distintos husos horarios con el fin de orientar el desarrollo a sitios web y proyectos internacionales.
- **RXDatabase**: Se trata de un paquete de clases que permiten abstraer la capa de datos en cualquier desarrollo de software y crear una capa independiente de los controladores y el propio gestor de base de datos. Realiza la gestión de inserciones, modificaciones, borrado y consulta de información sobre algunos de los distintos sistemas de bases de datos existentes en el mercado.
- **RXCurl**: Capa de abstracción de la librería Curl, explicada en esta misma sección del documento, más adelante. Debido a que por lo general la librería Curl utiliza un gran número de configuraciones necesarias para la realización de peticiones a servidores HTTP complejas, esta capa de abstracción facilita la utilización de esta librería sin perturbar la codificación de funcionalidades en nuestro producto de software.
- **RXCache**: Una capa de abstracción más para la utilización de caches en proyectos de software principalmente orientada a sitios web que precisen un alto rendimiento. Este conjunto de librerías permite la utilización de una manera sencilla de cachés basadas en

sistemas de fichero, en memoria o en estructuras de datos complejas y cuya estructura está orientada al almacenamiento clave-valor.

- **RXView**: Permite abstraer el tratamiento de la información previo a la representación de la vista por medio de un motor de templates o la utilización de uno de terceros. Este paquete de clases es compatible con el paquete de cachés, con el fin de la generación de vistas cacheadas de manera dinámica.
- **RXFile**: Gestión de ficheros general.
- **RXLog**: Gestión de logs, niveles de criticidad y alertas por correo electrónico cuando sea necesario.
- **RXForm**: Este paquete permite la generación de elementos y conjuntos de elementos que formen parte de un formulario HTML.

6.10.1.2. Librería LibCurl

Se trata de una librería usada comúnmente para la realización de peticiones HTTP a través de la red que vayan a ser interpretadas por un servidor y que se desea recuperar una respuesta. Esta librería es de uso muy común en diversos tipos de desarrollo de software y además de estar disponible en distintas plataformas y sistemas operativos, muchos lenguajes de programación ya la incluyen como funcionalidades nativas.

La librería ofrece una gran cantidad de posibilidades de configuración de las cabeceras HTTP que se envían junto con la petición, la gestión de los códigos de respuesta permitidos por el protocolo HTTP y multitud de diversas opciones añadidas. Además, la librería permite realizar peticiones a servidores que usen el protocolo seguro HTTPs y gestiona autenticaciones de HTTP Basic.

En este proyecto en concreto, esta librería deberá integrarse en el proceso FETCH que permite la recuperación de contenidos desde servidores de información. A través de ella se realizarán peticiones contra los recursos, documentos o colecciones de documentos que sean objetivo de su recuperación y ofrecer una respuesta al sistema del resultado de las peticiones.

6.10.1.3. Librería iconv

Se trata de una librería que permite convertir los juegos de caracteres de elementos textuales, ficheros y derivados por otros formatos que estén estandarizados. Esta librería, además de contemplar las codificaciones de caracteres estandarizadas, está disponible en diversas plataformas y sistemas operativos. Además, es posible realizar llamadas a funciones implementadas de manera nativa en numerosos lenguajes de programación que utilizan la librería de manera interna o se puede desplegar dentro de un desarrollo desde un recurso externo.

En este proyecto en concreto, esta librería deberá integrarse en el proceso FETCH que realiza la recuperación de los contenidos de un documento. Debido a que el sistema trabajará con la codificación de caracteres UTF-8, cualquier documento que esté codificado con otro juego de caracteres deberá ser convertido a UTF-8 previa detección de la configuración de caracteres del documento. Es necesario realizar esta comprobación y la posterior recodificación de aquellos documentos que no compartan la codificación UTF-8 debido a que cualquier tratamiento de esa información será relacionada con contenido que ya esté codificado con dicho juego de caracteres. Si se diese el caso de que no se puede recuperar la información de codificación de un documento concreto, debería desecharse para no ensuciar el contenido almacenado en la base de datos y en definitiva, ofrecer resultados que tengan caracteres mal formados.

6.10.2.Plataforma

El sistema será implementado en una plataforma con sistema operativo Linux. Esta decisión es tomada debido a que por un lado, será una plataforma que aunque pueda tener cabida dentro del ámbito comercial deberá ser gratuita. Muchas distribuciones Linux cumplen los requisitos necesarios para realizar la labor de servidor web. Además, los repositorios de librerías y de actualizaciones del

software son de fácil acceso y la gestión de los paquetes que fuesen necesarios instalar para realizar actualizaciones es muy cómoda.

6.10.3.Servidor HTTP

Un servidor HTTP es lo que permite realizar las comunicaciones entre el servidor y los clientes a través del protocolo HTTP. Ante una petición contra el servidor realizada por un usuario, es necesario que el servidor pueda procesarla y además ofrecer una respuesta. La gestión de estas peticiones y las respuestas realizadas, con el procesamiento necesario de la información y su devolución al usuario la realizará un servidor web.

Actualmente existen múltiples servidores web en el mercado entre los cuales existen algunos que son ampliamente utilizados y otros que además son de código abierto. Aquellos que se encuentran dentro del grupo de servidores HTTP de código abierto además ofrecen sus actualizaciones a través de repositorios de libre acceso por lo que el mantenimiento es bastante eficiente.

Es necesario tener en cuenta que para la elección de un servidor HTTP que de soporte a nuestro sistema debe ofrecer una eficiencia suficiente como para procesar una gran cantidad de peticiones de usuarios. Debido a que el lenguaje de programación que se utilizará para el desarrollo del sistema será PHP 5.3, el servidor HTTP no precisará de características por ejemplo de Java Servlets, sino que utilizará una configuración o módulos específicos para poder realizar la ejecución de PHP. Debemos considerar servidores web que contengan la posibilidad de activar ciertas características como:

- **Soporte SSL:** Con el fin de que las comunicaciones entre el servidor y los clientes se realicen de manera segura, se habilitará esta opción por medio de un certificado SSL emitido por una autoridad de certificación de confianza. Autoridades de certificación como Verisign (ahora Symantec), GeoTrust, Comodo o similares pueden ofrecer certificados SSL para cifrar las comunicación [19].
- **Virtual Hosting**: Mediante la configuración apropiada, se puede dotar a una sola instancia del servidor HTTP de distintas ejecuciones de servicios.
- **IPv6**: Es siempre interesante que el servidor HTTP esté actualizado a la última versión del protocolo IP.

A continuación, vamos a comparar algunos de los servidores HTTP que se han encontrado más interesantes dentro de las características que se han enumerado. Las comparaciones se realizarán a modo de análisis de eficiencia y de uso de recursos que utiliza el servidor HTTP. De esta manera tenemos los siguientes puntos a tener en cuenta:

- **Lighttpd:** Es un servidor web diseñado para ser rápido, seguro, flexible y que sigue los estándares. Su arquitectura le permite optimizar los recursos consumiendo menos CPU y memoria RAM que otros productos. Es un software libre y funciona en plataformas GNU/Linux.
- **Apache HTTP Server:** Es un servidor HTTP ampliamente conocido por su temprana aparición en 1995. Este servidor, es utilizado en el 54,2% de los sitios web del mundo. Se basa en una arquitectura modular multiproceso.
- **Nginx**: Es un servidor HTTP que ofrece un alto rendimiento con un gran número de peticiones al segundo. Está especialmente orientado a servir ficheros estáticos e índices.

A continuación deberemos decidir qué solución utilizar en la implementación del sistema. Para ello se realizará una investigación entre cuales de los sistemas ofrecen un mejor rendimiento. De esta manera, se establecerá una serie de pruebas contra estos servidores funcionando en una máquina con el fin de esclarecer las diferencias entre el uso de memoria, utilización del procesador y tiempo de respuesta.

Estas pruebas son realizadas mediante la ejecución de ApacheBench [20] Una herramienta que permite realizar pruebas de concurrencia contra servidores HTTP. De esta manera, se establece una

concurrencia de 10.000 peticiones HTTP contra una imagen localizada en el servidor y de manera concurrente por 50 clientes en ejecución simultánea.

```
ab –n 10000 –c 10 http://localhost/test_image.png
```

Los resultados obtenidos son los siguientes:

SERVIDOR	Apache 2.2.16	Nginx 0.7.67	Lighttpd 1.4.26
Uso medio CPU	13,52 %	0,77 %	0,74 %
Consumo medio de memoria	164,66 MBs	17 MBs	15,6 MBs
Peticiones por segundo	393,78	412,61	443,05
Tiempo medio por petición	2,53 ms	2,41 ms	2,29 ms

Tabla 8 - Comparativa de servidores HTTP

En los datos se puede observar como Apache queda muy por detrás en el uso de recursos contra Nginx o Lighttpd. Sin embargo es necesario tener en cuenta que el funcionamiento habitual de Apache es mediante hilos de ejecución lo que beneficiará su utilización en máquinas con varios procesadores. Además, Apache ofrece la posibilidad de utilizar módulos que pueden activarse y desactivarse para su utilización. En el caso que a nosotros nos interesa, PHP funcionaría como un módulo de Apache al contrario que con Nginx o Lighttpd, que funcionaría en modo CGI. Para entender esto se describen ambos modos en los siguientes apartados:

- **PHP en modo CGI**: Esto quiere decir que el servidor HTTP realizará por cada llamada una ejecución de PHP cuando así sea requerido lo que significa realizar la carga de la configuración de librerías y entorno de PHP por cada petición. Esto puede suponer una carga muy elevada para la máquina que estemos utilizando si el número de conexiones es demasiado alto.
- **PHP como módulo**: Las librerías, configuración y en general, el compilador de PHP serán cargadas junto con el servidor HTTP. Esto beneficiará a las respuestas de las llamadas y al uso de recursos de la máquina debido a que por cada petición no deberá realizarse ninguna carga adicional.

Tanto Nginx como Lighttpd necesitan de módulos adicionales para realizar la compilación de PHP en tiempo de ejecución lo cual supone un problema para la generación de contenido dinámico.

La mejor de las soluciones es combinar ambas herramientas y tener la posibilidad de configurar PHP como módulo de Apache y al mismo tiempo enriquecerse de la velocidad de otro servidor al mismo tiempo. En este caso, la elección ha sido Nginx.

Tanto Apache como Nginx son servidores HTTP muy potentes y aunque cada uno de ellos tenga carencias en ciertos aspectos, la unión permite generar unos resultados muy interesantes. De esta manera, podemos configurar Apache para que funcione junto a Nginx, teniendo así la velocidad de Nginx como servidor frontal y la potencia y versatilidad de Apache para procesar las peticiones por detrás.

A la hora de configurar Nginx como servidor frontal, podemos reconfigurar que todas las peticiones que se realicen contra elementos que deban ser procesados en PHP se realicen contra Apache a un puerto en concreto.

6.10.4.Bases de datos

Existen numerosas soluciones en cuanto a gestores de bases de datos se refiere. Vamos a analizar dos de ellos para averiguar cuál es el que deberíamos usar para el desarrollo de nuestro sistema. En

concreto, vamos a analizar PosgreSQL y MySQL. Ambos sistemas gestores de bases de datos son de código abierto y su distribución es gratuita. Aunque es cierto que MySQL dispone de soporte comercial si fuese necesario, ambos proyectos tienen una gran cantidad de documentación y son usados ampliamente por la comunidad de desarrolladores por lo que no se deberían encontrar problemas para utilizarlos. [21]

- **MySQL**:
    - Tiene la posibilidad de utilizar dos tipos de motores de generación para las tablas de la base de datos: InnoDB y MyIsam:
        - InnoDB: Permite realizar transacciones. Realiza un bloqueo sobre los registros para evitar problemas de carrera. Tiene un gran rendimiento cuando se realizan muchas consultas de modificación de los registros de la base de datos.
        - MyIsam: Está especialmente orientado a la consulta, evitando bloqueos de registro lo cual puede suponer problemas de concurrencia, pero elevando la tasa de recuperación de la información.
    - Tiene muy bajo consumo de recursos y su velocidad es muy alta a la hora de hacer operaciones frente a otros gestores.
    - Existen mutitud de herramientas para gestionar las bases de datos, tanto gráficas como en consola.
    - Los conectores, drivers y APIs de conexión contra las bases de datos están plenamente extendidos.
- **PostgreSQL**:
    - Posee una gran escalabilidad, permitiéndose ajustar los recursos utilizados así como el número de procesadores para su ejecución.
    - Implementa transacciones.
    - Tiene la capacidad de comprobar la integridad referencial, así como almacenar procedimientos en la propia base de datos.
    - Los registros están limitados a 8KBytes y aunque pueden aumentarse hasta 32KBytes, el rendimiento se ve muy resentido.
    - Puede llegar a ser un 50% más lento que MySQL en ciertas operaciones debido a su arquitectura.

Una vez analizados ambas posibilidades, nos decantaremos por MySQL ya que ofrece mayor versatilidad en cuanto a elección de motores y a su vez la velocidad de consulta es mayor. Debido a que los procesos del sistema tendrán que ejecutar desde distintas máquinas y realizarán muchas consultas de manera concurrente, es importante la elección del motor InnoDB en nuestra base de datos MySQL.

6.10.5.Cachés

Del mismo modo que ocurre con las bases de datos, existen gran multitud de productos de sistemas de cachés en el mercado. Muchos de ellos son gratuitos y sus características son muy diferentes. En este caso, necesitaremos un sistema de cachés que nos permita tener un rendimiento muy elevado para poder realizar tanto escrituras, actualizaciones y consultas de la manera más rápida posible.

Los sistemas de cachés seleccionados serán en esencia una base de datos, que por su tipología y en concreto su arquitectura ofrecen al usuario una gran capacidad y rendimiento. La mayoría de los sistemas de cachés de bases de datos son sistemas de bases de datos no relacionales y están orientados a almacenar su contenido en memoria para que el acceso sea mucho más rápido.

Para la elección del sistema de cachés que se utilizará en el sistema, se realizará un análisis de rendimiento y las características de varios sistemas de cachés, en concreto: Memcache, Tokyo Tyrant y MongoDB. Aunque existen muchos más productos en el mercado, nos decantaremos por estas que además de tener soporte para el lenguaje de programación elegido para el desarrollo, tienen gran cantidad de documentación, ejemplos y ayudas realizadas por la comunidad de desarrolladores.

- **Memcache**: Es un sistema de cachés que almacena información en la memoria de la máquina. Es uno de los más rápidos del mercado y es ampliamente usado en el sector. Se trata de un sistema de base de datos no SQL que almacena los registros de información en pares clave – valor. De esta manera, su acceso debe realizarse por una clave, impidiendo su consulta iterativa. El mayor inconveniente de Memcache es que es volátil, en el caso de que la máquina donde esté funcionando se reinicie o el propio servicio del sistema de caché deje de funcionar, la información se perderá.
- **Tokyo Tyrant**: Es un sistema similar a Memcache, no SQL y con estructura clave – valor. Sin embargo, permite realizar un respaldo de la información almacenada en ficheros físicos en el disco duro. Estos recursos físicos se pueden programar para que la copia de información se realice cada cierto tiempo y así mantener los datos en un lugar seguro en caso de que pueda haber problemas. Otro de las características interesantes de Tokyo Tyrant es la posibilidad de realizar consultas ya no solo por la clave, sino por ciertos elementos almacenados en el valor del registro.
- **MongoDB**: Es un sistema similar a los anteriores pero está orientado al almacenaje de documentos identificados por una clave. A diferencia de los anteriores, permite acceder a la información por medio de consultas similares al lenguaje empleado en otro tipo de bases de datos SQL. Por medio de estas consultas, se pueden realizar filtrados, listados y ordenaciones de los resultados obtenidos. Además, al igual que Tokyo Tyrant, permite realizar un respaldo de la información en ficheros físicos para su recuperación en caso de problemas.

Ambos sistemas de cachés permiten realizar consultas tanto de lectura como de escritura en modo de listado. Esto significa que una sola petición contra el servidor de caché nos podrá devolver varios resultados o realizar la escritura o actualización de registros de una sola vez.

Es muy importante tener en cuenta que para la utilización de cualquier sistema de caché es necesario tener un respaldo. Este respaldo ya no es solo de la propia caché sino también de la información. En nuestro caso, el respaldo se encuentra en la base de datos transaccional de MySQL por lo que la información se podría volver a procesar para su utilización desde caché.

Debido a que nuestro interés de poder tener un respaldo de la información es muy importante, deberemos decantarnos por Tokyo Tyrant o MongoDB. Ambos sistemas permiten realizar, además del respaldo de la información, una sincronización de las bases de datos entre varias instancias.

De esta manera, y según está establecido en la arquitectura del sistema, se tendrán dos instancias sobre las que se recuperará información en modo lectura únicamente y otra instancia sobre la que se escribirá. La instancia de escritura será la maestra y estará sincronizando la información con las otras instancias de acceso en modo lectura.

A continuación vamos a comparar el comportamiento en tiempo de respuestas de ambos sistemas mediante un experimento que calcula el tiempo que tarda cada uno de los sistemas en realizar operaciones de inserción, actualización y consulta sobre miles de registros:

Registros (en miles)	100	200	300	400	500	Espacio en Disco
Tokyo Tyrant inserción	19 s	38 s	57s	76s	94s	21 MBs
MongoDB inserción	23 s	45 s	69 s	99 s	121 s	977 MBs
Tokyo Tyrant consulta	10 s	19 s	29 s	38s	48 s	-
MongoDB consulta	40 s	79 s	118 s	157s	196 s	-
Tokyo Tyrant actualización	24 s	49 s	75 s	99s	124 s	-
MongoDB actualización	74 s	116 s	144 s	172s	197 s	-

Tabla 9 - Comparativa de operaciones contra sistemas de caché

En una visualización gráfica podemos ver lo siguiente:

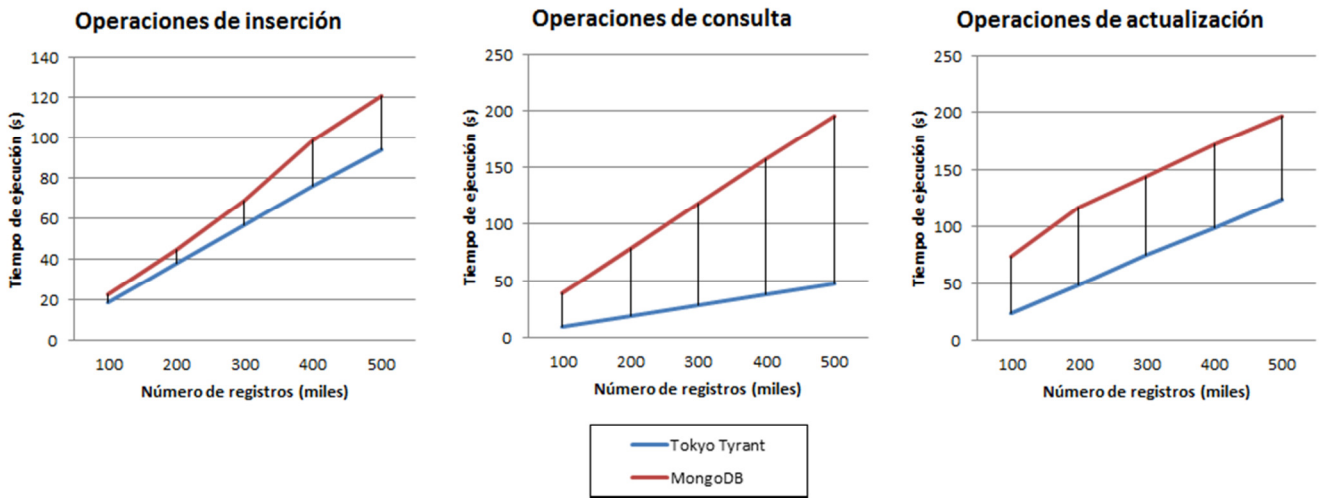

Figura 10 - Comparativa de operaciones contra sistemas de caché

Como se puede comprobar en las anteriores gráficas, Tokyo Tyrant la utilización de Tokyo Tyrant es una elección mucha más rápida que MongoDB y además, podemos comprobar como el uso de recursos para el almacenamiento del respaldo es mucho menor. La diferencia entre el uso de recursos entre Tokyo Tyrant y MongoDB es muy interesante y a su vez acusada. Debido a que tendremos una gran cantidad de registros almacenados, será preciso tener en cuenta el uso de recursos para realizar la implantación del sistema. Esto es necesario tenerlo en cuenta porque el número de registros que tendremos será muy alto.

Por otro lado es necesario tener en cuenta la necesidad de estás cachés. En las operaciones que se realicen contra el sistema las cachés almacenarán información relativa a las relaciones de los términos con las URLs y por otro lado los elementos de URL en sí. De esta manera, deberemos de dotar de cierta independencia a la hora de elegir las claves de los elementos que se almacenen en cachés para que no existan colisiones. Por esta razón, se utilizarán los identificadores tanto de los términos como las URLs para evitar estas colisiones y además, se introducirá el tipo en las claves para que entre ambos conjuntos tampoco se den problemas de integridad.

Como es necesario decantarse por un sistema de cachés en concreto, elegiremos aquel que consuma menos recursos y además ofrezca una mayor eficacia. La elección es utilizar Tokyo Tyrant en las máquinas de caché.

6.11. Sistema de ficheros en red

Como hemos comentado anteriormente, será necesario un sistema de ficheros que esté replicado en cada una de las máquinas. Este procedimiento de réplica se realizará a través de la red local a la cual están conectados cada uno de los terminales. Para ello necesitaremos una de las máquinas que esté funcionando como servidor de ficheros a la cual hemos llamado (en el diagrama de arquitectura) filer01 en el entorno de producción y filer02 en el entorno de desarrollo.

Esta máquina estará dedicada expresamente a realizar la sincronización de los ficheros existentes en el sistema de ficheros. Para ello deberá estar ejecutando un servidor de sincronización continuamente, y las máquinas tener un punto de montaje habilitado sobre dicho servidor de ficheros. Cuando el servidor de ficheros detecte que una máquina o desde algún otro punto un fichero dentro del sistema ha sido modificado deberá marcarlo como tal y actualizar dicha información para que cuando alguna máquina realice un acceso a él lo vea en su última versión.

Hay que tener en cuenta lo siguiente:

- Grandes números de ficheros y directorios generarán que el servidor de ficheros tenga que tener en cuenta dichos recursos.
- En caso de que tengan lugar sucesos de creación o modificación de ficheros masivos, se generarán picos de tráfico en la red local y en la CPU del servidor de ficheros.
- Cualquier estructura de directorios que se genere en los directorios compartidos por la red local, deberá tener una estructura con cierta dispersión que mantenga la integridad. Esto significa que los directorios existentes no deberán contener un gran número de ficheros en su interior y deberán estar dispersos en subdirectorios. La razón de esto es que el hecho de utilizar un servidor de ficheros avanzado, hará que el acceso a recursos como directorios donde por lo general será necesario listar los ficheros existentes, sea muy costoso si el número de ficheros es muy alto.

Para la integración del sistema de fichero en las distintas máquinas del sistema se utilizará el protocolo NFS. Este protocolo, según el modelo OSI, interviene en el nivel de aplicación y permite obtener un sistema de ficheros distribuido a través de la red local. Las máquinas conectadas a dicha red tendrán acceso a directorios compartidos con el resto de máquinas con el fin de unificar los elementos existentes.

El sistema de ficheros se basará en dos árboles de directorios, uno por entorno: producción y desarrollo. Las máquinas que estén conectadas a entornos de producción solo tendrán acceso a dicho sistema de ficheros, del mismo modo que las que estén conectadas a entornos de desarrollo. Las máquinas desde la cual debamos tener acceso al sistema de ficheros por razones de implementación, mantenimiento o desarrollo deberán tener acceso a ambos sistemas de ficheros si las autorizaciones son las suficientes y en caso del equipo de desarrollo, solo tendrá acceso al sistema de ficheros de desarrollo.

De esta manera las máquinas tendrán dos puntos de montaje:

- **Entorno de desarrollo**: `/mnt/filer/desarrollo/` que será un enlace simbólico al directorio real `/mnt/filer02/`
- **Entorno de producción**: `/mnt/filer/produccion/` que será un enlace simbólico al directorio real `/mnt/filer01/`

La razón fundamental de tener los enlaces simbólicos de `/desarrollo/` y `/produccion/` es por razones de simplicidad en el modelo y en la configuración a la hora de compartir desarrollos debido a

que con el cambio de una constante de configuración o una variable de entorno podremos ser capaces de acceder a un sistema de ficheros o a otro.

6.12. Configuración en tiempo de ejecución

Con el fin de que todos los procesos se ejecuten de igual manera y que en un momento dado se puedan modificar ciertos parámetros para cambiar su comportamiento, es necesario centralizar la configuración del sistema.

Los elementos que deberán ser configurados en el sistema estarán localizados en un directorio de configuración y deberán ser cargados cuando sea necesario acceder a ellos. Cada uno de los ficheros de configuración hará mención a lo que configure, es decir, si tenemos una configuración relativa a las vistas estará separada de aquella reservada a las conexiones de base de datos.

Estos son los ficheros de configuración con los que contará el sistema:

- **Fichero de configuración de rutas**: En este fichero se establecen las rutas de los directorios de todos los paquetes de clases incluidos en el sistema. Del mismo modo se configuran los recursos compartidos como plantillas, recursos de procesamiento, monitorización, etc.
- **Fichero de configuración de cachés**: Contendrá la configuración de conexión a los sistemas de cachés, así como otras posibles variables de entorno que sea necesario configurar para que la utilización de estos sistemas sea óptima.
- **Fichero de configuración del crawler**: Contendrá la configuración de todo lo relativo a la generación del índice de documentos, es decir, los pesos otorgados a los términos en función de su localización del documento, rutas a ficheros necesarios para el procesamiento como verbos, preposiciones, etc.
- **Fichero de configuración de base de datos:** En este fichero se establecerá la configuración de acceso a la base de datos, así como los credenciales necesarios para acceder a la misma.

6.13. Monitorización de procesos

Todos los procesos tendrán unos ficheros de monitorización llamados logs. En estos ficheros se registrará el hilo de ejecución de los procesos de los que se ha hablado en puntos anteriores. Las operaciones que aparecerán en estos logs o ficheros de monitorización tendrán relación primero con posibles momentos dentro de los programas a tener en cuenta y que sean registradas como los errores que puedan acontecer.

Es de especial importancia tener en cuenta que al ser un fichero donde varios procesos van a realizar operaciones de escritura, deberán ser bloqueados en el momento en que esta escritura tenga lugar y ser liberados para que otro proceso pueda escribir en ellos. En ningún momento dos o más procesos deberán realizar operaciones de escritura de manera simultánea porque de otra manera, los ficheros quedarán corrompidos.

Por otro lado, para que estos ficheros puedan ser útiles, deberán almacenar información en horquillas de tiempo. Esto significa que deberán ser relativos a un día, una semana o un periodo de tiempo acotado y que de esta manera el fichero no sea un enorme conglomerado de líneas anticuadas que no nos sirva para posibles tareas de mantenimiento.

Una posible alternativa para mantener este sistema de monitorización excluyendo la posible horquilla de tiempo es tener otros procesos en ejecución que realicen la rotación de estos logs y los almacenen en directorios de copia de seguridad. En un momento dado, si fuese necesario se podrían recuperar esas copias de seguridad para acceder al instante en el que ha sucedido un error y encontrar la causa para solucionarla.

6.14. Árbol de directorios

La estructura de directorios que tendrá la plataforma deberá tener las siguientes características:

- El entorno de producción, que estará separado del de desarrollo deberá ser igual en cuanto a funcionalidades, sin tener en cuenta las modificaciones de proyectos actuales o de mantenimiento que el equipo de desarrollo tenga pendientes de su puesta a producción.
- Los directorios de contenido estático, es decir, aquellos utilizados para generar la indexación de los documentos, deberán tener una dispersión tal que no coexistan más de quinientos ficheros o directorios en el mismo directorio. Esto también es aplicable a los contenidos multimedia, en general imágenes.
- El directorio raíz a partir del cual se sirven las páginas a los usuarios deberá estar separado de los directorios que contengan cualquier tipo de componente de software interno. Esto quiere decir que tanto la estructura de clases, scripts, ficheros de configuración y otros elementos que intervienen en las operaciones de procesamiento estarán localizados fuera del directorio raíz que sirva el servidor HTTP. Este directorio raíz deberá estar explícitamente indicado en la configuración del servidor HTTP.
- Todos y cada uno de los componentes del sistema deberán estar localizados en directorios que describan su existencia. Además, cada uno de los componentes deberán estar localizados en sus paquetes correspondientes.
- Los directorios relativos a logs estarán fuera de la estructura de directorios donde resida el código de la aplicación.
- Los directorios de templates, estarán localizados allá donde estén los controladores de visualización que utilicen dichos componentes.
- Los scripts que deberán estar en funcionamiento en las máquinas reservadas a tal efecto deberán estar agrupados dentro de un mismo directorio para poder identificarlos.
- Las librerías de terceros estarán localizadas dentro del árbol de directorios donde residan los paquetes de clases que conforman el sistema, dentro de un paquete específico para tal efecto.

A continuación se muestra una imagen donde se puede visualizar la estructura del árbol de directorios y una tabla donde se describen algunos de los elementos que aparecen en la imagen.

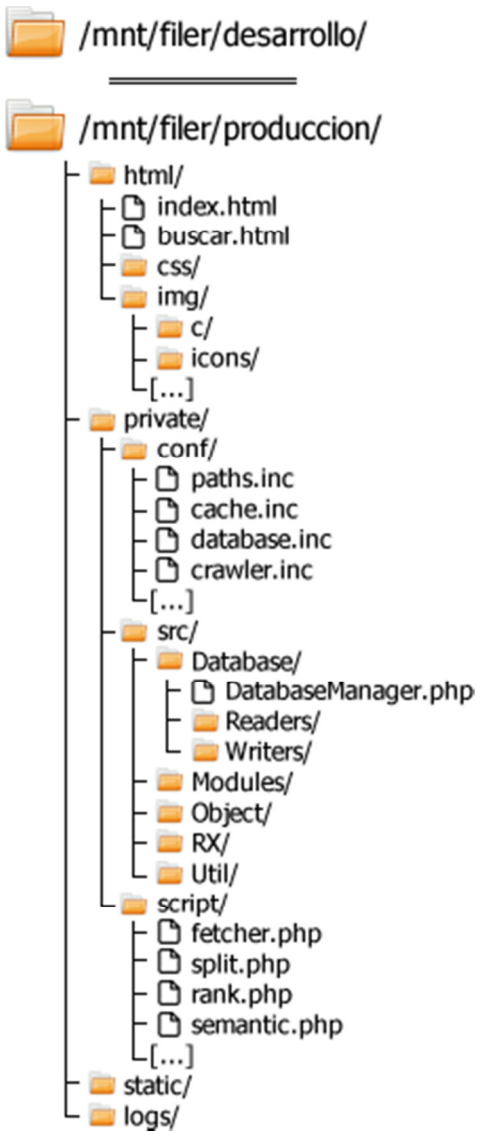

Figura 11 - Estructura de directorios del sistema

6.15. Sistema de control de versiones

Debido a que el sistema estará dividido en dos entornos distintos cuyos objetivos son distintos y aunque la gestión de cambios entre estos dos entornos se pudiese hacer de una manera más manual, la utilización de un sistema de control de versiones sería clave.

Un sistema de control de versiones, permite mantener la consistencia entre dos árboles de directorios distintos, pero para los cuales deseamos aplicar cambios entre ellos o comprobar sus diferencias. Además, hacen lo que propiamente indica su nombre, una gestión de las versiones. De esta manera, el sistema de control de versiones realizará un incremento en la versión del software que se actualice cada vez que suceda un cambio. Los cambios pueden aplicarse desde cualquiera de las ramas que se indiquen en la configuración y además, estos cambios implicará que las versiones sean autogestionadas y se realicen copias de seguridad. La necesidad de estas copias de seguridad es muy importante debido a que en caso de suceda algún imprevisto, mediante una simple tarea de mantenimiento, el sistema pueda volver rápidamente a una versión anterior y que sepamos que es estable. Esta característica fundamental afecta de manera directa al código de la aplicación y sus

modificaciones en el tiempo, pudiendo comparar estos cambios entre los componentes de versiones diferentes.

Existen un gran número de sistemas de control de versiones en el mercado. Aunque alguna de las soluciones son de pago, preferimos tender a un sistema de código abierto y gratuito. Dentro de este formato de sistemas de control de versiones, actualmente están aflorando aquellos que gestionan el almacenamiento de las versiones en la nube, pudiendo incluso consultar los cambios o el código de la aplicación a través de navegadores web. La solución escogida para ser integrada dentro de la plataforma es SVN, un sistema de control de versiones de código abierto y que permite realizar una gestión sencilla de las versiones y cambios en el software ya sea por comandos de consola o integrada dentro de los entornos de desarrollo.

La gestión de las versiones se realizará de manera transparente, almacenando en el sistema una serie de ficheros que indicarán las modificaciones aplicadas en el cambio de una versión a otra, quién la realizó, cuándo y a qué componentes afecta. Además, las entradas de estos ficheros deberán estar registras de manera detallada para saber qué se cambió con el fin de poder realizar un cambio de versión en caso de problemas con la aplicación intentando que afecte al menor número de funcionalidades.

6.16. Interfaz de usuario

A continuación se muestran algunas capturas de pantalla que muestran la interfaz de búsqueda y las páginas de resultados de búsqueda.

6.16.1. Página inicial

Esta sería la página inicial de la aplicación web, que nos permite, mediante un formulario introducir los términos de búsqueda.

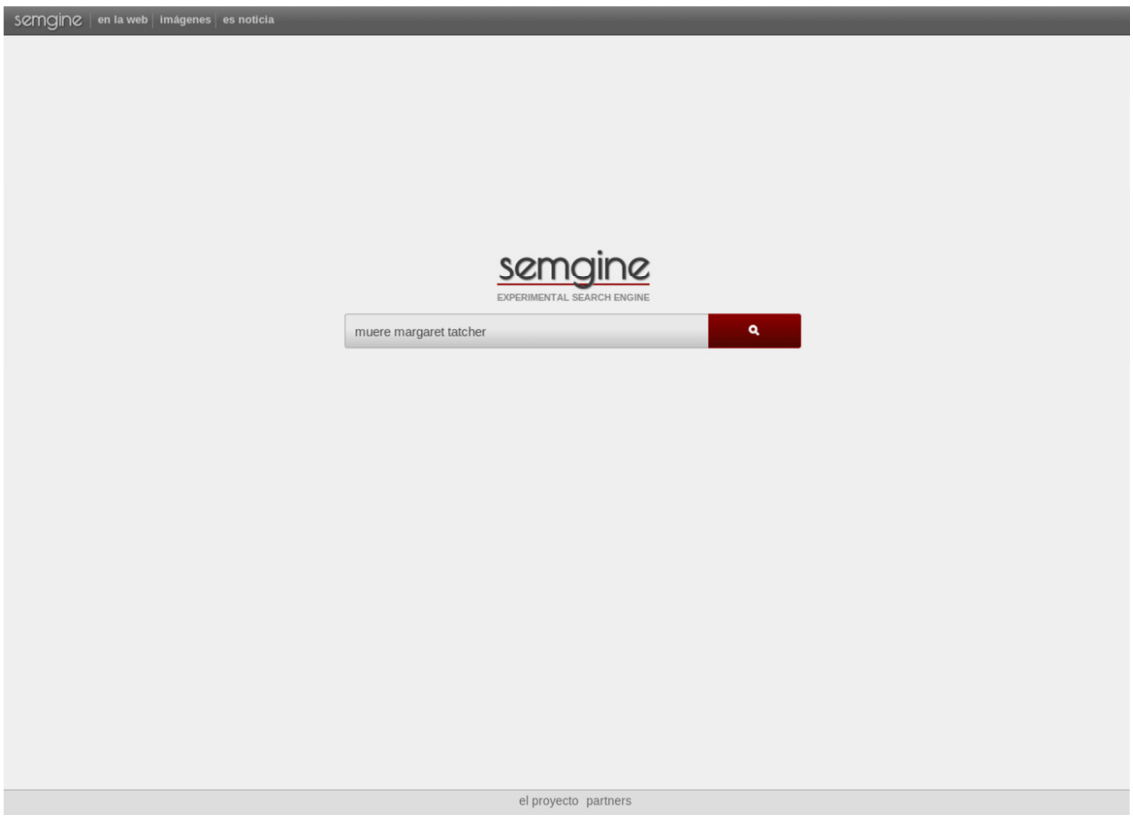

Figura 12 - Interfaz de usuario, página inicial

6.16.2.Resultados de una búsqueda

A continuación se muestra una página de resultados de una búsqueda. Esta será la interfaz más importante, que debe mostrar correctamente los datos más importantes de los documentos almacenados.

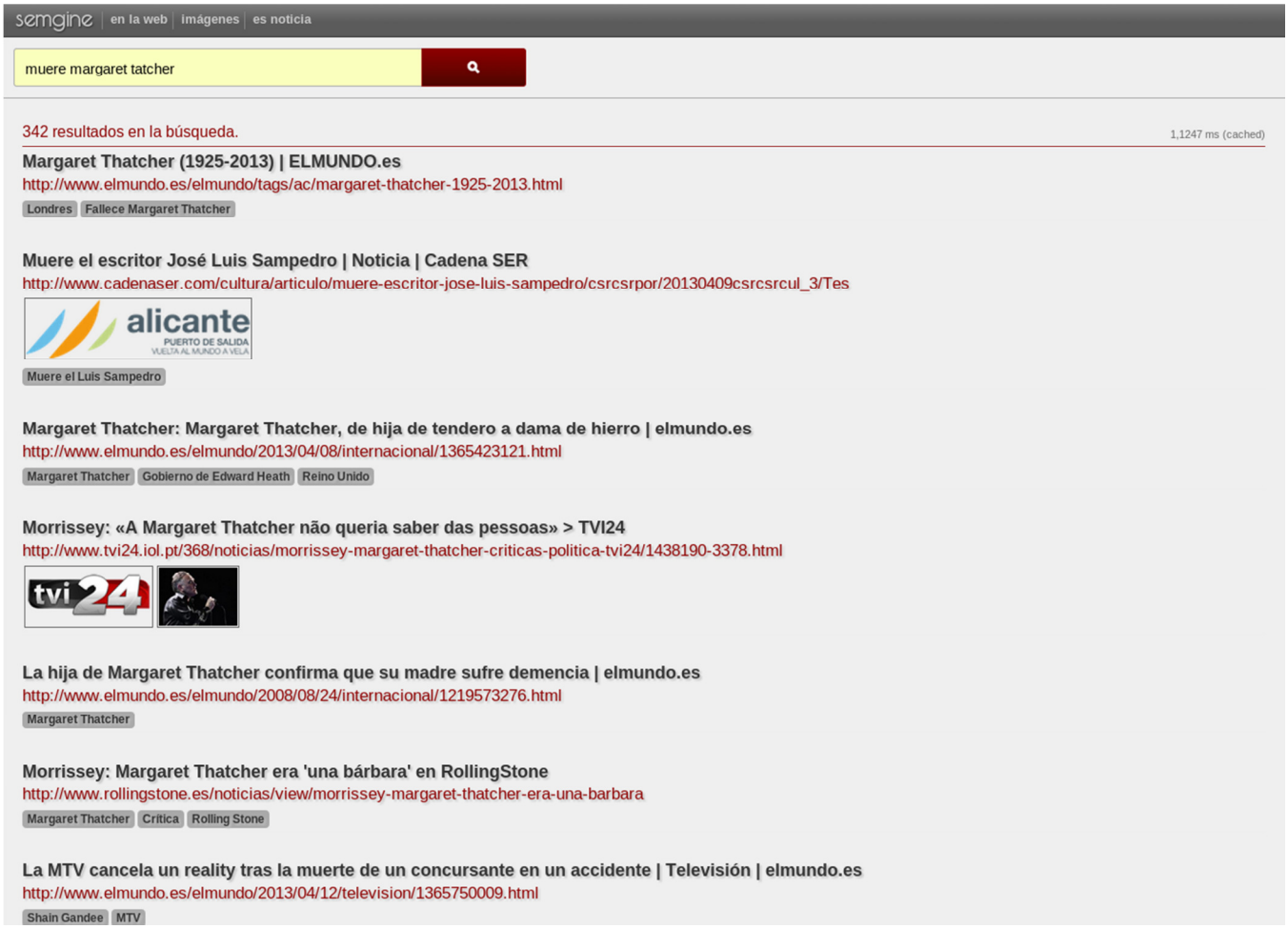

Figura 13 - Interfaz de usuario, resultados de una búsqueda

En la captura de pantalla se observa que se muestran los siguientes datos de un documento:

- Título
- URL del documento
- Resumen
- Entidades incluidas en el documento
- Imágenes relacionadas con el documento

Además, ante la búsqueda "muere margaret tatcher", se observa el número de documentos que coinciden con la búsqueda introducida y además, el tiempo de ejecución de la misma.

6.16.3.Resultados de búsqueda de imágenes

Ante una búsqueda por imágenes, podemos obtener los siguientes resultados ante la introducción de los términos que se introdujeron en la anterior búsqueda de documentos "muere margaret tatcher":

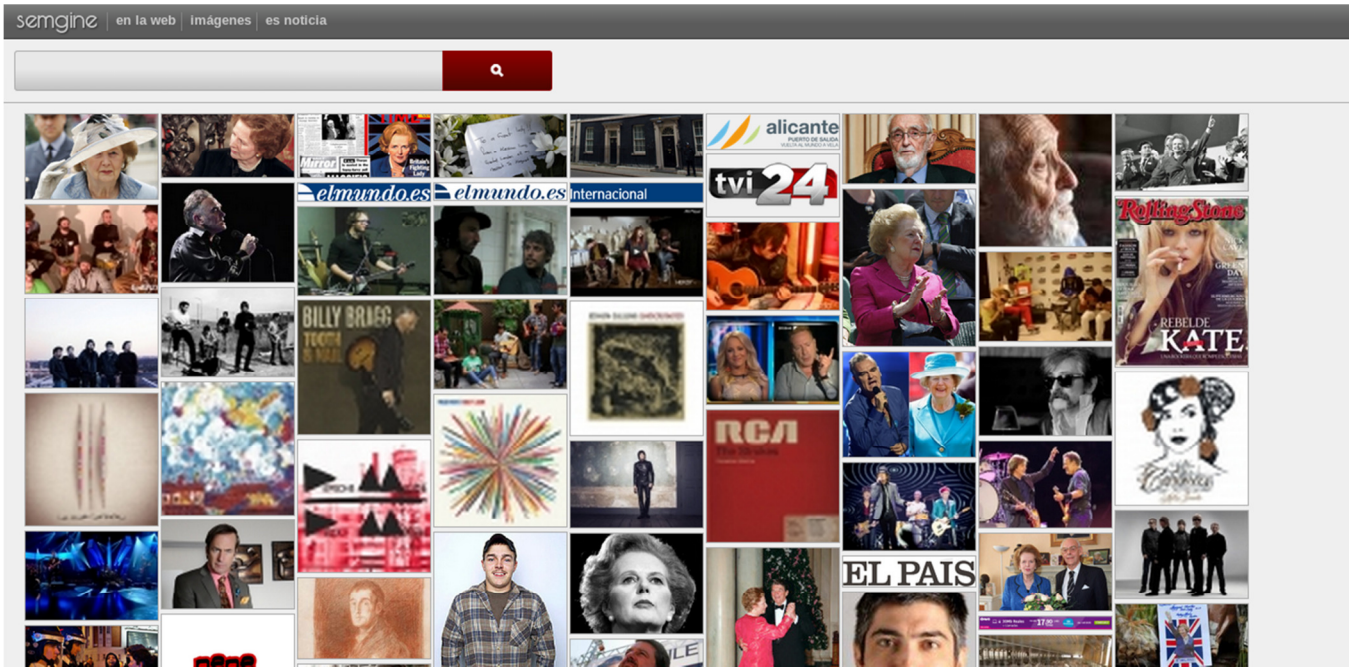

Figura 14 - Interfaz de usuario, búsqueda de imágenes

Para cada una de las imágenes recuperadas, se puede hacer acciones sobre ellas y obtener la información detallada de esa imagen:

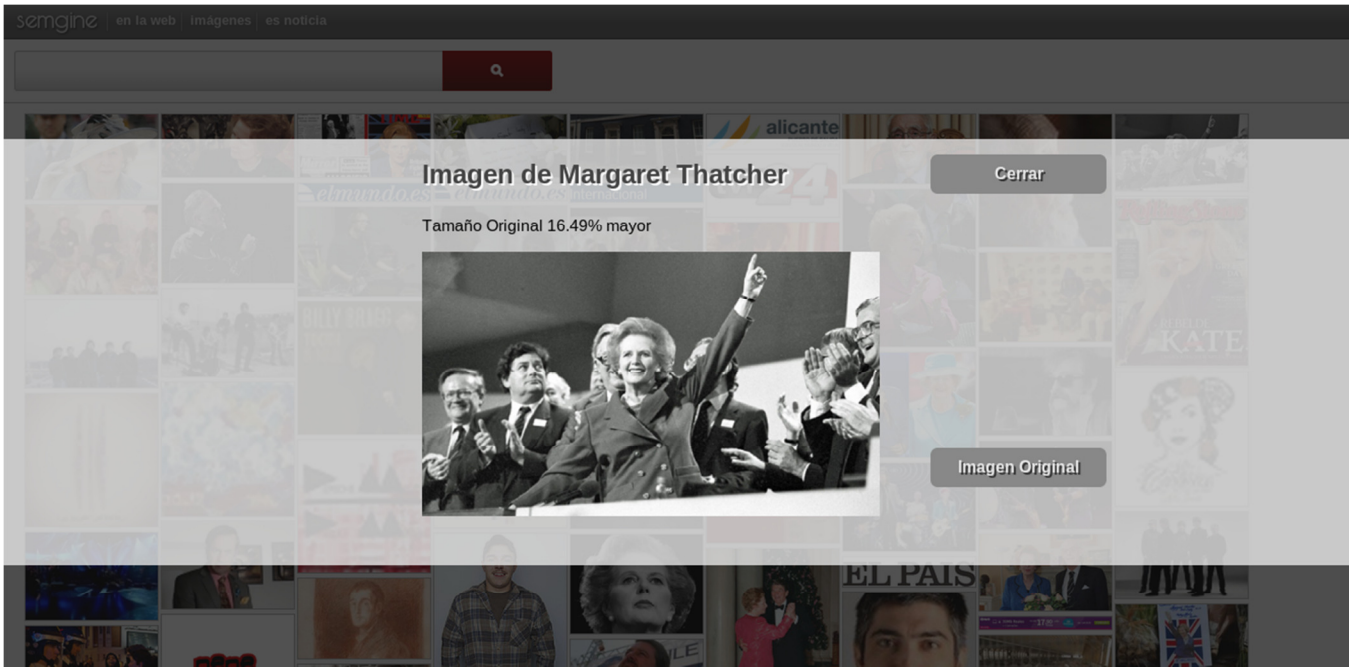

Figura 15 - Interfaz de usuario, resultado de imagen detallado

En el resultado de imagen detallado, se observa cómo se tienen los siguientes datos de la imagen recuperada:

- Título de la imagen, generalmente el texto que la rodea o el texto alternativo de la imagen del documento donde se recuperó.
- URL de la imagen original, que se puede acceder desde un botón situado a la derecha.
- Tamaño original de la imagen, que se muestra porcentualmente y relativo a la visualización del resultado.
- Imagen en miniatura que se recuperó durante el análisis.

7. ORGANIZACIÓN DEL PROYECTO

A continuación se elaborará la planificación del proyecto en los aspectos relativos a los recursos humanos que serán necesarios para su desarrollo e implantación y recursos económicos necesarios. Del mismo modo, se establecen los hitos primordiales del proyecto con el fin de establecer una duración del mismo.

7.1. Recursos humanos

Será necesario cubrir los siguientes recursos:

Tipo de recurso	Número	Descripción
Jefe de proyecto	1	Deberá ser capaz de gestionar los hitos del proyecto dentro de la planificación establecida, así como la supervisión del desarrollo de las soluciones especificadas en este documento.
Administrador de Sistemas	1	Será la persona encargada de establecer las pautas necesarias para la implantación de la plataforma y los entornos de trabajo del equipo de desarrollo. Del mismo modo será encargado de mantener dicha infraestructura con el fin de que su funcionamiento sea el adecuado.
Responsable de calidad	1	Será encargado de desarrollar los paquetes de pruebas destinados a establecer los criterios de calidad necesarios para el correcto funcionamiento del sistema.
Analista / Desarrollador	3 – 5	Desempeñarán las labores necesarias para desarrollar el producto final siguiendo las pautas de análisis establecidas por este documento y el Jefe de Proyecto.

Tabla 10 - Organización de recursos humanos

7.2. Planificación

Hitos	Duración
Establecimiento de la plataforma y pruebas de conectividad	1 semana
Generación de estructura del sistema básica	2 semanas
Implementación y pruebas de proceso FETCH	2 semanas
Implemetnación y pruebas de proceso SPLIT	2 semanas
Implementación y pruebas de proceso RANK	4 semanas
Implementación y pruebas de proceso SEMANTIC	4 semanas
Implementación y pruebas de proceso LINK	2 semanas
Implementación y pruebas de proceso PICTURE	1 semana
Pruebas de ciclo completo	2 semanas
Generación de interfaz de usuario y salidas	2 semanas
Puesta en producción	1 semana
TOTAL	**23 semanas**

Tabla 11 - Planificación

La planificación ofrece la necesidad de tener 23 semanas de desarrollo para la generación de los componentes necesarios para otorgar al sistema de un funcionamiento adecuado.

7.3. Recursos económicos

A continuación se establecen los recursos económicos necesarios por un lado para cubrir los costes relativos a los recursos humanos asignados al proyecto y por otro para aquellos relativos a la plataforma.

En el primero de los casos es necesario tener en cuenta que las tarificaciones se especifican en euros por hora, por tanto se tendrán un total de 920 horas, teniendo en cuenta que se tendrán 5 días laborables de 8 horas para las 23 semanas especificadas. Además hay que tener en cuenta los impuestos aplicables a estos costes.

En el caso de los costes relativos a la plataforma se establecerá una relación de los elementos disponibles en la infraestructura del sistema según las previsiones de costes del mercado actual.

Tipo de recurso	Tarifa	Coste
Administrador de Sistemas	25 € / h	23.000 €
Analista / Desarrollador	22 € / h	20.240 €
SUBTOTAL	**-**	**43.240 €**
Impuesto: IVA	21%	9.080,40 €
TOTAL	-	**52.320,40 €**

Tabla 12 - Recursos económicos relativos a recursos humanos

A continuación, como se comentaba anteriormente se establece la relación de precios de algunos elementos existentes en el mercado. En concreto, se ha analizado las posibilidades ofrecidas por la empresa Dell para dotar al sistema de una infraestructura adecuada según el diseño de la arquitectura del sistema.

A estos costes hay que añadirle los gastos relativos al ancho de banda utilizado a partir de que el desarrollo del sistema esté finalizado.

Elemento de infraestructura	Descripción	Coste unitario	Número	Coste
Servidor ud0X0Y	Dell PowerEdge R210 II Intel Core i3 3240 4GB RAM, 1 HDD de 2 TB	409 €	8	3.272 €
Servidores big0X	Dell PowerEdge T420 Intel Xeon E5 2450 16 GBs RAM, 1 HDD de 2TB	3,473 €	2	6.946 €
Servidor de Bases de Datos y Cachés	Dell PowerEdge R210 II Intel Core i3 3240 4GB RAM, 1 HDD de 2 TB	406 €	8	3.272 €
Servidor de Ficheros	Dell PowerEdge R520 Intel Xeon E5 2450	3.714 €	2	7.428 €

	4GB RAM, 4 HDD de 3 TBs			
Equipo de Desarrollo	Dell Optiplex 3020 MT Intel Core i5 4570 4GB RAM, 1 HDD de 2 TB Monitor	559 €	8	4.472 €
Switches Ethernet	Dell PowerConnect 2800	238 €	4	952 €
Redes (Firewalls, Cableado, …)	Estimativo	3.000 €		3.000 €
SUBTOTAL	**-**	**-**	**-**	**29.342 €**
Impuesto: IVA	21%			6.161,82 €
TOTAL	**-**	**-**	**-**	**35.503,82 €**

Tabla 13 - Costes relativos a infraestructura

A continuación se detalla los costes finales, en conjunto con los costes añadidos de manera arbitraria para mantener el servicio al menos un año. Estos costes estarán dentro de los costes de producción, donde también se incluyen los recursos básicos como el consumo de electricidad. Del mismo modo se detallan los costes relativos al personal que se mantenga tras haber desarrollado el sistema. Este coste es también arbitrario y se fija a modo de presupuesto.

Concepto	Coste
Recursos humanos	**52.320,40 €**
Infraestructura	**35.503,82 €**
Costes de producción	30.000 €
Costes de personal tras el desarrollo	50.000 €
TOTAL	**167.824,22 €**

Tabla 14 - Resumen de costes

8. EXPERIMENTACIÓN Y RESULTADOS

Con el fin de que el desarrollo del sistema tenga un resultado adecuado, se realizarán una serie de pruebas para comprobar tanto el correcto funcionamiento de todos los componentes, como posibles formas de mejorar la eficiencia y eficacia del proceso.

A continuación se establecen una serie de pruebas en dos vertientes, la primera enfocada a comprobar el funcionamiento de los procesos de análisis de documentos que estarán en ejecución, la segunda, una serie de pruebas realizadas contra el sistema que el usuario final será capaz de manejar con el fin de comprobar su funcionamiento y eficacia.

8.1. Plan de pruebas de procesos

En adelante se realizará una serie de pruebas a los procesos que efectuarán tanto la recuperación como el análisis de los documentos. En concreto, se realizarán pruebas de rendimiento y tiempo de ejecución para cada uno de los procesos y finalmente una comprobación de todo el conjunto de procesos de manera simultánea. Además, para cada uno de los procesos se analizarán los elementos más importantes de aquellos que intervengan en los mismos.

8.1.1. Pruebas realizadas del proceso FETCH

Las pruebas realizadas al proceso FETCH sobre un conjunto de 20 documentos, ofrecen unos resultados en cuanto al tiempo de ejecución del proceso de la siguiente forma:

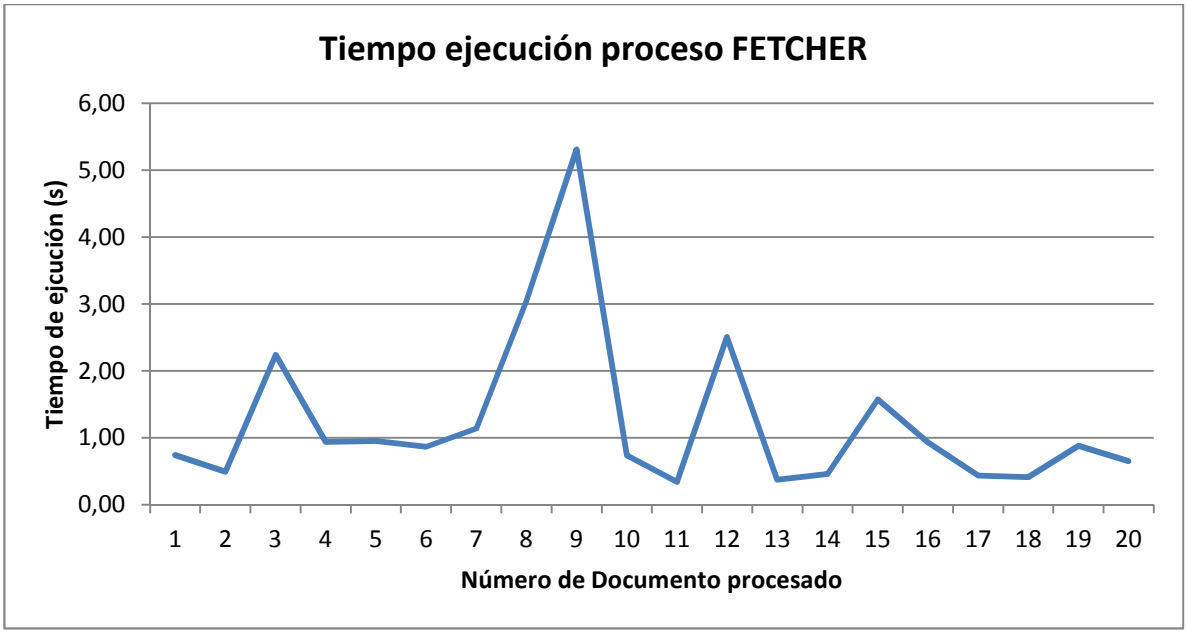

Figura 16 - Tiempo de ejecución del proceso FETCHER

En la siguiente tabla, se muestran los resultados obtenidos en la gráfica anterior:

Documento	T. Ejecución FETCHER (s)
1	0,74
2	0,49
3	2,24
4	0,94
5	0,95
6	0,87
7	1,14
8	3,04
9	5,31
10	0,74
11	0,34
12	2,51
13	0,38
14	0,46
15	1,57
16	0,94
17	0,44
18	0,41
19	0,88
20	0,65

Tabla 15 - Tiempo de ejecución del proceso FETCHER

En los resultados, se comprueba que el proceso tiene un tiempo de ejecución que suele ser normalmente por debajo de dos segundos y en otras ocasiones se amplía bastante. El caso del pico que se observa en la gráfica puede ser debido al tiempo de respuesta del servidor que se esté consultando para recuperar el documento. Se puede dar la posibilidad que el servidor no responda (no se observan casos en las pruebas realizadas para 20 documentos), o que por otro lado, se tengan unos tiempo de respuestas altos.

Aunque el tiempo de respuesta antes de catalogar la petición como timeout es relativamente bajo, se van a dar casos en los que ya sea por la saturación de la conectividad, la latencia entre los servidores u otras causas, el tiempo de ejecución de este proceso va a ser relativamente alto debido a la espera de la respuesta de los servidores.

En este proceso, debido a que no se realiza ningún análisis y simplemente se recogen documentos de los servidores web, no tiene demasiado sentido analizar más información. Sin embargo, un dato que puede interesarnos puede ser el número de documentos que se han intentado recuperar y que por alguna razón no se ha podido realizar dicha recuperación y si es posible alguna información del documento. Vamos a comprobarlo con la siguiente tabla:

Descripción	Valor	% del total
Número de documentos totales recuperados	4.120	-
Errores de recuperación (No se encuentra el documento)	176	4,27 %
Errores de recuperación (Error en el servidor)	72	1,74 %
Documentos con codificación UTF-8	2.428	58,93 %
Documentos con codificación ISO-8856-1	1.416	34,36 %

Tabla 16 - Distribución de documentos recuperados

8.1.2. Pruebas realizadas del proceso SPLIT

A continuación se comprobará el funcionamiento del proceso SPLIT, el cual se encarga de distribuir los fragmentos del documento recuperado en ficheros que se puedan analizar. El tiempo de ejecución analizado se ha realizado sobre la misma muestra que el proceso FETCHER, es decir, 20 documentos:

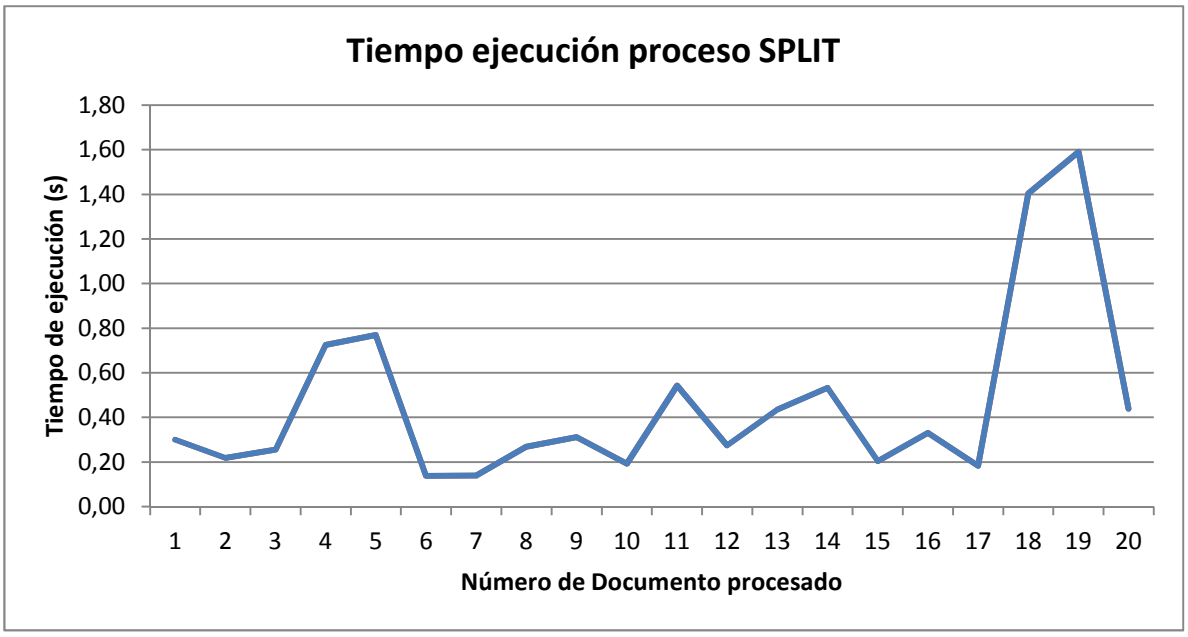

Figura 17 - Tiempo de ejecución del proceso SPLIT

El proceso no realiza una ejecución muy laboriosa y suele tardar en finalizar la ejecución para cada documento relativamente poco. El tiempo de ejecución variará en función de la longitud de los textos encontrados en el documento y sobre todo de los elementos que se deban limpiar en el fichero para generar fragmentos limpios que puedan ser analizados. En cualquier caso, el tiempo de ejecución no supera mucho más de un segundo de manera habitual.

Para ver los resultados de esta gráfica en detalle, se muestran los datos obtenidos durante la experimentación en la siguiente tabla:

Documento	T. Ejecución SPLIT (s)
1	0,30
2	0,22
3	0,26
4	0,73
5	0,77
6	0,14
7	0,14
8	0,27
9	0,31
10	0,19
11	0,54
12	0,28
13	0,44
14	0,53
15	0,20
16	0,33
17	0,18
18	1,40
19	1,59
20	0,44

Tabla 17 - Tiempo de ejecución del proceso SPLIT

Hay que tener en cuenta que este proceso únicamente se encarga de dividir los ficheros de los documentos recuperados en trozos más pequeños, por lo que el tiempo de ejecución del proceso, como se comprueba, no debe ser demasiado alto.

En el caso de este proceso no tiene sentido realizar un análisis más exhaustivo de ningún otro detalle que nos pueda ayudar a mejorarlo.

8.1.3. Pruebas realizadas del proceso RANK

A continuación se realiza el mismo análisis de carga sobre el proceso RANK con el procesamiento de 20 documentos. Se obtienen los siguientes resultados:

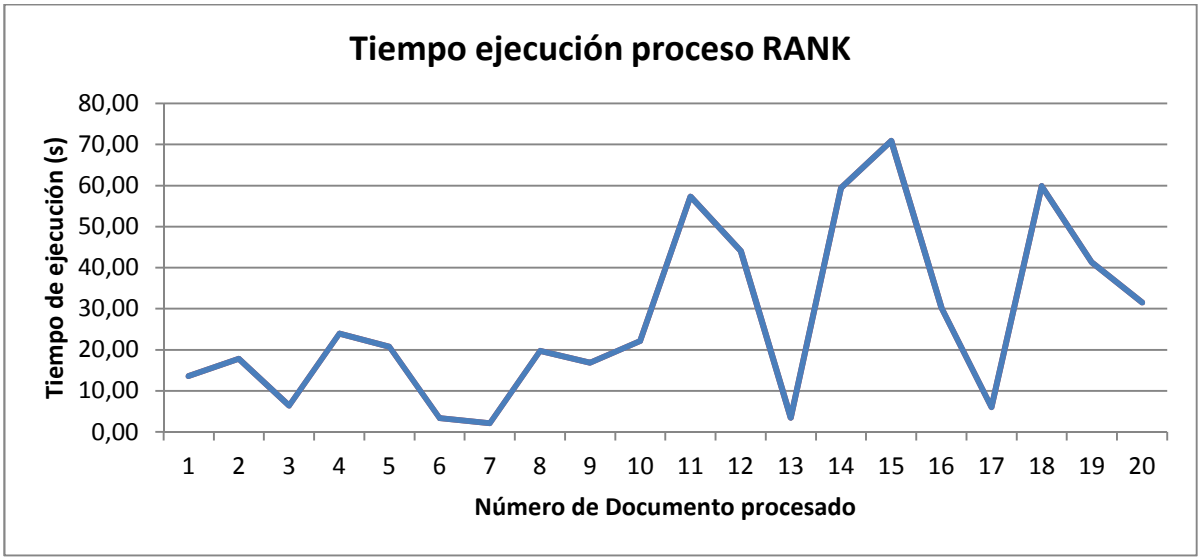

Figura 18 - Tiempo de ejecución del proceso RANK

El detalle de los resultados obtenidos en la anterior gráfica se representa en datos en la siguiente tabla:

Documento	T. Ejecución RANK (s)
1	13,56
2	17,75
3	6,39
4	23,94
5	20,76
6	3,35
7	2,10
8	19,76
9	16,88
10	22,12
11	57,27
12	44,08
13	3,41
14	59,43
15	70,91
16	30,20
17	6,03
18	59,88
19	41,29
20	31,52

Tabla 18 - Tiempo de ejecución del proceso RANK

Como se aprecia, el tiempo de ejecución es bastante variable y mucho mayor que en los procesos anteriores. Esto puede suponer un problema a la hora de realizar el lanzamiento del sistema debido a que este proceso parará los que vengan después, dejando los documentos en espera durante mucho tiempo.

La razón de la variabilidad de los tiempos de ejecución es debido a que algunos documentos tendrán muchos términos que ser analizados y contendrán textos demasiado extensos frente a otros documentos que su longitud sea bastante escueta.

Por otro lado, también afectará el hecho de que los términos no se encuentren ya registrados, es decir, que ya estén en la base de datos. Al no estar en los registros del sistema, los nuevos términos que vayan apareciendo, deberán ser introducidos, lo que supone que se realicen más operaciones contra la base de datos y significará más tiempo de ejecución.

Una posible solución a esto es dividir este proceso en varios subprocesos menos costosos que permitan realizar por partes el análisis de los términos que se realiza en el proceso completo.

A parte de este análisis, es interesante comprobar el número de elementos que se van introduciendo en el sistema y cómo este número deberá ir decreciendo a lo largo de las ejecuciones.

Cuantos más documentos hayan sido analizados, más diversidad de términos existirá en la base de datos y menos términos necesitarán ser introducidos en el sistema. Sobre una muestra de en torno a 200 documentos se obtienen el siguiente resultado:

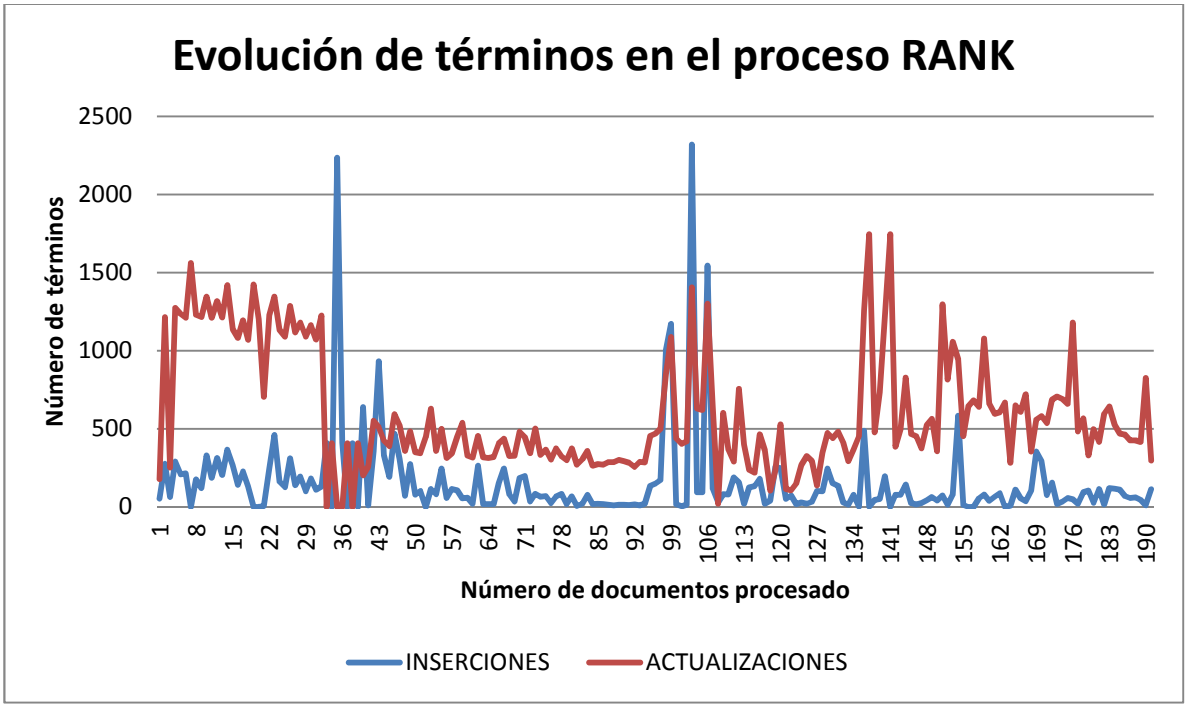

Figura 19 - Evolución de términos en el proceso RANK

Como se puede comprobar, las inserciones (en azul) van decreciendo en número y comienzan a mantenerse muy bajas con respecto a las actualizaciones de términos (en rojo). Esto significa, que a lo largo del tiempo, muchos términos no necesitarán ser insertados porque ya se tendrán en la base de datos y únicamente deberán ser actualizados para contabilizar que un documento más lo contiene. Como se comprueba en el gráfico y eso que la muestra es de solo 200 documentos, a partir del documento 40 el número de inserciones es mucho mayor que al final del proceso.

Además, puede parecer extraño que al principio de la muestra se observe una gran diferencia entre actualizaciones e inserciones y eso es debido a que en ese momento, esos documentos procesados habían sido parte de un experimento anterior, es decir, se habían procesado con anterioridad.

Por otro lado, es visible que existen picos de inserciones y esto es debido a que en ocasiones se tendrán documentos que contengan un gran número de términos que no tenemos en nuestros registros. Esto sucederá de vez en cuando, pero a la larga dejará de suceder.

8.1.4. Pruebas realizadas del proceso SEMANTIC

A continuación se muestra el análisis en el tiempo de procesamiento sobre 20 documentos del proceso SEMANTIC:

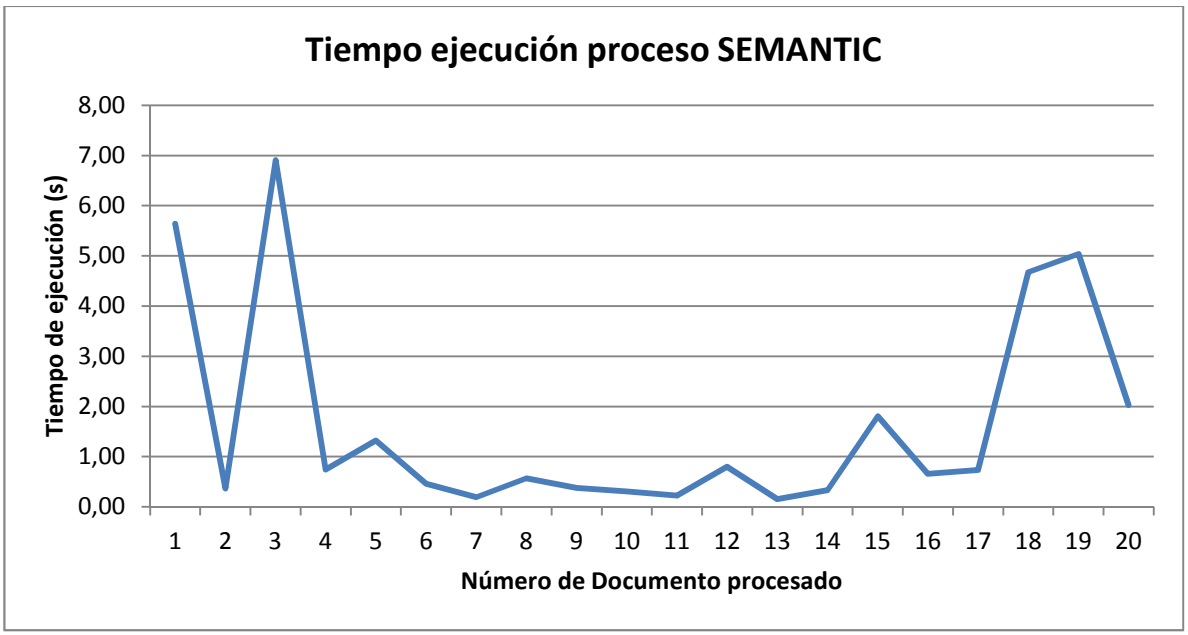

Figura 20 - Tiempo de ejecución del proceso SEMANTIC

En el análisis se comprueba que el tiempo de procesamiento es variable aunque no es demasiado grande como para necesitar una optimización de este proceso.

Es posible que en un futuro, el número de relaciones semánticas en el sistema sea muy elevado y la necesidad de revisar la optimización de este proceso sea más adecuada.

El detalle de la gráfica anterior se puede comprobar en la siguiente tabla:

Documento	T. Ejecución SEMANTIC (s)
1	5,64
2	0,36
3	6,91
4	0,74
5	1,32
6	0,46
7	0,19
8	0,57
9	0,38
10	0,31
11	0,23
12	0,80
13	0,15
14	0,33
15	1,81
16	0,66
17	0,74
18	4,68
19	5,04
20	2,03

Tabla 19 - Tiempo de ejecución del proceso SEMANTIC

En este experimento para 20 documentos, se visualiza que algunos documentos no contienen gran información semántica y el proceso no tarda apenas en procesarlo, sin embargo, algunos otros documentos contienen mucha más información sobre entidades, relaciones y otras valoraciones semánticas por lo que el tiempo de procesamiento es bastante mayor.

8.1.5. Pruebas realizadas del proceso PICTURE

A continuación se muestra el análisis en el tiempo de procesamiento sobre 20 documentos del proceso PICTURE:

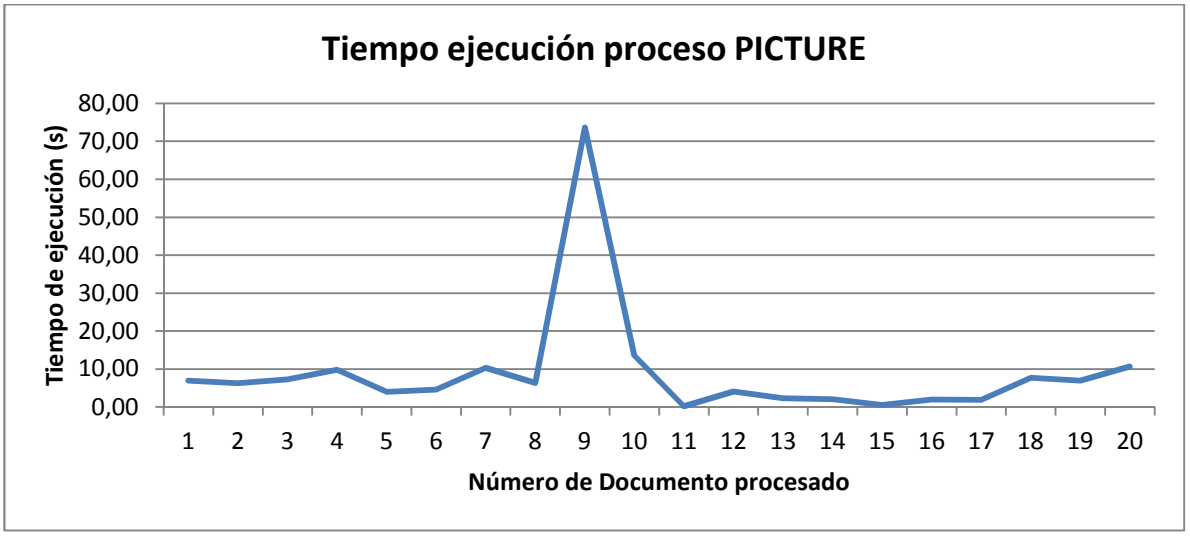

Figura 21 - Tiempo de ejecución del proceso PICTURE

El detalle de la experimentación realizada se muestra en la siguiente tabla:

Documento	T. Ejecución PICTURE (s)
1	6,97
2	6,26
3	7,24
4	9,82
5	4,02
6	4,61
7	10,29
8	6,32
9	73,64
10	13,63
11	0,19
12	4,11
13	2,31
14	2,02
15	0,50
16	1,92
17	1,89
18	7,72
19	6,98
20	10,62

Tabla 20 - Tiempo de ejecución del proceso PICTURE

Al igual que otros procesos, el tiempo de ejecución de este proceso depende del número de elementos a analizar. En este caso, dependerá drásticamente del número de imágenes que tenga un documento.

Si nuestro deseo es recoger todas las imágenes que aparezcan en un documento para relacionarlas con él, es muy necesario recuperarlas todas. Sin embargo, este proceso se podría optimizar de tal manera que únicamente se recuperasen ciertas imágenes, por ejemplo, aquellas que pertenezcan a servidores del mismo nombre de dominio que el documento que se analiza o aquellas que sean representativas o tengan unas dimensiones acotadas.

8.1.6. Pruebas realizadas del proceso LINK

A continuación se muestra el análisis en el tiempo de procesamiento sobre 20 documentos del proceso LINK o FETCH BUILDER:

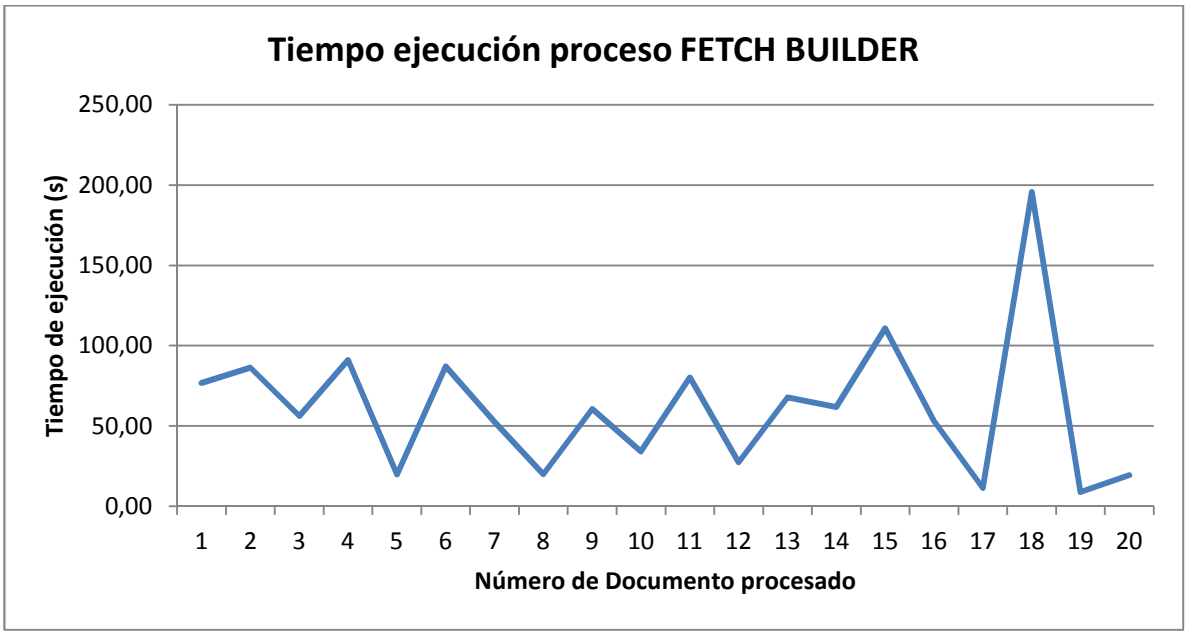

Figura 22 - Tiempo de ejecución del proceso LINK

En este proceso, vemos grandes diferencias entre unos documentos y otros. Esto es debido a la cantidad de enlaces que puede tener un documento frente a otros. Algunos documentos tendrán del orden de decenas de enlaces mientras que otros pueden tener cientos.

Este problema generará que el proceso tarde mucho en generar nuevas entradas para próximas revisiones de documentos o incluso recuperación de documentos nuevos.

Sin embargo, no es algo que podamos detener ni obviar. Aunque una posible solución a esto es no analizar todos los enlaces de un documento, esto derivaría en un problema debido a que no estaríamos recuperando toda la información del documento como debería ser y además, estamos cerrando el rango de documentos que recuperar. Por otro lado, no se podría decidir que fragmento del documento elegir para recuperar enlaces a otros documentos y cual no ya que la distribución de estos documentos difiere demasiado.

El detalle de la gráfica anterior que muestra la experimentación realizada con el proceso LINK se muestra en la siguiente tabla:

Documento	T. Ejecución LINK (s)
1	6,97
2	6,26
3	7,24
4	9,82
5	4,02
6	4,61
7	10,29
8	6,32
9	73,64
10	13,63
11	0,19
12	4,11
13	2,31
14	2,02
15	0,50
16	1,92
17	1,89
18	7,72
19	6,98
20	10,62

Tabla 21 - Tiempo de ejecución del proceso LINK

8.1.7. Pruebas realizadas del proceso completo y comparación de procesos

Para finalizar las pruebas de procesos, se realizará una comparativa de la información obtenida en la experimentación de todos los resultados. En estas pruebas se ha comprobado qué procesos necesitan más tiempo para ejecutarse y de este modo hemos podido averiguar qué sucederá en tiempo real durante la ejecución de ellos.

Sabiendo esta información podemos realizar una planificación de procesos más exhaustiva, consiguiendo así que los documentos se procesen correctamente sin tener que relegar algunos de los procesamientos a tiempos de espera demasiado altos.

A continuación se muestra un gráfico sobre un conjunto de procesos limitado (se han llegado a hacer pruebas con hasta 4.000 documentos y la evolución es similar. En este caso, el número de documentos a analizar es de 50 y en el proceso se comprueba el tiempo de ejecución de cada uno de los procesos:

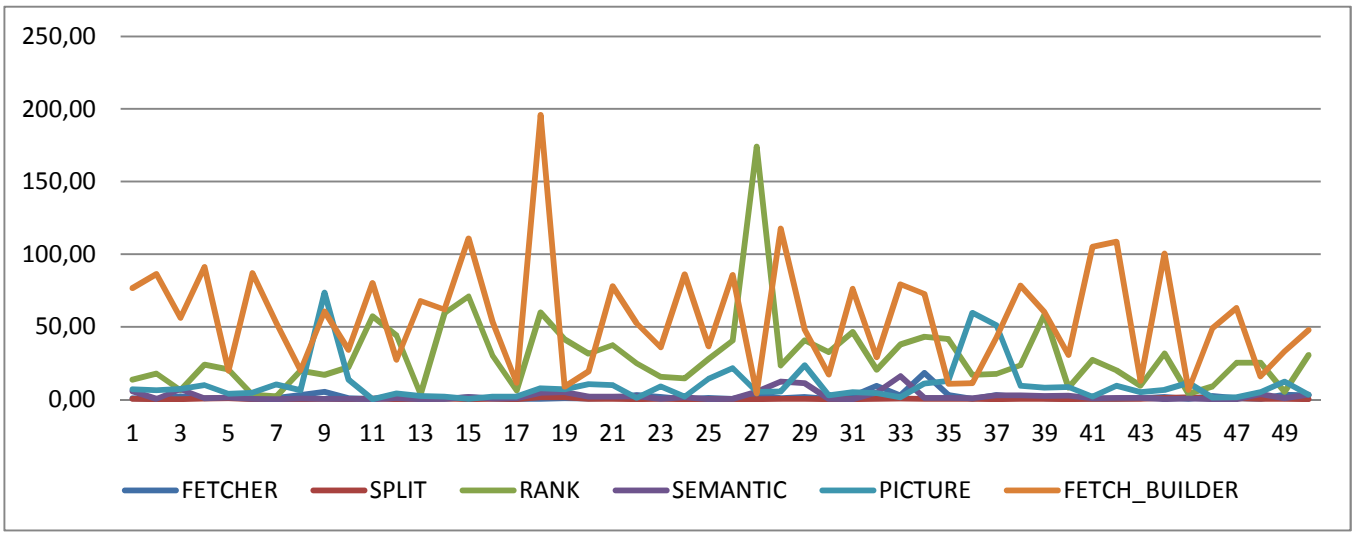

Figura 23 - Comparativa de tiempo de ejecución de los procesos

Los procesos más costosos, como se comprueba son los procesos RANK y FETCH_BUILDER (o también llamado LINK). Es evidente que el proceso LINK no es sencillo de optimizar debido a las razones expuestas en la experimentación reservada a dicho proceso. Sin embargo, el proceso RANK si sería posible optimizarlo debido a que en ese proceso se realiza diferentes labores que podrían ser particionadas para generar distintos subprocesos. En cualquier caso, cuando encontremos un proceso que sea más costoso que el resto, se le puede asignar más capacidad de procesamiento en las máquinas donde vayan a ejecutar. Es decir, en las máquinas se podrían configurar varios procesos RANK, en un número superior al resto para que existan más entidades de procesos ejecutando a la vez sobre distintos grupos de documentos y que no se generen cuellos de botella innecesarios.

Si expandimos el gráfico a un número de documentos mucho mayor obtenemos la siguiente visualización que se muestra a modo de curiosidad:

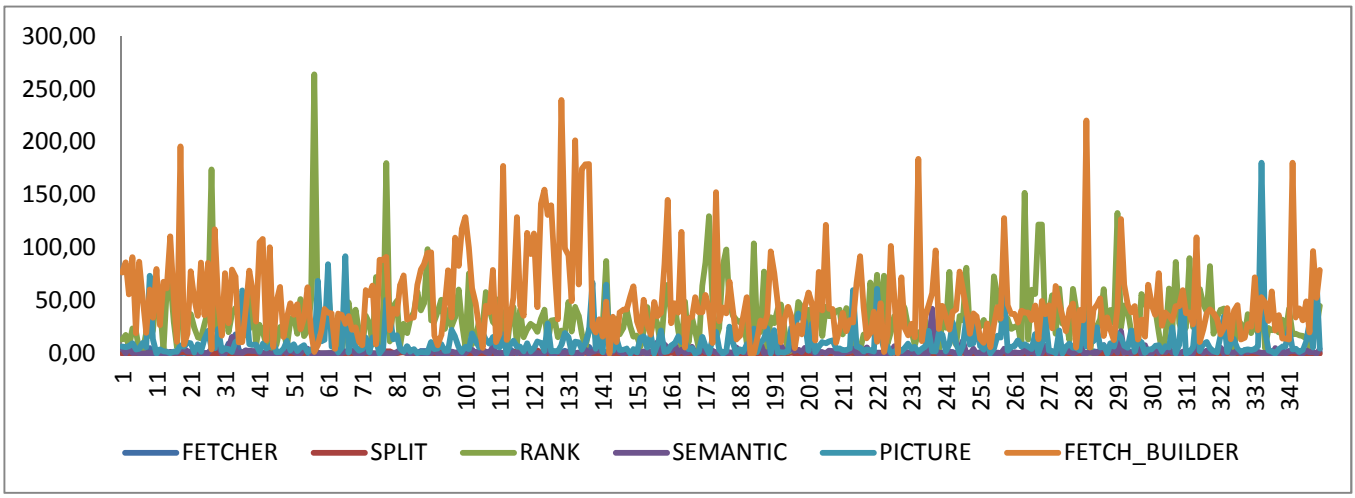

Figura 24 - Expansión de la comparativa de procesos

En el gráfico anterior se pueden comprobar los picos generados por los procesos a lo largo de en torno a 350 documentos analizados.

9. CONCLUSIONES

Ante el presente crecimiento que existe en el ámbito de las tecnologías de la información y en concreto de la propia información, la existencia de índices de búsqueda de documentos es cada vez más útil. Si bien es cierto que grandes empresas multinacionales están copando este mercado, el hecho de realizar experimentaciones sobre este campo es muy interesante para conocer cómo funcionan por dentro los algoritmos que dan vida a los buscadores que estas empresas regentan. La idea de este proyecto no es su culminación como un producto que pueda ser usado por los usuarios de manera habitual con el fin de intentar hacer sombra a este tipo de productos ya existentes. Estas grandes empresas tienen enormes equipos de desarrollo y los servicios que prestan son tales que el hecho de plantearse la generación de un producto de estas características no tiene demasiado sentido.

Por otro lado, este tipo de productos puede ser un claro aliciente para desarrollos acotados o pequeñas empresas que quieran realizar una indexación de la información en el ámbito privado, por ejemplo, documentación de software aplicada a los productos de la empresa, informes de auditorías o cualquier conjunto de información que una empresa almacene puede ser documentado. Este tipo de soluciones se está dando ya con otros productos similares que basan su potencia en tratar de manera semántica la información que se les inyecta.

Si bien es cierto que el desarrollo de este proyecto, así como su implantación puede llegar a tener unos costes elevados, como puro carácter académico y de experimentación, este proyecto resulta muy interesante. La categoría de los componentes que se precisan desarrollar para poner este proyecto en píe, al igual que el diseño de la arquitectura necesaria para su implementación puede representar ciertos retos de un calibre respetable como para tener en cuenta posibles soluciones alternativas.

Uno de los puntos clave dentro del ámbito de la generación de índices de relevancia entre documentos y su información es que la teoría existente en este ámbito está muy extendida y las prácticas apuntan casi siempre a la prueba y error. Si bien es cierto que la teoría existe y que multitud de estudios han conseguido encontrar fórmulas aplicables a estas soluciones, la capacidad de moldear estas propuestas es la más es una dirección muy interesante para tener en cuenta a la hora de avanzar. Las soluciones presentadas en este documento para su aplicación a este sistema son adecuadas para conseguir resultados óptimos dentro del rango esperado.

En cuanto a las soluciones de arquitectura ofrecidas en el documento es necesario apuntar que aunque existen otro tipo de alternativas o puedan ofrecerse otras posibilidades, las presentadas en este documento han sido ya experimentadas en otros proyectos similares y su eficiencia es altamente reconocida. Cualquier alternativa posible a las presentadas en este documento, en general referidas a la manera de implantar este sistema en un ámbito real, ha sido cotejada a la hora de formularse una posibilidad de aplicación y evidentemente, en los mismos puntos donde se tratan dichas soluciones, se exponen las alternativas posibles. Sin ninguna duda, la eficiencia en este tipo de sistemas es crucial y aunque esta característica depende mucho de la carga a la que esté expuesto el sistema final, su planificación es muy importante y así se intenta mostrar en este documento. El producto final, debe ser de tal manera que el usuario pueda obtener la información que requiera en el menor tiempo posible y esto puede conseguirse de distintas maneras. La primera es la posibilidad de utilizar elementos tecnológicos que por sus especificaciones permitan ofrecer este servicio. La segunda y más acertada es la de planificar los procedimientos que el sistema utilizará así como los recursos y sus componentes con el fin de orientar las soluciones de tal manera que se aprovechen las posibilidades tecnológicas de la manera más eficiente.

En lo que respecta a la tecnología y en general a los recursos de los que ya se ha hablado hay un problema a tener en cuenta que no se ha expuesto en los puntos donde se habla de ello. El principal problema que este sistema tendrá es el crecimiento de los documentos a consultar. Esto se traduce en problemas de almacenamiento de recursos, indexación de claves en bases de datos y en definitiva requerirá acciones de ahorro de espacio de almacenamiento y mantenimiento eficiente de recursos. Aunque el crecimiento de los documentos indexados crezca de manera exponencial, los propios términos, entidades y otros elementos que intervendrán en estas indexaciones no lo harán o no deberían hacerlo de la misma manera. Debido a que el conjunto terminológico dentro de un lenguaje

es limitado, las repeticiones de estos términos a lo largo del universo de documentos a consultar se verá incrementada. Aunque en un principio estos términos crezcan de una manera muy amplia, en el mismo rango temporal y muy seguramente según avance su tiempo de vida, los términos comenzarán a repetirse en los análisis realizados, generando una estabilidad aparente. Esta estabilidad será contrarrestada por la actualización de relaciones entre los términos y documentos que irán cambiando con el tiempo. Debido a esto último, la actualización de los análisis de cada documento es muy importante y deberá hacerse paulatinamente. La evolución de estos hechos se explicará en el apartado referente a los planteamientos futuros sobre el sistema.

Otro punto a tener en cuenta es el interés en focalizar el diseño del sistema y de la plataforma en torno a la eficiencia y a la escalabilidad. El diseño realizado en la arquitectura permite aumentar o disminuir según la necesidad el número de elementos intervinientes para equilibrar el uso del sistema. Por otro lado, el sistema es escalable en cuanto a procesos se refiere, de tal manera que muchos procesos serían capaces de funcionar de manera simultánea desde distintos puntos establecidos en la red. En la propia arquitectura del sistema se establecen dos máquinas para la ejecución de estos procesos y además los recursos están distribuidos a lo largo de la red. El mayor inconveniente en este tipo de sistemas paralelos es sin duda la utilización de los recursos compartidos de manera eficiente. Siguiendo con las elecciones realizadas a la hora de analizar el diseño de la arquitectura, no deberían existir problemas en los accesos a los recursos compartidos por parte de los procesos. Tanto los recursos almacenados en bases de datos, las cachés o los ficheros físicos, tienen operaciones bloqueantes de estos recursos con el fin de que el contenido no sea modificado a la vez por dos procesos distintos. A su vez, los gestores de almacenamiento de los recursos compartidos están implementados de manera eficiente.

10. FUTURAS LÍNEAS DE TRABAJO

El sistema, según está diseñado en este documento estaría preparado para ser implantado siguiendo las pautas establecidas para su utilización real. Según lo establecido, el diseño de la arquitectura incluye dos entornos de desarrollo, uno de ellos para producción y el otro para desarrollo. De esta manera, un equipo de personas podrían estar evolucionando el sistema cuando fuese necesario en cuanto a funcionalidades se refiere o las propias especificaciones del motor de indexación de documentos.

Por otro lado, la evolución del sistema en cuanto a plataforma se refiere tendrá que ver con ciertos factores que afecten tanto al rendimiento como a la eficiencia del propio sistema. El funcionamiento habitual y evolutivo del sistema parte de la base de la elección de una semilla o documento origen sobre el que tras ser procesado se recuperarán otros documentos para analizar. Esto sucederá sucesivamente consiguiendo un crecimiento prácticamente exponencial del número de documentos analizados. Finalmente, esto repercute en el número de términos asociados con los documentos indexados y sus relaciones con los mismos. A lo largo del tiempo, se verá que el número de documentos es tan grande que afectará de la siguiente manera:

- El espacio necesario para almacenar los recursos de cada documento, imágenes, ficheros de contenido asociado, términos y relaciones en las bases de datos aumentará considerablemente en número.
- Debido a que los documentos se tienen que volver a analizar cada cierto tiempo con el fin de comprobar que las relaciones establecidas en el índice son correctas en el momento del análisis, aumentará la necesidad de generar nuevos procesos de análisis.

La problemática del espacio de almacenamiento se puede resolver fácilmente ampliando el propio espacio, asignando nuevas máquinas que se dediquen a gestionarlo, siguiendo la arquitectura establecida en el diseño. Por otro lado, el incremento paulatino en el número de documentos indexados afectará al número de procesos que estarán ejecutando en las máquinas para analizar tanto documentos nuevos que sean introducidos en el sistema como los ya indexados. El diseño del software, así como el diseño de la arquitectura están pensados de tal manera que varios procesos puedan utilizar los mismos recursos de una manera ordenada, con una planificación que no genere bloqueos debido a los accesos. De esta manera, ampliar el número de procesos será una necesidad, sobre todo aquellos que sean más costosos como el proceso de análisis de términos y el proceso de análisis semántico. Las máquinas que se han establecido en el diseño de la arquitectura pueden ser suficientes para albergar el funcionamiento normal de un gran número de procesos de manera simultánea, sin embargo, estas máquinas pueden quedarse pequeñas y será necesario aumentar su número. Este hecho está contemplado en el diseño de la arquitectura, permitiendo ampliar el número de máquinas que ejecuten procesos si fuese necesario.

Otro caso que puede afectar a la utilización futura del sistema es el tráfico saliente generado. Esto es, en definitiva, el número de usuarios que realizan consultas contra el sistema para recuperar información. El problema puede llegar desde varios frentes:

- Se puede llegar a un punto en el que el tráfico generado no sea suficiente para las redes utilizadas en la conexión del sistema con Internet. En este caso, la modificación del sistema de redes o la ampliación de los soportes de la red en la arquitectura del sistema, bastaría para subsanar el problema.
- Por otro lado, está la utilización de recursos de acceso rápido. Las cachés que serán utilizadas para mostrar resultados ante la consulta de un usuario pueden quedarse pequeñas en cuanto a eficiencia se refiere. Esto puede subsanarse ampliando el número de máquinas y en definitiva de las cachés ofrecidas a los servidores frontales.

- El último posible caso que puede darse es el aumento en la carga de los propios servidores frontales. En este caso, el balanceo generado para que una sola rama del sistema no soporte toda la posible carga de peticiones debería ampliarse con un nuevo grupo de frontales que pueda ser accedido desde fuera.

Fuera ya de los problemas futuros que puede tener el sistema, existen también posibles opciones de aumentar tanto las funcionalidades como mejorar el análisis de la información:

- Por un lado, como se ha explicado en este documento, el sistema está orientado a analizar documentos en un idioma específico, el castellano. Una posible manera de mejorar el sistema en un futuro es ampliar el abanico de posibilidades del lenguaje a otros idiomas. Esto generaría documentos de distintos idiomas que será necesario categorizar como tales o incluso abrir una rama nueva de almacenamiento de información específica para otro idioma. Finalmente, la consulta realizada por el usuario debería ser capaz de recuperar información de todos los conjuntos de idioma existentes.

- Otra posible mejora o ampliación sería aumentar las posibilidades de análisis semántico sobre un idioma concreto que ya esté implementado. En este caso, debería ampliarse el análisis de relaciones semánticas entre entidades, mejorar el análisis morfológico y crear nuevas interrelaciones entre documentos en base a estos análisis.

APÉNDICE I: GLOSARIO DE TÉRMINOS

En el presente documento pueden encontrarse una serie de términos que pueden resultar ambiguos en su contexto si no se comprenden adecuadamente. En adelante se enumeran algunos de los términos que aparecen en el documento con una descripción de los mismos.

- **Tecnologías de la Información**: También conocido como TIC o tecnologías de la información y la comunicación, es un concepto relacionado con la informática y la tecnología. Se entiende como un conjunto de recursos, técnicas y procedimientos usados en el procesamiento, almacenamiento y transmisión de información.
- **Software**: Aunque es un anglicismo, se entiende como un elemento lógico o un equipamiento lógico de un sistema informático. Estos componentes lógicos están en contraposición con los componentes físicos, que son denominados hardware.
- **World Wide Web**: Es la definición del acrónimo WWW o comúnmente conocida como la web. Se trata de un sistema de distribución de documentos de hipertexto distribuidos por servidores informáticos conectados a redes. Por lo general está asociado a la distribución de dichos contenidos a través de Internet.
- **W3C**: Es la abreviatura del World Wide Web Consortium, un organismo internacional que desarrolla estándares y recomendaciones para elaborar de manera adecuada recursos a través de la red.
- **HTML**: Abreviatura de HyperText Markup Language que hace referencia al lenguaje de marcado utilizado en los documentos de hipertexto que dan forma a una página web. Se trata de un estándar que a lo largo del tiempo se ha utilizado como referencia para elaborar documento de hipertexto y que son inteligibles por una serie de programas informáticos llamados navegadores. Estos programas son capaces de interpretar el lenguaje de marcado y de mostrar visualmente el contenido de un documento de manera estructurada.
- **URL**: Es la abreviatura de Uniform Resource Locator – en castellano, localizador de recursos uniforme. Se trata de un identificador que indica la localización de manera inequívoca y única de un documento de hipertexto en la red.
- **Hipervínculo**: De manera análoga al concepto de URL, un hipervínculo es un enlace que mediante un protocolo de conexión permite llegar a un recurso o un documento de hipertexto a través de la red.
- **Framework**: Se trata de un conjunto estandarizados de conceptos, prácticas y criterios para enfocar un tipo de problema particular que utilizado como referencia, permite resolver otros problemas con características similares. En el desarrollo de software, se entiende como framework a una serie de componentes lógicos y estructuras tecnológicas que ofrecen un soporte adecuado sobre el que trabajar para ofrecer soluciones a problemas mediante la aplicación o la extensión de dichos componentes.
- **Google**: Google Inc. es una empresa norteamericana que ofrece productos y servicios dentro del ámbito tecnológico. Es mundialmente conocida por su motor de búsqueda a través de la red y por otras tecnologías en torno al correo electrónico, la distribución audiovisual a través de la red, etc.
- **Bing**: Es un motor de búsqueda a través de la red desarrollado por la empresa norteamericana Microsoft.
- **Información Meta**: En el ámbito de los documentos de hipertexto, la información meta es aquella que describe la información contenida en dicho documento.
- **CMS**: Abreviatura de Content Management System, en castellano, sistema de gestión de contenidos. Se trata de una solución software basada en componentes lógicos que unidos

ofrecen la posibilidad de gestionar los contenidos organizados en documentos de hipertexto ya sean editoriales o de otra índole.

- **XML**: siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por elWorld Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.
- **JSON**: acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
- **Sistema Multihilo**: Se trata de un sistema que está diseñado de tal manera que permite la ejecución simultanea de varias instancias de subprocesos similares para conseguir un propósito definido.
- **Base de Datos**: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Existen programas denominados sistemas gestores de bases de datos, abreviado DBMS, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.
- **Base de Datos Relacional**: Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California),
- **MySQL**: Se trata de un Sistema de gestión de bases de datos relacional, que permite a multiples usuarios gestionar un número libre de bases de datos. Actualmente es uno de los productos más utilizados en el mercado y presenta unas características apropiadas para su uso en servidores web. Es un software de código abierto bajo licencia GNU GPL.
- **PostgreSQL**: Es un Sistema de gestión de bases de datos relacional y orientado a objetos. Es un software de código abierto y ofrece la posibilidad de configurar múltiples bases de datos para ser usadas por múltiples usuarios. Se ofrece bajo licencia BSD.
- **Linux**: Es uno de los términos utilizados para referirse a la combinación del núcleo desarrollado en código abierto utilizado en ciertos sistemas operativos.
- **Timeout**: Se trata de un concepto propio de la comunicación entre redes que define el evento producido al no poderse establecer una conexión de manera adecuada donde el emisor de la comunicación no recibe respuesta del receptor potencial. Este concepto además se produce de manera premeditada, es decir, para detectar que este suceso tiene lugar es necesario definir un tiempo mínimo de respuesta para el cual una conexión ha tenido éxito. En caso de que el tiempo de respuesta sobre pase el definido se llegará a la situación explicada anteriormente.
- **Cookie**: Se trata de una estructura de información que es enviada por un servidor HTTP junto a la respuesta de la petición realizada por un navegador y que incluye información relevante tanto para la utilización de los servicios del sistema que se consulte como para almacenar información de la sesión del usuario. Esta información se almacena en espacios habilitados para tal efecto por parte de los programas clientes que realizan peticiones contra servidores HTTP a través de la red.
- **UTF-8**: Se trata de un formato de codificación de caracteres creado por Robert C. Pike y Kenneth L. Thompson. Está definido bajo un estándar y permite representar cualquier carácter Unicode.
- **Unicode**: Es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático y la visualización de textos de múltiples lenguajes, incluyendo lenguas muertas. El estándar es mantenido por el Consorcio Unicode al cual muchas empresas importantes del sector de la informática pertenecen.

- **Protocolo de comunicaciones**: Es un conjunto de reglas que permiten a dos o más entidades de un sistema de comunicación establecer procesos de transmisión de información por medio de un medio físico.
- **MongoDB**: Es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto y que forma parte de los sistemas de bases de datos NoSQL. En lugar de almacenar la información en tablas como se realiza en sistemas bases de datos relacionales, las estructuras de datos son documentos en formato JSON.
- **Tokyo Cabinet**: Es un sistema de base de datos NoSQL orientado a ofrecer un gran rendimiento en sistemas que así lo precisan. Este sistema ofrece la posibilidad de almacenar información en estructuras del tipo clave-valor, para los cuales se realiza el acceso al información a través de sus claves.
- **HTTP**: Es un protocolo de transmisión de información desarrollado por el W3C (World Wide Web Consortium) y que actualmente forma parte de un estándar. El protocolo define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web para comunicarse con los clientes a través de un navegador.
- **HTTPs**: Es un protocolo de transmisión de información basado en el protocolo HTTP pero que está destinado a una transferencia segura de la información. Actualmente y cada vez más empresas utilizan este protocolo para transmitir sus datos ya que su funcionamiento está basado en la utilización de una capa de cifrado de la información antes de su transmisión. Para realizar la ofuscación de la información se utilizan procedimientos de cifrado basados en certificados de autenticación.
- **HTTP Basic Authentication**: Es un método utilizado en el protocolo HTTP para autenticar un usuario por medio de un nombre de usuario y una contraseña para realizar la comunicación de la información. Representa la forma más básica de autenticación que un servidor web puede ofrecer a un usuario ya que no requiere de otros elementos como cookies, identificadores de sesión o páginas de inicio de sesión.
- **Software de código abierto**: Es un tipo de software conocido por ser distribuido y desarrollado libremente. Este tipo de software se beneficia mayoritariamente de los desarrollos realizados por los propios usuarios del software con el fin de ejecutar mejoras o solventar problemas encontrados en el código fuente. Aunque está relacionado con cuestiones éticas o de libertad, el software de código abierto se focaliza especialmente en sus beneficios prácticos.
- **Modelo MVC**: MVC son las siglas de "Model View Controller" que en castellano significa "Modelo Vista Controlador". Se trata de un patrón de diseño de arquitectura de software orientado a generar interfaces de usuario. El modelo permite abstraer a un usuario utilizar un "controlador" que manipula ciertos parámetros de un "modelo" que actualizará unas "vistas" que finalmente verá el usuario.
- **Logs**: Es un registro de sucesos catalogados durante la ejecución de un producto de software. Este registro puede estar acotado en el tiempo y representar el funcionamiento de muchos elementos de un sistema. En particular, estos registros se utilizan mayoritariamente con la intención de realizar un mantenimiento a lo largo del ciclo de vida de un software en concreto.
- **Motor de plantillas:** Se trata de una capa de abstracción incluida entre la solución de la vista final y del programa que la procesará. Es una solución software para agilizar el desarrollo de aplicaciones que contengan una visualización en sus funcionalidades y que permite al código fuente del producto ofrecer una salida visual por medio de un motor que entienden ambas partes.
- **CPU:** También llamado procesador es la unidad básica de un ordenador o dispositivo que utiliza cierta tecnología de procesamiento.

- **Modelo OSI**: Se trata de un modelo de interconexión entre sistemas para establecer una comunicación que les permita intercambiar información. Este modelo está basado en ciertas capas que establecen las pautas que los protocolos que lo adoptan deben cumplir. Las capas inferiores son capaces de procesar información para las capas superiores y todas ellas están ordenadas siempre de manera descendente de más abstracta a más básica siendo la más básica aquella que llega al nivel físico de la comunicación.
- **NFS**: Son las siglas de "Network File System" o "Sistema de ficheros en red". Es un protocolo que sigue el modelo OSI y que afecta al nivel de aplicación. Utilizado para sistemas de archivos distribuidos en un entorno de red de computadoras de área local.
- **Versión de Software**: Se trata de un procedimiento habitual utilizado para designar un identificativo único a un producto de software para indicar su nivel de desarrollo. En general la versión es un nombre o número compuesto por dos o más elementos que indican, tras el lanzamiento de nuevas versiones, las diferencias entre ambas.
- **Sistema de control de versiones**: Son aplicaciones generalmente orientadas a controlar y gestionar los cambios entre las distintas versiones del código fuente de un producto de software con el fin de poder revertir los cambios en caso de que sea necesario.
- **Almacenamiento en La Nube**: Es un modelo de almacenamiento de información que en lugar de realizarse en un ámbito local se realiza de manera distribuida a través de los protocoles de comunicación entre redes. Este hecho permite que un recurso almacenado pueda ser accedido desde dispositivos localizados en distintas redes.

APÉNDICE II: BIBLIOGRAFÍA

[1] Emma Jone, *The Literary Companion*. Robson books. p 115

[2] Kardex Remstar, *Case Study 029 – Frydenbo*. [Consulta: Abril 2014] <http://www.kardex-remstar.es/uploads/media/Case_Study_029_Frydenbo_ES_low.pdf>

[3] Vladislav Shkapenyuk y Torsten Suel, *Design and Implementation of a High-Performance Distributed Web Crawler*. Polythechnic University of Brooklyn: 2001. <http://cis.poly.edu/tr/tr-cis-2001-03.pdf>

[4] David Eichmann, *Balancing Effective Search Against Web Load*. University of Houston. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.8663&rep=rep1&type=pdf>

[5] Andreas Staeding, *List of User-Agents (Spiders, Robots, Crawler, Browser)*. [Consulta: Septiembre 2013] <http://user-agents.org>

[6] Kelvin Tan, *Lucence Scoring.* [Consulta: Marzo 2014] <http://www.lucenetutorial.com/advanced-topics/scoring.html>

[7] Joaquín Pérez-Iglesias, José R. Pérez-Agüera, Víctor Fresno, Yuval Z. Feinstein, *Integrating the Probabilistic Model BM25/BM25F into Lucene*. [Consultado: Marzo 2014] <http://nlp.uned.es/~jperezi/Lucene-BM25/>

[8][9] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press: 2008 - <http://nlp.stanford.edu/IR-book/html/htmledition/document-and-query-weighting-schemes-1.html>, <http://nlp.stanford.edu/IR-book/html/htmledition/hubs-and-authorities-1.html>

[10] Lawrance Page, Sergey Brin, Rajeev Motwani y Terry Winograd, The PageRank Citation Ranking: Bringing Order to the Web. Stanford University: 1999. <http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf>

[11] Sergey Brin y Lawrence Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. Stanford University - <http://infolab.stanford.edu/~backrub/google.html>

[12] Brian Pinkerton, Finding what people want: Experiences wht he WebCrawler. University of Washington: 1994. <https://web.archive.org/web/20010904075500/http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/pinkerton/WebCrawler.html>

[13] Hewlett-Packard Development Company, L.P., Autonomy IDOL Reference Architecture. [Consultado: Abril 2014] <http://www.ndm.net/archiving/pdf/20130424_PI_WP_HP_AUTN_IDOL_Architecture_web.pdf>

[14] Scrapy Developers, Scrapy at a glance. [Consultado: Mayo 2014] <http://doc.scrapy.org/en/latest/intro/overview.html>

[15] Sphinx Technologies Inc., About Sphinx [Consultado: Mayo 2014] <http://sphinxsearch.com/about/sphinx/>

[16] Wordnet, A lexical database for English. [Consultado: Octubre 2013] <http://wordnet.princeton.edu/>

[17] Real Academia Española. Real Academia Española. [Consultado: Octubre 2013] <http://www.rae.es/>

[18] Real Academia Española: Banco de datos (CORDE), Corpus diacrónico del español. [Consultado: Octubre 2013] <http://www.rae.es>

[19] Michael Gabriel Sumastre, Pluralsigh The Top 7 Most Reliable SSL Certificate Providers Pluralsight blog. <http://blog.pluralsight.com/top-reliable-ssl-certificates>

[20] The Apache Software Foundation, Apache Benchmarking Tool. [Consultado: Junio 2014] <http://httpd.apache.org/docs/2.0/programs/ab.html>

[21] Daniel Pecos Martínez, PosgreSQL vs MySQL [Consultado: Junio 2014] <http://danielpecos.com/documents/postgresql-vs-mysql/#AEN71>