Universidad de las Ciencias Informáticas

Facultad 1

Plugin para el cálculo de la popularidad de enlaces en Nutch.

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autor: Lisdaly Crespo Hernández

Tutores: Ing. Daneysi Hernández García

Ing. Eyeris Rodríguez Rueda.

La Habana, junio de 2017

"Año 59 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Lisdaly Crespo Hernández, 94090334277 soy el autor principal del trabajo titulado ‘’Plugin para el cálculo de la popularidad de enlaces en Nutch’’ y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los < > días del mes de junio del año 2017.

Autor:

_____

Lisdaly Crespo Hernández

Tutores:

_____ _____

Ing. Daneysi Hernández García Ing. Eyeris Rodríguez Rueda.


¨El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; lo que más estamos sembrando son oportunidades a la inteligencia(...)¨
Fidel Castro.

DEDICATORIA

Dedico este trabajo de diploma, este punto culminante de mis cinco años de universidad a todas aquellas personas que cuentan con las ansias de estudiar y de superarse pero que por distintas razones no han tenido la posibilidad. Especialmente a mi abuela Zenaida o como todos la conocemos "Chelo"; una mujer excepcional en toda la magnitud que la palabra admite, con una gran inteligencia y sabiduría, pero que las condiciones históricas de su juventud no le permitieron estudiar como ella deseaba. A ti, mi abuela del alma, mi guía, mi motor impulsor; dedico mi tesis, mi universidad, y todos los logros que alcance en la vida.

AGRADECIMIENTOS

Es difícil redactar esta página sin que se quede nadie fuera, pero trataré de hacerlo lo mejor posible:

Agradezco a Dios por permitirme llegar hasta estos días, por darme las fuerzas y la entereza que solo él puede darme.

Agradezco a mi país, a mi revolución, por darme esta oportunidad que tal vez en otro país no hubiese tenido.

A mis padres, que me han dado todo a su alcance, principalmente su apoyo y amor que es todo lo que necesito.

¿A mi tío Félix, que puedo decir de mi guía, mi otro papá? Pues, gracias por tus palabras que contienen el peso necesario para hacerme pensar, gracias por tu amor, por tu apoyo, por estar hay para mí y para todos, simplemente no puedo expresarte mi gratitud.

A mis abuelos maternos, más que abuelos ustedes son mis ángeles, especialmente tu mi abuelita Chelo. ¿Qué sería de mi sin ti? Gracias por tu amor que me nutre y me impulsa, gracias por ser mi abuela, no existe privilegio mayor en mi vida.

A mi hermano Lidier y mi cuñada, ustedes también han estado presente en toda esta carrera y son eslabón clave, los quiero.

A toda la familia Reyes González, mi punto de apoyo, la otra familia que la vida me ha regalado, gracias por permitirme entrar en sus vidas y estar pendientes de mí estos cinco años, los quiero con el alma.

A mis compañeros de aula y del día a día, gracias por soportarme, por cada momento que compartí a su lado, y por su ayuda siempre oportuna y desinteresada, siempre estarán conmigo.

A mis tutores, gracias por su paciencia, por su interés, y por dedicarme algo tan valioso como el tiempo, que ese nunca se recupera.

Al resto de mi familia y amigos, que todos han aportado algo que me sustenta, que me ha ayudado a seguir adelante.

A ti te dejo para el final, pero te he tenido presente desde que comencé a escribir estas líneas. ¿Qué pudiera escribir que ya no te haya dicho? Mi gratitud no puede expresarse con letras, pero por escrito debe quedar mi más profundo agradecimiento por permitirme ser parte de tu vida, por estar conmigo desde el primer año hasta el quinto, por no dormir cuando no duermo, por tu paciencia cuando no entiendo, por tu mano que me ha sustentado en los momentos más duros, gracias por amarme, sin ti esto no hubiese sido lo mismo; Elioenai Reyes González, gracias.

RESUMEN

El surgimiento de internet y la creciente publicación de artículos, han propiciado el aumento del conocimiento de las personas en diversos temas. Esto ha conllevado al aumento de la información recogida en la *web* con las publicaciones de documentos, las creaciones de páginas o sitios *web*, realización de portales, *blog*, etc. Este cúmulo de información ha dificultado la búsqueda para los usuarios de lo que puedan desear en momentos específicos. Para resolver estos problemas se diseñaron los Sistemas de Recuperación de Información. En la Universidad de las Ciencias Informáticas se desarrolla el buscador Orión, el cual utiliza como indexador Solr, Symfony para el desarrollo web, y tiene como mecanismo de rastreo a Nutch. Nutch pertenece a la fundación Apache, se encarga de recorrer la web para obtener información publicada en ella. La información que se muestra a los usuarios debe aparecer lo más ordenada posible según popularidad o importancia de las páginas. Actualmente Nutch le otorga el mismo nivel de popularidad a sus enlaces, lo cual dificulta la búsqueda para los usuarios. La presente investigación permitió desarrollar un *plugin* para Nutch, el cual posibilita establecer un peso a cada página o documento almacenado en Solr y así establecer la popularidad de cada enlace. Para la implementación de la propuesta de solución, guiada por la metodología AUP-UCI se seleccionaron como principales tecnologías Nutch y Solr, como lenguaje de programación Java para desarrollar en el IDE *NetBeans*, además utilizando el Lenguaje Unificado de Modelado (UML) se escogió Visual Paradigm para el modelado del proceso en general. El plugin para establecer la popularidad de los enlaces en Nutch, permite calcular un peso o popularidad a cada enlace según una serie de aspectos, que contribuyen a la disminución o aumento del *score* del enlace.

Palabras claves: Nutch, popularidad, Sistemas de Recuperación de Información.

Tabla de contenido

INTRODUCCIÓN

Desde 1969, con el nacimiento de la Red de la Agencia de Proyectos de Investigación Avanzada (ARPANET por sus siglas en inglés) a petición del Departamento de Defensa de los Estados Unidos, las comunicaciones a nivel mundial han trascendido de manera asombrosa. Debido a la importancia de la comunicación, como una actividad fundamental presente en cualquier proceso social, Internet[1] se ha convertido en un medio imprescindible para el desarrollo del mundo moderno. La creciente interacción de los usuarios con la red, ha propiciado el auge de nuevas estrategias para el intercambio de información digital, revolucionando la forma de hacer política, construir economías e influir positivamente en la cultura de millones de personas (Alarcón y otros, 2009).

La cantidad de recursos[2] presentes en Internet muchas veces puede desorientar a los usuarios a la hora de buscar algún tema de interés. La falta de información ya no es un problema, lo que muchas veces pasa es que los usuarios no saben dónde está la información deseada. Los Sistemas para la recuperación de información intentan ayudar a los usuarios en situaciones como esta. (Rueda, y otros, 2012)

La recuperación de información son técnicas empleadas para almacenar y buscar grandes cantidades de datos y ponerlos a disposición de los usuarios según (Martínez, 2004). En el centro de Ideoinformática (CIDI), perteneciente a la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI), se desarrolla el buscador Orión. Este buscador cuenta con tres componentes fundamentales: mediante su interfaz web posibilita las consultas para las búsquedas a los usuarios, Solr como servidor de indización[3] de contenidos y Nutch, se encarga de rastrear la información en la web. Orión brinda servicios de búsqueda y recuperación de información, permitiendo a los usuarios un acceso a los recursos publicados en la intranet de la universidad.

[1] El nombre Internet procede de las palabras en inglés ***Inter****connected* ***Net****works*, que significa "**redes interconectadas**". Internet es la unión de todas las redes y computadoras distribuidas por todo el mundo, por lo que se podría definir como una **red global en la que se conjuntan todas las redes que utilizan protocolos TCP/IP y que son compatibles entre sí.**

[2]Se hace referencia a todo objeto que ofrece información. Puede ser en formato mp3, pdf, png, html, etc.

[3] Registrar ordenadamente datos e informaciones para elaborar su índice.

Nutch es un spider[4] o web *crawler* libre y de código abierto, desarrollado totalmente en Java por *The* Apache *Software Foundation.* Ha sido implementado sobre la base de Apache Lucene[5], una librería de alto rendimiento para la búsqueda basada en texto y que utiliza una modificación del algoritmo *"Vector Space Model*[6]" (Modelo de Espacio Vectorial en español), con un enfoque booleano que restringe las estimaciones de los resultados obtenidos (Alemán y otros, 2015).

Debido a que cada página es una fuente de información valiosa, es de especial interés poder establecer una popularidad dentro de la red de páginas donde se encuentra. Para los buscadores que rastrean la *web*, los enlaces son las vías de acceso entre las páginas, los *spiders* o *crawler* se encargan de recorrer la *web* siguiendo los enlaces presentes en las páginas o documentos, este recorrido puede ser realizado en profundidad o a lo ancho (Rueda y otros, 2012).

Desde finales de 1990 los enlaces han servido también como votos, representando la democracia de la opinión de la *web* sobre qué páginas son importantes y populares. A través de los enlaces es que los buscadores no sólo analizan la popularidad de un sitio web basado en el número y popularidad de las páginas que lo enlazan, sino que también pueden analizar métricas como confianza, *spam*[7] y autoridad. Los sitios confiables tienden a enlazar a otras páginas confiables, mientras que las páginas de *spam* reciben muy pocos enlaces de fuentes confiables (Silva, 2016).

La popularidad de un enlace o recurso puede representarse como una función matemática en la cual intervienen varias variables. Entre los algoritmos más conocidos para el cálculo de esta popularidad se encuentra el *PageRank* de Google. Nutch utiliza varios *Plugin* para el cálculo de ranking de los enlaces, entre los que se encuentran *Scoring-depth*, *Scoring-link*, y el *Scoring-Opic*. Con la utilización de los anteriores plugins, ante cualquier inconsistencia en la base de datos, el valor de popularidad de algunos enlaces se incrementa desmedidamente, provocando una relevancia errónea a los enlaces y dificultando así la búsqueda para los usuarios. Actualmente en el buscador Orión se utiliza una propiedad del sistema

4 Un spider o araña, puede definirse como una aplicación que se conecta a Internet periódicamente y recorre la Web en busca de información pública (Kumar, 2010).

5 Es un Api de código abierto. Es útil para cualquier aplicación que requiera indexado y búsqueda a texto completo.

6 Se conoce como **modelo de espacio vectorial** a un modelo algebraico utilizado para filtrado, recuperación, indexado y cálculo de relevancia de información

7 Se define **SPAM** a los mensajes no solicitados, habitualmente de tipo publicitario, enviados en forma masiva. La vía más utilizada es la basada en el correo electrónico pero puede presentarse por programas de mensajería instantánea o por teléfono celular.

para otorgar el mismo nivel de popularidad a todos sus enlaces. Un correcto valor de popularidad otorgado a un enlace, proporciona una adecuada relevancia, y así facilita la búsqueda a los usuarios. Considerando la situación problemática anteriormente descrita, se plantea: **¿Cómo establecer la popularidad de los enlaces en Nutch?,** interrogante que constituye el problema de investigación del presente trabajo. Constituyendo **como objeto de estudio:** los procesos de Recuperación de información; enfocándose en el proceso de asignación de popularidad a los enlaces en Nutch como **campo de acción**.

Como respuesta al problema antes planteado se define como **objetivo general**: desarrollar un *plugin* utilizando tecnologías de la web y de recuperación de información que establezca la popularidad de los enlaces en Nutch.

Para una realización gradual del objetivo general este se ha descompuesto en los siguientes **objetivos específicos**:

1. Caracterizar el marco teórico conceptual y el estado del arte respecto a las tecnologías actuales para el desarrollo de algoritmos para la popularidad de enlaces.
2. Diseñar un *plugin* que permita establecer la popularidad de los enlaces identificados por Nutch.
3. Implementar un *plugin* que permita la asignación de popularidad a los enlaces identificados por Nutch.
4. Validar el correcto funcionamiento del *plugin* de popularidad de enlaces.

A partir de lo anterior, se establecen un conjunto de tareas de investigación que guiarán la realización del trabajo:

- Caracterización de los fundamentos teóricos de los procesos de Recuperación de información y los SRI para elaborar el marco teórico de la investigación.
- Caracterización de los mecanismos de rastreo utilizados en los SRI para establecer niveles de popularidad en Nutch.
- Definición de las herramientas, metodologías y lenguaje de programación para el diseño e implementación de la propuesta de solución.
- Elaboración de los artefactos necesarios para el diseño y la implementación de la propuesta de solución.
- Implementación de la propuesta de solución para establecer popularidad en Nutch.
- Diseño de las pruebas de *software* para validar la propuesta de solución.
- Ejecución de las pruebas de *software* para la validación de la propuesta de solución.

Par dar cumplimiento a las tareas de investigación, se utilizaron los siguientes métodos de investigación:

- **Histórico –Lógico:** con el objetivo de constatar teóricamente cómo han evolucionado en el tiempo los Sistemas de Recuperación de Información, específicamente sus mecanismos para establecer un nivel de popularidad a sus documentos.
- **Analítico –Sintético:** empleado para el análisis de los elementos esenciales referentes a las teorías, documentos y literatura en general relacionada con los Sistemas de Recuperación de Información.

Estructura del documento:

Capítulo1. Fundamentación teórica de los Sistemas de Recuperación de Información y sus métodos para establecer niveles de popularidad en sus documentos. Se realiza el levantamiento bibliográfico, el estudio de los fundamentos teóricos, además se exponen características a considerar, sobre los Sistemas de Recuperación de Información y sus métodos para el cálculo de popularidad de sus documentos. Además, se define la metodología, herramientas y lenguaje de programación que se utilizarán.

Capítulo 2. Diseño de la propuesta de solución: se exponen las características del *plugin*, además de los requisitos funcionales y no funcionales, patrones de diseño y arquitectura utilizada; y algunos de los artefactos que requiere la metodología de desarrollo utilizada.

Capítulo 3. Implementación y validación del *plugin* de cálculo de *ranking* de enlaces en Nutch: se exponen algunos aspectos asociados con la implementación de la solución informática, y los componentes que la integran. Además, se presentan los diseños de casos de prueba a utilizar en la validación del *plugin* y se analizan los resultados de las pruebas realizadas que permiten evaluar la calidad de la propuesta de solución.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo con el objetivo de favorecer una mejor comprensión de la investigación que se presenta, se realiza una revisión bibliográfica acerca de los elementos y áreas del conocimiento que engloban al objeto de estudio y al campo de acción. Además, se hace un análisis y evaluación de los principales algoritmos utilizados para establecer un nivel de popularidad o peso a los enlaces en Nutch y se exponen las características esenciales de las herramientas, las tecnologías y la metodología de desarrollo de software que se utilizarán para la implementación de la solución.

La información está presente en imágenes, textos, señales, sonidos, la cual constituye un mensaje que cambia el estado de conocimiento de la persona o sistema que la recibe. Como sabemos el entorno va cambiando constantemente debido a los avances tecnológicos, sociales, culturales y políticos que va teniendo la sociedad y es por eso que los medios de comunicación han tenido que ir renovándose para no perder de vista su objetivo principal, el cual es informar y generar contenidos. (Roastbrief, 2015)

La tendencia actual de las publicaciones es la sustitución del formato impreso por el medio electrónico, es un proceso acelerado, que se inició en la década de los años 1990, con el desarrollo de *World Wide Web* (WWW) y el HTML. Estos avances produjeron un incremento notable de la difusión del conocimiento, con formas muchas veces incontrolables. (Aguilar, 2003)
Teniendo en cuenta las palabras del científico británico o también conocido como el padre de la *web*, Berners-Lee: *"El concepto de la Web integró muchos sistemas de información diferentes, por medio de la formación de un espacio imaginario abstracto en el cual las diferencias entre ellos no existían. La Web tenía que incluir toda la información de cualquier tipo en cualquier sistema."*

Es evidente que el aumento de información en la *web*, la velocidad en que se transforma y su uso desmedido, pueden dificultar la búsqueda para los usuarios. Es aquí donde juegan un papel importante los sistemas de recuperación de la información. Estos sistemas son capaces de encontrar la información de una forma rápida y organizada.

1.1 Fundamentos Teóricos

La Recuperación de Información (RI) no es un área nueva, sino que se viene desarrollando desde finales de la década de 1950. Sin embargo, en la actualidad adquiere un rol más importante debido al valor que tiene la información. Se puede plantear que disponer o no de la información justa en tiempo y forma puede resultar en el éxito o fracaso de una operación (Tolosa, 2007).

La recuperación de información se ha asociado a muchos campos de la ciencia, como: la Informática, Ciencia de la Información, Documentación, Lingüística, entre otras, por esta razón muchos investigadores han aportado su visión propia del tema. Aunque se puede hacer sin el uso de herramientas informáticas no es lo más común hoy en día.

Según Korfhage la Recuperación de Información (RI) es: "la localización y presentación a un usuario de información relevante a una necesidad de información expresada como una pregunta" (Korfhage, 1997).

Aunque desde puntos de vista individuales y enfrentándose al transcurso del tiempo, sus definiciones han cambiado, mantiene una esencia no lejana de lo que plantean los profesores Tolosa y Bordignon: "*la RI consiste en encontrar y ordenar documentos relevantes que satisfagan la necesidad de información de un usuario, expresada en un determinado lenguaje de consulta"* (Tolosa, 2007).

De lo antes expresado se puede decir que la Recuperación de Información se lleva a cabo mediante un lenguaje de interrogación adecuado a la base de datos donde se almacena la información estructurada. Para un mejor desarrollo de este proceso surgen los Sistemas de recuperación de información.

1.1.2 Sistemas de Recuperación de Información

Debido a la cantidad de información que se puede encontrar en internet, como consecuencia al uso desmedido de la misma, puede resultar difícil para un usuario encontrar lo que realmente necesita en el tiempo deseado. Los Sistemas de Recuperación de Información (SRI) pueden resultar una respuesta a esta situación, debido a que localizan y procesan todo tipo de documentos en la red. Según Salton "*cualquier SRI puede ser descrito como un conjunto de ítems*[8] *de información, un conjunto de peticiones y algún mecanismo que determine qué ítems satisfacen las necesidades de información expresadas por el usuario en la petición"* (Salton, 1986).

[8] Cada uno de los elementos que forman parte de un dato.

Con el transcurso del tiempo los sistemas de recuperación de información han ido adecuándose a las nuevas tecnologías y a las demandas de los usuarios.

1.1.3 Clasificación de Los Sistemas de Recuperación de información

Baeza-Yates afirma que hay básicamente tres formas de buscar información en la *web*: “dos de ellas son bien conocidas y frecuentemente usadas. La primera es hacer uso de los motores de búsqueda, que indexan una porción de los documentos residentes en la globalidad de la *web* y que permiten localizar información a través de la formulación de una pregunta. La segunda es usar directorios, sistemas que clasifican documentos web seleccionados por materia y que nos permiten navegar por sus secciones o buscar en sus índices. La tercera es buscar en la web a través de la explotación de su estructura hipertextual (de los enlaces de las páginas web)” (Baeza-Yates, 1999).

La evolución lógica de los SRI ha sido hacia la *web*, donde han encontrado una alta aplicación práctica y un aumento del número de usuarios, especialmente en el campo de los directorios y motores de búsqueda (Martínez, 2004).

Los directorios son unos de los primeros sistemas de búsqueda existentes, se dice que el primero que hubo fue el directorio de Yahoo!, el cual opera desde 1993, hoy día se estima que tiene alrededor de 3.000.000 de páginas indexadas. Se considera una barata tecnología, que es ampliamente utilizada por la cantidad de programas *scripts* en el mercado. Este tipo de buscador presenta la información de la *Web* en forma de una colección de categorías, y las búsquedas se realizan en profundidad, una vez que se selecciona una categoría, se muestran las subcategorías contenidas en ella, además de los sitios que se relacionan directamente con dicha categoría. La indexación se realiza de forma manual por lo editores del sistema. Presenta una estructura en forma de árbol, en el que se agrupa en las ramas principales las categorías o temas más generales y amplios, por ejemplo (artes, deportes, computación), obsérvese que estas categorías son capaces de encerrar dentro de ellas a muchas otras. No se requieren muchos recursos de informática. Están muy extendidos en la red por estos motivos. En cambio, se requiere más soporte humano y mantenimiento. El hecho de que los sitios se incluyan de forma manual implica que los resultados de la búsqueda sean mejores y que el ruido sea casi nulo, esto quiere decir que los resultados se corresponden en un gran porciento al tema de búsqueda, una desventaja es que el costo de operación es relativamente alto al ser el factor humano fundamental en el funcionamiento del mismo. A diferencia de los buscadores, se centran en la calidad y no en la cantidad. Como principal ventaja está la gran calidad

de sus contenidos y desventaja con respecto al crecimiento que tiene la *Web* y la velocidad de clasificación (Ecured, 2006). Sus algoritmos son mucho más sencillos, presentando la información sobre las webs registradas como una colección de directorios. No recorren las webs ni almacenan sus contenidos. Solo registran algunos de los datos de las páginas. Más que buscar información sobre contenidos de la página, los resultados serán presentados haciendo referencia a los contenidos y temática de la web. Son apropiados para buscar categorías, más que informaciones específicas. Para visitar sitios de temática común. Es la tecnología que utilizan portales y buscadores de sectores especializados. Ejemplos de directorios: Antigus directorios, Yahoo!, Terra. Ahora, ambos utilizan tecnología *spider*, y Yahoo!, conserva su directorio. (Fernández, 2017). Existen directorios temáticos, o sea enfocados en un contenido específico y otros abiertos, en los cuales es posible encontrar referencias sobre cualquier tema. Normalmente funcionan permitiendo que les sean sugeridos sitios a incluir en su base de datos, los que son verificados manualmente y si son adecuados se incluyen con una corta descripción de su uso (Carrodeguas, 2010). A continuación, se muestran distintas clasificaciones referentes a los directorios.

- **Directorios de pago.** Es necesario pagar para que incluyan nuestro enlace. Son muy beneficiosos para el posicionamiento ya que al poseer menos enlaces transmiten mayor autoridad y en ocasiones *PageRank*.
- **Directorios gratis.** Es posible dar de alta a una *web* de forma gratuita. Son de poco interés para el posicionamiento por estar muy saturados y en muchos casos son considerados por los buscadores como emisores de *webspam* (Una excepción es DMOZ).
- **Directorios por registro.** Son gratis, pero es necesario registrarse, sugerir un sitio y esperar por una revisión. Los beneficios son similares a los anteriores.
- **Directorios recíprocos.** Son gratis, pero exigen a cambio un vínculo para ser listados o sea se crea un intercambio de enlaces que favorece a ambos. Son más relevantes que los gratuitos.
- **Directorios por pujas**. Es una modalidad nueva. Son de pago, en ellos se ocupa una posición dependiendo del precio que se pague mediante subasta. Cuanto más se pague mejor posicionada estará la página.

Los metabuscadores son sistemas desarrollados para mitigar el problema de tener que acceder a varios motores de búsqueda con el fin de recuperar una información más completa sobre un tema, siendo estos mismos sistemas los que se encargan de efectuarlos por el usuario. Un metabuscador colecciona las

respuestas recibidas y las unifica, la principal ventaja de los metabuscadores es su capacidad de combinar los resultados de muchas fuentes y el hecho de que el usuario pueda acceder a varias fuentes de forma simultánea a través de una simple interfaz de usuario. Estos sistemas no almacenan direcciones y descripciones de páginas en su base de datos, en lugar de eso contienen registros de motores de búsqueda e información sobre ellos. Envían la petición del usuario a todos los motores de búsqueda que tienen registrados y obtienen los resultados que les devuelven (Méndez, 2004). Algunos de los metabuscadores más utilizados son (Moya, 2012):

- **Zapmeta:** es uno de los que más gusta por su rapidez y por la cantidad de buscadores a los que consulta. Además, incluye búsqueda organizada en *clústeres* que pueden ayudar a filtrar aún más los resultados.
- **Zuula**: la diferencia con Zapmeta es que permite saber de dónde está extrayendo la información. Además, tiene la ventaja de poder buscar por *tags* y buscar dentro de *blogs*.
- **Soovle**: es uno de los meta-buscadores más originales en su visualización. Lo que le diferencia de los demás, además de cómo muestra el contenido, es que en "*Engine*" se puede definir en qué buscadores quiere el usuario que consulte.
- **Metacrawler**: es uno de los más conocidos porque lleva mucho tiempo y es muy estable. Consulta en Google, Bing y Yahoo!. Permite buscar datos de contacto en "Páginas amarillas" y "Páginas blancas".
- **Entireweb**: es muy ágil e incluye búsquedas en tiempo real dentro de Twitter. La configuración avanzada es muy completa permitiendo idiomas, países y línea temporal.
- **Iboogie**: es otro de los mejores. Realiza búsquedas organizadas en el *clúster* y tiene un motor muy potente. Además, puedes añadir pestañas solicitando buscar en bases de datos concretas que tienen a disposición del usuario (no sólo de buscadores, sino también de webs).

Los buscadores consideran la *Web* como una gran Base de Datos de texto y tienen como objetivo "Indexar toda la Web". Actualmente son los más utilizados, emplean un mecanismo que recorre automáticamente la *Web* en busca de nuevos documentos y actualizaciones en los ya indexados. Este mecanismo es realizado por un s*oftware*, "*Spider*" que de cada documento recupera su información y la almacena en una base de datos. Una de las principales ventajas que presenta es que de cada url visitada es posible guardar la mayoría de sus elementos, título, texto íntegro, texto HTML, vínculos, permitiendo

realizar búsquedas sobre cada uno de estos campos, otra ventaja importante es que mediante el *Spider* es posible actualizar rápidamente los cambios producidos en Internet. Para realizar una búsqueda utiliza lo que se conoce como “Motor de Búsqueda”, esta herramienta es muy útil y tiene un gran peso en el éxito de cualquier buscador, el motor de búsqueda trabaja tomando los términos introducidos por la persona que está realizando la búsqueda y devuelve los documentos que se encuentran relacionados con dichos términos, la forma más habitual de hacer esta consulta es con lenguaje natural *(*Tramullas, 2001*).* Ejemplos: Google, Terra, Lupa, etc.

Los buscadores o motores de búsqueda son Sistemas de Recuperación de Información que permiten obtener aquellos documentos de mayor relevancia, a partir de un criterio de búsqueda introducido por el usuario. Desde la perspectiva del usuario, los buscadores deben cumplir dos requisitos fundamentales: “un tiempo corto de respuesta y una gran colección de documentos web disponibles en su índice. La calidad de un buscador reside en lo abundante, relevante y actualizada que sea su colección” (Baeza-Yates, 1999).

Los buscadores, entre otros componentes, constan de grandes bases de datos donde se almacena la información que sus *robots* de búsqueda extraen de la red. Cada buscador está configurado bajo una serie de parámetros que cada compañía considera los más adecuados para ordenar las búsquedas y mostrar los resultados (Moya, 2012).

“Los buscadores, motores de navegación o motores de búsqueda (MB), son aquellos programas o herramientas interactivas que facilitan la búsqueda y recuperación de información. Los motores de búsqueda ofrecen formularios para introducir los datos mediante una interfaz de fácil comprensión para el usuario, el cual teclea una palabra clave o frase y recupera una lista de recursos que se corresponden con el criterio indicado” (Camiño, 2003).

Se puede decir, de manera simplificada que un motor de búsqueda consta de cuatro partes (Archanco, 2015):

- Un interfaz para el usuario para hacer peticiones de búsqueda.
- Un *robot* o *spider* que busca la información en Internet.

- Un algoritmo que conecta las peticiones de los usuarios con la base de datos.
- Y una base de datos donde se han indexado los contenidos.

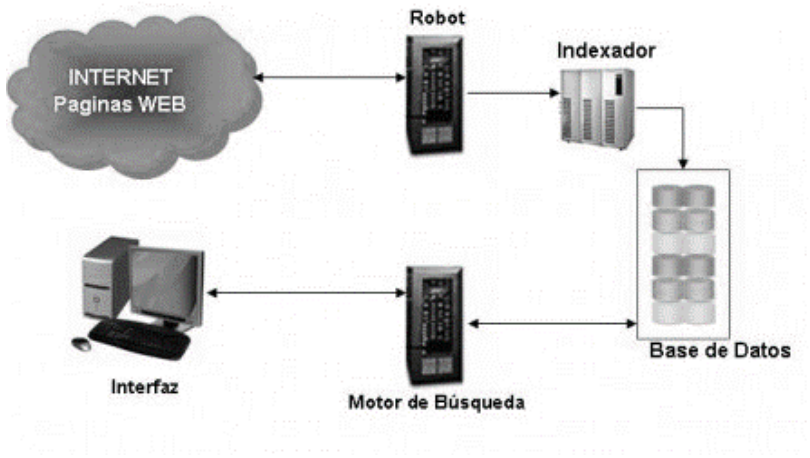


Figura 1: Arquitectura de un buscador (Archanco, 2015).

También se deben de tener en cuenta los buscadores verticales que son especializados en un sector o nicho concreto, lo que le permite analizar la información con mayor profundidad que un buscador genérico, disponer de resultados más actualizados y ofrecer al usuario herramientas de búsqueda avanzadas (Ricciardi, 2009).

Se puede concluir que los buscadores son programas o sistemas que facilitan la búsqueda de información en internet a los usuarios. Para facilitar o hacer más rápido y eficiente esta búsqueda cuentan con distintas funcionalidades y servicios especializados. A continuación, se muestra un estudio realizado sobre los buscadores más populares a nivel internacional y nacional.

1.1.4 Estudio referente a buscadores internacionales.

- **Google** es una compañía estadounidense fundada en septiembre de 1998 cuyo producto principal es un motor de búsqueda creado por Larry Page y Sergey Brin. El término suele utilizarse como sinónimo de este buscador, el más usado en el mundo. La característica más destacada de Google como buscador es su facilidad de uso. La página principal se limita a presentar el logotipo de la

empresa en ocasiones, adornado con imágenes que permiten recordar algún evento o realizar un homenaje, un cuadro de búsqueda para introducir los términos a buscar y algunos enlaces hacia otros servicios de la firma (Pérez y otros, 2104).

- **Bing** es un motor de búsqueda de Microsoft. Utilizar Bing es muy parecido a utilizar otros buscadores, en el que se puede ingresar una o más palabras clave en la barra de búsqueda para encontrar páginas web que mencionen las palabras o frases que se ingresaron. En Bing, se puede llevar a cabo una búsqueda de artículos, imágenes, videos, noticias, sitios de compras y más. También se puede ingresar comandos específicos con palabras claves para reducir los resultados de búsqueda. Para una búsqueda avanzada de documentos en el buscador Bing, se puede contar con filtros que limitan los resultados según el sitio, tipos de archivos, título, idioma, región, búsqueda segura, ubicación, mostrar cantidad de resultados en cada página. Bing ayuda a identificar los resultados de búsqueda relevantes a través de funciones como Best Match., tiene Quick Preview, con el cual se despliega una ventana que se expande sobre el título de un resultado de búsqueda y que da una idea mejor de la relevancia del sitio. La opción de *Web Groups* agrupa los resultados de manera intuitiva tanto en el panel de exploración como en los resultados reales (Rautenstrauch, 2010).

1.1.5 Estudio referente a buscadores nacionales.

- Según el sitio oficial redcuba.cu **C.U.B.A** o Contenidos Unificados para Búsqueda Avanzada (C.U.B.A.) es una plataforma que integra los servicios Web disponibles en la red cubana. Está basada en la tecnología del motor de búsqueda Orión, desarrollado por la Universidad de las Ciencias Informáticas (UCI). Surge como respuesta a la necesidad de mostrar a los usuarios la información existente en el dominio cubano y brinda la posibilidad de acceder a sitios de interés cultural, informativos e investigativos. Esta plataforma facilita al usuario la búsqueda de contenidos en varios formatos digitales (páginas web, imágenes y documentos), proporciona referencias concretas de un tema alojado en varias fuentes. Analizando la información proporcionada por el motor de búsqueda que utiliza la plataforma, el usuario puede tener una visión más amplia acerca de un mismo tema al contar con varias fuentes de información y diferentes materiales de consulta.

- **Orión** es un buscador diseñado e implementado con tecnologías libres y hace uso de los componentes fundamentales de un sistema de recuperación de información (interfaz, indexador,

spider). Contiene un gran número de funcionalidades que lo convierten en una herramienta potente para la búsqueda de información alojada en la web cubana, posee servicios desarrollados y otros en proyección de desarrollo que permitirán un avance significativo en la economía del país.

Los buscadores o motores de búsqueda cuentan con distintos programas para cada vez tratar de optimizar sus servicios a los usuarios. Además, interpretan la web de una forma particular la cual ayuda a su propio rastreo e indexación. Observar la web como un grafo dirigido facilita estos procesos para los buscadores y al proceso de recuperación de información en general.

1.2 La web como un grafo

Se puede ver la WWW como un grafo dirigido. Las páginas web son los vértices. Un enlace de una página a otra forma un borde dirigido. Los nodos corresponden a las páginas estáticas, y cuyos arcos son los enlaces entre ellas. Se decide que una página es importante si es señalada por páginas importantes. El estudio de la Web como un grafo permite una nueva e interesante mirada a los algoritmos de *crawling*, búsqueda y recuperación de la información (Durante, 2007).

La Teoría de Grafos sirve como un modelo matemático para estructuras en cualquier campo. Pero una de su más importante área de aplicación es la ciencia de la computación. Estos modelos son aplicados especialmente en lenguajes de computación, circuitos lógicos, redes e interconexión de redes para procesadores paralelos, inteligencia artificial entre otros. Además de ser de gran ayuda para la creación de redes sociales (Franco, 2012).

Todo lo que hay en la red está enlazado entre sí, pudiendo asociarla a un gran grafo y los buscadores también lo ven de esta manera. Por las características propias de las páginas web necesariamente el vínculo entre páginas tiene una orientación definida ya que se parte desde la página x y se dirige hacia la página y, en consecuencia, el grafo es orientado. No solo lo buscadores lo ven de esta forma, sino también las redes sociales. La diferencia es que para un buscador el grafo se compone de documentos y enlaces y para las redes sociales de personas y sus relaciones entre sí (Maciá, 2013).

Aún más, al considerar la totalidad de vínculos existentes en una página hacia cualquier otro conjunto de páginas, fácilmente se puede asignar la ponderación correspondiente a cada vínculo. La forma más común, es realizarlo de la manera equitativa, aunque nada impide que existan casos en los cuales existan otros factores que incidan en las ponderaciones, y por lo tanto éstas no sean iguales. (Barriola, 2016).

Una vez presentada la utilidad de la teoría de grafos para representar la red de páginas web, ahora es preciso indagar en cómo se recorre este conjunto de páginas para encontrar lo que desea.

1.3 *Spiders* o arañas como mecanismo de rastreo de información en la web.

Los buscadores se auxilian de un programa denominado "*spide*r" que se encarga de ir moviéndose por todas las páginas Web, simulando la confección de una gran telaraña e ir recolectando la información de los documentos por donde va pasando y almacenándola. La mayoría de los buscadores internacionales que se usan y conocen se auxilian de *spiders*. Estos requieren muchos recursos para su funcionamiento y no están al alcance de cualquiera.

Los *spiders* recorren las páginas recopilando información sobre los contenidos de las mismas. Principalmente el texto que en ellas aparece. Cuando se busca una información en los buscadores, ellos consultan su base de datos, con la información que han recogido de las páginas, y la presentan clasificada por su importancia. Cada cierto tiempo, los *spiders* revisan la *web*, para actualizar los contenidos de su base de datos, por lo que no es infrecuente, que los resultados de la búsqueda no estén actualizados, de forma que la información o la página no existan (Fernández, 2009).

Según los doctores Heydon y Najork, al escribir "Mercator: *a scalable, extensible Web crawler*", describieron a los *spiders* como programas que descargan documentos de Internet de manera automatizada, los cuales también se nombran con los términos: *web spiders*, *web crawler*, *bots*, *robots*, caminantes o vagabundos. Al mencionar cualquiera de los vocablos anteriores en el presente trabajo, se hará alusión al término *spider* o araña, por su significado en español.

Entre las principales ventajas se encuentran:

1. La rápida actualización de los cambios producidos en Internet.

2. Al indexar prácticamente todos los términos del documento, la búsqueda se puede llevar a cabo por todos estos.

Como principal desventaja, señalar que la precisión en la identificación del documento no es tan exacta como la manual (Tramullas, 2001).

En la actualidad los motores de búsqueda suelen contar con varias instancias de su *spider*, distribuidos estratégicamente. Algunos ejemplos de *spiders web* son: Mnogosearch, Wire y Nutch (Rueda, 2012). Otros ejemplos de *spiders* son: (Román, 2007):

Googlebot-Image: robot indexador de imágenes para Google.

Googlebot-News: robot indexador de noticias para Google News.

Existen varios usos para los rastreadores web, pero esencialmente se utiliza un rastreador web para recopilar o extraer datos de Internet. La mayoría de los motores de búsqueda lo utilizan como un medio de proporcionar datos actualizados y encontrar lo nuevo en Internet. Las compañías analíticas y los investigadores de mercado usan rastreadores web para determinar las tendencias de los clientes y del mercado en una geografía dada (Baiju, 2015).

Entre los *spiders* más utilizados se encuentran:

Heritrix: se caracteriza por tener buena documentación y fácil configuración. La plataforma la cual sustenta se caracteriza por ser estable por más de una década. Se puede considerar escalable pero no dinámicamente escalable. Su rastreo se encuentra limitado según su memoria RAM. No tiene un buen rendimiento en entornos distribuidos (Yadav, Goyal,2015).

Nutch: se caracteriza por ser altamente escalable, obedece regla de robots.txt. La calidad del rastreo puede estar sesgado para buscar páginas “importantes” primero. Es de fácil integración con el motor de búsqueda apache Solr (Yadav, Goyal,2015).

Scrapy: se considera de fácil uso para quien tiene conocimiento de Python, se puede considerar rápido. No es escalable como Nutch y Heritrix (Yadav, Goyal,2015).

Aspseek: puede rastrear unos pocos millones de enlaces. Puede trabajar con múltiples lenguajes, está desarrollado en C++. Realiza búsqueda por frases y palabras. Los resultados de búsqueda pueden limitarse según el período de tiempo dado del sitio (Yadav, Goyal,2015).

JSpider: analiza la estructura de los sitios en cuanto a la existencia de errores y los enlaces entrantes y salientes del mismo**.** Además, se utiliza para descarga completa de sitios web (Yadav, Goyal,2015).

MnoGoSearch: es un *spider* de código abierto para sistema informáticos de tipo Unix, desarrollado en C**.** Se especializa en rastreo de sitios web e intranet, soporta numerosos idiomas. Se basa principalmente en el rastreo de HTTP. Para el almacenamiento de los sitios indexados puede utilizar las bases de datos MySQL o PostgreSQL (Baiju, 2015).

Tabla 1: Spiders de software libre más utilizados.

Spider	Lenguaje	Plataforma	Licencia
Heritrix	Java	Linux	Apache Licencia
Nutch	Java	Multiplataforma	Apache Licencia
Mnogosearch	C	Multiplataforma	GNU/ General Public License
Aspseek	C++	Linux	GPL
JSpider	Java	Windows	GNU/ Library
Scrapy	Paython	Multiplataforma	BSD[9] Licencia

Luego del estudio realizado y teniendo en cuenta las características evidenciadas en la tabla 1 se selecciona el *spider* Nutch. Como se evidencia anteriormente este *spider* es altamente escalable, posee una licencia de código abierto, desarrollado en java, es multiplataforma y posee buen rendimiento en entornos distribuidos.

1.4 Apache Nutch

9 Beberkely Software distribution.

Nutch es un *spider* o *web crawler* libre y de código abierto, desarrollado totalmente en Java por The Apache Software Fundación. Ha sido implementado sobre la base de Apache Lucene, una librería de alto rendimiento para la búsqueda basada en texto y que utiliza una modificación del algoritmo "Vector Space Model" (Modelo de Espacio Vectorial en español), con un enfoque booleano que restringe las estimaciones de los resultados obtenidos (Mahecha, 2009; Apache Nutch, 2013).

1.4.1 Arquitectura y funcionamiento de Nutch

Para llevar a cabo las tareas de rastreo e indización de los documentos, Nutch realiza una serie de procesos, los cuales son ejecutados por algunos de los componentes fundamentales de su arquitectura, en los que destacan: el *spider*, que descubre y recupera páginas web, el indizador, se encarga de diseccionar las páginas y construir palabras clave basadas en sus índices.

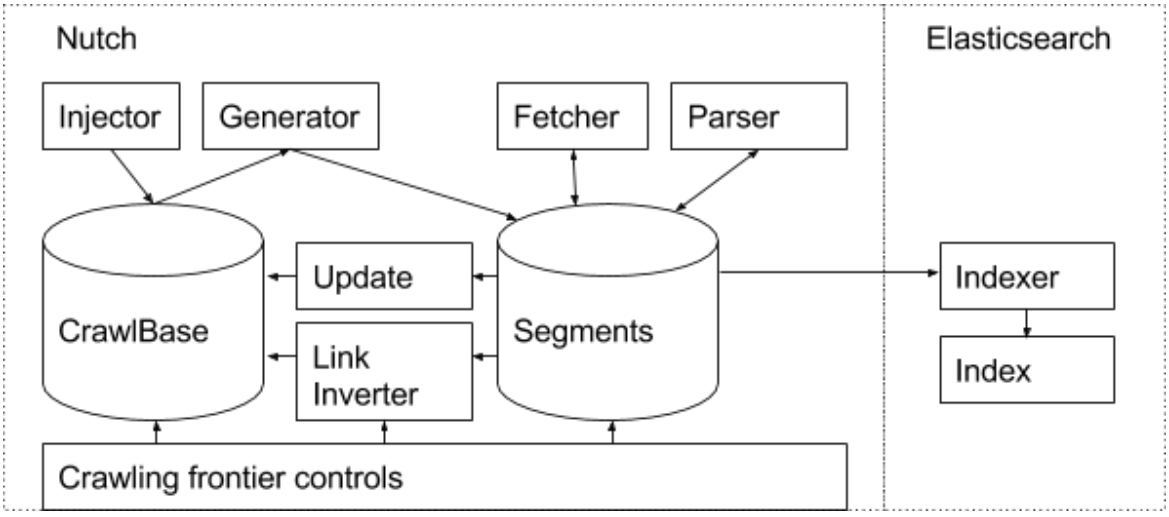


Figura 2: Arquitectura de Nutch. (Kofler, 2015)

Según (Nagel, 2014) los procesos fundamentales que esta arquitectura sostiene son:

- **Inicializar** la *CrawlDb.*
- **Generar** la lista de búsqueda o seleccionar las URL del *CrawDB.*
- **Recuperar** las URL de la lista de búsqueda, trata de establecer una conexión con la página para su descarga y la almacena en el segmento de rastreo.

- **Análisis** Una vez descargada la página, se procede al análisis de su contenido (texto y metadatos). Es aquí donde se descubren los nuevos enlaces en el contenido.
- **Invertir enlaces** Se encarga de asignar los textos de anclaje a los documentos que apuntan a los enlaces.
- **Calcular** la popularidad de enlaces en el gráfico web, actualizar las puntuaciones de CrawlDb.
- **Deduplicar** los documentos.
- **Actualización de la Base de Datos:** al terminar el análisis, Nutch actualiza la base de datos web con el contenido encontrado y añade los nuevos enlaces a la base de datos de URLs.

1.4.2 Cálculo de popularidad de los documentos en Nutch

Los robots utilizan los enlaces en las páginas recuperadas para descubrir nuevas páginas. Todas las páginas de la *web* no tienen la misma importancia. Por ejemplo, la página principal de *Le Louvre* es más importante que la página web de una persona desconocida. La información de importancia de la página es muy valiosa. Se utiliza por los motores de búsqueda para mostrar los resultados en el orden de importancia. (Abiteboul, 2001).

La popularidad de enlaces se refiere tanto a la cantidad y la calidad de los enlaces que apuntan a una página web determinada, como la página de inicio del sitio web. En primer lugar, el sitio tiene que ser accesible y estar indexado por los principales motores de búsqueda. Las arañas que crean índices de los motores de búsqueda siguen enlaces de un sitio web al otro. Si no hay sitios web que enlazan con el que se analiza, entonces es poco probable que sean indexados. La mayoría de los motores de búsqueda utilizan actualmente la popularidad de una página web como uno de los criterios para determinar su clasificación cuando un usuario realiza una búsqueda (Silva, 2015).
Nutch utiliza distintos *plugins* para el cálculo de popularidad de sus enlaces, estos son *Scoring-Depth*, *Scoring-Link* y el *Scoring-Opic*. El valor de popularidad de los enlaces se guarda en el campo *boost* y luego son enviados a Solr. Actualmente Nutch utiliza la propiedad *db.score.injected*, con la cual le otorga valor 1.0 a todos los enlaces, dificultando así la búsqueda para los usuarios ya que no se muestran ordenados.

1.5 Algoritmos para el análisis de enlaces

El análisis de la web como un grafo dirigido, conjuntamente con el análisis de los enlaces, brinda información valiosa que es utilizada con diferentes fines, entre los más destacados se encuentran los algoritmos que utilizan dicha información para determinar la importancia de las páginas, entre otros.
Los algoritmos para el análisis de enlaces 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 popularidad de los mismos.

Algunos de los algoritmos encargados para el análisis de enlaces son el Trust-Rank y el Author-Rank.

- **Trust-Rank** fue creado en 2005 por Google y uno de los principales objetivos era reducir el *SPAM* en los buscadores mediante las páginas de mayor credibilidad en función de los enlaces recibidos. Se puede definir el TrustRank como la credibilidad y como la confianza que tiene un sitio de cara a Google. Para medir el TrustRank, se puede decir que vendría a ser como medir una página a través de enlaces que se encuentren apuntando a la web. Google distingue o más bien categoriza páginas que garantizan "*Good links*" y "*Bad links*". Da por sentado que raramente una página garantizada como "*Good links*" realice enlaces hacia una página de *SPAM* (Consultor-SEO, 2017).
- **Author-Rank** es una de las últimas creaciones de Google para optimizar el posicionamiento de los enlaces. Determina la autoridad de las personas en un área de conocimiento en particular. Esta autoridad la determinan su vez otras personas expertas en dicha área con su número de referencias. Los niveles de autoridad se pueden tener según las temáticas (Consultor-SEO, 2017).
- **Sistema para la gestión de url o enlaces maliciosos:** sistema desarrollado en Cuba por el Ingeniero Daniel Ramón Torres Miyare y el Master en Ciencias Nelson William Gamazo Sánchez en el 2014. Este sistema se encarga de encontrar, descargar y dar seguimiento a las URLs que apuntan a programas malignos. El sistema además permite detectar nuevos repositorios de programas malignos a partir de los enlaces previamente conocidos.

Algunos de los algoritmos más conocidos para el cálculo de popularidad de enlaces son:

- **HITS**: 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 (*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 popularidad 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 el conjunto completo. (Méndez, 2004)

- **PageRank**: Page y Brin propusieron una noción de importancia de página basada en la estructura de enlaces de la web. Esto fue utilizado por Google con un éxito notable. Intuitivamente, una página es importante si hay muchas páginas importantes que apuntan a ella. Es posiblemente el algoritmo de análisis de enlaces más conocido. 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 popularidad 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 (Méndez, 2004).

Las ideas principales de estos dos algoritmos son similares. Mientras que *PageRank* define un orden estático entre las páginas web en función de su prestigio, HITS establece un orden dinámico que depende de la consulta concreta.

- **Scoring-Opic**: este algoritmo representa la importancia de la página en línea. Les otorga dos valores a las páginas visitadas, el primer valor es el mismo para todas y seguidamente se va actualizando a medida que el algoritmo va avanzando por las páginas. Proporciona una manera más eficaz de calcular la importancia de la página que otros algoritmos antiguos, ya que:

  - Trabaja en línea con una gran cantidad de gráficos dinámicos, por lo que refina continuamente su estimación de la importancia de las páginas.

  - Utiliza menos recursos.

  - No requiere almacenar la matriz de enlace.

OPIC converge de forma independiente en cómo se seleccionan las páginas (suponiendo que cada página se selecciona infinitamente). Sin embargo, la velocidad de conversión depende de la elección. Considerando las dos siguientes estrategias de conversión: *Random* y Goloso, la primera escogiendo aleatoriamente y la segunda escogiendo al enlace con mayor valor, siendo esta última quien ofrece mayor velocidad de convergencia (Abiteboul y otros, 2001).

Tabla 2: Comparación entre los algoritmos para cálculo de popularidad de enlaces.

Características	Hits	PageRank	ScoringOpic
Lenguaje	Java	Java	Java
Licencia	Privativa	Privativa	Libre

Robot	No	Googlebot	Nutch
Online	No	Si	Si

Luego de un estudio referente a los algoritmos más conocidos para el cálculo de popularidad de enlaces se puede concluir que el algoritmo creado por Jon Kleinberg solo se emplea sobre un subconjunto de documentos y se ve afectado según el criterio de búsqueda además de que no funciona on-line. El PageRank a pesar de que es un algoritmo muy eficiente su principal desventaja es que no es de código libre. Se selecciona el ScoringOpic ya que además de que trabaja on-line ya se encuentra implementado bajo la arquitectura de Nutch.

1.6 Metodología, tecnologías y herramientas seleccionadas para la realización de la propuesta de solución.

Para desarrollar la propuesta solución, surgida del estudio realizado de algoritmos homólogos para el cálculo de popularidad, se hace necesario el estudio de tecnologías, herramientas y lenguajes disponibles.

1.6.1 Metodología de desarrollo

Una metodología de desarrollo proporciona una base de procesos para llevar a cabo un proyecto informático. Estas pueden ser incluidas entre dos grupos, robustas o ágiles. En dependencia de su clasificación tienen sus propios procedimientos o enfoques para definir la forma en que se realizan las actividades. Las metodologías brindan un soporte a la toma de decisiones.

AUP-UCI

Esta metodología puede ser considerada una variación de Proceso Unificado Racional (RUP), pensada específicamente para la Universidad de las Ciencias Informáticas. El AUP aplica técnicas ágiles incluyendo:

Desarrollo Dirigido por Pruebas.

Modelado ágil.

Gestión de Cambios ágil.

Refactorización de Base de Datos para mejorar la productividad.

Además, cuenta con tres fases consecutivas:

1.Inicio: el objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

2.Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.
3.Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Entre los artefactos que genera esta metodología se encuentran:

1.6.2 Modelo de dominio

Un modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, que contiene conceptos propios de la realidad física. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno del sistema. El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema, por lo cual puede ser tomado como el punto de partida para su diseño (González, 2012).

1.6.3 Especificación de los requisitos de software

Los requisitos del software permiten establecer lo que el sistema debe hacer, sus características fundamentales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo del software. De manera general, estos requisitos expresan las necesidades objetivas que presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio (Sommerville, 2005).

1.6.4 Historias de usuario (HU)

Las HU utilizan los requerimientos de los clientes según el negocio y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación. Las HU son escritas por el equipo de trabajo en conjunto con los clientes en base a lo que se estima que es necesario para el sistema. Están escritas en un formato de oraciones en la terminología del cliente, sin necesidad de sintaxis técnicas. También son utilizadas para poder crear las pruebas de aceptación. Las HU solo proveen suficiente detalle para poder realizar la estimación de cuánto tardará en ser implementada dicha funcionalidad. Una gran diferencia entre las HU y los documentos tradicionales es que se centran en lo que el cliente necesita (Balarezo, 2013).

1.6.5 Entorno de Desarrollo Integrado.

Un entorno de desarrollo integrado o IDE (siglas en inglés de *Integrated Development Environment*) consiste en un conjunto de herramientas de programación puestas a disposición como un programa informático. En otras palabras, puede estar compuesto por un editor de código, un compilador, un intérprete, un sistema de apoyo al control de versiones, un constructor de interfaces gráficas, entre otras herramientas comúnmente utilizadas para el desarrollo de software. *NetBeans* fue Liberado bajo el licenciamiento dual de CDDL y GPL (versión 2), NetBeans es un potente IDE para programadores que proporciona una plataforma ideal para escribir, compilar, depurar y ejecutar programas informáticos (ORACLE, 2014). Aunque inicialmente fue ideado para Java, puede ser empleado para la codificación de aplicaciones en múltiples lenguajes de programación. Este, además de ser gratuito y sin restricciones de uso, posee versiones para los distintos sistemas operativos del mercado, convirtiéndolo en una alternativa con grandes ventajas para los desarrolladores. La estructura modular de NetBeans le proporciona estabilidad y grandes posibilidades de ser extendido gradualmente por desarrollos comunitarios, permitiendo agregar continuamente nuevas funcionalidades.

1.6.6 Lenguaje de programación.

Java es un lenguaje de programación orientado a objetos ideado por *Sun Microsystem*. Es un lenguaje de propósito general por lo que con él se podría crear cualquier tipo de aplicación. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de

punteros o memoria (Belmonte Fernández, 2005). Se escogió este lenguaje ya que Nutch está desarrollado en Java.

Como parte de las tendencias de desarrollo actuales, Java presenta una serie de beneficios:

- Es independiente a la plataforma de desarrollo.
- Existen dentro de su librería, clases gráficas que permiten crear objetos visuales

comunes altamente configurables y con una arquitectura independiente de la plataforma.

- Permite utilizar la flexibilidad de la programación orientada a objetos para el diseño de sus aplicaciones.

Entre sus características principales se encuentran:

- Simple: ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de estos. Elimina muchas de las características de otros lenguajes como C++, para mantener reducida la especificación del lenguaje.
- Orientado a objetos: implementa la tecnología de C++ y soporta las tres características del paradigma orientado a objetos. Encapsulamiento: implementa información oculta. Polimorfismo: El mismo mensaje se envía a diferentes objetos, resultando en comportamientos que dependen de la naturaleza del objeto que recibió el mensaje. Herencia: puede definir nuevas clases y comportamientos (métodos) basados en clases existentes.
- Robusto: realiza verificaciones en busca de problemas, tanto en tiempo de compilación, como de ejecución. La comprobación de tipos ayuda a detectar errores. Obliga a la declaración explícita de los métodos.
- Seguro: la seguridad tiene dos facetas: se eliminan características como los apuntadores y el casting implícito para prevenir el acceso ilegal a la memoria El código Java pasa por muchas verificaciones antes de ser ejecutado en una máquina mediante el classloade.
- Distribuido: presenta extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Por si sólo no es distribuido, pero proporciona herramientas para que nuestros programas puedan serlo.

1.6.7 Visual Paradigm

Visual Paradigm es una herramienta CASE multiplataforma, que soporta el ciclo completo de desarrollo de software: análisis, diseño, implementación y pruebas. Facilita la construcción de aplicaciones informáticas con un menor coste que destacan por su alta calidad y contribuye a mejorar la experiencia de usuario mediante el diseño de un gran número de artefactos de ingeniería de software. Permite la generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería directa e inversa, la gestión de requisitos de software y la modelación de procesos del negocio (Visual Paradigm, 2014).

1.6.8 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es el lenguaje estándar especificado por el Object Management Group (OMG) para visualizar, especificar, construir y documentar los artefactos de un sistema, incluyendo su estructura y diseño. Utiliza un conjunto de símbolos y notaciones para representar gráficamente los diversos componentes que forman parte de la arquitectura de software. Permite el modelado de procesos de negocio y el modelado de requisitos apoyándose en el análisis orientado a objetos (Object Management Group, 2013).

1.6.9 Modelado de proceso de negocio.

BPMN (Notación de Modelado de Procesos de Negocio) es utilizada como un estándar para la representación de procesos de negocios, independientemente de la metodología empleada para automatizar dichos procesos. Esta notación ha sido diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes en las actividades. Proporciona un lenguaje común para que las partes involucradas en el análisis de negocios (cliente y desarrollador) puedan intercambiar sobre un mismo modelo de forma clara, precisa y eficiente (Baisley, 2006).

1.6.10 Servidor de indexación Solr.

Es una plataforma de búsquedas basada en Apache Lucene, que funciona como un "servidor de búsquedas". Sus principales características incluyen búsquedas de texto completo, resaltado de resultados y manejo de documentos (como Word y PDF). Solr es escalable, permitiendo realizar búsquedas distribuidas y replicación de índices (Seta, 2010).

Según la página principal de Solr, las características más destacables de este servidor de búsquedas son:

- Capacidades avanzadas de búsqueda de texto completo.
- Optimizado para un tráfico Web elevado.
- Interfaces abiertas basadas en estándares – XML, JSON y HTTP.
- Completas interfaces Web de administración.
- Estadísticas del servidor expuestas mediante JMX para su monitorización.
- Escalable linealmente, replicación automática del índice, recuperación automática.
- Indexación cuasi-inmediata.
- Flexible y adaptable a través de una configuración en formato XML.
- Arquitectura extensible mediante *plugins*.

Solr está escrito en Java y se puede ejecutar como un servidor de búsqueda de texto completo independiente dentro de un contenedor de *servlets* como Tomcat. Solr Lucene utiliza la biblioteca Java de búsqueda en su base para la indexación de texto completo y de búsqueda, y tiene como REST HTTP / XML y JSON Apis que hacen que sea fácil de utilizar desde prácticamente cualquier lenguaje de programación. Potente configuración externa de Solr permite que sea adaptado a casi cualquier tipo de aplicación Java sin codificación, simplemente hay que utilizarlo con peticiones GET para realizar las búsquedas en el índice, y POST para agregar documentos (Apache, 2015).

1.6.11 Mecanismo para la recolección de Información.

Como spider se utiliza Nutch debido a las múltiples ventajas que ofrece a lo largo de todo el proceso de rastreo y almacenamiento de la información. Presenta web crawler libre y de código abierto desarrollado en Java bajo la licencia de Apache. Este proporciona interfaces extensibles para implementaciones personalizadas. Inicialmente fue implementado sobre la base de Apache Lucene, aunque ya la versión actual es independiente de Lucene, una librería de alto rendimiento para la búsqueda basada en texto y que utiliza una modificación del algoritmo “Vector Space Model” (Modelo de Espacio Vectorial en español), con un enfoque booleano que restringe las estimaciones de los resultados obtenidos. La arquitectura de Nutch es flexible permitiendo realizarle mejoras por parte de los usuarios a través de plugins. Este es independiente del servidor de indización lo que permite la integración con Solr (Apache Nutch, 2013).

1.7 Conclusiones parciales.

En este capítulo se trataron los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

- ✓ El estudio de los diferentes sistemas de los conceptos asociados al cálculo de la popularidad de enlaces permitió lograr un mejor entendimiento de la investigación que se realiza.
- ✓ La investigación realizada a las tecnologías informáticas definió la base tecnológica que se utilizará en el desarrollo del Plugin, seleccionando a AUP-UCI como metodología para guiar el desarrollo, UML como lenguaje de modelado y Visual Paradigm 8.0 como herramienta CASE para el modelado del sistema. Además, se utilizará el lenguaje de programación Java, para el desarrollo del componente en entorno de desarrollo NetBeans.
- ✓ El estudio de los diferentes sistemas homólogos permitió identificar diferencias y similitudes entre los mismos, de manera que se escogió el algoritmo Scoring-Opic para realizar el cálculo de popularidad de los enlaces en Nutch, identificando como mayo r ventaja de este que ya se encuentra implementado para Nutch.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DEL PLUGIN PARA EL CÁLCULO DE LA POPULARIDAD DE ENLACES EN NUTCH

INTRODUCCIÓN

En este capítulo se abordarán los aspectos fundamentales relacionados con el diseño y el análisis del Plugin a desarrollar. Entre los elementos a destacar se encuentran el diagrama del modelo del dominio, mediante el cual se representan los objetos reales que intervienen en el proceso de cálculo de popularidad de enlaces en Nutch. Como vía para definir las futuras funcionalidades del Plugin. Se generaron los artefactos relacionados a la especificación de los requerimientos funcionales y no funcionales que deberá poseer el software; así como la especificación de los casos de uso del sistema. Como parte del diseño de la aplicación se definieron los estilos y patrones de arquitectura y diseño que se emplearán para lograr buenas prácticas de diseño y programación. A lo largo del capítulo se mostrarán los principales artefactos de ingeniería de software correspondientes a las funcionalidades.

2.1 Modelo de dominio

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos de software con responsabilidades, sino que modela clases conceptuales significativas en un determinado problema (Larman, 2004).

Para lograr un mejor entendimiento de la presente investigación se hace necesario describir el procedimiento de cálculo de popularidad de enlaces en Nutch, entidades y sus relaciones, agrupándose en un modelo del dominio con el fin de contribuir a la comprensión del contexto actual del problema.

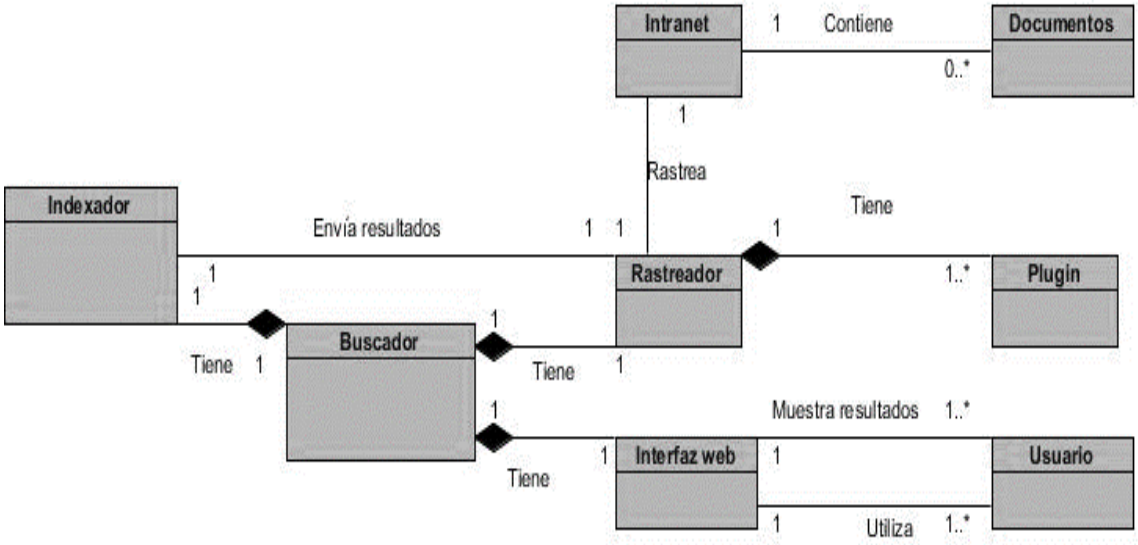


Figura 3: Diagrama de clases de dominio.

2.2 Descripción de Clases del Modelo de Dominio

La modelación del dominio constituye la herramienta fundamental para garantizar la comprensión y descripción de las clases o conceptos y sus relaciones más importantes dentro del contexto del problema. A continuación, se presenta la descripción de los conceptos identificados.

Tabla 3: Descripción de las clases del modelo del dominio.

Usuario	Persona que realiza las búsquedas
Buscador	Constituye una herramienta de recuperación de información en la Web.
Rastreador	Mecanismo que se encarga de recopilar los documentos en la Web.
Interfaz web	Constituye la vista mediante la cual se le muestran los resultados al usuario.
Servidor de índice	Mecanismo que se encarga de indexar los documentos.

Plugins	Componentes externos que se le agregan a Nutch para extender su funcionamiento.
Documento	Recursos publicados en la Web (páginas web, imágenes, videos, documentos ofimáticos)

2.3 Especificación de los requisitos del software

Los requisitos del software constituyen las descripciones de los servicios y funcionalidades que brinda una aplicación informática, teniendo en cuenta además las restricciones operativas a las cuales están sujetas. De manera general,

estos requisitos expresan las necesidades objetivas que presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio (Somerville, 2005).

2.3.1 Requisitos Funcionales

- RF1: Identificar número de enlaces salientes.
- RF2: Identificar número de enlaces entrantes.
- RF3: Evaluar calidad de la url.
- RF4: Calcular antigüedad del enlace.
- RF6: Determinar texto ancla del enlace.
- RF7: Diagnosticar penalización.

2.3.2 Requisitos No funcionales.

Los requisitos no funcionales, son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (Sommerville, 2005). A continuación, se presentan los requisitos no funcionales del Plugin para cálculo de la popularidad de los enlaces en Nutch.

Requerimientos de software.

- RNF1: Se requiere una distribución GNU/Linux como Sistema Operativo.
- RNF2: Se requiere la instalación de la Máquina Virtual de Java (JVM, por sus siglas en inglés) para el correcto funcionamiento del rastreador.

Requerimientos de diseño e implementación.

- RFN3: Se requiere el uso del paradigma de Programación Orientada a Objetos (POO).
- RNF4: Se requiere el uso de NetBeans como entorno de desarrollo.
- RNF5: Como lenguaje de programación para el Plugin se deberá utilizar Java.

Requerimientos de hardware.

- RNF6: Para los servidores de rastreo e indexación: 4 GB RAM, CPU de 4 núcleos y al menos 150 GB de Disco Duro.

Requerimientos de soporte.

- El soporte del sistema se debe gestionar mediante el Centro de Soporte de la UCI.

2.4 Propuesta solución.

En el presente trabajo se propone el desarrollo de un Plugin que permita calcular la popularidad de los enlaces en Nutch. Para el cálculo de la popularidad se desarrollará una variante del Scoring-Opic. El algoritmo Scoring-Opic, teniendo un grafo G, el cual representa los enlaces en la web y su relación entre

sí, en la matriz L, donde si hay un enlace del nodo i al nodo j, entonces L(i,j)=1. La importancia o popularidad de cada nodo está dada por la fórmula: Imp(j)=Σ(L'[i,j]*Imp(i))

El cumplimiento de cada requisito funcional aportará un valor numérico, el cual intervendrá en una suma total de la popularidad del enlace. A continuación, se describe cómo influye cada requisito funcional en la popularidad del enlace.

- Repartir el score de nodo(i), a cada uno de los **n** enlaces a los cuales redirecciona: Score(i) /n.
- Aumentar el score del nodo(i) según el score que le aporten los enlaces que le redireccionan (r): Score(i)=Score(i)+Σ Score(r).
- Restar valor al score del nodo(i), según la longitud (l) de su url y la cantidad de caracteres especiales (ce) que presente la misma: Score(i)= Score(i)+-(0.01) +-(0.01)
- Amentar score del nodo(i) según un valor(v) dependiente de la antigüedad(x) del nodo (i): Score(i)=Score(i)+v.
- Descontar del score del nodo(i) según el texto ancla de sus enlaces (tx): Score(i)= Score(i)- (0.1*tx)
- Descontar o aumentar el score del nodo(i), según la penalización: Score(i)= Score(i)-(0.1*Cantidad de enlaces repetidos).

Una vez calculada la popularidad de cada enlace rastreado por Nutch, este enviará el valor al servidor de indexación Solr, el cual lo actualizará en el atributo boot. A continuación, se muestra el seudocódigo del ScoringOpic agregando en la etapa específica del algoritmo los aportes que proporcionan los requisitos funcionales.

A continuación, se muestra el pseudocódigo del algoritmo Scoring-Opic.

for each i let C[i] := 1/n ;

for each i let H[i] := 0 ;

let G:=0 ;

do forever

begin

choose some node i ;

% Por cada nodo seleccionado se calcula su popularidad según los requisitos funcionales anteriormente descritos.

H[i] += C[i];

%% single disk access per page

for each child j of i, do C[j] += C[i]/out[i];

%% Distribution of cash

%% depends on L

G += C[i];

C[i] := 0 ;

end

2.5 Historia de usuarios (HU).

Las HU sirven para registrar los requerimientos de los clientes según el negocio y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación. Las HU son escritas por el equipo de trabajo en conjunto con los clientes en base a lo que se estima que es necesario para el sistema

Tabla #4: Historias de usuario # 1

Historias de usuario	
Número: HU_1	Nombre: Identificar número de enlaces salientes.

Prioridad en negocio: media	
Descripción: permite identificar la cantidad de enlaces que salen de la página que se analiza, interviene en el proceso de repartir score a los enlaces salientes de la página.	

. Tabla #5: Historias de usuario # 2

Historias de usuario	
Número: HU_2	Nombre: Identificar número de enlaces entrantes.
Prioridad en negocio: alta	
Descripción: permite identificar la cantidad de enlaces que entran a la página que se analiza, cada enlace aporta score a la página a la cual direcciona.	

Tabla #6: Historias de usuario # 3

Historias de usuario	
Número: HU_3	Nombre: Evaluar calidad de la url.
Prioridad en negocio: media	
Descripción: permite realizar una evaluación de la url del documento que se analiza, en cuanto a longitud y contenido de caracteres especiales.	

Tabla #7: Historias de usuario # 4

Historias de usuario	
Número: HU_4	Nombre: Calcular antigüedad del enlace.

Prioridad en negocio: alta
Descripción: permite analizar la edad del documento desde que fue identificado por primera vez por Nutch.

Tabla #8: Historias de usuario # 5.

Historias de usuario	
Número: HU_5	Nombre: Determinar texto ancla del enlace.
Prioridad en negocio: baja	
Descripción: permite determinar si la palabra que constituye el texto ancla está lo más relacionada posible con el enlace.	

Tabla #9: Historias de usuario # 6.

Historias de usuario	
Número: HU_6	Diagnosticar penalización
Prioridad en negocio: media	
Descripción: permite determinar si una página contiene enlaces duplicados.	

2.6 Descripción de los estilos arquitectónicos y los patrones de diseño.

En la arquitectura de software, los estilos definen la forma de articulación u organización arquitectónica de los subsistemas informáticos que conforman una aplicación en general. Estos, indican los tipos de componentes y conectores involucrados, las restricciones de interconexión entre ellos, y el empleo de buenas prácticas para el diseño de software. Las buenas prácticas para el diseño de arquitecturas de sistemas informáticos, han sido descritas en forma de patrones arquitectónicos, los cuales han condicionado históricamente la toma de decisiones estructurales coherentes. A partir de la identificación

de los estilos, se recomienda realizar un análisis crítico para su reutilización en las situaciones semejantes que se presenten en el futuro (Reynoso, 2004).

2.6.1 Estilos arquitectónicos.

Se determina como estilo arquitectónico, (ver figura4) y así poder lograr una mejor integración entre sus componentes.

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Peláez, 2009).

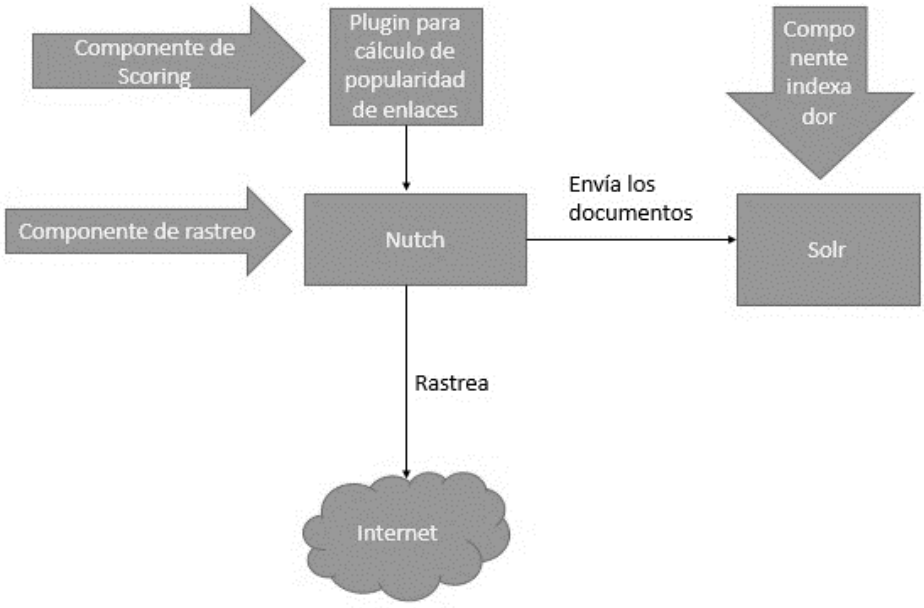


Figura #4. Estilo arquitectónico propuesto.

El estilo de arquitectura basado en componentes tiene las siguientes características (Peláez, 2009):

- Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.
- Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. Los principios fundamentales cuando se diseña un componente es que estos deben ser (Peláez, 2009):

- ✓ Reusable. Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- ✓ Extensible. Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- ✓ Encapsulado. Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado.
- ✓ Independiente. Los Componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto, los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

Los siguientes son los principales beneficios del estilo de arquitectura basado en componentes:

• **Facilidad de Instalación**. Cuando una nueva versión esté disponible, usted podrá reemplazar la versión existente sin impacto en otros componentes o el sistema como un todo.

• **Costos reducidos**. El uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.

• **Facilidad de desarrollo**. Los componentes implementan una interface bien definida para proveer la funcionalidad definida permitiendo el desarrollo sin impactar otras partes del sistema.

• **Reusable**. El uso de componentes reutilizables significa que ellos pueden ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.

2.6.2 Patrones de diseño.

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos, y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí (Larman, 2003).

Los Patrones GRASP son patrones de diseños que describen cada uno de los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Bajo Acoplamiento

El patrón bajo acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto (Larman, 2004). En otras palabras, que una clase no dependa de muchas otras, lo cual potencia la reutilización y disminuye la dependencia entre estas.

La Clase OPICScoringFilter es responsable de la realización del cálculo de la popularidad para cada uno de los enlaces, esta clase su vez solo depende en casos muy necesarios de otras clases. Un ejemplo de este patrón se puede observar en la funcionalidad distributeScoreToOutlinks de dicha clase.

Se realiza un conteo de los enlaces salientes y se reparte el valor del Scoring entre todos estos.

```
// internal and external score factor
float internalScore = score * internalScoreFactor;
float externalScore = score * externalScoreFactor;
for (Entry<Text, CrawlDatum> target : targets) {
    try {
        String toHost = new URL(target.getKey().toString()).getHost();
        String fromHost = new URL(fromUrl.toString()).getHost();
        if (toHost.equalsIgnoreCase(fromHost)) {
            target.getValue().setScore(internalScore);
        } else {
            target.getValue().setScore(externalScore);
        }
    } catch (MalformedURLException e) {
        LOG.error("Error: ", e);
        target.getValue().setScore(externalScore);
    }
}
```

Figura #5. Ejemplo de utilización del patrón Bajo Acoplamiento.

Alta cohesión

En el diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Siguiendo este principio se han diseñado las clases de este sistema y sus relaciones.

A continuación, se muestra un ejemplo de la utilización en la clase Age, para el cálculo del aumento según el intervalo de tiempo.

```
    public double AumentarPorEdad() {
        double aumento = 0;

//de 0 a 7 dias
        if ((intervalo > 0) && (intervalo <= 10080)) {
            aumento = 0.1;
        }
//de 7 dias a 15
        if ((intervalo > 10080) && (intervalo <= 21600)) {
            aumento = 0.2;
        }
        // de 15 dias a 30
        if ((intervalo > 21600) && (intervalo <= 43200)) {
            aumento = 0.3;
        }
        // de 30 dias a 90
        if ((intervalo > 43200) && (intervalo <= 129600)) {
            aumento = 0.4;
        }
        //mayor que 90 dias
        if (intervalo > 129600) {
            aumento = 0.5;
        }
        return aumento;
    }
```

Figura #6. Ejemplo de utilización del patrón Alta Cohesión.

Creador

La instanciación de una clase es una de las actividades fundamentales en un sistema orientado a objetos. Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, con lo que se logra menos dependencia y mayores oportunidades de reutilización de código (Larman, 2004).

```
public Age(Date date) {
    try {
        this.init = date;
        this.current = new Date();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public Age(String date, String format) {
    try {
        this.init = new SimpleDateFormat(format).parse(date);
        this.current = new Date();
    } catch (ParseException e) {
        e.printStackTrace();
    }
```

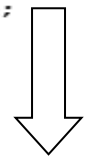
Figura #7: Ejemplo de utilización del patrón Creador.

Cadena de responsabilidades

Se debe de usar cuando más de un objeto puede manejar una solicitud, el manejador no se conoce a priori y debe ser averiguado automáticamente. Además, se desea enviar una solicitud a uno de varios objetos sin especificar explícitamente el receptor y el conjunto de objetos que pueden manejar una solicitud debe especificarse dinámicamente. La cadena de responsabilidad tiene los siguientes beneficios y responsabilidades (Design Patterns, 1994):

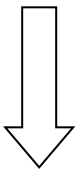
- Acoplamiento reducido. El patrón libera un objeto de saber qué otro objeto maneja una petición.
- Mayor flexibilidad en la asignación de responsabilidades a los objetos.
- El recibo no está garantizado. Dado que una solicitud no tiene un receptor explícito, no hay garantía de que se manejará, la solicitud puede caer al final de la cadena sin ser manipulada.

```
//aumentamos el valor del score
@Override
public float score() {
    double aumento=AumentarPorEdad();
    this.score+=aumento;
    return score;
}
```

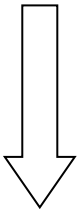


```
public float calculateScore(){
 float score = 0;
      float numForRest = 0;
    for (Map.Entry<Text, CrawlDatum> target : targets) {
        String urlinterna = target.getKey().toString();
        if (anchor.contains(target.getKey().toString())) {
            score -= 0.01;
        }
    }
    return score;
}
```

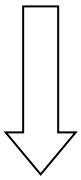
```
    @Override
    public float score() {
        float result = DEFAULT_VALUE_INIT;
        if (links != null)
            for (CrawlDatum crawlDatum : links)
                result += crawlDatum.getScore();
        return result;
    }
}
```



```
    @Override
    public float score() {
        return anchor.score() + url.score();
    }
}
```



```
    @Override
    public float score() {
        float score = 0.0f;
        for (Link  link :links) score += link.score();
        return score;
    }
}
```



```
@Override
public float score() {
    float score = 0.0f;
    if(url.length() <= charactersSize)
        score += 0.01f;
    else score -= 0.01f;
   // System.out.println("*****************************
    //System.out.println("El score segun la longitud es :
    List<String> specials = getSpecialCharactersInUrl();
    if(specials.size() <= specialCharactersSize)
        score += 0.01f;
    else score -= 0.01f;
   // System.out.println("*****************************
   //System.out.println("El score con caracteres especial
    return score;
}
```

Figura #8. Ejemplo de utilización del patrón Cadena de Responsabilidades.

Singleton

Este patrón se utiliza cuando debe haber exactamente una instancia de una clase, y debe ser accesible a los clientes desde un punto de acceso bien conocido. Además, cuando la única instancia debe ser extensible por subclase, y los clientes deben ser capaces de utilizar una instancia extendida sin modificar su código (Design Patterns, 1994).

```
public void updateDbScore(Text url, CrawlDatum old, CrawlDatum datum, List<CrawlDat
    DataInput input = new DataInput(inlinked);
    if (old == null) old = datum;
    datum.setScore(old.getScore() + input.score());
}
```

Figura #9: Ejemplo de utilización del patrón Singleton.

2.7 Modelo de Diseño

El modelo de diseño se utiliza como medio de abstracción del modelo de implementación y el código fuente del *software*. Su objetivo fundamental es transmitir, a través de la representación mediante diagramas, una comprensión en profundidad de los aspectos relacionados con los requerimientos no funcionales y restricciones concernientes a los lenguajes de programación (Larman, 2004).

2.7.1 Diagrama de clases.

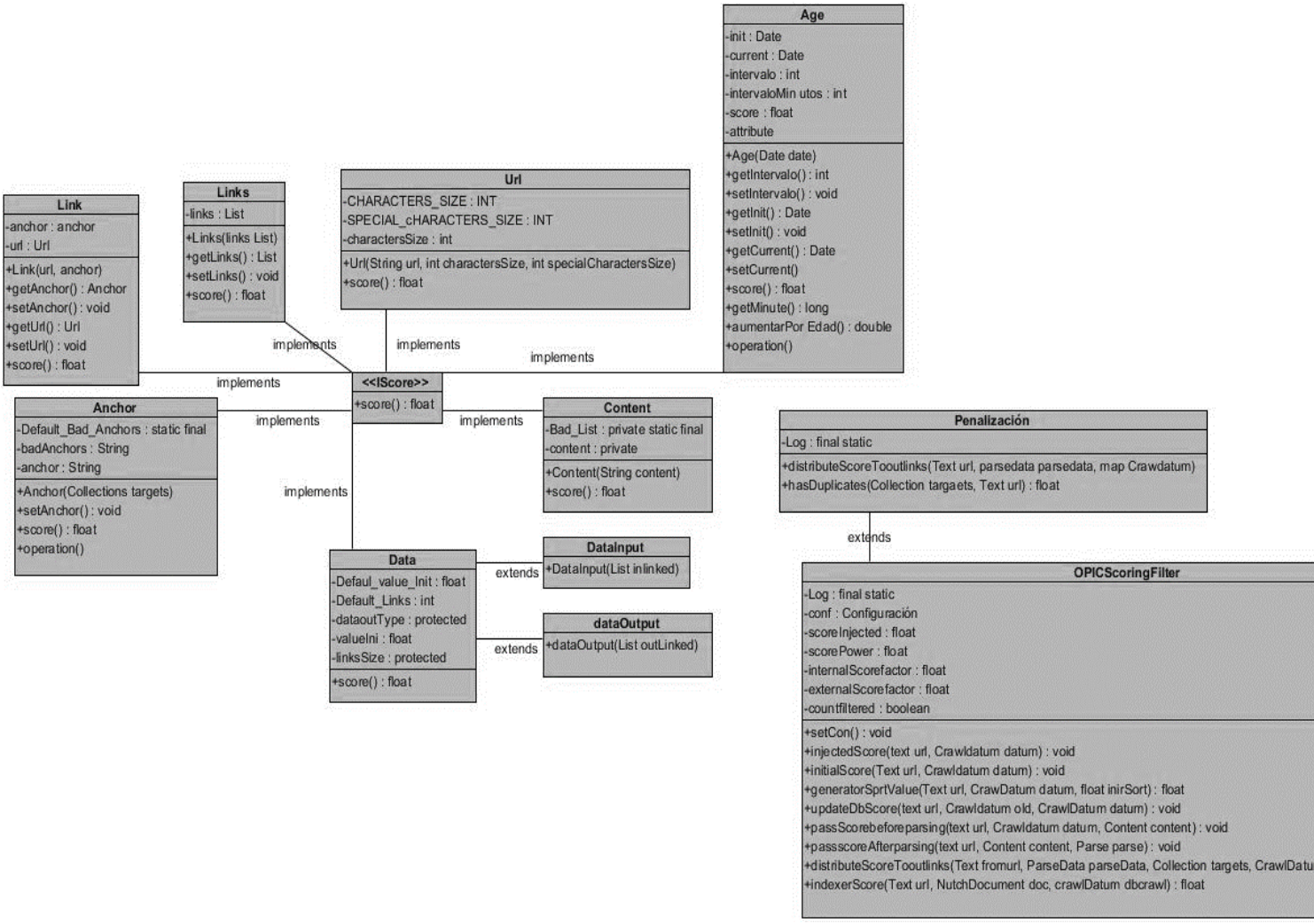


Figura 10: diagrama de clases.

2.7.2 Diagrama de despliegue.

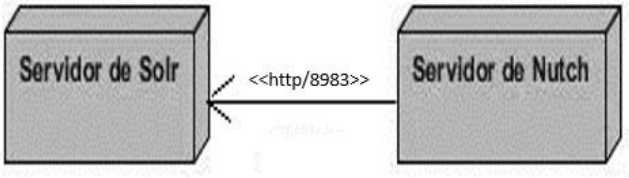


Figura 11: Diagrama de despliegue.

El diagrama de despliegue se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el *hardware* y el *software* que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno (SparxSystems, 2014). En la figura anterior puede visualizarse el diagrama de despliegue definido para la solución propuesta. En la cual se evidencia la utilización de dos servidores, uno para el rastreador y otro para indexación.

2.8 Conclusiones del capítulo.

Con la especificación de los requisitos funcionales y no funcionales del Plugin, se logró un mejor entendimiento, de los resultados que se pretenden obtener y sirvieron de guía para la implementación del Plugin. La representación y descripción de los artefactos generados garantizaron una mejor comprensión de los flujos de trabajos presentes en el proceso de cálculo de popularidad de enlaces. La definición de la arquitectura y los patrones de diseño a utilizar, permitieron establecer las bases para fomentar la reutilización y las buenas prácticas de programación durante la fase de implementación, así como disminuir el impacto de los cambios futuros en el código fuente. La elaboración del diagrama de despliegue permitió identificar la disposición física de los artefactos del producto informático a desarrollarse.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DEL PLUGIN PARA EL CÁLCULO DE LA POPULARIDAD DE LOS ENLACES EN NUTCH.

3.1 Introducción

La etapa de implementación es fundamental durante el proceso de desarrollo de un software, es en este momento donde se define y organiza el código de la propuesta de solución. Es durante esta etapa que se materializan, en forma de código, de todos los artefactos de implementación, descripciones y arquitectura propuestos en la etapa de análisis y diseño; conformando así el producto final requerido por el cliente. Además, en este capítulo se realizará la etapa de prueba, es donde se garantiza que el software cumpla con los estándares requeridos, con los requerimientos del cliente.

3.2 Estándares de codificación utilizados.

La legibilidad del código fuente evidencia la comprensión del programador del sistema de software. Una buena práctica que potencia la legibilidad del código es la utilización de estándares de codificación. Un estándar de codificación comprende todos los aspectos de la generación de código, la legibilidad del código, reflejando un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.
Estas normas son muy útiles por muchas razones, entre las que destacan:

Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.

Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.

Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

Para la implementación del Plugin para cálculo de popularidad de los enlaces en Nutch, se utilizó Java como lenguaje de programación. A continuación, se mencionan los estándares de codificación de dicho lenguaje:

Organización de ficheros:

Las clases en Java se agrupan en paquetes. Estos paquetes se deben organizar de manera jerárquica.

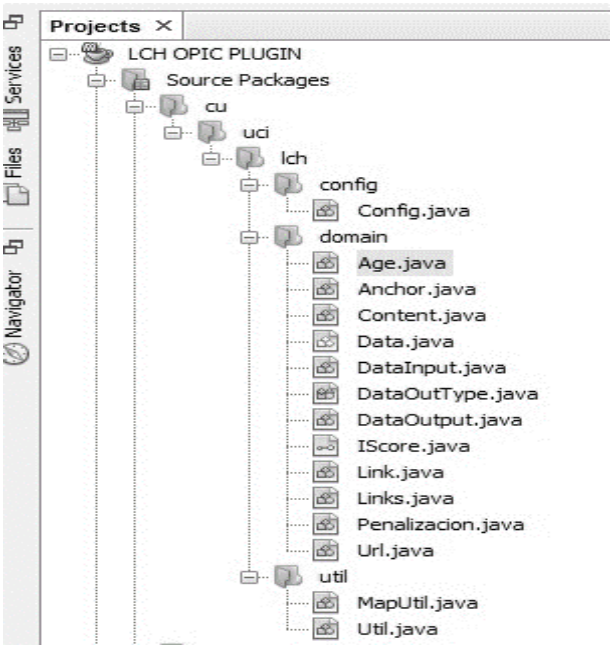


Figura #12: Ejemplo de estándar de codificación.

Cada fichero fuente Java debe contener una única clase o interfaz pública.

```
import java.util.Date;

public class Age implements IScore {

    private static final int MILI_SECONDS_DAY = 86400000;

    private Date init;
    private Date current;

    /* Format Date "YYYY-MM-DD" */
    public Age(Date date) {
        try {
            this.init = date;
            this.current = new Date();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Figura #13: Ejemplo de estándar de codificación.

Todo fichero fuente debe comenzar con un comentario que incluya el nombre de la clase.

```
// Clase Edad,calcula edad de dominio

package cu.uci.lch.domain;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Age implements IScore {
```

Figura 14: Ejemplo de Estándar de codificación.

Tras la declaración del paquete se incluirán las sentencias de importación de los paquetes necesarios.

```
package cu.uci.lch.domain;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
```

Figura 15: Ejemplo de Estándar de codificación.

Declaración de clases e interfaces.

Tabla#12: Declaración de tablas e interfaces (Java Foundations, 2010)

Elementos de declaración de una clase / interfaz	Descripción
Comentario de documentación de la clase/interfaz /** ... */	Permite describir la clase/interfaz desarrollada. Necesario para generar la documentación de la api mediante javadoc.
Comentario de implementación de la clase/interfaz, si es necesario /* ... */	Este comentario incluye cualquier información que no pueda incluirse en el comentario de documentación de la clase/interfaz.
Variables de clase (estáticas)	En primer lugar las variables de clase públicas (public), después las protegidas (protected), posteriormente las de nivel de paquete (sin modificador), y por último las privadas (private).
Variables de instancia	Primero las públicas (public), después las protegidas (protected), luego las de nivel de paquete (sin modificador), y finalmente las privadas (private).
Métodos	Deben agruparse por funcionalidad en lugar de agruparse por ámbito o accesibilidad. Por ejemplo, un método privado puede estar situado entre dos métodos públicos. El objetivo es desarrollar código fácil de leer y comprender.

Variables

Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas.

```
private static final String[] DEFAULT_BAD_ANCHORS =

private String[] BadAnchors =  DEFAULT_BAD_ANCHORS;
private Map<String,String> blackListAnchors = Util.

private String anchor;
```

Figura 16: Ejemplo de Estándar de codificación.

Se deben evitar las asignaciones de un mismo valor sobre múltiples variables en una misma sentencia, ya que dichas sentencias suelen ser difíciles de leer.

3.5 Validación del Plugin.

Las pruebas son un elemento crítico para la garantía de la calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. El objetivo fundamental de las pruebas es descubrir diferentes clases de errores con la menor cantidad de tiempo y de esfuerzo. Aunque las pruebas no pueden asegurar la ausencia de defectos; sí pueden demostrar que existen defectos en el software. (Pressman, 2002).

3.6 Nivel de pruebas.

Se selecciona el de sistema para validar el plugin de cálculo de la popularidad de enlaces en Nutch. El propósito de este nivel es probar que el sistema funciona correctamente como un todo. Es la actividad dirigida a verificar el programa final, después que todos los componentes del software y hardware han sido integrados, asegurando que el sistema realiza las debidas funciones. (Pressman, 2002).

A partir del nivel de prueba escogido se determinó aplicar los siguientes tipos de pruebas.

3.6.1 Pruebas de aceptación Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, ya que tienen como objetivo obtener la aceptación final del cliente antes de la entrega del producto para su utilización. Son realizadas por el usuario final permitiendo la valoración del producto, donde el cliente confirma que las funcionalidades exigidas y descritas en la HU funcionan

correctamente. Intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos, errores de rendimiento y errores de inicialización y terminación (Fontela, 2003).

Para una mejor validación del plugin se insertaron códigos para una visualización del algoritmo. Mientras se fuera ejecutando el plugin en Nutch, este mostrara en terminal, como el score o popularidad de los enlaces se va modificando. Para una mejor comprensión se ofrecen las siguientes imágenes.

Figura 17: Pruebas de validación.

En la anterior figura se evidencia como cambia el *score* del enlace php.uci.cu, según la fecha de identificación por el *spider*. Además, se muestra como el enlace pasa por las distintas funcionalidades del algoritmo, como son: identificar si tiene enlaces duplicados o si presenta textos anclas que representen una disminución del score.

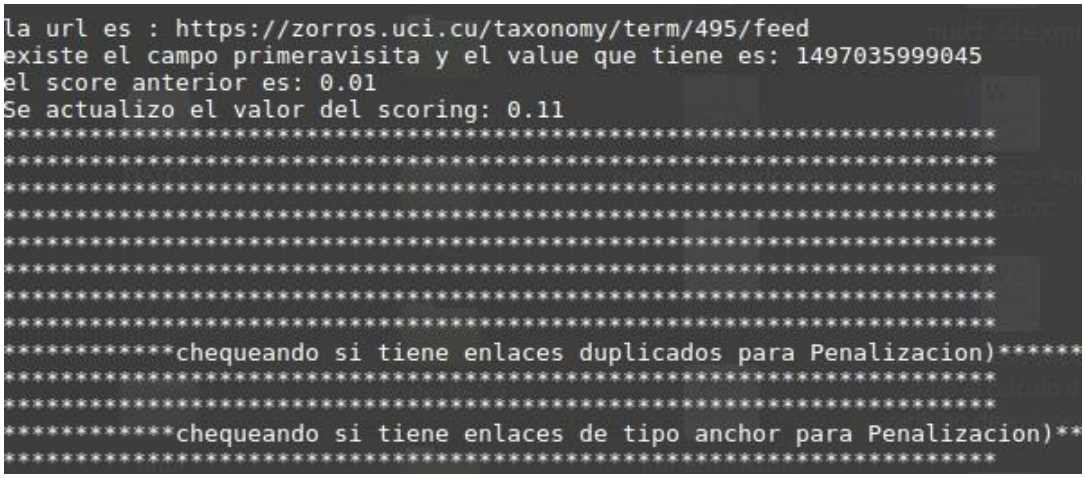


Figura 18: Pruebas de aceptación.

En la figura anterior se evidencia la identificación del sitio zorros.uci.cu, como varía su score luego de identificada su fecha de identificación por el spider, además como va pasando las distintas fases del algoritmo donde se determina si posee enlaces duplicados o textos anclas para posteriormente diagnosticar penalización.

3.6.2 Pruebas Unitarias

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código como métodos o clases. Generalmente son realizadas por los mismos programadores puesto que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo. Por último, es importante que las funcionalidades de cada componente unitario sean cubiertas, por al menos, dos casos de prueba, los cuales deben centrarse en probar al menos una funcionalidad positiva y una negativa (Myers, 2011). Facilitan que el programador cambie el código para mejorar su estructura (refactorización), puesto que permiten hacer pruebas sobre los cambios y asegurarse de que no han introducido errores.

Resultado de las pruebas Unitarias

A continuación, se presenta la figura 17 donde se puede apreciar con una mayor precisión el comportamiento de los errores detectados en cada una de las iteraciones.

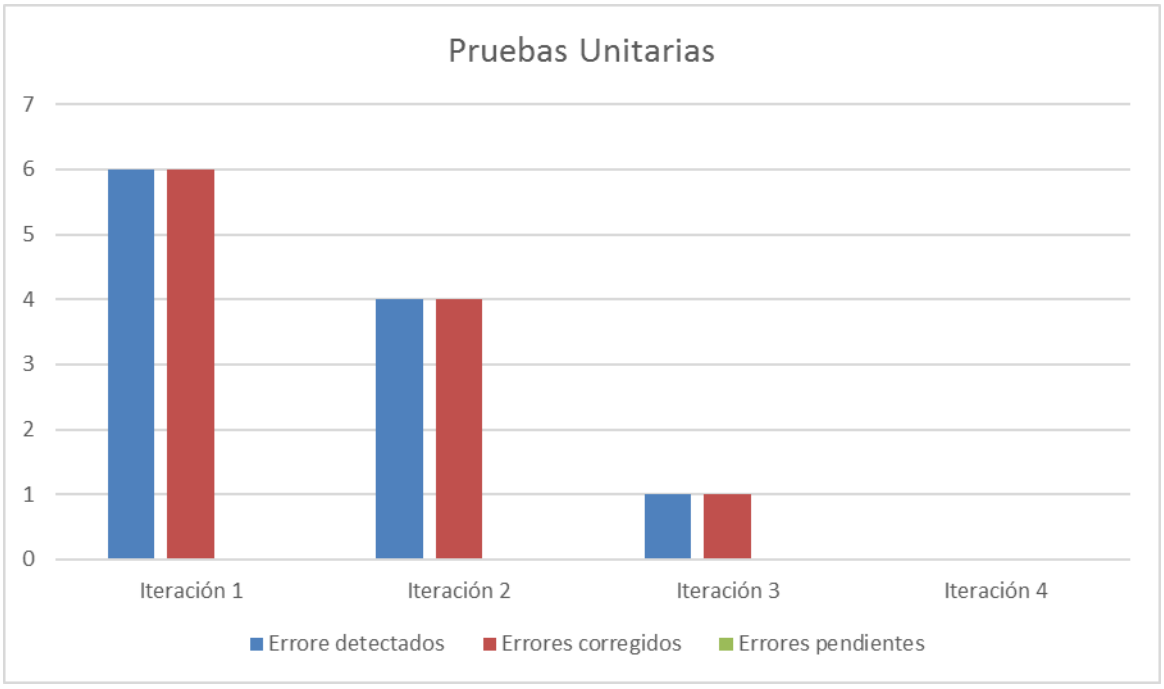


Figura 19: Resultados de las pruebas unitarias.

En las pruebas unitarias se realizaron cuatro iteraciones. En las tres primeras se detectaron 11 no conformidades, una vez dada solución a las mismas se realizaba la iteración siguiente. Finalmente se realizó una cuarta iteración donde no se presentaron no conformidades

Algunas de las principales no conformidades identificadas fueron:

- No se confeccionaba el mapa de caracteres especiales para la url.
- No se lograba analizar correctamente el formato de la fecha de detección del documento en Nutch.
- No se modificaba el score de la página de acuerdo al intervalo de tiempo de su identificación por el spider.
- No se detectaban enlaces repetidos en páginas que si lo contenían.

3.7 Conclusiones

Durante el desarrollo del capítulo se implementaron las funcionalidades necesarias para dar cumplimiento a los requisitos funcionales determinado por el cliente. Se logró una mayor legibilidad, limpieza y organización del código, gracias a los estándares de codificación utilizados. También se realizaron una serie de pruebas que permitieron la detección de errores que posteriormente fueron solucionados, todo esto arrojó como resultado un mejor desempeño del *plugin*.

CONCLUSIONES

- ✓ Luego del estudio del estado del arte se logró identificar el algoritmo a extender para el cálculo de la popularidad de los enlaces en Nutch.
- ✓ La realización de rastreos de prueba demostró que algunas de las páginas web no siguen buenas prácticas SEO, afectando el cálculo de la popularidad de los enlaces.
- ✓ Se identificó que muchas páginas presentan problemas en el tiempo de disponibilidad, lo cual llevó a reajustar la propuesta de solución.
- ✓ Se realizaron una serie de pruebas que permitieron la detección de errores que posteriormente fueron solucionados, permitiendo un mejor desempeño del *plugin.*

RECOMENDACIONES

Se recomienda el uso en el buscador Orión de un categorizador de documentos para posteriormente determinar la relevancia temática de los mismos.

REFERENCIAS BIBLIOGRÁFICAS.

ABITEBOUL, SERGE; PREDA, MIHAI; COBENA, GREGORY. Adaptive On-Line Page Importance Computation, 2001.

ALARCÓN SOTO, NATALIA M, REYES PILOTO, YASNIEL, MACÍAS SOTOLONGO, EDUARDO M. Sistema para la toma de decisiones del Webmaster en el ámbito del Posicionamiento Web. Pregrado, Universidad de las Ciencias Informáticas, La Habana, 2009. https://repositorio.uci.cu/jspui/handle/ident/TD_2350_09

ALEMÁN JIMÉNEZ, YULIO, THOMAS SOSA, YONIEL JORGE, ESTRADA VELAZCO, AYLIN, RODRÍGUES RUEDA, EYERIS, GUILLARD VARA YANISEL. Módulo de configuración para el mecanismo de rastreo del buscador Orión. Pregrado, Universidad de las Ciencias Informáticas, La Habana, 2015. https://repositorio.uci.cu/jspui/handle/ident/8919

APACHE NUTCH. Sitio oficial de Apache Nutch. [En línea]. The Apache Software Foundation, 2013. [Citado el: 14 de diciembre de 2016]. Disponible en: http://nutch.apache.org/index.html.

Apache, Solr. Apache Software Fundation. [En línea] 2015. [Citado el: 8 de diciembre de 2016.]. Disponible en: http://httpd.apache.org

ARCHANCO, RAMÓN. Como funciona un motor de búsqueda. [en línea]. Papeles de inteligencia, 2015. [citado el: 20 de octubre de 2016]. Disponible en: http://papelesdeinteligencia.com/como-funciona-un-motor-de-busqueda/

BAEZA-YATES, R. AND RIBEIRO-NETO, B. Modern information retrieval. New York: ACM Press; Harlow [etc.]: Addison-Wesley, 1999. ISBN 0-201-39829-X.

BAIJU, NT. Top 50 open source web crawlers for data mining. [en línea]. Nueva York, 2015. [Consultado el 30 de marzo de 2017]. Disponible en: http://bigdata-madesimple.com/top-50-open-source-web-crawlers-for-data-mining/

BAISLEY, B. 2006. *Unified Modeling Language Infraestructure.* s.l. : Pearson, 2006.

BALAREZO. *Metodologías Ágiles,Programación Extrema XP.* s.l. : Facultad de Ciencias Físicas y Matemáticas., 2013.

BARRIOLA, JUAN MANUEL; DOTTA, MILENA. How does google work? PageRank algorithm, graph diagrams and markov chains. Facultad de Ciencias Económicas, Buenos Aires, 2016. p.30.

CAMIÑO, RR. Motores de búsqueda sobre salud en internet. [En línea]. Motores de búsqueda,2003. [Consultado el 15 de marzo de 2017]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352003000500002

CARRODEGUAS, NORFI. Los directorios web de enlace en español en internet.[En línea].Publicación web, 2010.[Consultado el 30 de marzo de 2017]. Disponible en: https://norfipc.com/web/que-son-los-directorios-web-internet-funcion-utilidad-beneficios.html

CONSULTOR-SEO. Rankin de autor de Google. ¿Qué es? Herramientas SEO, 2017. [Consultado el 1 de abril de 2017]. Disponible en: http://www.consultor-seo.com/author-rank-google-que-es/

DURANTE F, MARCELO. Medidas y análisis del grafo de la web. Buenos Aires, 2007.

FERNÁNDEZ, H. Motores de búsqueda. [en línea]. Artículos, 2009. [Consultado el 24 de marzo de 2017]. Disponible en: http://especializacion.una.edu.ve/Internet/paginas/Lecturas/Fernandez.pdf

FONTELA, C; SUAREZ, P. Documentación y pruebas Antes del paradigma de objetos.2003

FRANCO X, SANTILLIAN. Aplicación de grafo en una red social**.** [en línea]. Grafo Facebook, 2012. [Consultado el: 30 de marzo de 2017]. Disponible en: https://es.slideshare.net/xsantillanf/grafo-facebook

GAMMA, ERICH; HELM, RICHARD; JOHNSON, RALPH; VLISSIDES, JHON. Design Patterns elements of reusable object oriented software. Grandy Booch, 1994. 429p

GÓNZALEZ, LIANET CABRERA. 2012. Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de apliaciones de gestión de información. [En línea] 2012. [Consultado el 30 de noviembre de 2016]. Disponible en: https://www.redib.org/recursos/Record/oai_articulo983403-extensión-visual-paradigm-uml-desarrollo-dirigido-modelos-aplicaciónes-gestión-información.

KOFLER, ROLAND. Scrapin the web with Nutch for elastic search, 2015. [en línea] [consultado el 22 de noviembre de 2016]. Disponible en: https://qbox.io/blog/scraping-the-web-with-nutch-for-elasticsearch

KORFHAGE, R, R. Information Storage and Retrieval. Michigan, Norton, 1997. p.368.

KOSTAKOPOULOU, D. Matters of Control: Integration Tests, Naturalisation Reform and Probationary Citizenship in the United Kingdom. Journal of Ethnic and Migration Studies, Vol. 36. 2010, 829p-847p.

KUMAR, S P. Integration of Web mining and web crawler: Relevance and State of Art. s.l. International Journal on Computer Science and Engineering, 2010. p. 772-776. Vol. 2. ISSN: 0975-3397.

MACIÁ, FERNANDO. La teoría de grafos: la más importante en búsqueda y de la que nadie habla. [en línea]. Londres, 2013. [Citado el: 15 de enero de 2017]. Disponible en: http://www.humanlevel.com/noticias/eventos/la-teoria-de-grafos-busqueda-search.html

MAHECHA, I. A. N. Buscador web open-source: Nutch. [En línea]. Universidad Nacional de Colombia, 2009. [Citado el: 3 de Diciembre de 2016]. Disponible en: http://dis.unal.edu.co/profesores/eleon/cursos/tamd/presentaciones/nutch.pdf/

MARTÍNEZ MÉNDEZ, FRANCISCO JAVIER. Recuperación de Información: Modelos, Sistemas y Evaluación. Murcia, José María Carbonell Arias, 2004. 100p.

MOYA, EVA. 14 meta-buscadores que te harán la vida más fácil. [En línea]. Monitorización, 2012.[Consultado el 30 de marzo de 2017]. Disponible en: http://inteligenciacomunicacionononline.blogspot.com/2012/03/10-meta-buscadores-meta-search-que-te.html

MYERS,G,J. The Art of Software Testing, Second Edition. Badgett and Todd M. Thomas with Corey Sandler. 2011

ORACLE. Características del IDE NetBeans. [En línea]. Sitio oficial del IDE NetBeans, 2014. [Consultado el: 25 de enero de 2017]. Disponible en: https://netbeans.org/features/index.html.

PELAEZ, JUAN CARLOS. Arquitectura basada en componentes.2009 [en línea]. [Consultado el 15 de febrero de 2017]. Disponible en: https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/

PÉREZ PORTO, JULIÁN; GARDEY, ANA. Definición de.[En línea]. Definición de Google, 2014. [Consultado el 20 de marzo de 2017]. Disponible en: http://definicion.de/google

PRESSMAN, S, R. Ingeniería del software un enfoque práctico. México. McGRAW- HILL INTERAMERICANA EDITORES, S.A. DE C.V. 2002. 57-70p

RAUTENSTRAUCH, RAMÓN. 2010. Opciones avanzadas en las búsquedas de Google y Bing.[En línea], 28 de Octubre de 2010. [Citado el: 6 de febrero de 2017.] Disponible en: http://www.apasionadosdelmarketing.es.

RICCIARDI, AGUSTÍN. Buscadores verticales, 2009. [En línea].[Consultado el 30 de marzo de 2017]. Disponible en: http://blog2puntocero.wordpress.com/2009/01/28/buscadores-verticales

ROASTBRIEF. La evolución de los medios de comunicación. [en línea]. Medios de comunicación, 2015. [Citado el: 16 de noviembre de 2016]. Disponible en: http://www.roastbrief.com.mx/2015/07/la-evolucion-los-medios-comunicacion/

ROMÁN HERNÁNDEZ, JOSÉ. Robots.txt. Todo lo que deberías saber.[en línea].Artículos, 2007.[Citado el: 20 de febrero de 2017].Disponible en: https://www.emezeta.com/articulos/robots-txt-todo-lo-que-deberia-saber

RUEDAS RODRÍGUEZ, EYERIS, IDALGO DELGADO YUSNIEL. Los spiders y su función en los motores de búsqueda. En: UciCiencia2012. La Habana: universidad de las Ciencias Informáticas, 2012, p. 1-12.

SALTON, GERARD; MCGILL, MICHAEL J. Introduction to Modern Information Retrieval. New York, Macgraw, 1986. P.353.

SILVA, HENRY. Incrementando la popularidad y los enlaces. [en línea]. Guía SEO para principiantes, 2016. [Consultado el: 16 de noviembre de 2016]. Disponible en: https://seo.pe/guia-seo-para-principiantes/incrementando-popularidad-enlaces/

Seta. Apache Solr:una introducción. [En línea] 2010. [Citado el: 22 de enero de 2017.] Disponible en: http://wwwdosideas.com/noticias/java/913-apache-solr-unaintroducción.html

SOMMERVILLE, IAN. *ngenieria de Software.* Madrid : Pearson Eduación S.A , 2005. 84-7829-074-5.

TOLOSA, G. H.; BORDIGNON, F. R. A. Introducción a la Recuperación de Información. 2007

TRAMULLA, JESÚS; OLVERA-LOBO, M.DOLORES. Recuperación dela información en Internet, 2001. Madrid ISBM 9788478974580.

TRAVIESO AGUILAR, MAYELÍN. Las publicaciones electrónicas: Una revolución en el siglo XXI.ACIMD[en línea].2003, vol. 11, n.2, p 1-2. http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1024-94352003000200001

YADA MONIKA, GOYAL NEGA. Comparison of open source Crawlers-A Review. International Journal of Scientific & Engineering Research, vol.6. September,2015. ISSN 2229-5518.